Preface

LANGUAGE technologies is an area of computer science, more specifically, of artificial intelligence, devoted to the computational analysis of human language, in its written or spoken form, and the corresponding practical applications in which computers reasonably deal with language data.

Applications of natural language technologies include search, translation, or summarization of texts; automatically obtaining answers to questions basing on the information present in huge collections of documents or in Internet, detecting trends and prevailing opinions or sentiments of people expressed in social networks with respect to a company's products or a government's actions, interfaces between the users and computers using normal language, detection of plagiarism, especially important in educational or academic settings, as well as numerous other applications.

Such applications save people time on reading or looking for information, improve quality of life of the users by, say, recommending them appropriate products or making it easy for people to use computers or other devices, increase the income for businesses by helping them better understand the consumers' needs, and improve democracy by timely informing decision makers, political parties and government of the citizens' opinions.

Many of these applications are based on a varying degree of understanding of the contents of the corresponding texts. Computational semantics is a branch of natural language processing that studies the techniques for extraction of meaning from natural language texts or building computational structures that reflect elements of the meaning of the text important for its computational treatment in practical applications.

This issue of Polibits features a special section on computational semantics and language technologies, which consists of the following six papers.

Arda Çelebi and Arzucan Özgür from Turkey introduce a novel concept on n-gram parsing and show that it improves the performance of a classical constituency parser. In addition, n-gram parsing is capable of producing partial parsing even if the whole sentence is not grammatically correct. This will be especially useful in applications related to user-contributed contents such as comments or blogs or to social networks such as Facebook and Twitter, given that in many cases texts from such sources cannot be parsed with classical parsers.

Marzieh Fadaee, Hamidreza Ghader, Heshaam Faili, and Azadeh Shakery from Iran present a statistical-based method for automatic construction of WordNet-like dictionaries. WordNet is a dictionary that provides information vital for most of the current language processing technologies. However, the original WordNet was developed for English. While WordNet-like dictionaries of varying quality and completeness exist for a small number of major languages, the great majority of languages, spoken by a considerable part of humankind, lack such a very important resource. Fadaee et al. show that for such a language as Persian, their method produces highly accurate WordNet-like dictionary.

Yanhui Gu, Zhenglu Yang, Miyuki Nakano, Masaru Kitsuregawa from Japan aim at boosting both the effectiveness and efficiency of measuring sentence similarity by combining various existing sentence similarity measuring techniques. They also introduce a number of optimization strategies and give insights into balancing of effectiveness and efficiency of measuring sentence similarity, a task of crucial importance for many natural language processing applications such as information retrieval, question answering, information extraction, text mining, text summarization, and machine translation, to name only a few.

Carlos Cobos and Martha Mendoza, and Elizabeth León from Colombia, Milos Manic from the USA, and Enrique Herrera-Viedma from Spain address the task of web page clustering. Web page clustering improves the user experience in information retrieval by presenting the user with a whole cluster of roughly equivalent or similar documents instead of individual documents, which helps the user to grasp all the alternatives at a glance and not dig into a long list of very similar pages. Cobos et al. report that 90% of the users agreed on that their method improves the search process.

Prashant Mathur, Nick Ruiz, Marcello Federico from Italy show how automatic translation of text segments can be used to help human translators in manual translation. Generally, automatic translation output needs extensive editing effort from human translators to reach the quality of manual translation. In many cases it is faster for the translator to type the translated text than to correct a suggestion from an automatic system; in other cases it is faster to correct the suggestion than to type the translation from scratch. Reading each suggestion and deciding whether it is easier to correct it or to discard it takes even more time and effort. Mathur et al. show how to automatically decide in which cases suggestions are likely to be useful and thus are to be shown to the translator for editing, and it which cases it is better to allow the user to type the translation without asking him or her explicitly.

Amitava Kundu, Dipankar Das, and Sivaji Bandyopadhyay from India address the problem of automatic understanding of the contents of movies. This task is important on the one hand for contents-based recommender systems in order to

recommend specific movies to users, and, on the other hand, it can be a source of information on real-world situations for computers to automatically learn common-sense knowledge. Kundu et al. present a method for detection the change of scene by analyzing the movie script.

The next four papers are regular papers on diverse topics of artificial intelligence, computer science, and automatic control.

Amir Elalouf and Eugene Levner from Israel and T.C.E. Cheng from Hong Kong discuss the problem of routing mobile agents in computer networks. Congestion in the network environment can cause unpredictable delays in information transfer and, in particular, in routing of mobile agents, which can be undesirable in time-sensitive applications. Elalouf et al. consider a problem of finding a path in the network that with some probability guarantees that the information will be passed within the specified time. They show that this is an NP-hard problem, and present algorithms for its exact or approximate solution.

María Bárbara Calva-Yáñez, Paola Andrea Niño-Suárez, Miguel Gabriel Villarreal-Cervantes, Gabriel Sepúlveda-Cervantes, and Edgar Alfredo Portilla-Flores from Mexico consider a problem of optimal control or a four-bar mechanism with spring and damping forces. They show how this dynamic optimization problem can be solved to with evolutionary techniques, specifically, a differential evolution algorithm. The efficiency of the proposed method is verified with a simulation experiment.

Mailyn Moreno Espino, Alternán Carrasco Bustamante, Alejandro Rosete Suárez and Marta Dunia Delgado Dapena from Cuba propose software development patterns for agent-based programs. Agents are active software components, which, in contrast to objects, not only react to external requests but perform actions when they deem appropriate judging on the state of the system and the internal state of the agent. While object-oriented programming is now a mature and stable area with decades of experience, agent-oriented programming is still novel and lacks the variety of well-established methodologies that object-oriented programming has. Moreno Espino et al. show how agent-oriented elements can be incorporated into well-established object-oriented methodology.

L.E. Gómez, J.H. Sossa, R. Barrón, F.J. Cuevas, J.F. Jiménez from Mexico present their improvements to music recommendation, which is an application with very high commercial demand and which affect the quality of life of a considerable part of humankind. Music recommendation systems help people choose, or just automatically offer, the music pieces or songs to listen in their everyday environment, basing on the knowledge of their preferences extracted from various sources, such as the history of songs the user has chosen in the past. Most often the selection is done basing on the metadata associated with the song, such as genre, author, singer, year, etc., which are numbers or text strings. Gómez et al. show that with the user of artificial neural networks, namely dynamic neural networks, it is possible to compare and recommend music basing on the acoustic properties only and not on metadata. This can lead to better recommendation of songs lacking metadata at all, and to improved recommendation of songs that have them.

This issue of Polibits will no doubt be useful for researchers, students, and general public interested in artificial intelligence, especially computational semantics and natural language processing, as well as other areas of computer science and automatic control.

Efstathios Stamatatos

Assistant Professor,
Department of Information and
Communication Systems Engineering,
University of the Aegean, Greece

N-gram Parsing for Jointly Training a Discriminative Constituency Parser

Arda Çelebi and Arzucan Özgür

***Abstract*—Syntactic parsers are designed to detect the complete syntactic structure of grammatically correct sentences. In this paper, we introduce the concept of n-gram parsing, which corresponds to generating the constituency parse tree of n consecutive words in a sentence. We create a stand-alone n-gram parser derived from a baseline full discriminative constituency parser and analyze the characteristics of the generated n-gram trees for various values of n. Since the produced n-gram trees are in general smaller and less complex compared to full parse trees, it is likely that n-gram parsers are more robust compared to full parsers. Therefore, we use n-gram parsing to boost the accuracy of a full discriminative constituency parser in a hierarchical joint learning setup. Our results show that the full parser jointly trained with an n-gram parser performs statistically significantly better than our baseline full parser on the English Penn Treebank test corpus.**

***Index Terms*—Constituency parsing, n-gram parsing, discriminative learning, hierarchical joint learning.**

I. Introduction

PARSING a natural language sentence is a process of characterizing the syntactic description of that sentence based on the syntax of its language. Over the last half-century, there have been many techniques developed to improve parsing accuracy. Some of the studies have targeted the model that the parser relies on, such as by replacing rule-based approaches [1], [2] with statistical models like generative [3], [4] and discriminative ones [5], [6]. Others introduced external ways of boosting the parser, such as by using a reranker [7], [8], by bootstrapping it with itself in a self-training setup [9], or by using partial parsing in a co-training setup [10]. Another recent thread of research is about a more specialized form of the co-training approach, where multiple models from different domains are jointly trained together and help each other to do better. One example is [11], where they introduce the Hierarchical Joint Learning (HJL) approach to jointly train a parser and a named entity recognizer. Their HJL model achieved substantial improvement in parsing and named entity recognition compared to the non-jointly trained models.

In this paper, we aim to improve the accuracy of a discriminative constituency parser by training it together with another parser in the HJL setup. While our actual parser works on complete sentences, its accompanying parser tackles the parsing task in a less complex way, that is, by parsing n-grams instead of complete sentences. To the best of our knowledge, this is the first study that introduces the concept of n-gram parsing. Even though syntactic parsers expect grammatically correct and complete sentences, an n-gram parser is designed to parse only n consecutive words in a sentence. An outputted n-gram tree is still a complete parse tree, but it covers only n words instead of the whole sentence. We derive our n-gram parser from a discriminative parser which was implemented based on [12]. After analyzing the characteristics of n-gram parsing, we train the full parser together with the n-gram parser. Our underlying hypothesis is that the n-gram parser will help the full parser at cases where the n-gram parser does better. We performed experiments with different n-gram sizes on the English Penn treebank corpus [13] and obtained a statistically significant increase in the accuracy of the jointly trained full parser over the original (non-jointly trained) full parser.

This paper continues with the related studies. Following that, in Section III, we introduce the concept of n-gram parsing and the characteristics of the n-gram trees. In Sections IV and V, we describe how we perform discriminative constituency parsing and how we use the HJL approach, respectively. Before discussing the experiments, we introduce the data and the evaluation methods that we used in Section VI. We present the experimental results obtained with the n-gram parser alone and the jointly trained parser. We conclude and outline future directions for research in the last section.

II. Related Work

In this paper we tackle the problem of improving the performance of a discriminative constituency parser by training it with an n-gram parser using hierarchical joint learning. Although generative models [3], [4] still dominate the constituency parsing area due to their faster training times, a number of discriminative parsing approaches have been proposed in the recent years motivated by the success of discriminative learning algorithms for several NLP tasks such as part-of-speech tagging and relation extraction. An advantage of discriminative models is their ability to incorporate better feature rich representations. There are three different approaches for applying discriminative models to the parsing task. The first and perhaps the most successful one

Manuscript received on December 7, 2012; accepted for publication on January 11, 2013.

Arda Çelebi and Arzucan Özgür are with Department of Computer Engineering, Boğaziçi University, Bebek, 34342 İstanbul, Turkey (e-mail: {arda.celebi, arzucan.ozgur}@boun.edu.tr).

is to use a discriminative reranker to rerank the n-best list of a generative parser [5], [7], [6]. To our knowledge, the forest-based reranker in [8] is the best performing reranker that helps its accompanying parser to achieve an F_1 score of 91.7%[1]. The second approach considers parsing as a sequence of independent discriminative decisions [14], [15]. By discriminative training of a neural network based statistical parser, an F_1 score of 90.1% is obtained in [15]. The third approach, which we adapted for this paper from [12], is to do joint inference by using dynamic programming algorithms in order to train models and use these to predict the globally best parse tree. With this, an F_1 score of 90.9% is achieved in [12]. Due to their notoriously slow training times, however, most discriminative parsers run on short sentences. That is why we use sentences that have no more than 15 words for full sentence parsing in this study.

One of the aspects of our research is that it involves working with multiple parsers at the same time, that is the n-gram parser and the full sentence parser. There have been a couple of studies that experimented with multiple parsers in the literature. One example is [16] which extended the concept of rerankers by combining k-best parse outputs from multiple parsers and achieved an F_1 score of 92.6%. In [17], Fossum and Knight worked on multiple constituency parsers by proposing a method of parse hybridization that recombines context-free productions instead of constituents in order to preserve the structure of the output of the individual parsers to a greater extent. With this approach, their resulting parser achieved an F_1 score of 91.5%, which is 1 point better than the best individual parser in their setup.

Another thread of research related to ours is jointly training multiple models. This evolved from the concept of multi-task learning, which is basically explained as solving multiple problems simultaneously. In the language processing literature, there have been a couple of studies where this concept is adapted for multi-domain learning [18], [19], [20]. In these studies, they make use of labelled data from multiple domains in order to improve the performances on all of them. In [11], for example, a joint discriminative constituency parser alongside a named-entity recognizer is used and substantial gains over the original models is reported. Like most of the prior research, a derivative of the hierarchical model, namely the Hierarchical Joint Learning (HJL) approach is used. In this paper, we adapted this approach by replacing the named-entity recognizer with an n-gram parser, which to our knowledge hasn't been attempted before in the literature.

The most distinguishing contribution of our research is the introduction of n-gram parsing. To the best of our knowledge, n-gram parsing has never been considered as a stand-alone parsing task in the literature before. One reason might be that n-gram trees have no particular use on their own. However, they have been used as features for statistical models in either lexicalized or unlexicalized forms. For example, in [9] they are used to train the reranking model of the self-training pipeline. There are only a couple of studies in the literature comparable to our notion of n-gram trees. One of them is the stochastic tree substitution grammars (TSG) used in Data Oriented Parsing (DOP) models in [21]. However, unlike TSG trees, our n-gram trees always have words at the terminal nodes. Another related concept is the Tree Adjunct Grammar (TAG) and the concept of local trees proposed in [22]. As in the case of TSG trees, TAG local trees also differ from our n-gram trees by not having words at all terminal nodes but one. Another related study was performed in [23], where significant error reductions in parsing are achieved by using n-gram word sequences obtained from the Web.

In the literature, the concept of n-grams is used in a number of contexts to represent a group of n consecutive items. This can be, for instance, characters in a string or words in a sentence. In our research, we consider an n-gram as n consecutive words selected from a sentence. n-gram parsing, then, refers to the process of predicting the syntactic structure that covers these n words. We call this structure an n-gram tree. In this paper, we study the parsing of 3- to 9-grams in order to observe how n-gram parsing differs with length and at which lengths the n-gram parser helps the full parser more. A sample 4-gram tree which is extracted from a complete parse tree is shown in Figure 1. Compared to complete parse trees, n-gram trees are smaller parse trees with one distinction. That is, they may include constituents that are trimmed from one or both sides in order to fit the constituent lengthwise within the borders of the n-gram. We call such constituents incomplete, and denote them with the -INC functional tag. n-gram parsing is fundamentally no different than the conventional parsing of a complete sentence. However, n-grams, especially the short ones, may have no meanings on their own and/or can be ambiguous due to the absence of the surrounding context. Even though the relatively smaller size of n-gram trees makes it easier and faster to train on them, their incomplete and ambiguous nature makes the n-gram parsing task difficult. Despite all, n-gram parsing can still be useful for the actual full sentence parser, just like the partial parsing of a sentence used for bootstrapping [10]. In this paper, we train the full parser together with an n-gram parser and let the n-gram parser help the full parser at areas where the n-gram parser is better than the full one.

A. N-gram Tree Extraction Algorithm

In order to generate a training set for the n-gram parser, we extract n-gram trees out of the complete parse trees of the Penn Treebank, which is the standard parse tree corpus used in the literature [13]. Since we use sentences with no more than 15 words (WSJ15) for complete sentence parsing, we use the rest of the corpus (WSJOver15) for n-gram tree extraction.

The pseudocode of our n-gram tree extraction algorithm is given in Fig. 2. It takes a complete parse tree as

[1]The performance scores reported in this section are for Section 23 of the English Penn Treebank.

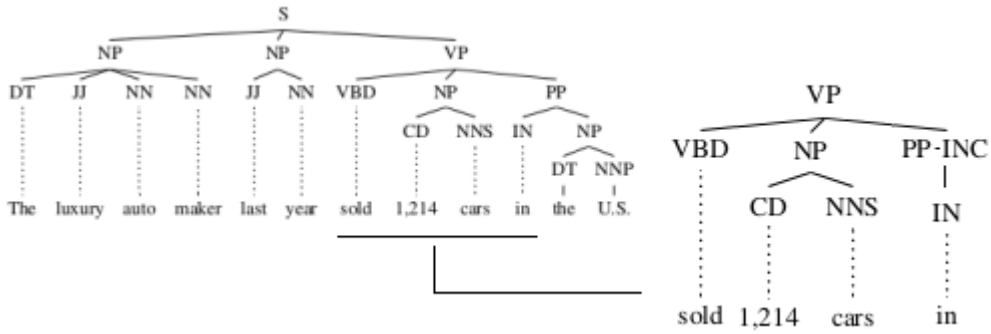


Fig. 1. Sample 4-gram tree extracted from a complete parse tree

Require: n, width of the n-gram trees
Require: $tree$, parse tree of a sentence
  $len \leftarrow$ length of the given sentence
  $i \leftarrow 0$
  while $i < len - n$ **do**
    $subtree \leftarrow$ get subtree that covers $[i, i+n]$ span
    $trimmed \leftarrow$ trim $subtree$'s constituents outside the $[i, i+n]$ span, if any
    if $trimmed$ has any const. with no head child **then**
      i++ and continue
    end if
    $markedtree \leftarrow$ mark all trimmed consts. as incomplete
    $filtered \leftarrow$ filter out incomplete unary rule chain from the ROOT, if any
    save $filtered$ tree as n-gram tree
    i++
  end while

Fig. 2. Extracting and storing n-gram trees from a parse tree

input and returns all the extracted valid n-gram trees. It starts by traversing the sentence from the first word and preserves the minimum subtree that covers n consecutive words. While doing that, it may trim the tree from one or both sides in order to fit the constituents lengthwise within the borders of the n-gram. Hence, the extracted n-gram trees may contain incomplete and thus ungrammatical constituents, which is not something that a conventional parser expects as input. Nevertheless, we assume that not all of the trimmed constituents are ungrammatical according to the concept of generatively accurate constituents that we introduce in this paper. This concept stems from the concept of the head-driven constituency production process [3] where a constituent is theoretically generated starting from its head child and continuing towards the left and right until all children are generated. If the head child is the origin of the production, then it is safe to say that it defines the constituent. Therefore, our n-gram tree extraction process makes sure that the head child is still included in the trimmed constituents. Otherwise, the whole n-gram tree is considered generatively inaccurate and is thus discarded. If all the heads are preserved, the algorithm marks the trimmed constituents with the -INC functional tag. For example, the PP constituent of the 4-gram tree in Figure 1 is trimmed from right hand side and since the head child "IN" is still included in the constituent, it is considered generatively accurate. The corresponding constituent is marked with an -INC tag and the extracted tree is stored as a valid 4-gram tree. However, if we try to extract the next 4-gram in the same sentence, it would have failed due to not being able to keep the head of the rightmost NP. The -INC tags are later used in the features in order to let the n-gram parser better predict such incomplete cases. Following these steps, the extraction process filters out the incomplete chains of unary rules that can be reached from the $ROOT$ constituent. The algorithm also keeps the parent information of the $ROOT$ constituent of the n-gram tree as an additional context information.

B. Characteristics of n-gram Trees

As we apply the extraction algorithm on the WSJOver15 portion of the Penn Treebank, we get hundreds of thousands of n-gram trees for each value of n in $\{3..9\}$. The analysis of these data sets reveals interesting points about the characteristics of such n-gram trees. Table I gives the percentages of the most common constituents in each n-gram training set along with the corresponding numbers obtained from the complete parse trees of the WSJ15 portion, which we use for training our full parser. The comparison indicates that the percentages of the noun (NP), verb (VP), and prepositional (PP) phrases in the n-gram trees are higher than the ones in the complete parse trees. On the other hand, the percentages of long-range constituents like S are lower for the n-gram trees, which is expected as the extraction process disfavors such constituents. Nonetheless, we see higher percentage in case of another long-range constituent SBAR, which exemplifies how the extraction process may still favor some long-range constituents. Based on this analysis, we may postulate that the increasing percentage of the NPs, VPs, and PPs per parse tree may help the n-gram parser do a better job in addition to the fact that they are smaller, thus less complex phrases.

TABLE I
PERCENTAGE OF THE MOST COMMON NON-TERMINAL CONSTITUENTS IN TRAINING SETS

Model	NP	VP	PP	ADJP	ADVP	S	SBAR	QP
3-gram	21.20	10.82	6.25	1.04	1.03	4.68	1.43	0.96
4-gram	20.69	10.87	6.44	1.05	1.07	4.93	1.64	0.85
5-gram	20.39	10.87	6.45	1.06	1.10	5.11	1.75	0.80
6-gram	20.11	10.81	6.49	1.04	1.10	5.21	1.84	0.79
7-gram	19.86	10.78	6.49	1.02	1.11	5.29	1.92	0.78
8-gram	19.68	10.74	6.50	1.01	1.11	5.35	1.99	0.77
9-gram	19.54	10.68	6.50	1.00	1.10	5.39	2.04	0.76
Full (WSJ15)	18.74	9.51	4.21	1.05	1.69	6.91	1.00	0.60

III. DISCRIMINATIVE CONSTITUENCY PARSING

In order to parse the n-grams and the complete sentences, we implemented a feature-rich discriminative constituency parser based on the work in [12]. It employs a discriminative model with the Conditional Random Field (CRF) based approach. Discriminative models for parsing maximize the conditional likelihood of the parse tree given the sentence. The conditional probability of the parse tree is calculated as in Equation 1, where $\mathcal{Z}_s$ is the global normalization function and $\phi(r|s;\theta)$ is the local clique potential:

$$P(t|s;\theta) = \frac{1}{\mathcal{Z}_s} \prod_{r \in t} \phi(r|s;\theta), \tag{1}$$

where

$$\mathcal{Z}_s = \sum_{t' \in \tau(s)} \prod_{r \in t'} \phi(r|s;\theta), \tag{2}$$

$$\phi(r|s;\theta) = exp \sum_i \theta_i f_i(r,s). \tag{3}$$

The probability of the parse tree t given the sentence s is the product of the local clique potentials for each one-level subtree r of a parse tree t which is normalized by the total local clique potential of all possible parse trees defined by $\tau(s)$. Note that the clique potentials are not probabilities. They are computed by taking the exponent of the summation of the parameter values θ for the features that are present for a given subtree r. The function $f_i(r,s)$ returns 1 or 0 depending on the presence or absence of feature i in r, respectively. Given a set of training examples, the goal is to choose the parameter values θ such that the conditional log likelihood of these examples, i.e., the objective function $\mathcal{L}$ given in Equation 5, is minimized:

$$\begin{aligned} \mathcal{L}(\mathcal{D};\theta) &= \sum_{(t,s) \in \mathcal{D}} \left(\sum_{r \in t} \langle f(r,s), \theta \rangle - logZ_{s,\theta} \right) \\ &- \sum_i \frac{\theta_i^2}{2\sigma^2}. \end{aligned} \tag{4}$$

When the partial derivative of our objective function with respect to the model parameters is taken, the resulting gradient in Equation 5 is basically the difference between the empirical counts and the model expectations, along with the derivative of the L_2 regularization term to prevent over-fitting. These partial derivatives which are calculated with the inside-outside algorithm by traversing all possible parse trees for a given sentence are then used to update the parameter values at each iteration. As in [12], we use stochastic gradient descent (SGD), which updates parameters with a batch of training instances instead of all in order to converge to the optimum parameter values faster:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \sum_{(t,s) \in \mathcal{D}} \left(\sum_{r \in t} f_i(r,s) - E_\theta[f_i|s] \right) - \frac{\theta_i}{\sigma^2}. \tag{5}$$

We use the same feature templates of [12] and the same tool [24] to calculate the distributional similarity clusters which are used in the feature definitions. However, we use a different combination of corpora to calculate these clusters. We gathered an unlabelled data set of over 280 million words by combining Reuters RCV1 corpus [25], Penn treebank, and a large set of newswire articles downloaded over the Internet. Despite the difference, we tried to keep the size and the types of contents comparable to [12]. We use the default parameter settings for the tool provided by [24] and set the number of clusters to 200. In order to handle out-of-vocabulary (OOV) words better, we also introduce a new lexicon feature template $\langle$ *prefix, suffix, base(tag)* $\rangle$, which makes use of the most common English prefixes and suffixes. A feature is created by putting together the prefix and suffix of a word, if available, along with the base tag of that word. If it does not have any prefixes or suffixes, NA is used, instead. As for n-gram parsing, we did not include or exclude any features. Like in [12], we also implemented chart caching and parallelization in order to save time.

IV. HIERARCHICAL JOINT LEARNING

In this section, we show how we jointly train the n-gram parser and the full parser. We use an instance of the multi-task learning setup called the Hierarchical Joint Learning (HJL) approach introduced in [11]. HJL enables multiple models to learn more about their tasks due to the commonality among the tasks. By using HJL, we expect the n-gram parser to help the full parser in cases where the n-gram parser is better.

As described in [11], the HJL setup connects all the base models with a top prior, which is set to zero-mean Gaussian in our experiments. The only requirement for HJL is that the base models need to have some common features in addition to the set of features specific to each task. As both parsers employ the same set of feature templates, they have common features

through which HJL can operate. All the shared parameters between base models are connected to each other through this prior. It keeps the values of the shared parameters from each base model close to each other by keeping them close to itself.

The parameter values for the shared features are updated by incorporating the top model feature into the parameter update function as in Equation 6. While the first term is calculated by using the update value from Equation 5, the second term ensures that the base model m is not getting apart from the top model by taking the difference between the top model and the corresponding shared parameter value. The variance σ_d^2 is a parameter to tune this relation:

$$\frac{\partial \mathcal{L}_{hier}(\mathcal{D};\theta)}{\partial \theta_{m,i}} = \frac{\partial \mathcal{L}_{hier}(\mathcal{D}_m,\theta_m)}{\partial \theta_{m,i}} - \frac{\theta_{m,i} - \theta_{*,i}}{\sigma_d^2}. \quad (6)$$

As shown in Equation 7, the updates for the top model parameter values are calculated by summing the parameter value differences divided by the base model variance σ_m^2, and then by subtracting the regularization term to prevent overfitting:

$$\frac{\partial \mathcal{L}_{hier}(\mathcal{D};\theta)}{\partial \theta_{*,i}} = \left(\sum_{m \epsilon \mathcal{M}} \frac{\theta_{*,i} - \theta_{m,i}}{\sigma_m^2} \right) - \frac{\theta_{*,i}}{\sigma_*^2}. \quad (7)$$

As in the case of the discriminative parser described in the previous section, SGD is used for faster parameter optimization. At each epoch of SGD, a batch of training instances is selected uniformly from each model in the setup. The number of training instances coming from each set, hence, depends on the relative sizes of the training sets.

V. Experimental Setup

A. Data

We evaluate our models by using the English Penn treebank corpus [13]. Like previous studies in this field, we use sections 02–21 for training, 22 for development, and 23 for testing. For complete sentence parsing, we use only the sentences that have no more than 15 words, that is WSJ15. To train the n-gram parsers, on the other hand, we use the rest of the Penn treebank, which we call WSJOver15. To test the n-gram parsers, we use the n-gram trees extracted from the development and the test sets of the WSJ15 in order to make the results more comparable with the full parser. By using our n-gram tree extraction algorithm, we extract n-gram trees for $n = [3, 9]$. Table II gives the number of parse trees in the training, development and test sets of each parser.

B. Evaluation

We use the *evalb* script[2] to get labelled precision, recall, and F_1 scores. These are calculated based on the number of nonterminals in the parser's output that match those in the standard/golden parse trees. We also report the percentage of completely correct trees, the average number of brackets crossing with the actual correct spans, and the percent of guessed trees that have no crossing brackets with respect to the corresponding gold tree. In order to better understand the n-gram parser and the jointly trained parser, we also evaluate how accurately these parsers handle different types of constituents.

[2]The *evalb* script is available on http://nlp.cs.nyu.edu/evalb.

TABLE II
Number of parse trees for each parser

Model	Training Set	Dev. Set	Test Set
3-gram	384,699	1,742	2,495
4-gram	318,819	1,341	1,916
5-gram	267,155	1,050	1,506
6-gram	227,505	807	1,158
7-gram	195,229	635	891
8-gram	168,075	486	667
9-gram	145,040	349	491
Full	9,753	421	603

VI. Experiments and Discussion

A. Baseline Parser

Our baseline parser is a discriminative constituency parser that runs on complete sentences. In order to make it run at its best, we set the learning factor η to 0.1 and the variance σ^2 to 0.1. We do 20 passes over the training set and use the batch size of 15 for the purpose of SGD. Table III shows the results we obtained with these settings on the development and test sets of WSJ15 portion of the Penn treebank. Our baseline parser achieves an F_1 score of 87.8% on the development set and 86.4% on the test set. Compared to the results obtained in [12], our implemented version's performance is a couple of points behind. The difference might be caused by small implementation details as well as by the different corpus that we used to calculate the distributional similarity clusters as discussed in Section IV.

TABLE III
Results on the Penn treebank

Dataset	Precision	Recall	F_1 score
Dev. Set	87.5	88.1	87.8
Test Set	86.4	86.4	86.4

B. N-gram Parser

Before training the full parser with the n-gram parser using HJL, we test the stand-alone n-gram parsers in order to understand where they are good at or where they fail, especially with respect to the full parser. We experimented with seven different n-gram sizes, i.e., with $n = [3, 9]$. Even though there are hundreds of thousands of training instances for each parser available from the WSJOver15 portion of the Penn treebank, we train our models with $20,000$ instances due to time and computational constraints. For a statistically more reliable evaluation, we report the averages of the scores obtained from five randomly selected versions of each training set.

TABLE IV
RESULTS OF THE n-GRAM PARSERS FOR THE DEVELOPMENT SET

Model	Precision	Recall	F_1 score	Exact	Avg CB	No CB	TagAcc
3-gram	86.37	86.60	86.49	73.13	0.02	98.15	86.80
4-gram	85.55	85.89	85.72	66.50	0.08	94.86	87.88
5-gram	86.91	86.29	86.60	61.68	0.10	93.02	89.95
6-gram	86.68	86.41	86.54	54.76	0.16	89.33	90.67
7-gram	87.49	87.00	87.24	51.29	0.21	86.52	92.17
8-gram	87.12	86.77	86.94	49.48	0.28	83.05	92.63
9-gram	87.69	86.96	87.33	46.68	0.35	79.41	92.91

TABLE V
F_1 SCORES ON THE MOST COMMON CONSTITUENTS FROM THE DEVELOPMENT SET

Model	NP	VP	PP	S	SBAR	ADVP	ADJP	QP
3-gram	89.26	89.86	92.04	84.69	56.31	73.19	46.50	75.27
4-gram	88.02	89.79	90.72	83.52	61.43	73.23	48.23	75.77
5-gram	88.35	90.42	89.81	85.77	72.39	76.31	52.22	72.96
6-gram	87.65	91.04	89.03	86.11	71.49	75.16	52.00	74.34
7-gram	87.80	91.73	88.40	87.39	77.91	76.87	50.09	84.45
8-gram	87.22	92.29	87.81	87.18	76.43	77.84	49.05	86.73
9-gram	87.79	91.76	87.33	87.46	76.37	72.21	53.66	86.43
Full	88.77	89.81	88.79	90.91	80.56	79.42	59.31	94.21

We use the same experimental setup of the baseline full parser. However, we optimize the parameters specifically for n-gram parsing. To make the n-gram parser run at its best, we set the learning factor η to 0.05 and the variance σ^2 to 0.1. Instead of doing 20 iterations like we did for the full parser, we observe that 10 iterations are enough. We choose a batch size of 30, instead of 15 for the SGD. Both decisions are related to the fact that the n-gram trees are relatively smaller compared to the complete parse trees. Thus, an n-gram parser requires a larger batch of training instances, but takes fewer iterations to get to its best performance.

Table IV shows the averaged F_1 scores obtained with all seven n-gram parsers on the development set. The comparison of our n-gram parsers with each other reveals a couple of interesting points. Firstly, using bigger n-gram trees in general leads to slightly higher F_1 scores, but the increase in precision is more apparent. Secondly, the fact that the 3-gram parser achieves an F_1 score of 86.5% by guessing 73.13% of the parse trees exactly, suggests that finding the exact n-gram tree is mostly an easy job, yet a small set of 3-gram trees contain most of the errors. This observation does not hold for larger n values, since the parsing task becomes more difficult for bigger trees.

In order to do further analysis, we investigate how accurately the n-gram parsers handle the different types of constituents. Table V shows the average F_1 scores of each n-gram model for the most common constituents. The first thing to notice is the degrading performance of handling noun (NP) and prepositional (PP) phrases, and the improving performance of handling verb phrases (VP) and declarative clauses (S) as n increases. When n increases, longer as well as more complex NPs and PPs are introduced. This results in degrading performance for such phrases.

On the other hand, as the sizes of the n-gram trees increase, it becomes easier to handle long-range constituents like VPs and Ss, since the parser sees more of them in the training set. The same argument holds for the remaining types of constituents in Table V. Another interesting point is the significantly lower accuracies of the n-gram parsers on QPs, especially with smaller n-gram trees.

TABLE VI
ACCURACY ON THE INCOMPLETE CONSTITUENTS IN THE DEVELOPMENT SET

Model	% of Incomplete Constituents in Golden Trees	Incomplete Constituent Accuracy	% of Unidentified Incomplete Const. w.r.t. All Unidentifieds
3-gram	22.0	86.65	26.2
4-gram	17.4	85.78	21.1
5-gram	14.5	84.20	16.7
6-gram	12.3	83.41	15.2
7-gram	10.8	83.32	13.9
8-gram	9.5	83.92	11.5
9-gram	8.7	84.94	10.4

Table VI shows the performances of the n-gram parsers on the incomplete constituents, as well as the percentages of constituents that are incomplete and the percentages of unidentified constituents from the golden trees that are incomplete. In most cases, as n increases, the accuracies on the incomplete constituents decreases. The contribution of the incomplete constituents to the number of unidentified constituents decreases as well.

However, this is more attributed to the fact that their percentage with respect to all constituents drops as n increases. Another point to notice is that despite its high performance, more than a quarter of the constituents unidentified by the 3-gram parser are incomplete. Considering that the 3-gram parser predicts 73.13% of the parse trees completely, it is highly likely that the performance of the 3-gram parser is affected by such constituents.

TABLE VII
AVERAGED F_1 SCORES OF THE BASELINE FULL PARSER (B) JOINTLY TRAINED WITH EACH n-GRAM MODEL

	Results for Dev. Set of the WSJ15				Results for Test Set of the WSJ15			
Model(s)	1K	2K	5K	10K	1K	2K	5K	10K
B+3-gram	87.60	87.69	87.97	87.91	86.37	86.07	86.23	86.21
B+4-gram	87.93	87.98	**87.99***	87.70	86.52	**86.42*****	86.44	86.54
B+5-gram	87.72	87.67	88.00	87.72	86.36	85.84	86.35	86.33
B+6-gram	87.88	87.73	**88.12****	87.66	86.55	85.95	86.24	86.31
B+7-gram	87.83	87.94	88.05	87.72	86.58	86.16	86.24	86.42
B+8-gram	87.93	87.91	87.96	87.78	**86.57****	86.45	86.16	86.43
B+9-gram	**88.19***	87.89	87.89	87.86	86.46	86.42	86.44	**86.59*****

TABLE VIII
F_1 SCORES OF THE JOINTLY TRAINED PARSER ON THE MOST COMMON CONSTITUENTS IN THE DEVELOPMENT SET

Model(s)	NP	VP	PP	S	SBAR	ADVP	ADJP	QP
B+3-gram	88.91	89.87	89.39	90.20	80.09	80.05	64.51	92.44
B+4-gram	88.82	90.30	89.43	90.26	81.00	79.51	61.65	92.14
B+5-gram	89.10	90.21	89.23	90.22	79.44	79.56	61.23	92.68
B+6-gram	89.19	90.15	89.30	90.26	80.55	79.75	63.18	93.07
B+7-gram	89.04	90.19	89.15	90.35	80.73	79.31	62.93	93.03
B+8-gram	88.96	90.17	88.90	90.27	80.91	79.26	61.11	92.48
B+9-gram	88.86	90.14	89.06	90.12	79.45	79.07	62.38	92.74
Baseline (B)	88.77	89.81	88.79	90.91	80.56	79.42	59.31	94.21

C. Jointly Trained Parser

In order to boost the accuracy of the full parser, we train it along with each n-gram parser. For the full and n-gram models, we use the previously used variance settings, that is 0.1. We set the top model variance σ_*^2 to 0.1 as well. We set the learning factors for the n-gram models and the top model to 0.1, whereas we use 0.05 for the full parsing model. With a lower learning rate, we make sure that the full parsing model starts to learn at a slower pace than usual so that it doesn't directly get into the effect of the accompanying n-gram model. As in the case of the baseline full parser, we do 20 passes over the training set and at each iteration, we update the parameters with a batch of 40 training instances gathered from all training sets in the setup.

In order to evaluate the effect of the training set size for each n-gram model, we use training sets of four different sizes for the n-gram parsers. We execute each experiment three times with randomly selected training sets. Table VII shows the averaged F_1 scores obtained by the jointly trained full parser on the development and test sets of the WSJ15. The rows indicate which models are trained together, whereas each column corresponds to a training set of different size for the n-gram model. In case of the full parser, we use the standard training set of the Penn treebank, which contains $9,753$ instances.

Scores in bold in Table VII indicate that the value is significantly[3] better than the baseline value according to the t-test. When we compare the results with the baseline F_1 score of 87.8% on the development set and 86.4% on the test set, we observe slight improvement at some of the configurations. In general, the jointly trained full parser outperforms the baseline parser when it is trained alongside an n-gram parser that uses a relatively smaller training set, like 5,000 instances for the development set and 1,000 instances for the test set. The best results though, are obtained by jointly training the baseline parser with the 9-gram parser. These results are statistically significantly better than the ones of the non-jointly trained full parser both for the development and test sets. In addition to these comparisons, we also observed that within 20 iterations, the jointly trained full parser reaches its best performance faster with respect to the baseline parser, which shows the push of the n-gram parser over the full parser.

We also analyze how accurately the jointly trained parser[4] handles different constituent types. Table VIII shows the averaged F_1 scores for the most common constituent types in the development set. The results indicate a couple of interesting reasons behind the slight improvement of the jointly trained full parser over the baseline. The first one is the slight improvement on NPs as the n-grams are getting bigger, which is especially visible with the best performing configuration among them, that is the one with the 6-gram model. PPs and VPs are also better processed with almost all jointly trained models. The biggest improvement is seen with the adjective phrases (ADJPs), especially when smaller n-grams are used. Even though the impact of ADJPs to the overall result is small compared to the other phrase types like NPs and PPs, this improvement is still worth mentioning. It is interesting to note that the same analysis on the stand-alone n-gram parsers reveals that they are not that good with ADJPs. Another thing to notice is the degrading performance over the QPs, as well as SBAR and S type constituents due to the fact that the n-gram parsers perform relatively worse on them (see Table V).

[3]The superscript * adjacent to the F_1 scores indicates that its significance is $p < 0.01$. In case of ** and ***, it is $p < 0.005$ and $p < 0.001$, respectively.

[4]Each accompanying n-gram parser in the HJL setup uses 5,000 training instances.

VII. Conclusion and Future Work

In this paper, we introduced n-gram parsing and analyzed how it is different than full sentence parsing. We observed that the bigger n-grams we use, the better accuracies we get, mostly due to increasing context information. We showed that the n-gram parsers are better than the full parser at parsing NPs, VPs, and PPs, but worse at parsing Ss and SBARs. After analyzing the stand-alone n-gram parsers, we used them for jointly training a full discriminative parser in the HJL setup in order to boost the accuracy of the full parser. We achieved statistically significant improvement over the baseline scores. The analysis of the results obtained with the jointly trained parser revealed that the resulting parser is better at processing NPs, VPs, PPs, and surprisingly ADJPs. However, it is negatively influenced by the performance of the n-gram parser over constituents like S and SBAR. Furthermore, it achieves its best performance faster than the baseline parser, indicating yet another benefit of training alongside an n-gram parser.

As future work, we plan to improve our baseline parser in order to make the jointly trained parser more competitive with respect to its peers in the literature. We will explore new approaches for selecting better n-gram trees to improve the quality of the training data. We also plan to use multiple n-gram parsers in joint training instead of just one. In addition, we will use the n-gram trees and the HJL setup to build a self-trained parser by expanding the n-gram parser's training data with n-gram trees extracted from the output of the full sentence parser. This will enable the full sentence parser to be indirectly trained with its own output.

Acknowledgments

We thank Brian Roark and Suzan Üskudarlı for their invaluable feedback. This work was supported by the Boğaziçi University Research Fund 12A01P6.

References

[1] T. Kasami, "An efficient recognition and syntax-analysis algorithm for context-free languages," *Technical report, Air Force Cambridge Research Lab*, 1965.

[2] J. Earley, "An effcient context-free parsing algorithm," *Communications of the ACM*, vol. 13(2), pp. 94–102, 1970.

[3] M. Collins, "Head-driven statistical models for natural language parsing," Ph.D. dissertation, Department of Computer and Information Science, University of Pennsylvania, 1999.

[4] E. Charniak, "Statistical parsing with a context-free grammar and word statistics," *Proceedings of AAAI-97*, pp. 598–603, 1997.

[5] A. Ratnaparkhi, "Learning to parse natural language with maximum entropy models," *Machine Learning*, vol. 34(1–3), pp. 151–175, 1999.

[6] E. Charniak, "A maximum-entropy-inspired parser," *Proceedings of the North American Association of Computational Linguistics*, 2000.

[7] M. Collins, "Discriminative reranking for natural language parsing," *Proceedings of ICML-2000*, pp. 175–182, 2000.

[8] L. Huang, "Forest reranking: Discriminative parsing with non-local features," *Proceedings of Ninth International Workshop on Parsing Technology*, pp. 53–64, 2005.

[9] D. McClosky, E. Charniak, and M. Johnson, "Effective self-training for parsing," *Proceedings of HLT-NAACL*, 2006.

[10] S. Abney, "Part-of-speech tagging and partial parsing," *Corpus-Based Methods in Language and Speech Processing, Kluwer Academic Publishers, Dordrecht*, 1999.

[11] J. R. Finkel and C. D. Manning, "Hierarchical joint learning: Improving joint parsing and named entity recognition with non-jointly labeled data," *Proceedings of ACL 2010*, 2010.

[12] J. R. Finkel, A. Kleeman, and C. D. Manning, "Efficient, feature-based conditional random field parsing," *Proceedings of ACL/HLT-2008*, 2008.

[13] M. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of English: The Penn Treebank," *Computational Linguistics*, vol. 19(2), pp. 313–330, 1993.

[14] A. Ratnaparkhi, "A linear observed time statistical parser based on maximum entropy models," *Proceedings of EMNLP*, pp. 1–10, 1997.

[15] J. Henderson, "Discriminative training of a neural network statistical parser," *42nd ACL*, pp. 96–103, 2004.

[16] H. Zhang, M. Zhang, C. L. Tan, and H. Li, "K-best combination of syntactic parsers," *Proceedings of EMNLP 2009*, pp. 1552–1560, 2009.

[17] V. Fossum and K. Knight, "Combining constituent parsers," *Proceedings of NAACL 2009*, pp. 253–256, 2009.

[18] H. Daume III and D. Marcu, "Domain adaptation for statistical classifiers," *Journal of Artificial Intelligence Research*, 2006.

[19] J. R. Finkel and C. D. Manning, "Nested named entity recognition," *Proceedings of EMNLP 2009*, 2009.

[20] ——, "Joint parsing and named entity recognition," *Proceedings of the North American Association of Computational Linguistics*, 2009.

[21] R. Bod, R. Scha, and K. Sima'an, "Data oriented parsing," *CSLI Publications, Stanford University*, 2003.

[22] A. Joshi, L. Levy, and M. Takahashi, "Tree adjunct grammars," *Journal of Computer and System Sciences*, vol. 10:1, pp. 136–163, 1975.

[23] M. Bansal and D. Klein, "Web-scale features for full-scale parsing," *Proceedings of 49th Annual Meeting of ACL: HLT*, pp. 693–702, 2011.

[24] A. Clark, "Combining distributiona and morphological information for part of speech induction," *Proceedings of the tenth Annual Meeting of the European Association for Computational Linguistics (EACL)*, pp. 59–66, 2003.

[25] T. Rose, M. Stevenson, and M. Whitehead, "The Reuters corpus volume 1 - from yesterday's news to tomorrow's language resources," *Proceedings of the 3rd international conference on language resources and evaluation.*, 2002.

Automatic WordNet Construction Using Markov Chain Monte Carlo

Marzieh Fadaee, Hamidreza Ghader, Heshaam Faili, and Azadeh Shakery

***Abstract*—WordNet is used extensively as a major lexical resource in information retrieval tasks. However, the qualities of existing Persian WordNets are far from perfect. They are either constructed manually which limits the coverage of Persian words, or automatically which results in unsatisfactory precision. This paper presents a fully-automated approach for constructing a Persian WordNet: A Bayesian Model with Markov chain Monte Carlo (MCMC) estimation. We model the problem of constructing a Persian WordNet by estimating the probability of assigning senses (synsets) to Persian words. By applying MCMC techniques in estimating these probabilities, we integrate prior knowledge in the estimation and use the expected value of generated samples to give the final estimates. This ensures great performance improvement comparing with Maximum-Likelihood and Expectation-Maximization methods. Our acquired WordNet has a precision of 90.46% which is a considerable improvement in comparison with automatically-built WordNets in Persian.**

***Index Terms*—Semantic network, WordNet, ontology, Bayesian inference, Markov chain Monte Carlo, Persian.**

I. Introduction

NOWADAYS WordNet as an ontology, where the relations between word senses are interpreted as relations between concepts, is widely used in different areas of information retrieval and linguistic researches such as machine translation, text classification, word sense disambiguation, and text retrieval.

Princeton university constructed the first WordNet in English in 1995 employing human experts [1]. In Princeton WordNet (PWN) English words have been grouped into sets of cognitive synonyms called synsets. Synsets in PWN are also interlinked by means of conceptual semantics and lexical relations. Each English word may appear in several synsets in PWN, which are realized as senses of that word.

Acknowledgment of the practicality of PWN leads many researchers to develop a WordNet in languages other than English. The obvious obstacle of developing a WordNet from scratch is that it is very labor intensive and time consuming, so different methods were proposed to construct a WordNet automatically or semi-automatically. Constructing a WordNet automatically can be categorized into two approaches: merging methods, and expanding methods. The merging methods build the WordNet in a specific language based on monolingual resources in that language, and then map the created WordNet to existing WordNets of other languages, oftentimes PWN. The expanding methods use a basis WordNet (commonly PWN) so that they preserve the original WordNet structure, and construct the WordNet in a specific language by translating the synsets or applying different methods of learning. Resources in this category can be bilingual or multilingual. Either way having links between a WordNet in secondary language with PWN can improve the usability of said WordNet.

For example the BabelNet project [2], which uses PWN as the lexicographic resource and Wikipedia pages in different languages as the encyclopedic knowledge. It utilizes machine translation methods in order to enrich the lexical information and defines links between Wikipedia pages and WordNet items.

Although there have been several attempts in constructing a WordNet for Persian language, the lack of a sizable WordNet is still noticeable. Some of the most significant researches on Persian WordNet are introduced in [3], [4], [5].

In [3] the authors established a scoring function for ranking synsets and respective words automatically and selected the highest scores as the final WordNet. This method achieved a precision of 82.6% with manually judged candidates. In [4] an unsupervised learning approach was proposed, which constructed a WordNet automatically using Expectation-Maximization (EM) method. This research collects a set of words and their possible candidate synsets, and assigns a probability to each candidate. By applying an EM method these probabilities are updated in each iteration of the algorithm until the changes in probabilities are minute. The final WordNet was built by selecting the 10% of highly probable word-synsets and achieved a precision of 86.7%. Another project of building a Persian WordNet was FarsNet, which uses a semi-automatic method for building the Persian WordNet with some predefined heuristics and then judges each entry manually with human experts' knowledge [5].

The automatic approaches of constructing a Persian WordNet still need improvements in precision, and the manual approaches are time consuming and slow-growing. Our work aims for constructing a scalable Persian WordNet with better quality by defining a Bayesian Inference for estimating the probabilities of links between words and synsets. The proposed model is independent of language and can be applied in any language with basic resources: a raw corpus, a bilingual dictionary, and PWN.

Manuscript received on December 7, 2012; accepted for publication on January 11, 2013.

All authors are with the School of ECE, College of Engineering, University of Tehran, Tehran, Iran; Heshaam Faili and Azadeh Shakery are also with the School of Computer Science, Institute for Research in Fundamental Science (IPM), P.O. Box 19395-5746, Tehran, Iran (e-mail: {m.fadaee, h.ghader, hfaili, shakery}@ut.ac.ir).

We propose a model that utilizes a Markov chain Monte Carlo technique, namely Gibbs Sampling, in estimating the probabilities. This model iteratively disambiguates words according to their neighbors in a context and assigns probabilities to each possible sense of a word. After a certain number of iterations, the mathematical expectation of probabilities is the concluding value for each link.

In this research we construct a Probabilistic Persian WordNet, in which each Persian word is associated with relative PWN synsets and a probability that signifies these links. Using these links, the relations defined in the PWN is also applicable in our Persian WordNet.

Our proposed unsupervised approach to create a WordNet is very similar to the approach of [4] in some aspects. The main difference between these two approaches is that the Expectation-Maximization method in the research of [4] has been replaced by a fully Bayesian inference via Markov chain Monte Carlo. The Bayesian inference tries to estimate and update the probabilities assigned to word-synsets links, in an iterative process as Expectation-Maximization does. But this iterative process is a Markov chain Monte Carlo algorithm that estimates the posterior distribution. Each iteration of the algorithm includes 2 steps: (i) assigning correct senses to the words in the corpus using current estimate of the probabilities via a word sense disambiguation method, (ii) estimating new probability values according to the conditional posterior distribution, which has recently assigned senses in its condition.

Our model is expected to do better than the state-of-the-art EM method for two reasons: it incorporates the prior knowledge that the multinomial distribution over possible senses of a word is a sparse one, in its estimation of the posterior distribution, and it generates lots of samples from posterior distribution and use the expected value of these samples to give the final estimate, while the Expectation-Maximization (EM) method finds a local maximum in the search space and returns it as the final estimate. Thus, our approach takes the parameter values that may have generated the observed data with less probability into account, while the EM method fails to consider them.

Our WordNet does not have the obstacles of time and expert knowledge like the manual methods of constructing a Persian WordNet. By establishing a specified size for our WordNet (approximately 10,000 word-synset pairs) we achieve a precision of 90.46%. This is an improvement in comparison with the EM method, the state-of-the-art in constructing a Persian WordNet automatically, which achieved a precision of 86.7% with approximately the same size.

The rest of the paper is organized as follows: Section II presents an overview on several methods that have been proposed in the area of automatic and semi-automatic WordNet construction. Section III presents the details of the proposed model for automatically constructing the WordNet, and the training algorithm. Section IV explores experimental results and evaluations. Lastly, the work is concluded and some future works are suggested.

II. Background

WordNet is a semantic network providing machine-readable lexicographic information, first developed in Princeton University [1] . Princeton WordNet is a lexical database consisting of syntactic categories of English words—nouns, verbs, adjectives and adverbs, grouped into lexicalized concepts. Each cognitive synonym (synset) conveys a conceptual meaning and is part of a relational network. In WordNet several hierarchal relations are defined between synsets and words, such as synonymy (similar), antonymy (opposite), hyponymy (subordinate) and meronymy (part).

Princeton WordNet is currently the most advanced English WordNet containing a wide range of English words and word senses. It was created manually by English linguists, but manual construction of WordNet is a time consuming task and requires linguistic knowledge. In order to achieve comprehensive WordNet in languages other than English, many automatic and semi-automatic approaches were presented.

EuroWordNet was a similar project but with the goal of enriching the resources of Western European languages [6]. It started with four languages: Dutch (at the University of Amsterdam), Italian (CNR, Pisa), Spanish (Fundacion Universidad Empresa), and English (University of Sheffield, adapting the Princeton WordNet); and later Czech, Estonian, German, and French were added.

By applying a common framework between all WordNets and integrating them into a single database, EuroWordNet became a multilingual semantic network which could be used in many multilingual applications.

In order to maintain a similar coverage in all languages, first a set of 1,024 base concepts were created. Since these concepts were not lexicalized in all languages, iteratively, the base concepts were selected based on the common concepts between the majority of European languages. The base concepts were classified with the aid of a language-independent top ontology.

EuroWordNet is not used widely due to licensing issues and lack of further extensions. In [7] a freely-available French WordNet (WOLF) was built automatically from multilingual resources like Wikipedia and thesaurus. In the proposed approach, they constructed a multilingual lexicon by aligning a parallel corpus for five languages. By using multiple languages, polysemous lexicon entries are disambiguated. WOLF contains all four parts of speeches, including 32,351 synsets corresponding with 38,001 unique literals. The average polysemy in WOLF is 1.21 synsets per literal but the core vocabulary of it is sparse.

The resulting WordNet was evaluated both automatically and manually. In the former approach, they compared WOLF with the French WordNet, created as part of the EuroWordNet project, in regard to the words that appeared in WOLF so as

not to penalize the WOLF for not containing some words in the utilized corpora. WOLF achieved a precision of 77.1% and recall of 70.3%. In the latter approach they randomly selected 100 literals and the corresponding 183 synsets to judge them by hand and achieved 80% correctness in assigned synsets. In this paper it is shown that building a WordNet with the alignment approach provides more basic synsets.

BalkaNet was another European project focusing on Central and Eastern European languages [8]. The method of constructing BalkaNet is comparable to EuroWordNet with added features such as independence of every WordNets. It uses individual monolingual WordNets that have already been developed for the participant languages, including Greek, Turkish, Romanian, Bulgarian, Czech and Serbian. BalkaNet contains 15,000 comparable synsets in each language, and 30,000 literals. BalkaNet concept sets are very dense in the sense that for any concept in the BalkaNet concept sets, all of its hypernyms are also in the BalkaNet. Turkish WordNet is a side-result of the BalkaNet project [9] containing 11,628 synsets and 16,095 literals. It has an average polysemy of 1.38.

Word sense disambiguation techniques are applied in many approaches of constructing or expanding a WordNet. In [10] they defined multiple heuristics including maximum similarity, prior probability, sense ordering, IS-A relation, and co-occurrence, for linking Korean words to English synsets. The heuristics were then combined using decision tree learning for non-linear relationship. To evaluate each of their proposed heuristics separately, they manually judged the candidate synsets of 3260 senses. The decision tree based combination of the heuristics achieved 93.59% in precision and 77.12% in recall. Their generated Korean WordNet contains 21,654 synsets and 17,696 nouns.

There are other attempts in constructing WordNets for Asian languages. A Thai WordNet was constructed utilizing machine-readable dictionaries [11]. In this semi-automatic approach several criteria were defined to extract and evaluate relations between translated words. To evaluate the candidate links in each criterion they apply the stratified sampling technique [12]. The final WordNet contains 19,582 synsets and the corresponding 13,730 words and provides 80% coverage and 76% accuracy.

Arabic WordNet was first introduced in [13]. By considering three main criteria—connectivity, relevance and generality, synsets were extracted and manually validated. In this project they also generated a machine-understandable semantics in first order logic for word meanings. The Arabic WordNet consists of 11,270 synsets and 23,496 Arabic expressions. There were several extensions of Arabic WordNet, particularly the semi-automatic approach in [14]. They designed a Bayesian network with four layers to equate Arabic words and English synsets by using lexical and morphological rules. The resulting WordNet has a precision of 67%.

There were several researches on constructing WordNet in Persian language in recent years, focusing on Persian adjectives [15], verbs [16], or nouns [17]. These methods either use lexicographers' knowledge in constructing the WordNet manually, or proposing semi-automatic approaches. PersiaNet was a project of Princeton University for a comparable Persian WordNet with Princeton WordNet [17]. This work, which is strictly based on a volunteering participation of experts, has a scarce lexical coverage. It uses Persian orthography for representing words and also supports a parallel Roman writing system in order to facilitate searching for Persian words.

In [18] a semi automatic method was proposed using human annotators to make the decision of relativeness of each word and candidate synsets. In this work they introduced FarsNet which consists of two parts: semantic lexicon and lexical ontology. They used a bilingual dictionary, a syntactic lexicon including the POS tags of the entries, a Persian POS tagged corpus and WordNet in order to develop an initial lexicon and perform word-sense disambiguation. A linguistic expert reviewed the results to evaluate the method which gained a 70% accuracy.

They expanded their work later, completing FarsNet by applying some heuristics and word sense disambiguation techniques in an automated method with human supervision [5]. It consists of 9,266 synsets and 13,155 words. In this paper we use FarsNet as the baseline in evaluating the quality of our WordNet.

In [3] an automatic method was presented in which they compute a similarity score between each Persian word and the candidate English synsets and select the highest score as the respective link. The score, containing the mutual information of words, is computed from bilingual dictionaries and Persian and English corpora. To evaluate the constructed Persian WordNet they randomly selected 500 Persian words and assessed the accuracy of the selected synsets. The precision of unambiguous links between words and synsets is 95.8%, and of ambiguous links is 76.4%. In total they achieved an accuracy of 82.6%.

In [4] an unsupervised learning approach was proposed for constructing a Persian WordNet. In this work they first assemble a list of candidate English synsets for each Persian word using a bilingual dictionary and Princeton WordNet. In the next step they automatically connect English synsets with Persian words using Expectation-Maximization method and eliminates unrelated links. The probabilities of each link are calculated in the Expectation step from the information extracted from a Persian corpus. In the Maximization step, the probabilities of selected candidate synsets is updated. In order to evaluate the resulting WordNet they manually judged 1365 randomly selected links between words and synsets. By accepting the top 10% of the probable links as the final Persian WordNet 7,109 literals (from 11,076 words appeared in the corpus) and 9,427 synsets were selected. The WordNet accuracy is 89.7% for adjectives, 65.6% for adverbs and 61.2% for nouns. This approach strongly depends on the initial Persian corpus that is used in the Expectation step and the initial values of probabilities of links.

A. Markov chain Monte Carlo

Using Bayesian inference to estimate posterior over a model, one may come across a situation in which the posterior or an intermediate distribution could not be computed analytically. In these cases, a widely used method is to estimate the intended distribution using Markov chain Monte Carlo techniques. The works [19], [20] are examples of dealing with this kind of situation in Bayesian inference. In [20] two MCMC techniques are used to induce a probabilistic context free grammar in an unsupervised setting. In this work, the MCMC methods are employed to sample from posterior distribution over probabilities of CFG rules and sequence of parse trees conditioned on the observed data. In [19] an MCMC technique is put in action to find a MAP estimation of a posterior distribution over POS tag sequence conditioned on the observed data.

In order to estimate a distribution using MCMC techniques, the techniques are used to generate sufficient samples from the distribution. The MCMC techniques construct a Markov chain whose desired distribution is equal to the distribution intended to be sampled. This means that they provide the transition probability between states of Markov chain so that the probability of visiting a state $St_{\mathbf{i}}$ of the chain be $p(St_{\mathbf{i}})$, according to the desired distribution. Then, they generate samples by moving between states of the Markov chain. Of course, some runout steps are required for the model to take, before that the generated samples being from the stationary distribution of the Markov chain. After generating sufficient samples from the desired distribution, these probabilistic choices can be used to calculate expectation over states. This makes the method a Monte Carlo technique. For example in [20], the sample values for θ, a random variable corresponding to the probability of CFG rules, are used to compute the expectation over it. Then, the resulted expectation is used as the estimated value for probability of CFG rules.

1) Gibbs Sampling: Gibbs sampling [21] is a sampling technique from class of Markov chain Monte Carlo techniques. This technique is used in situations that the state of the model is comprised of multiple random variables [22]. In other words, the situations in which the joint distribution of multiple random variables is intended to be sampled. If we assume that each state in the model has k dimension or is a joint of k random variables, the basic idea in this technique is to sample each random variable involved in the state of the model separately, but conditioned on the other $k-1$ dimensions [22]. That is to sample each random variable from the following distribution:

$$P(rv_i|rv_1^t, ..., rv_{i-1}^t, rv_{i+1}^{t-1}, ..., rv_k^{t-1}).$$

Here the superscript corresponds to time. It also provides the information that how many samples are generated from a random variable until current time. After sampling each random variable once, using the conditional distribution above, we will have one sample from the joint distribution of random variables. Repeating this action for a sufficient number of times, we will generate sufficient samples from the join distribution. These samples can be used in a variety of ways to compute an estimation of the intended distribution.

B. Dirichlet Priors

In recent years, the Bayesian methods for estimating probabilities are widely favored over the maximum likelihood estimation method among scientists working in computational linguistics [19], [23]. One reason for this, is the fact that Bayesian methods provide a way to take the prior knowledge about the model into account when doing estimation. As a standard example, taken from [23], suppose that we are given a coin to decide whether it is fair or not. Tossing the coin 10 times, we observe this sequence of heads and tails: (T T T T T T H H H H). Maximum likelihood estimation gives an estimate of 0.6 for the probability of observing tail in next toss. Maximum likelihood results this estimate by calculating the parameter value that is most likely to generate observed data. If we take θ as the probability of observing tail, that means

$$\hat{\theta} = \arg\max_{\theta} P(D|\theta).$$

As one can see, no prior knowledge is incorporated in this estimation. This is while the Bayesian methods take the prior knowledge into account using Bayes rule. For example maximize a posteriori estimation gives an estimate of θ as follows:

$$\begin{aligned} \hat{\theta} &= \arg\max_{\theta} P(\theta|D) \\ &= \arg\max_{\theta} \frac{P(D|\theta)P(\theta)}{P(D)} \\ &= \arg\max_{\theta} P(D|\theta)P(\theta). \end{aligned}$$

This estimation provides the possibility that our expectation of what θ could be, affect the final estimated value for θ. Here, by using a Beta distribution, which is a two dimensional version of dirichlet distribution, we can put a prior expectation toward fairness or unfairness of the coin into the estimation. If we choose parameters of the Beta distribution to be near zero, this will put a prior in favor of unfairness of the coin into the estimation. This means that an estimate of θ nearer to 0 or 1 is more desirable. This characteristic of Bayesian methods makes them more appropriate than maximum likelihood estimation, because it provides the possibility of using linguistically appropriate priors.

III. Modeling the Problem of Automatically Constructing a WordNet

A. The Approach

In this work, we create a probabilistic WordNet in which each link between a word and its synsets has a probability assigned to it. This probability signifies the relatedness of each synset to the word.The proposed approach consists of the following steps:

1) Collect candidate synsets as possible senses for each Persian word
2) Compute the probability of relatedness of each synset to a particular word (iteratively)
3) Choose the highest word-synset probabilities as the final WordNet

For the first step, we use a bilingual dictionary to find all possible definitions of a Persian word in English. Then, we use the Princeton WordNet to find all senses of the English words and consider them as potential senses of the Persian word.

In the next step, we compute the probabilities of the Persian word having each of the senses extracted in the previous step. These probabilities are computed according to different senses of a word that appear in a raw corpus. This corpus contains Persian words and the POS tags for each word, and so it aids in building a POS-aware Persian WordNet.

We use a word sense disambiguation technique, previously utilized in [4], as part of a Bayesian model in an unsupervised configuration to assign the correct sense of each word based on the context in which the word appears. During the process of sense disambiguation of words in the corpus, we compute the probabilities of assigning different senses to a word. As a result, some senses of a word will be assigned very small probabilities in comparison with other senses.

The algorithm iteratively computes the new probabilities using Gibbs Sampling, which will be discussed in the following section.

The algorithm uses a Markov Chain Monte Carlo (MCMC) method, Gibbs sampling, and iteratively computes the new probabilities. MCMC methods are widely used to estimate probability distributions that could not be computed analytically.

In the final step, we eliminate the senses assigned small probabilities according to some measures, and use the remaining senses to build up the Persian WordNet.

B. The Model

We define the probabilities of assigning synsets $s_1, ..., s_n$ to a word w as

$$\theta_w : [\theta_{ws_1}, \theta_{ws_2}, ...\theta_{ws_n}]. \quad (1)$$

If t_w indicates a synset assigned to word w, $t_w|context$ will have a multinomial distribution whose parameters are in the vector θ_w. For ease of reference, we present our notation in Table 1. For a multinomial with parameters $\theta_w = [\theta_{ws_1}, ..., \theta_{ws_k}]$ a natural choice of prior is the K-dimensional Dirichlet distribution, which is conjugate to the multinomial [19]. If we assume that the dirichlet distribution is symmetric and its K parameters are equal to α, $\alpha < 1$ will favor sparse multinomial distributions for θ_w. As the distribution of senses of a word in a context is a sparse multinomial distribution, a Dirichlet distribution with $\alpha < 1$ will be a linguistically appropriate prior in our model.

So we can safely assume that θ_w has a $Dirichlet(\alpha_w)$ distribution:

$$\theta_w \sim Dirichlet(\alpha_w). \quad (2)$$

Suppose that θ is the vector of θ_w for all words. The goal of constructing the WordNet in our model is obtaining a wise θ which can be computed as follows:

$$P(\theta|W) = \sum_t P(t, \theta|W), \quad (3)$$

with W being the corpus we use for sense disambiguation and t is a tag sequence of synsets. The distribution on the right side of the equation could be written as follows:

$$\begin{aligned} P(t, \theta|W) &= \frac{P(W|\theta, t)P(\theta, t)}{P(W)} \\ &= \frac{P(W|\theta, t)P(\theta, t)}{\sum_{t,\theta} P(W|\theta, t)P(\theta, t)}, \end{aligned} \quad (4)$$

which is intractable because of the large possible sets of t and θ, which should be observed to compute the denominator. If we take the Dirichlet prior into account, the posterior distribution will change as follows:

$$P(t, \theta|W, \alpha), \quad (5)$$

where α is a vector of parameters of the prior distributions which are Dirichlet distributions.

Since computing the probability distribution of Equation 5 is intractable, we propose to use a Markov chain Monte Carlo algorithm, Gibbs sampling, to generate samples from this distribution and use the samples to compute a wise value for θ. To use the Gibbs sampling method to generate samples from $P(\theta, t|W, \alpha)$, we have to sample each random variable conditioned on the current value of the other random variables constituting the state of the model. This means that we have to sample from the following two probability distributions at each step:

$$P(t|\theta, W, \alpha), \quad (6)$$

$$P(\theta|t, W, \alpha). \quad (7)$$

Formula (6) illustrates the sense assignments' probabilities, and Formula (7) illustrates the candidate senses' probabilities. In the following section the sampling process of these two distributions and how we estimate the probabilities are discussed in detail.

C. Estimating the Probabilities

In order to generate samples from (7) we can independently sample each multinomial distribution $P(t_i|w_i, context_{w_i}, \theta)$ for all possible i. Then we use resulted value for each t_i as sense tag of w_i. In the next step, given a value for t we generate sample from (8).

TABLE I
NOTATION

w	Persian word.
$wordlist$	the list of all Persian words we want to include in our WordNet.
s_i	Princeton WordNet synset.
ws_i	assigning synset s_i to Persian word w as a possible sense of that word.
θ_{ws_i}	probability of relateness of ws_i
θ_w	vector of probabilities of candidate senses for word w
θ	vector of θ_ws for all words.
α_w	Dirichlet parameter for the distribution of θ_w
α	vector of α_ws for all words w in $wordlist$
W	corpus, providing words w for disambiguation.
t_w	a random variable whose possible values are candidate senses s_i of word w
t	vector of t_ws for all words w in $wordlist$

1) Sense Assignment Probability Estimation: In the literature, a key assumption to induce correct sense of a word is that the context surrounding the word is indicative of its correct sense [23]. Making the same assumption, the distribution of senses of a word conditioned on the word itself and its context will be independent from the other words of the corpus and their senses. So we can write:

$$P(t|\theta, W, \alpha) = \prod_i P(t_{w_i}|w_i, context_{w_i}, \theta). \quad (8)$$

Hence we involve the context in computing the probabilities by considering a window of words rather than every individual word.

Finding multinomial distribution $P(t_{w_i}|w_i, context_{w_i}, \theta)$ for each possible i, and using the distributions to interpret the correct sense of each word w_i could be viewed as a word sense disambiguation task.

Word sense disambiguation is the task of interpreting senses of a word in a context via supervised or unsupervised methods. Most words in the Persian language are polysemous, so, in order to differentiate between individual meanings of a word we need to consider disambiguating its senses. To attain this goal we use an ancillary Persian corpus, Bijankhan [24], as our corpus for extracting statistical information. For every word in the training set, windows of words are obtained from the corpus containing the neighbors of that particular word for every occurrence of it in the corpus.

The score of each word w and synset s is calculated from the following formula:

$$score(w, s) = \frac{\sum_{w'} \sum_v \theta_{w',s} \times PMI(w', v)}{n}, \quad (9)$$

where w' is a word that has s as its candidate synset, n is the number of these words, v is a word in the window of w, *PMI* is point-wise mutual information, and $\theta_{w',s}$ is the probability assigned to word w' and sense s in the previous iteration of the procedure. This score function disambiguates a word by considering the senses of the neighbors of the word in a context [4].

Using the scores computed in Equation 9, we can find the distributions

$$P(t_{w_i}|w_i, context_{w_i}, \theta)$$

for all possible i by means of the following formula:

$$P(t_{w_i}|w_i, context_{w_i}, \theta) = \frac{score(w_i, t_{w_i})}{\sum_j score(w_i, s_j)}. \quad (10)$$

By involving all contexts in which a word is used in a corpus—windows of words—individual senses of the word have the chance of obtaining acceptable probabilities in the computation. For better determining individual senses of every word we consider the parts of speech of them. One of the main attributes of every synsets in Princeton WordNet is the part of speech of that sense. To take heed of this attribute we consider individual parts of speech for every word and perform the sampler on words and synsets in regard to parts of speech.

2) Candidate Sense Probability Estimation: In order to compute the second distribution we assume that θ_w for each word is independent from the others and as we discussed prior distribution on θ_w is Dirichlet distribution. So the prior probability on θ could be written as follows:

$$\begin{aligned} P(\theta) &= \prod_{w \in wordList} P(\theta_w|\alpha_w) \\ &= \prod_{w \in wordList} \Big(\prod_{s \in senses_w} \frac{1}{B(\alpha_w)} \theta_s^{\alpha_s - 1} \Big), \end{aligned} \quad (11)$$

where

$$B(\alpha_w) = \frac{\prod_{s \in senses_w} \Gamma(\alpha_s)}{\Gamma(\sum_{s \in senses_w} \alpha_s)}, \quad (12)$$

wordList is the list of all words we want to include in our WordNet. As a result to formulation above, $P(\theta)$ will be Dirichlet distribution and could be written as $P_D(\theta|\alpha)$, where α is a vector containing α_w for all words in $wordList$ and is the parameter of the prior Dirichlet distributions of θ_w.

Since the prior distribution of θ is conjugate prior to the likelihood of the sense tags, the posterior on θ conditioned on sense tags will be a Dirichlet distribution:

$$P(\theta|t, W, \alpha) \propto P(t, W|\theta, \alpha) P(\theta|\alpha); \quad (13)$$

$$\begin{aligned} &\propto \Big(\prod_{w \in wordList, s \in senses_w} \theta_s^{Cn_{w \to s}(t)} \theta_s^{\alpha_s - 1} \Big) \\ &= \prod_{w \in wordList, s \in senses_w} \theta_s^{Cn_{w \to s}(t) + \alpha_s - 1}, \end{aligned}$$

which could be written as

$$P(\theta|t,W,\alpha) = P_D(\theta|Cn(t)+\alpha) = \prod_{w\in wordList} P_D(\theta_w|Cn_w(t)+\alpha_w), \quad (14)$$

where $Cn_w(t)$ is a vector of the number of times the assignment $w \rightarrow s_i$, where $s_i \in senses_w$, happened in the context and $Cn(t)$ is a vector of $Cn_w(t)$ for all w.

For this to be done, we sample each Dirichlet distribution, $P_D(\theta_w|Cn_w(t)+\alpha_w)$, independently and put the results together to constitute a sample from θ. To create sample from the Dirichlet distribution, we use a Gamma distribution and sample γ_{s_j} from $Gamma(\alpha_{s_j}+Cn_{w\rightarrow s_j}(t))$ for all $\alpha_{s_j} \in \alpha_w = (\alpha_{s_1},...,\alpha_{s_m})$ and finally set each $\theta_{s_j} \in \theta_w = (\theta_{s_1},...,\theta_{s_m})$ as follows:

$$\theta_{s_j} = \frac{\gamma_{s_j}}{\sum_{i=1}^{m}\gamma_{s_i}}. \quad (15)$$

So we use Formulas (8), (10), and (14) to generate samples from Formulas (6) and (7). This will result in samples from $P(\theta,t|W,\alpha)$ as was discussed before. After sampling the acceptable number of values for θ, we can compute the Expectation of θ over these values which would grant us the wise θ we were looking for.

IV. Experiments and Evaluation

In this paper we have conducted different experiments to evaluate our proposed method. This section carries out with introducing the environment of the experiments and details on different trials and the methods of evaluation applied in this project.

A. Train Dataset

We use Bijankhan dataset [24] to take into account the context for every word in order to perform an unsupervised word sense disambiguation and compute θ values in the iterative process. Bijankhan is a POS tagged corpus in the Persian language consisting of news and colloquial texts. It contains 500 documents and around 2.6 million manually tagged words. The tag set consists of 40 different Persian POS tags, however we only use the four main POS tags in this experiment; verb, noun, adjective, and adverb.

B. Test Dataset

To evaluate the accuracy of our WordNet we use a manual approach of assessing our results, in view of the fact that if we wanted to automatically evaluate the WordNet we had to compare the results with the existing Persian WordNets, which wasn't fair to our WordNet; by comparing our results with the previous WordNets we would penalize the correctly assigned word-synset pairs that do not exist in the earlier WordNets.

To avoid this, we opt for building a test set which we have based on FarsNet, the semi automatically constructed WordNet. FarsNet links are added to this test set as the *correct links*. We also judged a subset of assigned words - synsets links manually and added this information to the test set. Our final gold data contains 3482 incorrect links and 17393 correct links.

TABLE II
Statistics of the test data set

Number of incorrect links (0)	3482
Number of correct links (1)	17393
size of the test dataset	20874

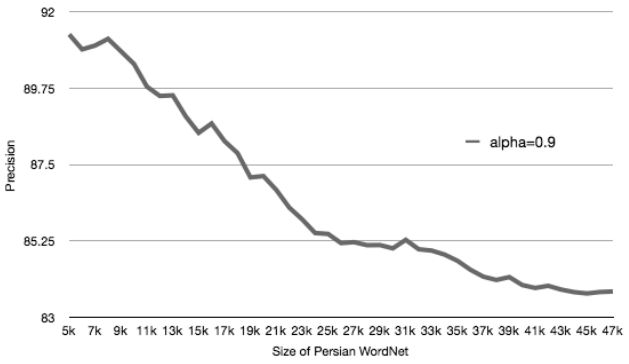


Fig. 1. Precision of Persian WordNet with respect to N, the size of WordNet (the N most probable word-synset links) after 100 iterations

C. Results

Upon the termination of the algorithm, a WordNet in target language and the probabilities of assigning each candidate synsets to each word are acquired and are sorted based on the probabilities, so by selecting the $top-N$ most probable word-synset pairs we obtain our Persian WordNet. The parameter N determines the size of the WordNet; there is a trade-off between precision of the WordNet and the coverage over all Persian Words i.e. the size of the WordNet, N.

We define the precision of the WordNet as the number of assigned links in the WordNet which appeared in the test data as correct links divided by the total number of assigned links in the WordNet which appeared in the test data. This definition of precision for WordNet was also used in BabelNet project [2].

Figure 1 demonstrates the precision of our WordNet with respect to size of the WordNet. We can observe that by increasing the size of the WordNet, precision decreases which is expected. By selecting the first 10,000 word-synset pairs we have a WordNet of precision 90.46%. This is an improvement in comparison with the state-of-the-art automatically built Persian WordNet which gained precision of 86.7% with approximately the same size of the WordNet [4].

D. Dirichlet Prior Parameter Tuning

In this section we evaluate the effect of the Dirichlet parameter in our proposed model. As we have stated earlier, dirichlet prior is taken into service to provide the possibility

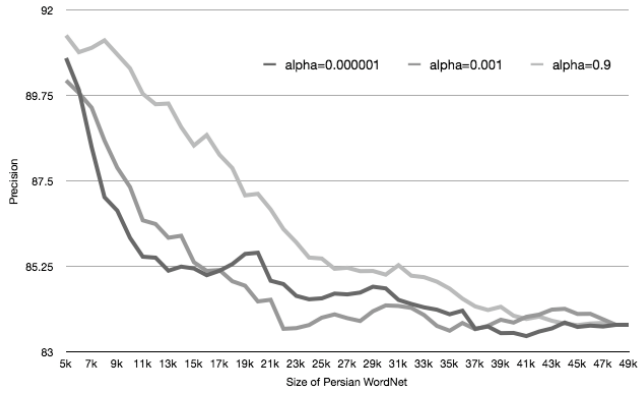


Fig. 2. Precision of Persian WordNet with respect to N, the size of WordNet after 100 iterations

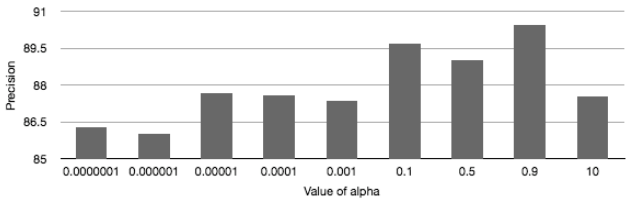


Fig. 3. Precision of Persian WordNet of size 10,000 with different values of α

of incorporating linguistically appropriate priors. According to the discussion, a valid assumption about the distribution of senses of a word is to assume that it is a sparse multinomial distribution.

Dirichlet distribution with parameters smaller than 1 is a natural prior over parameters of a sparse multinomial distribution. So, we assume a K-dimensional Dirichlet distribution over parameters of multinomial distribution with K dimensions. For simplicity, we assume that all dirichlet distributions are symmetric and its parameters are equal to α. As we prefer sparse multinomial distributions over senses of a word, we set $\alpha < 1$ for all Dirichlet prior distributions, but we also experiment with some large αs to observe the differences.

In order to observe the effect of the Dirichlet parameter, Figure 3 presents different values of precision of the WordNet with a fixed size of 10,000 word-sense pairs for different values of α. We can observe that the precision of the WordNet increases with the increase of the Dirichlet parameter. With optimum value of α, we obtained a precision of 90.46%.

The precision of the automatically built WordNet in this paper is calculated based on the joined test set containing annotated gold data, FarsNet, and the set of randomly judged words by human experts. N top demonstrates the size of the WordNet, for instance at $N = 10000$ we are selecting the 10,000 top links of word-sense and regarding them as our WordNet. It is clear that by expanding the size of the WordNet

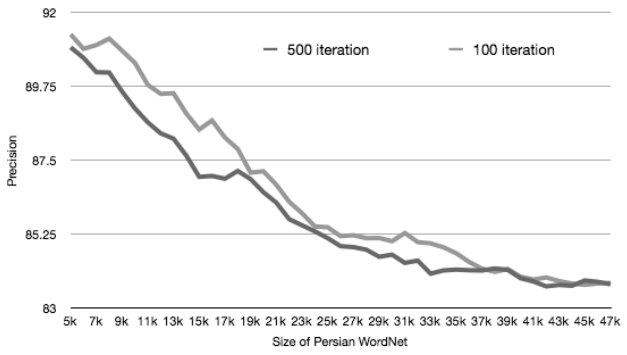


Fig. 4. Comparison of precision of Persian WordNet with respect to N for different number of iterations

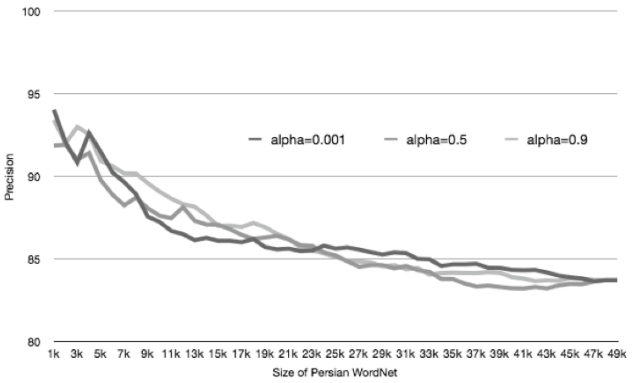


Fig. 5. Precision of Persian WordNet with respect to N, the size of WordNet after 500 iterations

and introducing more senses into our selected links we lose precision.

E. Number of Iterations

As stated earlier, the number of iterations of the proposed algorithm has an effect on the final results. In order to observe the effect of number of iterations on the results, we choose the approximate optimum value of 0.9 for α and present Figure 4. It is clear from this figure that the higher number of iterations acquire roughly the same results as lower number of iterations. The probabilities of the word-sense links are already converged with 100 iterations and we can trust our results with 100 iterations.

This figure shows that even with higher number of iterations we achieve better precision in the first 1000 links, but the value of precision gradually decreases with respect to lower number of iterations, hence, with 100 iterations of the algorithm we achieve better precision after the first 4000 links.

F. Coverage of the WordNet

To evaluate the coverage of our WordNet over all Persian words we perform two types of assessments: Considering

all words appearing in a Persian corpus, Bijankhan, as the baseline for Persian words, and analyzing the coverage of our WordNet over FarsNet as the baseline.

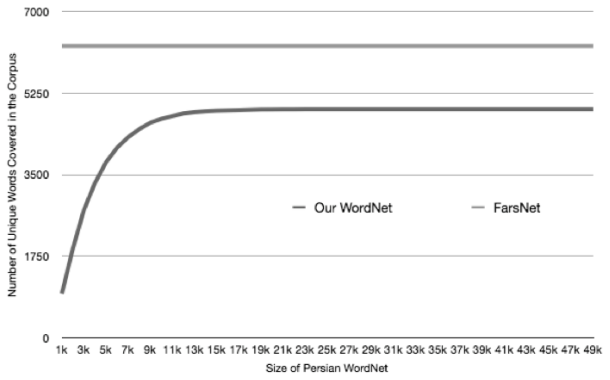


Fig. 6. Coverage of FarsNet and our Persian WordNet with respect to N, the size of WordNet, for $\alpha = 0.9$ over Persian words that appear in Bijankhan corpus

Figure 6 shows the number of unique words of the corpus, covered by our WordNet and also covered by FarsNet. We can observe that with high precision at the size of 10,000, our method covers a little less than FarsNet, which is a semi-automatically built WordNet.

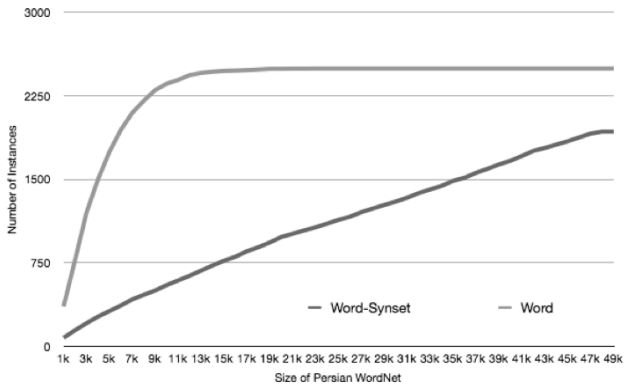


Fig. 7. Coverage of our Persian WordNet with respect to N the size of WordNet, for $\alpha = 0.9$ over FarsNet

The coverage of our WordNet in comparison with FarsNet as the baseline is displayed in Figure 7. In this figure we perceive two types of coverage: word, and word-synset pair. The former testifies to the number of words that both our WordNet and FarsNet have in common, and the latter testifies to the common sense coverage between two WordNets.

Figure 7 illustrates this experiment. We can note that by selecting 10,000 links, we cover 2,357 unique words in FarsNet, and this value only increases slightly by the increase of the size of our WordNet. However, the number of word-sense pairs covered in both FarsNet and our WordNet gradually increases with the increase of the size of our WordNet, signifying that we are adding new senses to the existing words with increase of the size and including new links.

V. Conclusion

In this paper, we have presented a method for constructing a Persian WordNet automatically. This method, which is based on a Bayesian Inference, uses Gibbs Sampling as a Markov chain Monte Carlo technique in order to estimate the probabilities of senses for each word in Persian. The final WordNet is established by selecting the pairs of word-synsets with highest probabilities. Our experiments show that this WordNet has satisfactory coverage over Persian words and maintains higher precision in comparison with published automatically-built WordNets in Persian. The resulting WordNet is freely released and can be downloaded from our website.[1]

In this paper, we assumed sparse multinomial distributions over senses of all words and used the same value for the parameters of all Dirichlet priors. In reality, the degree of sparsity of multinomial distributions differs for different words, and we should take this into account when setting parameter values of Dirichlet distributions as priors.

Another proposal for future work is to use variational Bayes as inference method for training the model. This will mitigate the problem of slow convergence of training step, which is the result of using Gibbs sampling as the inference algorithm. This makes the model capable of learning semantic nets with larger amount of words in relatively shorter time.

Acknowledgments

We want to acknowledge the support of Research Institute for ICT. This research was in part supported by a grant from IPM (No. CS1391-4-19).

References

[1] G. A. Miller, "WordNet: A lexical database for English," *Commun. ACM*, vol. 38, pp. 39–41, November 1995. [Online]. Available: http://doi.acm.org/10.1145/219717.219748

[2] R. Navigli and S. P. Ponzetto, "BabelNet: Building a very large multilingual semantic network," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, Uppsala, Sweden, 2010, pp. 216–225.

[3] M. Montazery and H. Faili, "Automatic Persian WordNet construction," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, ser. COLING '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 846–850. [Online]. Available: http://dl.acm.org/citation.cfm?id=1944566.1944663

[4] ——, "Unsupervised learning for Persian WordNet construction," in *RANLP*, G. Angelova, K. Bontcheva, R. Mitkov, and N. Nicolov, Eds. RANLP 2011 Organising Committee, 2011, pp. 302–308.

[5] M. Shamsfard, A. Hesabi, H. Fadaei, N. Mansoory, A. Famian, S. Bagherbeigi, E. Fekri, M. Monshizadeh, and M. Assi, "Semi automatic development of FarsNet, the Persian WordNet," in *5th Global WordNet Conference (GWA2010)*, Mumbai, India, 2010.

[6] P. Vossen, Ed., *EuroWordNet: A multilingual database with lexical semantic networks*. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[1] *http://ece.ut.ac.ir/nlp*

[7] B. Sagot and D. Fišer, "Building a free French WordNet from multilingual resources," in *OntoLex 2008*, Marrackech, Morocco, 2008.

[8] S. Stamou, K. Oflazer, K. Pala, D. Christoudoulakis, D. Cristea, D. Tufis, S. Koeva, G. Totkov, D. Dutoit, and M. Grigoriadou, "Balkanet: A multilingual semantic network for the balkan languages," in *Proceedings of the 1st Global WordNet Association conference*, 2002.

[9] O. Bilgin, Ö. Ç. Glu, and K. Oflazer, "Building a Wordnet for Turkish," pp. 163–172, 2004.

[10] C. Lee, G. Lee, S. JungYun, and G. Leer, "Automatic WordNet mapping using word sense disambiguation," in *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC)*, 2000.

[11] P. Sathapornrungkij, "Construction of Thai WordNet Lexical Database from Machine Readable Dictionaries," *English*, pp. 87–92, 2005.

[12] R. V. Krejcie and D. W. Morgan, "Determining sample size for research activities," *Educational and Psychological Measurement*, vol. 30, no. 3, pp. 607–610, 1970. [Online]. Available: http://eric.ed.gov/ERICWebPortal/recordDetail?accno=EJ026025

[13] W. Black, S. Elkateb, A. Pease, H. Rodriguez, and M. Alkhalifa, "Introducing the Arabic WordNet project," *Word Journal Of The International Linguistic Association*, 1998.

[14] H. Rodriguez, D. Farwell, J. Farreres, M. Bertran, M. Alkhalifa, and A. Marti, *Arabic WordNet: Semi-automatic Extensions using Bayesian Inference*. European Language Resources Association (ELRA), 2008, pp. 1–3. [Online]. Available: http://www.lrec-conf.org/proceedings/lrec2008/

[15] A. Famian, "Towards Building a WordNet for Persian Adjectives," *International Journal of Lexicography*, no. 2000, pp. 307–308, 2006.

[16] M. Rouhizadeh, M. Shamsfard, and M. Yarmohammadi, "Building a WordNet for Persian verbs," in *the Proceedings of the Fourth Global WordNet Conference (GWC '08)*. The Fourth Global WordNet Conference, 2008, pp. 406–412.

[17] F. Keyvan, H. Borjian, M. Kasheff, and C. Fellbaum, "Developing PersiaNet: The Persian WordNet," in *3rd Global wordnet conference*. Citeseer, 2007, pp. 315–318. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.148.7473\&rep=rep1\&type=pdf

[18] M. Shamsfard, "Towards semi automatic construction of a lexical ontology for Persian," in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*. Marrakech, Morocco: European Language Resources Association (ELRA), may 2008, http://www.lrec-conf.org/proceedings/lrec2008/.

[19] S. Goldwater and T. Griffiths, "A fully Bayesian approach to unsupervised part-of-speech tagging," in *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*. Prague, Czech Republic: Association for Computational Linguistics, Jun. 2007, pp. 744–751.

[20] M. Johnson, T. Griffiths, and S. Goldwater, "Bayesian inference for PCFGs via Markov chain Monte Carlo," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*. Rochester, New York: Association for Computational Linguistics, April 2007, pp. 139–146. [Online]. Available: http://www.aclweb.org/anthology-new/N/N07/N07-1018.bib

[21] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," in *Readings in computer vision: issues, problems, principles, and paradigms*, M. A. Fischler and O. Firschein, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987, pp. 564–584. [Online]. Available: http://dl.acm.org/citation.cfm?id=33517.33564

[22] P. Resnik and E. Hardisty, "Gibbs sampling for the uninitiated," University of Maryland, Tech. Rep., Oct. 2009.

[23] S. Brody and M. Lapata, "Bayesian word sense induction," in *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, ser. EACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 103–111. [Online]. Available: http://dl.acm.org/citation.cfm?id=1609067.1609078

[24] M. BijanKhan, "The role of the corpus in writing a grammar: An introduction to a software," *Iranian Journal of Linguistics*, vol. 19, 2004.

Exploration on Effectiveness and Efficiency of Similar Sentence Matching

Yanhui Gu, Zhenglu Yang, Miyuki Nakano, and Masaru Kitsuregawa

***Abstract*—Similar sentence matching is an essential issue for many applications, such as text summarization, image extraction, social media retrieval, question-answer model, and so on. A number of studies have investigated this issue in recent years. Most of such techniques focus on effectiveness issues but only a few focus on efficiency issues. In this paper, we address both effectiveness and efficiency in the sentence similarity matching. For a given sentence collection, we determine how to effectively and efficiently identify the top-k semantically similar sentences to a query. To achieve this goal, we first study several representative sentence similarity measurement strategies, based on which we deliberately choose the optimal ones through cross-validation and dynamically weight tuning. The experimental evaluation demonstrates the effectiveness of our strategy. Moreover, from the efficiency aspect, we introduce several optimization techniques to improve the performance of the similarity computation. The trade-off between the effectiveness and efficiency is further explored by conducting extensive experiments.**

***Index Terms*—String matching, information retrieval, natural language processing.**

I. Introduction

SIMILAR sentence matching is an essential issue because it is the basis of many applications, such as as text summarization, image extraction, social media retrieval, question-answer model, and so on.

Traditional techniques for measuring the similarity between documents (long texts), e.g., TF-IDF, have been introduced based on an intuitive assumption that a large number of common words exist in similar documents. However, these methods are inappropriate for measuring similarities between sentences because in short texts common words are few or even nonexistent [1]–[3]. To address this issue, numerous strategies have been proposed to measure the similarity between sentences. These strategies can be classified into four categories: (1) knowledge-based [2], [3]; (2) string similarity based [4], [5]; (3) corpus-based [6], [7]; and (4) hybrid strategies [1], [6].

As far as we know, the most comprehensive framework for sentence similarity calculation is introduced in [6]. The authors integrate several representative string-based and corpus-based (i.e., BNC) similarities. It is well known that WordNet and Wiki are important semantic resources and have been extensively studied on the measurement of semantic similarities [1], [2], [8]. An intuitive idea is to incorporate these semantic resources (i.e., WordNet and Wiki) into the general framework (i.e., [6]) to improve the effectiveness. In the first part of this paper, we thoroughly explore the idea, that evaluates the effect of different measurements on calculating sentence similarities. We believe that this is the first work which comprehensively studies the sentence similarity measurement by using most semantic resources.

In addition to the effectiveness aspect, efficiently searching similar sentences from a large number of data has become an important issue [9], [10] in the literature. From a given sentence collection, such queries aim to identify sentences that are most semantically similar to a given one. A naive approach can employ the following procedure: we first measure the semantic similarity score between the query and each sentence in the data collection using state-of-the-art techniques. The sentences are then sorted based on the score. Finally, the top-k sentences are identified and returned to the user. However, as the data collection size increases, the scale of the problem likewise increases, thus rendering state-of-the-art techniques impractical [10], [11], which highlights the importance of the efficiency issue. Several works explored optimization strategies for similarity measurement. In [12], the author addressed the efficiency issue by optimizing the string similarity, WordNet similarity and semantic similarity of words. An efficient method for the extraction of similar sentences was proposed in [9], where different strategies were combined by applying the threshold algorithm. In this paper, taking into account the new similarities (i.e., WordNet and Wiki), we introduce the corresponding optimization strategies to improve the efficiency. The trade-off between effectiveness and efficiency is also studied in this paper.

The contributions of this paper are as follows:

- We introduce several representative similarity measurement strategies and evaluate the effectiveness of each strategy individually as well as that of different combinations.
- We propose a dynamic weight tuning strategy to improve the effectiveness of the similarity measure. In addition, we also study the weight setting of the combination of different similarity strategies.
- We introduce optimization strategies for the new semantic resources (i.e., WordNet and Wiki) to improve the efficiency of sentence similarity matching.

Manuscript received on December 15, 2012; accepted for publication on January 11, 2013.

All authors are with the Institute of Industrial Science, University of Tokyo, Komaba 4-6-1, Meguro, Tokyo, 153-8505, Japan (e-mail: {guyanhui,yangzl,miyuki,kitsure}@tkl.iis.u-tokyo.ac.jp).

– We conduct comprehensive experiments to evaluate the performance of the proposed strategies. The results show that the proposed strategies outperform the state-of-the-art method. We also illustrate the trade-off between effectiveness and efficiency.

II. Preliminaries

To measure the similarity $sim(Q, P)$ between two sentences Q and P, we apply state-of-the art strategies by assembling multiple similarity metric features [1], [6]. Given that we cannot evaluate all the similarity measurement strategies in this paper, we select several representative features based on the framework presented in [6]. Notably, considering that a sentence comprises a set of words, the similarity score between two sentences denotes the overall scores of all word pairs, the components of which belong to each sentence. See [6] for detail on computing sentence similarity based on word similarity.

A. Similarity Measurement Strategies

1) String Similarity: String similarity measures the difference in syntax between strings. An intuitive idea is that two strings are similar to each other if they have adequate common subsequences (e.g., LCS [13]). String similarity measurement strategies, including edit-distance, hamming distance and so on. We focus on three representative string similarity measurement strategies introduced in [6], namely, NLCS, NMCLCS_1 and NMCLCS_n[1].

2) Corpus-based Similarity: The corpus-based similarity measurement strategy recognizes the degree of similarity between words using large corpora, e.g., BNC, Wikipedia, Web and so on. Corpus-based similarity measurement strategies are of several types: PMI-IR, LSA, HAL, and so on. In this paper, we apply the Second Order Co-occurrence PMI (SOC-PMI) [6], [14] which employs PMI-IR to consider important neighbor words in a context window of the two target words from a large corpus. They use PMI-IR to calculate the similarities between word pairs (including neighbor words). High PMI scores are then aggregated to obtain the final SOC-PMI score.

3) Common Word Order Similarity: Common word order similarity measures how similar the order of the common-words is between two sentences, as either the same order, almost the same order, or very different order. Although [15] indicates that syntactic information is less important during the semantic processing of sentences, we incorporate this similarity measurement strategy to test how much order similarity affects the whole sentence similarity. See [1], [6] for detail.

[1]NLCS: Normalized Longest Common Substring; NMCLCS_1: Normalized Maximal Consecutive LCS starting at character 1; NMCLCSn: Normalized Maximal Consecutive LCS starting at any character n. See [6] for detail.

B. General Framework for Measuring Sentence Similarity

To measure the overall similarity between two sentences, a general framework is presented by incorporating all similarity measurement strategies. To the best of our knowledge, [6] presented the most comprehensive approach that incorporates representative similarity metrics. They construct a similarity matrix and recursively extract representative words (maximal-valued element) which are then aggregated to obtain the similarity between two sentence.

III. Effectiveness Improvement with Additional Semantic Resources

Hybrid approaches incorporate different similarity strategies, such as string similarity, knowledge-based similarity, corpus-based similarity, etc. It is well known that WordNet and Wiki are two representative semantic resources and have been extensively studied on the measurement of semantic similarities [1], [2], [8]. Based on the general framework which is introduced in [6], in this paper we propose to take into account the additional important semantic resources (i.e., Wordnet and Wiki), to improve the effectiveness of the sentence similarity measurement. We explore the effect of different similarity metrics by using equal-weight setting (Section III), cross-validation (Section IV), and dynamic weight tuning (Section V). Efficiency optimization on sentence similarity matching is introduced in Section VI.

A. Two Additional Semantic Resources

1) WordNet-based Similarity Strategy: A word thesauri such as WordNet, constitutes the knowledge base for text-related research. An intuitive idea to determine whether two words are semantically similar to each other is by finding if the shortest path between them is small. This edge-counting approach has been extended by incorporating additional features in the knowledge base, such as depth, information content, or semantic density. We select one representative metric proposed in [16], that is, Leacock and Chodorow strategy. We take two words w_i,w_j, the similarity of which is determined as follows:

$$Sim_{lch}(w_i, w_j) = -ln\frac{length(w_i, w_j)}{2 * D},$$

where $length(w_i, w_j)$ is the length of the shortest path between two concepts (by applying node-counting strategy). D is the maximum depth of the taxonomy.

2) Wiki-based Similarity Strategy: Unlike taxonomy-based methods, such as the WordNet-based strategy, Wiki-based similarity cannot employ a new entity. An representative strategy, ESA [8], which applies the Wiki encyclopedia as a knowledge base to map text into Wiki-based concepts. In this approach, each Wiki concept is represented as an attribute vector of words that occur in the corresponding article. Entries of these vectors are assigned weights by using the TF-IDF scheme which quantifies the strength of

association between words and Wiki concepts. ESA measures similarity between sentences (arbitrary length) by aggregating each word distribution on concepts, i.e., sentence is a vector based on concepts with weight of each concept c_i calculated as: $\sum_{w_i \in T} v_i \cdot k_j$, where v_i is TF-IDF weight of w_i and k_j quantifies the strength of association of word w_i with Wiki concept c_j.

B. Experimental Evaluation

In this section, we first evaluate the single strategy and then the different combination of similarity strategies. Besides $Sim_{Baseline}$ (baseline is the combination of string similarity and BNC-based semantic similarity), we incorporate two different strategies, such as $Sim_{WordNet}$ and Sim_{Wiki} into the framework with equal weight. We apply $benchmark$ dataset (Miller-Charles' dataset) which has also been used in [1] to evaluate the effectiveness in this and the following sections. Figure 1 illustrates the results of the correlation coefficient with human ratings.

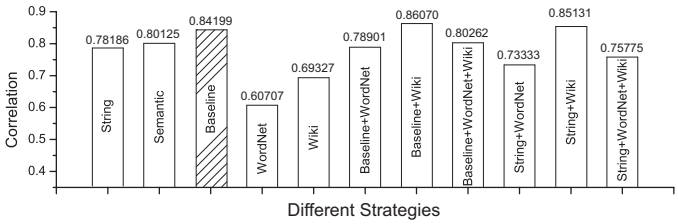


Fig. 1. Results of different strategies combination with equal weight

C. Result Explanation

The figure shows that all single similarity strategies are worse than the baseline strategy. Wiki is a good semantic resource because the combination of Wiki achieve better results than others but WordNet is not good on this dataset. However, the combination strategies are better than single similarity strategies but still falls short of the baseline strategy. Given their equal weights, all similarity strategies have the same proportion of similarity aggregation, that is, the weight is static and set arbitrarily.

IV. Effectiveness Improvement with Cross-Validation

Considering that the weight in Section III is set to be equal, in this section, we test each possible weight of the combination strategy. Obtaining the optimal values of these weights is certainly an important issue. We argue that this work is orthogonal to the existing effectiveness oriented studies in a complementary manner. We try to achieve the best effectiveness by testing each possible weight.

A. Cross-Validation

Cross-validation strategy can test all the possible weight combination results with human ratings. In this paper, we apply a 10-fold cross-validation strategy and study what weight can achieve the best effectiveness.

B. Experimental Evaluation

We conduct experiments on the benchmark dataset. Table I shows that, by applying cross-validation, we can obtain better results from combination strategies. From Table I, we observe that the WordNet strategy still has extremely low effectiveness. In addition, combination strategies that contain the WordNet strategy are also outperformed by other strategies.

C. Result Explanation

The experiment results show that setting weight arbitrarily and equally is improper for similarity measurement, especially in combination strategies. We can achieve better results by using cross-validation strategy. However, from the experiment results of Section III and Section IV, we can see that, the results of combination of WordNet are still low.

V. Effectiveness Improvement with Dynamic Weight Tuning

From Section IV we can see that, WordNet is not a good semantic resource when measuring the semantic similarity under the benchmark dataset. Because of the omission of two word pairs in WordNet, similarity score of these words are "0" which affect the whole similarities. In this section, we reduce the weight of these words which are not included in WordNet by dynamically weight tuning.

A. Dynamic Weight Tuning

To address this issue, one possible solution is to remove these words when calculating the similarity score. However, such a strategy may affect the similarity score of other strategies because of the reduction in the number of words. Another solution is by dynamically tuning the combination weight. We take the $String + WordNet$ strategy as an example. If the similarity score of some word pairs are "0", but this value holds a rather large weight, which can reduce the final similarity score. We propose a dynamic combination weight tuning strategy to address this issue. We denote two sentences Q and P, which have m and n words, respectively. A total of γ words are not included in WordNet of sentences P and Q. Based on the strategy in [6], at least $\gamma * min(m,n)$ or at most $\gamma * max(m,n)$ word pairs are "0" (we apply average value $\frac{(m+n)}{2} \cdot \gamma$). Therefore, we mitigate the effect of WordNet by tuning the weight to $\frac{1}{k} \cdot (1 - \frac{m+n}{2mn} \cdot \gamma)$, where k is the number of combination strategies. So, $Sim_{String+WordNet}$ = $\frac{1}{k} \cdot \frac{2mn(k-1)+(m+n)\cdot\gamma}{2mn} \cdot Sim_{String}$+ $\frac{1}{k} \cdot \frac{2mn-(m+n)\cdot\gamma}{2mn} \cdot \gamma \cdot Sim_{WordNet}$, where k = 2 in this case.

TABLE I
CORRELATION COEFFICIENT ON COMBINATION WITH CROSS-VALIDATION

Strategy	Correlation	Weight			
		String	Semantic	WordNet	Wiki
Baseline+WordNet	0.82019	0.377	0.531	0.092	–
Baseline+Wiki	0.86073	0.470	0.210	–	0.320
Baseline+WordNet+Wiki	0.81002	0.406	0.301	0.072	0.221
String+WordNet	0.77815	0.699	–	0.301	–
String+Wiki	0.86132	0.579	–	–	0.421
String+WordNet+Wiki	0.80126	0.470	–	0.110	0.420

Note: Baseline strategy involving string and semantic strategy.

B. Experimental Evaluation

We apply this strategy and conduct experiments on the benchmark dataset. Table II shows the results by dynamically tuning the weight.

Table II shows that, we obtain better correlation coefficient by dynamically tuning the weight. For $String + WordNet + Wiki$, we obtain a better result than $Baseline + Wiki$ and $String + Wiki$.

C. Result Explanation

From the experiment results, we can see that dynamically tuning the weight of each similarity strategy is a possible solution to improve the effectiveness. However, all the weight tuning is conducted on the benchmark dataset. The weight may be changed if we apply the strategy to another dataset. Supervised learning techniques could solve such an issue but are out of the scope of this paper.

Common word order is an important strategy in similarity measurement. To evaluate the effect of common word order similarity, we incorporate such similarity strategy into the framework. We incorporate common word order similarity into baseline strategy. Experiments conducted on the benchmark dataset illustrated that only 19 of 30 pairs have common words. Of all these 19 pairs, 15 pairs have exactly the same order (including 10 pairs which have only "1" common word). Such similarity only affects 4 pairs. We set weight of common word order similarity from 0 to 0.5 with the granularity "0.01" and we obtain the best correlation coefficient "0.81972" at 0.01 which is less than baseline.

VI. EFFICIENCY IMPROVEMENT

Searching for similar sentences from a large amount of data has become an important issue [9], [17] in the literature. From a given sentence collection, such queries aim to identify sentences that are most semantically similar to a given one. A naive approach could be: we first measure the similarity score between the query and each sentence in the data collection using state-of-the-art techniques [1], [2], [6]. The sentences are then sorted based on a score. Finally, the top-k statements are identified and returned to the user. However, as the size of the collected data, testing each candidate sentence becomes time consuming. In [9], the author proposed a solution which optimized state-of-the-art techniques to retrieve top-k sentences. Techniques that optimize string similarity and semantic similarity is proposed. From the analysis in Section V, we select a best similarity combination strategy that has the best effectiveness. We select one representative strategy which achieves best performance, that is, $string$, $WordNet$ and $Wiki$, with the weight "0.420", "0.210" and "0.370", respectively.

A. Optimization on WordNet

We apply the Leacock and Chodorow strategy as a WordNet evaluator which is an efficient technique [12].

Lemma 1 (Ordering in WordNet): Let Q be the query. Let P and S be two candidates that exist in the same taxonomy of Q, that is, T_P and T_Q. The shortest path between Q and P (or S) is L_P in T_P (or L_S in T_S). The maximum depth of T_P is D_P (or D_S of T_S). P is more similar to Q compared with S. Thus, we have $\frac{D_P}{L_P} > \frac{D_S}{L_S}$.

The lemma tells us that the similarity ordering between candidates in WordNet depends on the integration of the shortest path and the maximum depth of the taxonomy. For example, $father$ is in both a noun taxonomy (i.e., $\frac{D}{L} = 19$) and a verb taxonomy (i.e., $\frac{D}{L} = 14$)[2]. Thus, $father$ in a noun taxonomy should be accessed before that in a verb taxonomy. Sequentially we access the synonyms set between two taxonomies successively based on the value of $\frac{D}{L}$. Based on this lemma, we index all the candidates together with their neighbors and maximum taxonomy depth. We sequentially access nodes based on Lemma 1 and obtain the top-k results in a progressive manner.

B. Optimization on Wiki

ESA measures the similarity between sentences (arbitrary length) by aggregating each word distribution on concepts, that is, a sentence is a vector based on concepts with the weight of each concept c_i calculated as: $\sum_{w_i \in T} v_i \cdot k_j$, where v_i is TF-IDF weight of w_i and k_j quantifies the strength of association of word w_i with Wiki concept c_j. The traditional approach has to test each candidate in the data collection. In our optimized strategy, we first calculate all the similarity scores between each word in Wiki and between sentences in

[2]The maximum depths of the two taxonomies are 19 for noun and 14 for verb by querying WordNet during preprocessing.

TABLE II
CORRELATION COEFFICIENT ON DYNAMICALLY WEIGHT TUNING

Strategy	Correlation	Weight			
		String	Semantic	WordNet	Wiki
Baseline+WordNet	0.83033	0.408	0.375	0.217	–
Baseline+Wiki	0.86073	0.470	0.210	–	0.320
Baseline+WordNet+Wiki	0.84752	0.334	0.314	0.157	0.195
String+WordNet	0.79378	0.649	–	0.351	–
String+Wiki	0.86132	0.579	–	–	0.421
String+WordNet+Wiki	0.86201	0.420	–	0.210	0.370

the data collection to obtain a set of lists during preprocessing which is illustrated in Figure 2. Then we build a weighted inverted list, here each list indicates a word with sorted corresponding sentences based on the similarity score. Given a query sentence Q, each word in Q corresponds a list of sentences. Therefore, we apply the threshold algorithm [18] with TF-IDF weight to retrieve the top-k sentences. This manner accesses a small number of components of the data collection without need to test every candidate sentence.

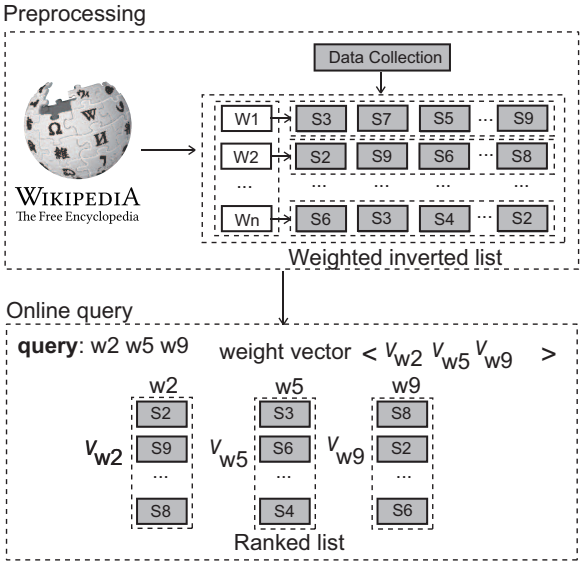


Fig. 2. Optimization on Wiki based strategy

C. Assembling Similarity Features

We introduce an efficient assembling approach to accelerate the process of searching for top-k similar sentences [18]. In [9], the author illustrated the method by using a concrete example. In this paper, we apply three different similarity measurement strategies, $String$, $WordNet$ and $Wiki$. We apply the threshold based strategy in assembling different similarities as well as in assembling words into a sentence to obtain the top elements. Given the page limitation, we do not include detailed explanations here.

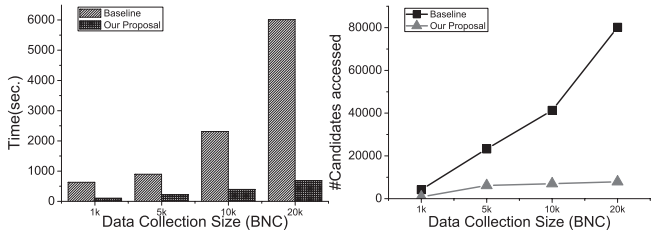
D. Experimental Evaluation on Efficiency

To evaluate the efficiency, we conduct extensive experiments on two large real datasets: BNC dataset (extracted from British National Corpus); MSC dataset (extracted from Microsoft Research Paraphrase Corpus).[3] Table III shows the statistics of these two datasets. (Statistics after preprocessing is italicized.)

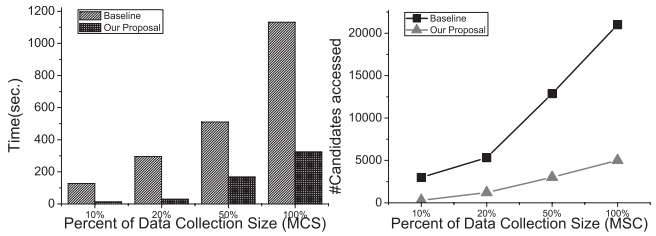
TABLE III
DATASET STATISTICS

	BNC		MSC	
Avg. sentence length	11.72	*7.19*	15.23	*9.31*
Min. sentence length	3	*2*	5	*3*
Max. sentence length	107	*38*	29	*17*
Max word length	17	*17*	13	*13*

1) Evaluation on Effect of Data Collection Size: Figure 3 shows the top-5 results after 10 randomly selected queries. We can see that our proposed optimized strategy is significantly faster than the baseline strategy for both datasets because our strategy substantially reduces the number of candidates tested. As the size of collected data increases, the query time of our proposed strategy also increases linearly and proportionately well.



(a) BNC dataset



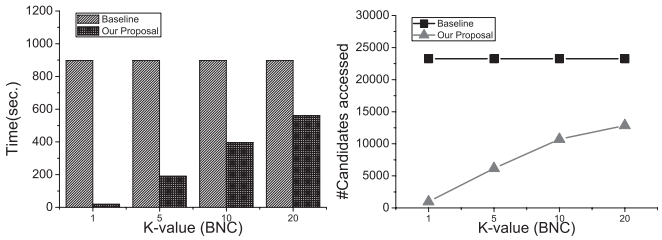
(b) MSC dataset

Fig. 3. Effect of data collection size

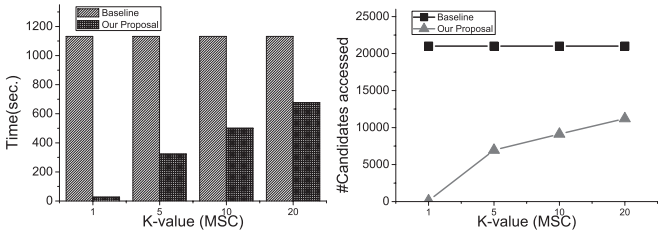
2) Evaluation on Effect of k value: In addition, we also verify the effect of k value. We randomly chose 10 queries

[3] 1k, 5k, 10k, 20k sentences are extracted from BNC and 10%, 20%, 50%, 100% of MSC are divided. After removing the duplicated sentences in MSC, 11,212 sentences are remained.

from both datasets and fix the data size to be 5k for BNC and the whole size for MSC. Figure 4 shows that the baseline has to access all candidate sentences, such that, the query time is the same for all the situations. For our proposed method, the top-1 can be returned almost instantly. The query time increases when k increases because more candidates need to be accessed.



(a) BNC dataset



(b) MSC dataset

Fig. 4. Effect of k-value

VII. Trade-off between Effectiveness and Efficiency

In Section VI-D, we prove that, our proposed optimization strategy can significantly reduce the execution time when retrieving top-k values. However, effectiveness and efficiency require a trade-off. We conducted experiments on the benchmark dataset by using $Baseline$ and $Baseline + WordNet + Wiki$ strategies to retrieve top-5 results. Table IV tells us combining several strategies can achieve high precision but may be time consuming. Therefore, designing an effective similar sentence matching framework with high efficiency remains a challenge.

TABLE IV
Trade-off between effectiveness and efficiency

Strategy	Effectiveness	Efficiency
	Correlation	Execution Time(Sec.)
Baseline(String+Semantic[BNC])	0.84019	2.90
Baseline+WordNet+Wiki	0.86201	3.32

VIII. Related Work

Measuring similarity between long texts has been extensively studied [7], [19]. However, only a few of them can be directly applied to sentence similarity measurement [1], [2], [6]. Based on the different strategies applied, existing works on similarity measurement between sentences can be classified into several categories:

String similarity based strategy. Numerous strategies estimate the string similarity between two texts [20]. One representative q-gram based strategy calculates the edit distance between words. In [21] the authors proposed several strategies, including adaptive q-gram selection, for the efficient retrieval of the top-k results. In [22], the authors introduced deliberated techniques, e.g., divide-skip-merge, to extract similar strings.

Knowledge-based strategy. Knowledge base (sometimes called word thesauri), e.g., WordNet, contains the labeled (or semi-labeled) data for text related research tasks. In [3], they firstly create semantic networks from word thesauri and then measure the relatedness between words based on these semantic networks. The hierarchy property of WordNet has been explored in [1]. The word pair similarity is estimated from the hierarchy based on a node counting strategy, i.e., calculating the number of nodes between the target words.

Corpus-based strategy. Statistics information of large corpus can be used to calculate the similarity between two words or texts. Some well known methods in corpus-based similarity are LSA (Latent Semantic Analysis) and HAL (Hyperspace Analogues to Language), etc. One representative strategy ESA (Explicit Semantic Analysis) [8] which applies machine learning techniques to explicitly represent the meaning of any text as a weighted vector of Wiki-based concepts.

Hybrid strategy. To tackle the drawback of single strategy, the hybrid strategy was proposed [1], [6]. The combination of knowledge based strategy and word order based strategy was proposed in [1]. In [6], the author applies string based, common word order based, and corpus based strategies to measure the similarity between sentences.

Currently, several works [9], [12] explore efficiency issue to optimize state-of-the-art similarity strategy. Efficient extraction on semantic similar words is presented in [12] by optimizing string-based, WordNet-based and corpus-based similarity strategies. In [9], the authors address efficiency issue to efficiently search for semantic similar sentences on three string similarity strategies and corpus-based strategy.

IX. Conclusion

In this paper, we have studied both effectiveness and efficiency aspect in the sentence similarity matching. The optimal strategies have been proposed through cross-validation and dynamically weight tuning. We also introduced several efficient techniques to improve the performance of the similarity computation. The trade-off between effectiveness and efficiency is also explored by conducting extensive experiments.

References

[1] Y. Li, D. McLean, Z. Bandar, J. O'Shea, and K. A. Crockett, "Sentence similarity based on semantic nets and corpus statistics." *IEEE*

Transactions on Knowledge and Data Engineering, vol. 18, no. 8, pp. 1138–1150, 2006.

[2] R. Mihalcea, C. Corley, and C. Strapparava, "Corpus-based and knowledge-based measures of text semantic similarity," in *Proceedings of the AAAI Conference on Artificial Intelligence*, ser. AAAI'06, 2006, pp. 775–780.

[3] G. Tsatsaronis, I. Varlamis, and M. Vazirgiannis, "Text relatedness based on a word thesaurus." *Journal of Artificial Intelligence Research*, vol. 37, pp. 1–39, 2010.

[4] W. W. Cohen, "Integration of heterogeneous databases without common domains using queries based on textual similarity," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD'98, 1998, pp. 201–212.

[5] G. Navarro, "A guided tour to approximate string matching," *ACM Computing Surveys*, vol. 33, no. 1, pp. 31–88, 2001.

[6] A. Islam and D. Inkpen, "Semantic text similarity using corpus-based word similarity and string similarity," *ACM Transactions on Knowledge Discovery from Data*, vol. 2, no. 2, pp. 1–25, 2008.

[7] T. K. Landauer and S. T. Dumais, "A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge," *Psychological Review*, vol. 104, pp. 211–240, 1997.

[8] E. Gabrilovich and S. Markovitch, "Computing semantic relatedness using wikipedia-based explicit semantic analysis," in *Proceedings of the International Joint Conference on Artifical Intelligence*, ser. IJCAI'07, 2007, pp. 1606–1611.

[9] Y. Gu, Z. Yang, M. Nakano, and M. Kitsuregawa, "Towards efficient similar sentences extraction," in *Proceedings of Intelligent Data Engineering and Automated Learning*, ser. IDEAL'12, 2012, pp. 270–277.

[10] P. Pantel, E. Crestan, A. Borkovsky, A.-M. Popescu, and V. Vyas, "Web-scale distributional similarity and entity set expansion." in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP'09, 2009, pp. 938–947.

[11] A. Goyal and H. Daumé, III, "Approximate scalable bounded space sketch for large data nlp," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP'11, 2011, pp. 250–261.

[12] Z. Yang and M. Kitsuregawa, "Efficient searching top-k semantic similar words," in *Proceedings of the International Joint Conference on Artificial Intelligence*, ser. IJCAI'11, 2011, pp. 2373–2378.

[13] D. S. Hirschberg, "A linear space algorithm for computing maximal common subsequences," *Communications of ACM*, vol. 18, no. 6, pp. 341–343, 1975.

[14] A. Islam and D. Inkpen, "Second order co-occurrence pmi for determining the semantic similarity of words," in *Proceedings of the International Conference on Language Resources and Evaluation*, ser. LREC'06, 2006, pp. 1033–1038.

[15] P. Wiemer-Hastings, "Adding syntactic information to lsa," in *Proceedings of the Annual Conference of the Cognitive Science Society*, ser. COGSCI'00, 2000, pp. 989–993.

[16] C. Leacock and M. Chodorow, "Combining local context and wordnet similarity for word sense identification," in *WordNet: An Electronic Lexical Database*. In C. Fellbaum (Ed.), MIT Press, 1998, pp. 305–332.

[17] M. B. Blake, L. Cabral, B. König-Ries, U. Küster, and D. Martin, *Semantic Web Services: Advancement through Evaluation*. Springer, 2012.

[18] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," in *Proceedings of the ACM SIGMOD symposium on Principles of Database Systems*, ser. PODS'01, 2001, pp. 102–113.

[19] V. Hatzivassiloglou, J. L. Klavans, and E. Eskin, "Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning," in *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, ser. EMNLP/VLC'99, 1999, pp. 203–212.

[20] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Physics Doklady*, vol. 10, no. 8, pp. 707–710, 1966.

[21] Z. Yang, J. Yu, and M. Kitsuregawa, "Fast algorithms for top-k approximate string matching," in *Proceedings of the AAAI Conference on Artificial Intelligence*, ser. AAAI'10, 2010, pp. 1467–1473.

[22] S. Sarawagi and A. Kirpal, "Efficient set joins on similarity predicates," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD'04, 2004, pp. 743–754.

TopicSearch—Personalized Web Clustering Engine Using Semantic Query Expansion, Memetic Algorithms and Intelligent Agents

Carlos Cobos, Martha Mendoza, Elizabeth León, Milos Manic, and Enrique Herrera-Viedma

***Abstract*—As resources become more and more available on the Web, so the difficulties associated with finding the desired information increase. Intelligent agents can assist users in this task since they can search, filter and organize information on behalf of their users. Web document clustering techniques can also help users to find pages that meet their information requirements. This paper presents a personalized web document clustering called TopicSearch. TopicSearch introduces a novel inverse document frequency function to improve the query expansion process, a new memetic algorithm for web document clustering, and frequent phrases approach for defining cluster labels. Each user query is handled by an agent who coordinates several tasks including query expansion, search results acquisition, preprocessing of search results, cluster construction and labeling, and visualization. These tasks are performed by specialized agents whose execution can be parallelized in certain instances. The model was successfully tested on fifty DMOZ datasets. The results demonstrated improved precision and recall over traditional algorithms (k-means, Bisecting k-means, STC y Lingo). In addition, the presented model was evaluated by a group of twenty users with 90% being in favor of the model.**

***Index Terms*—Web document clustering, intelligent agents, query expansion, WordNet, memetic algorithms, user profile.**

I. Introduction

In recent years, web document clustering has become a very interesting research area among academic and scientific communities involved in information retrieval (IR) and web search [1]. Web document clustering systems seek to increase the coverage (amount) of documents presented for the user to review, while reducing the time spent in reviewing documents [2]. In IR, these web document clustering systems are called web clustering engines. Among the most prominent ones are Carrot (www.carrot2.org), SnakeT (snaket.di.unipi.it), Yippy (yippy.com, originally named as Vivisimo and then as Clusty), iBoogie (www.iboogie.com), and KeySRC (keysrc.fub.it) [3]. Such systems usually consist of four main components: search results acquisition, preprocessing of input, cluster construction and labeling, and visualization of resulting clusters [1] (see Fig 1).

The **search results acquisition** component begins with a query defined by the user. Based on this query, a document search is conducted in diverse data sources, in this case in the traditional web search engines such as Google, Yahoo! and Bing. In general, web clustering engines work as meta search engines and collect between 50 to 200 results from traditional search engines. These results contain as a minimum a URL, a snippet and a title [1].

The **preprocessing** of search results comes next. This component converts each of the search results (as snippets) into a sequence of words, phrases, strings or general attributes or characteristics, which are then used by the clustering algorithm. There are a number of tasks performed on the search results, including: removing special characters and accents, the conversion of the string to lowercase, removing stop words, stemming of the words and the control of terms or concepts allowed by a vocabulary [1].

Once the preprocessing is finished, **cluster construction and labeling** is begun. This stage makes use of three types of algorithm [1]: data-centric, description-aware and description-centric. Each of these builds clusters of documents and assigns a label to the groups.

Data-centric algorithms are the algorithms traditionally used for data clustering (partitional, hierarchical, density-based, etc.) [1, 4-10]. They look for a solution in data clustering, but lack in their capabilities presentation of the labels and in providing explanations of the groups obtained. These algorithms address the problem of web document clustering as merely another data clustering problem.

Description-aware algorithms put more emphasis on one specific feature of the clustering process. For example, they might put a priority on the quality of the labeling of groups and as such achieve results that are more easily interpreted by the user. The quality of these algorithms however deteriorates during the cluster creation process. An example of this type of algorithm is Suffix Tree Clustering (STC) [8], which incrementally creates labels easily understood by users, based on common phrases that appear in the documents.

Manuscript received on March 13, 2013; accepted for publication on May 23, 2013.

Carlos Cobos and Martha Mendoza are with the University of Cauca, Colombia (e-mail: {ccobos,mmendoza}@unicauca.edu.co).

Elizabeth León is with the Universidad Nacional de Colombia, Colombia (e-mail: eleonguz@unal.edu.co).

Milos Manic is with the University of Idaho at Idaho Falls, USA (e-mail: misko@uidaho.edu)

Enrique Herrera-Viedma is with University of Granada, Spain (e-mail: viedma@decsai.ugr.es)

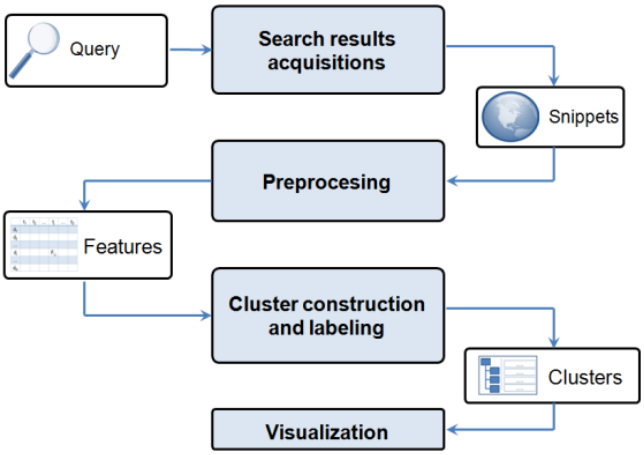


Fig 1.The components of a web clustering engine (adapted from [1])

Description-centric algorithms [1, 7, 11-15] are designed specifically for web document clustering, seeking a balance between the quality of clusters and the description (labeling) of clusters. An example of such algorithms is Lingo [11] (implemented by www.carrot2.org), which makes use of Singular Value Decomposition (SVD) to find the best relationships between terms, but groups the documents based on the most frequent phrases in the document collection.

Finally, in the **visualization** step, the system displays the results to the user in folders organized hierarchically. Each folder seeks to have a label or title that represents well the documents it contains and that is easily identified by the user. As such, the user simply scans the folders that are actually related to their specific needs. The presentation folder tree has been adopted by various systems such as Carrot2, Yippy, SnakeT, and KeySRC, because the folder metaphor is already familiar to computer users. Other systems such as Grokker and Kart004 use a different display scheme based on graphs [1].

In order to obtain satisfactory results in web document clustering, the algorithms must meet the following specific requirements [1, 8]: Automatically define the number of clusters that are going to be created; generate relevant clusters for the user and assign these documents to appropriate clusters; define labels or names for the clusters that are easily understood by users; handle overlapping clusters (the document can belong to more than one cluster); reduce the high dimensionality of document collections; handle the processing time i.e. less than or equal to 2.0 seconds; and handle the noise frequently found in documents.

Another important aspect of web document clustering algorithms is the document representation model. The most widely used models are [16]: Vector space model [2, 4], Latent Semantic Indexing (LSI) [2, 11], Ontology-based model [7, 17], N-gram [8], Phrase-based model [8], and Frequent Word (Term) Sets model [7, 18].

In Vector space model (VSM), the documents are designed as bags of words. Document collection is represented by a matrix of D-terms by N-documents. This matrix is commonly called Term by Document Matrix (TDM). In TDM, each document is represented by a vector of normalized frequency term by document inverse frequency for that term, in what is known as the TF-IDF value. In VSM, the cosine similarity is used for measuring the degree of similarity between two documents or between a document and the user's query. In VSM as in most of the representation models, a process of stop word removal and stemming [2] should be done before re-presenting the document. Stop word removal refers to the removal of very common words (like articles and prepositions, so can yield over 40% reduction on TDM matrix dimensionality), while stemming refers to the reduction of words to their canonical stem or root form.

The two predominant problems with existing web clustering are inconsistencies in cluster content and inconsistencies in cluster description [1]. The first problem refers to the content of a cluster that does not always correspond to the label. Also, the navigation through the cluster hierarchies does not necessarily lead to more specific results. The second problem refers to the need for more expressive descriptions of the clusters (cluster labels are confusing).This is the main motivation of the present work, in which a personalized web clustering engine modeled by agents is put forward. This model is developed to work on-line and off-line, which means that users can define the processing time (for example, it can be fixed at a value of less than two seconds to work on-line) of agents in the entire process of search, clustering and visualization. To the best of our knowledge, this research is the first to integrate synergistically web document clustering, the semantic query expansion process (based on WordNet and user profile), and memetic algorithms through a model of intelligent agents.

The remainder of the paper is organized as follows. Section 2 presents related work. Section 3 presents a personalized web document clustering model, i.e. the query expansion process, the clustering and labeling algorithm, and the user profile. Section 4 shows the experimental results. Finally, some concluding remarks and suggestions for future work are presented.

II. Related Work

In general, clustering algorithms can be classified into [19]: hierarchical, partitional, density-based, grid-based, and model-based algorithms, among others. The algorithms most commonly used for web document clustering have been the hierarchical and the partitional [4]. The hierarchical algorithms generate a dendogram or a tree of groups. This tree starts from a similarity measure, among which are: single link, complete link and average link. In relation to web document clustering, the hierarchical algorithm that brings the best results in accuracy is called UPGMA (Unweighted Pair-Group Method using Arithmetic averages) [5]. UPGMA was

devised in 1990 [7] and is based on the vector space model, using an average link based on the clusters cosine similarity divided by the size of the two clusters that are being evaluated. UPGMA has the disadvantage of having a time complexity of $O(n^3)$ and being static in the process of assigning documents to clusters.

In partitional clustering, the algorithms perform an initial division of the data in the clusters and then move the objects from one cluster to another based on the optimization of a predefined criterion or objective function [19]. The most representative algorithms using this technique are: K-means, K-medoids, and Expectation Maximization. The K-means algorithm is the most popular because it is easy to implement and its time complexity is $O(n)$, where n is the number of patterns or records, but it has serious disadvantages: it is sensitive to outliers, it is sensitive to the selection of the initial centroids, it requires a prior definition of the number of clusters, and the obtained clusters are only hyper spherical in shape [8]. In 2000, a Bisecting K-means [4, 7] algorithm was devised. This algorithm combines the strengths of the hierarchical and partitional methods reporting better results concerning the accuracy and the efficiency of the UPGMA and the K-means algorithms.

The first algorithm to take the approach based on frequent phrases shared by documents in the collection was put forward in 1998 and called Suffix Tree Clustering (STC) [7, 8]. Later in 2001, the SHOC (Semantic, Hierarchical, Online Clustering) algorithm was introduced [12]. SHOC improves STC and is based on LSI and frequent phrases. Next in 2003, the Lingo algorithm [11, 20] was devised. This algorithm is used by the Carrot2 web searcher and it is based on complete phrases and LSI with Singular Value Decomposition (SVD). Lingo is an improvement of SHOC and STC and (unlike most algorithms), tries first to discover descriptive names for the clusters and only then organizes the documents into appropriate clusters.

NMF (also in 2003) is another example of these algorithms, it is based on the non-negative matrix factorization of the term-document matrix of the given document corpus was made available [21]. This algorithm surpasses the LSI and the spectral clustering methods in document clustering accuracy but does not care about cluster labels.

Another approach was proposed by the Pairwise Constraints guided Non-negative Matrix Factorization (PCNMF) algorithm [22] in 2007. This algorithm transforms the document clustering problem from an un-supervised problem to a semi-supervised problem using must-link and cannot-link relations between documents. In 2007, the Dynamic SVD clustering (DSC) [14] algorithm was made available. This algorithm uses SVD and minimum spanning tree (MST). This algorithm has better performance than Lingo. Finally, in 2008, the CFWS (Clustering based on Frequent Word Sequences) and the CFWMS (Clustering based on Frequent Word Meaning Sequences) [7] algorithms were proposed. These algorithms represent text documents as frequent word sequences and frequent concept sequences (based on WordNet), respectively and they are mostly used in text clustering.

In relation to a frequent word sets model for web document clustering, in 2002, FTC (Frequent Term-Based Text Clustering) and HFTC (Hierarchical Frequent Term-Based Text Clustering) algorithms became available [15]. These algorithms use combinations of frequent words (association rules approach) shared in the documents to measure their proximity in the text clustering process.

Then in 2003, FIHC (Frequent Item set-based Hierarchical Clustering) was introduced [13] which measures the cohesion of a cluster using frequent word sets, so that the documents in the same cluster share more of the frequent word sets than those in other groups. These algorithms provide accuracy similar to that reported for Bisection K-means, with the advantage that they assign descriptive labels to associate clusters.

Finally, looking at partitional clustering from an evolutionary approach: in 2007, three hybridization methods between the Harmony Search (HS) [23] and the K-means algorithms [24] were compared. These were: Sequential hybridization method, interleaved hybridization method and the hybridization of K-means as a step of HS. As a general result, the last method was the best choice of the three. Later, in 2008 [9, 23, 25], based on the Markov Chains theory the researchers demonstrated that the last algorithm converges to the global optimum.

Next, in 2009, a Self-Organized Genetic [17] algorithm was devised for text clustering based on the WordNet ontology. In this algorithm, a modified LSI model was also presented, which appropriately gathers the associated semantic similarities. This algorithm outperforms the standard genetic algorithm [26] and the K-means algorithm for web document clustering in similar environments. In 2010, two new algorithms were put forward. The first one, called IGBHSK [27] was based on global-best harmony search, k-means and frequent term sets. The second one, called WDC-NMA [28] was based on memetic algorithms with niching techniques. These two researches outperform obtained results with Lingo (Carrot2) over few datasets.

III. The Proposed Model: TopicSearch

TopicSearch is a personalized web clustering engine based on semantic query expansion, user profile, and memetic algorithms through a model of intelligent agents. In the proposed model, the web document clustering problem was transformed from an on-line scenario to an on-line and off-line scenario. With this change, users can execute queries with instant response and users can send a query and see results later. In both scenarios, the results are very promising, but quality of results increases when the clustering algorithm is executed more time.

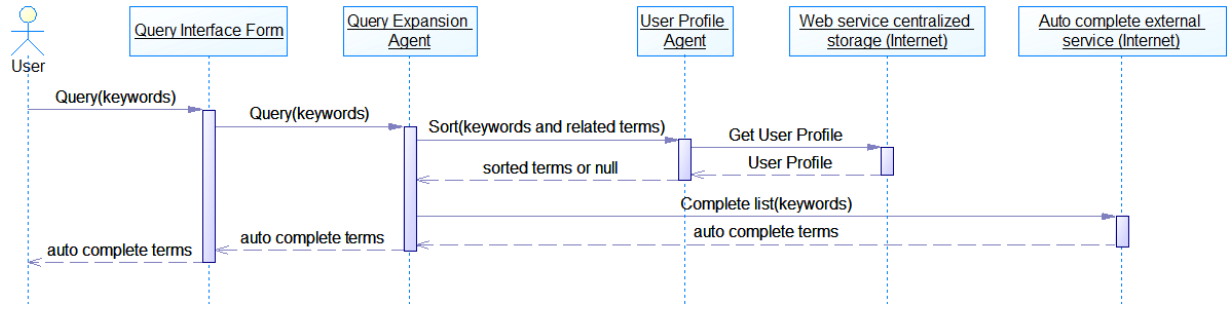


Fig 2. Agents in the Query Expansion Process

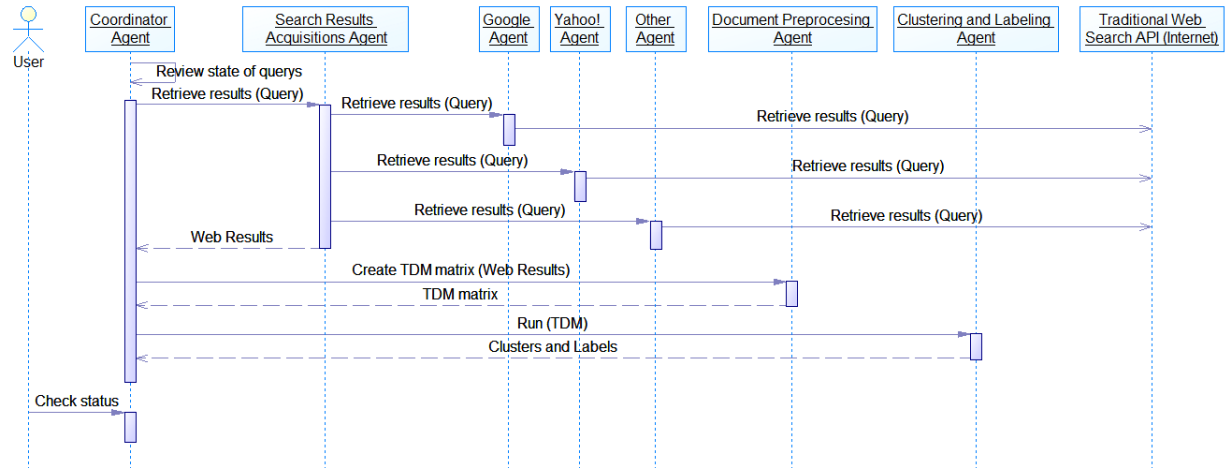


Fig 3. Agents in the Search and Clustering Process

The main actor in TopicSearch is the user. The user can execute multiple queries at the same time. Each query is handled by a group of agents. To present the model, the entire search process was organized into three sub processes: query expansion, search and clustering, and the visualization of results.

Query Expansion: When the user is typing the query it is supported by an interface agent called ***Query Expansion Agent***, which is responsible for displaying a list of terms that help the user to complete the query.

This agent uses the ***User Profile Agent*** which is responsible for finding through a web service the user profile data—in this case, a set of terms with its relevance and correlation among those terms. If the User Profile Agent has no information from the user, the Query Expansion Agent uses an external service of auto-complete, for example, the auto-complete Google service (Fig 2).

Search and Clustering: When the user starts the process of searching, a ***Coordinator Agent*** is activated for each specific query. This agent activates a ***Search Results Acquisitions Agent*** in order to retrieve the results of traditional web search engines like Google, Yahoo!, Bing, etc. At this point the Search Results Acquisitions Agent generates many agents as external web search services registered in the model, thereby achieving a parallel job which reduces the processing time at this stage of the process. In Fig 3 these agents are called ***Google Agent***, ***Yahoo! Agent*** and ***Other Agent***. When the results acquisition process ends, Coordinator Agent activates a ***Document Preprocessing Agent*** in charge of creating a matrix of terms by documents or TDM matrix. After the construction of the TDM matrix, Coordinator Agent activates the ***Clustering and Labeling Agent***, which is responsible for creating clusters of documents based on a memetic algorithm called Agents-WDC and assigns labels to the clusters based on a frequent phrases approach.

As a result, the model obtains a Clustered Documents and Cluster Labels which can be viewed by the user at any time (Fig 3).

Visualization of results: The user visualizes a form with a list of queries that he/she had previously registered in the system. Each query shows a status (Started, Completed, in-Evolution). When the Coordinator Agent is in the process of acquisition or pre-processing of results the query is in the Initiated state and cannot be stopped. When running the Clustering and Labeling Agent, the status of the query is in-Evolution and the process can be stopped. In this case the system generates the cluster labels of the best result found so far and goes on to state Completed. Finally, the completed state occurs when the Coordinator Agent ends the process for the query.

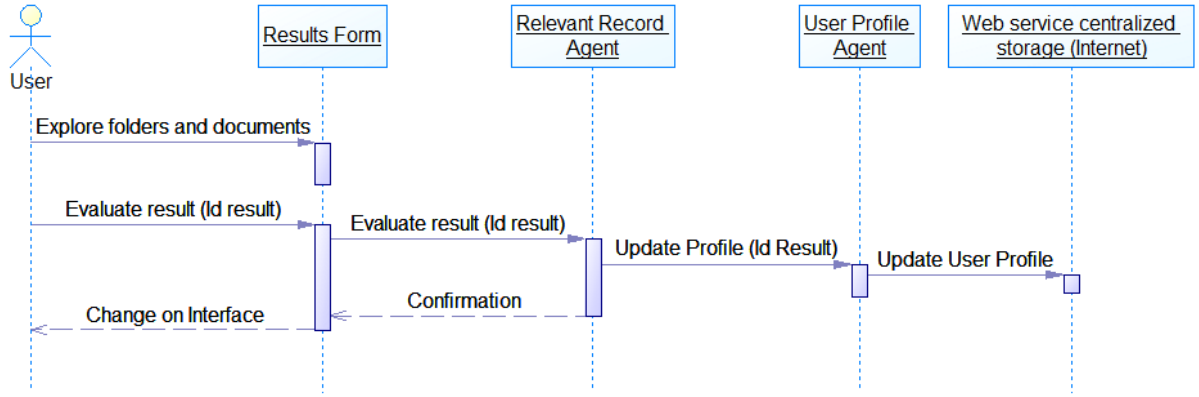


Fig 4. Agents in the Visualization of Results Process

When the user selects the results of a query, the system displays a form (interface) divided into two parts, the left side with a list of cluster labels and the right side with the list of documents belonging to each cluster label. When the user marks a document as relevant or not relevant, the ***Relevant Record Agent*** processes the document terms and through the ***User Profile Agent*** updates the user profile in the web service which centralizes the storage of users (see Fig 4).In this way, future queries can enjoy a more relevant search expansion process based on an updated profile.

The ***Web Service: Centralized Storage*** component allows users to log in from different computers and makes updated user profile information always available. Moreover, with the Windows Client Application (or Smart Client Application), the system takes advantage of the processing capacity of personal computers to reduce the workload of the centralized application server.

On the other hand, the deployment (installation and updates) for smart client applications is becoming increasingly easy to do. Next, we present a detailed description of different components of the model.

A. Query Expansion

In VSM, it has been shown that the process of query expansion improves the relevance (as measured by the accuracy) of the results delivered to users [2, 29, 30]. The expansion of the query in a web search system is usually made from one of two perspectives: user relevance feedback (URF) or automatic relevance feedback (ARF) [2, 29, 30]. URF requires the user to mark documents as relevant or not relevant.

The terms in these marked documents that the system has found to be relevant or not are added to or removed from each of the new user queries [2, 29, 30]. Rocchio proposes formula (1) to generate the expanded query, where q is the query typed by the user initially, R is a set of relevant documents, R′ is a set of non-relevant documents, α, β and γ are tuning constants for the model and q is the expanded query [2, 29, 30]:

$$q_e = \propto * q + \frac{\beta}{|R|}\sum_{d \in R} d - \frac{\gamma}{|R'|}\sum_{d \in R'} d. \quad (1)$$

In contrast, ARF (also known as pseudo feedback) expands the queries automatically based on two methods: considering global documents and considering partial documents [2, 29, 30].

In the global document-based methods, all documents in the collection are analyzed and relationships established between terms (words). As such, these methods are typically performed based on thesauri.

The disadvantage of these methods is that they need all the documents. In addition, the process of updating the thesaurus can be expensive and complex [2, 29, 30].

In the methods based on partial documents, the query is originally sent to the search engine. With the results delivered, a group of documents is selected (the first results, the most relevant) and with these the query is reformulated (Rocchio's formula with $\gamma = 0$) and re-sent to the search engine. The results of the second (or expanded) search are those which are actually presented to the user [2, 29, 30].

Both expansion models have some problems: for example one assumes that the user is always going to mark documents as relevant or not and the other assumes that the first results from the original query are all relevant [2, 29, 30].

Carpineto *et al*. [1] presented the need for giving more importance to the query expansion process in web clustering engines. TopicSearch offers a query expansion process that gives greater importance to the semantic similarity between terms (words), but leaves option for users to feedback into the model which documents are relevant, and which are not.

TopicSearch starts the search process with a user query (based on key words.) This query is expanded explicitly with the help of the user, through an auto-complete option. This option is based on a dynamic, drop-down list of terms that are displayed in a similar way to those of Google.

The auto-complete option is generated based on the list of terms that have been relevant to the user in earlier queries.

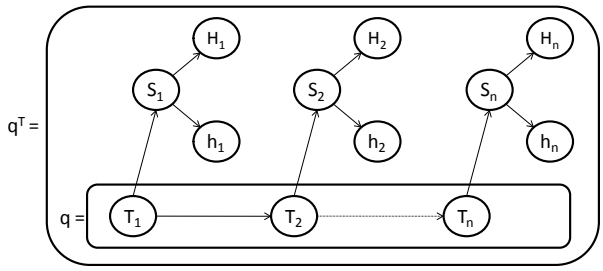


Fig 5. Expanded query for each term: synonyms (S), hypernyms (H) and hyponyms (h).

The process involves three steps described as follows: (1) pre-processing and semantic relationship, (2) terms listed in the profile and (3) using an external service.

1. **Pre-processing and semantic relationship**: It first takes the user query and removes special characters, converting each term to lower case and eliminating stop words. Then, it finds the most common synonyms (S, set of terms or synset in different languages that are used to represent the same concept), hypernyms (H, set of terms in the next level above in the hierarchy of the ontology, generalizations of the concept) and hyponyms (h, set of terms in the next level below in the hierarchy of the ontology, specializations of the concept) of the terms that the user has typed, based on WordNet (see Fig 5). In this research—as in WordNet—a synset is a set of terms to describe the same concept. The terms are searched in a general ontology, thesaurus or lexical database such as WordNet, based on partial matching on the new term in query. In summary, from the vector of original terms that make up the user query $q = \{T_1, T_2,\ldots,T_n\}$ each of the terms of the search vector is taken and concepts are formed so that each concept C is equal to (T, S, H, h). Each concept is equal to the term typed by the user and the semantically related terms that were retrieved from WordNet.
2. **Terms listed in the profile**: In the previous step we obtained a temporary extended search, but not all of these terms should be presented to the user in an auto-complete list. This is why it is necessary to define the order of presentation of the terms, so that to a greater degree they relate to the needs of the user. The aim of this step is just that, to set the order of presentation in relation to the user profile. To achieve this, the user's ***term co-occurrence matrix (matrix S)*** is consulted—the degree of correlation between each term and its related terms (S, H and h) for the current user (U), placing them in order of declining correlation (the most correlated to the least correlated).
   The first item in the drop-down list that is shown to the user is obtained by concatenating the original query without any processing and the term (S, H or h) with the highest degree of correlation. The second is obtained in a similar manner, the original query and the term with the second highest degree of correlation and so on up to a maximum number of terms to be presented on the interface (the model parameter is known as AutoComplete List Size or ACLS). In the event that the user has no information in the S matrix, the drop-down list gives priority to the terms written most recently, adding line by line first the synonyms, then the hyponyms and finally the hypernyms.
3. **Using an external service**: If in Step 1 (Pre-processing and semantic relation) of the model there is no information listed in WordNet, it goes to an external auto complete service, such as that of Google (based on the analysis of query logs of its users, with a focus on collaborative filtering). At this point, and as a future work, the model could incorporate an automatic approach of relevance feedback based on the Top-N documents retrieved (using an automatic relevance feedback based on partial documents).

The ***user profile*** is a fine-grained structure that relates for each user the number of documents reviewed by the user as relevant and irrelevant (N) (user feedback), the number of documents containing a term i (n_i), the number of relevant documents (R) and the number of relevant documents containing the term i (r_i). Moreover, for each document, the presence (1) or absence (0) of terms is recorded. From this information a matrix of term co-occurrence for each user is generated. This co-occurrence matrix, called S, is calculated as shown in Fig 6.

```
01  For each document d Є D do
02      For each term t_i Є d do
03          For each window w_z centered in term t_i do
04              For each term t_j Є w_z where t_j != t_i do
05                  S_i,j = C_i,j * IDF_i * IDF_j
06                  S_j,i = S_i,j
07              End-for
08          End-for
09      End-for
10  End-for
```

Fig 6. Algorithm for generating the term co-occurrence matrix (S).

The correlation factor $C_{i,j}$ is a normalized factor traditionally used in information retrieval [2].

It is defined by (2), where F_i is the frequency of term i, F_j is the frequency of term j and $F_{i,j}$ is the frequency of co-occurrence of terms i and j:

$$\mathrm{idf}_i = \frac{F_{i,j}}{F_i + F_j - F_{i,j}}. \quad (2)$$

The relative importance of a term in information retrieval is given by its IDF (inverse document frequency) value.

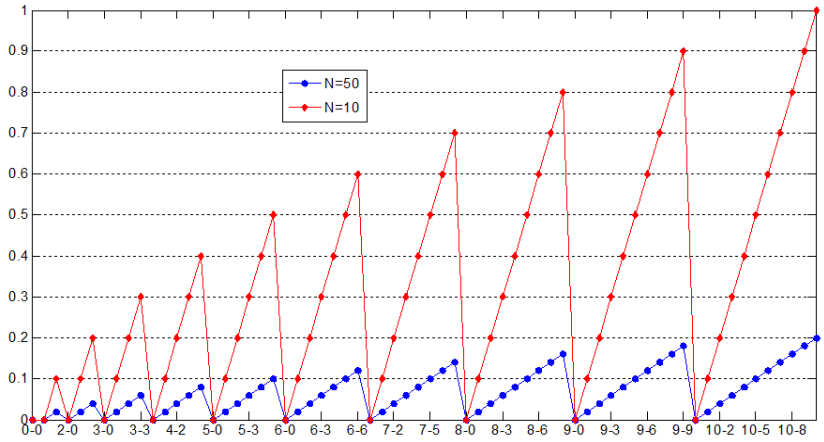


Fig 7. Graph of the IDF function used to calculate the S matrix. The function with N = 10 is shown by the marker in the form of squares and the function with N = 50 is shown by the ovals. The X axis shows different values of n_i and r_i, beginning with (0-0), passing for example through (6,3) and finishing at (10-6). The graph shows values of n_i between 0 and 10 and values of r_i between 0 and 6. For both functions the maximum is achieved when $n_i = r_i$, in this case (6,6) and the minimum when $r_i = 0$, regardless of the value of n_i.

To define this value, a range of formula can be used, e.g. the Robertson and Spärck-Jones (RSJ) proposal [31] one of the most cited in the literature. For our research, the RSJ formula was not suitable for construction of the S matrix. A new function based on formula (3) was defined instead. This IDF function (see Fig 7) defines the importance of a term in relation to the number of documents reviewed by the user (N), the number of documents relevant to the user (R), the number of documents in which the term i appears (n_i) and the number of relevant documents in which the term appears i (r_i):

$$\mathrm{idf_i} = \begin{Bmatrix} \frac{r_i}{N} \text{ si } n_i \leq R \\ \frac{r_i R}{n_i N} \text{ si } n_i > R \end{Bmatrix} \quad (3)$$

The IDF function proposed has a continuous range of values between zero and one [0,1]: zero when the term is not relevant at all and one when it is considered entirely relevant. The degree of relevance is in relation to the range of relevant documents, i.e. if there are many documents reviewed (as in the graph of N=50) and among these the term appears in only a few documents (e.g. 6) and all are relevant, the function has a value of 0.1, compared with a smaller number of documents (for example in the graph N=10), which a value of 0.6 would be obtained.

This IDF function was compared with the traditional Rocchio algorithm in three scenarios (without memory, with memory of the session and a long term memory) using the Communications of the ACM data set, and it obtains better results (see [32] for details).

The term co-occurrence matrix (S) of the user allows the ordered generation of the list of terms that complement those used by the user in the search expansion process as explained above in Step 2 "Terms listed in the profile."

B. Search Results Acquisition and Preprocessing

After performing the search expansion process, there follows the process of ***search results acquisition***. In this step, the query consists of ***key words*** typed by the user (those directly typed by the user and those selected from the auto complete list).

The Acquisition process conducts in parallel (different threads of execution) the collection of results in the various traditional search engines. In the initial model, Google, Yahoo! and Bing are used. As results are returned by the traditional search engines, ***pre-processing of entries*** is carried out. This process includes removing special characters and converting the text to lower case, among others; removing stop words; stemming; and filtering duplicate documents (documents reported concurrently by more than one traditional search engine). In addition, for each document the observed frequency of its terms is calculated and the document is marked as processed.

When all results have been acquired, documents are organized in a ***Terms by Documents Matrix*** using formula (4), which takes into account the relative importance (IDF value) of each term in the retrieved results from traditional search engines. This matrix is the original source of data for the clustering algorithm:

$$w_{i,j} = \left(\frac{F_{i,j}}{\max(F_i)}\right) Log\left(\frac{N}{n_j + 1}\right). \quad (4)$$

C. Cluster construction and labeling

Once the acquisition of search results has finished, the process of ***Cluster construction and Labeling*** follows. This process can be carried out using a variety of existing algorithms, among them Lingo [11], STC [8], SHOC [12],

Dynamic SVD [14]. But, because it should be improve the usefulness of the groupings and clarity of the labeling—as it was mentioned above—, a new algorithm called Agents-WDC was developed.

Agents-WDC is a description-centric algorithm [1] for web document clustering [1] based on Memetic Algorithms (MA) (MAs "are population-based meta-heuristic search methods inspired by both Darwinian principles of natural evolution and Dawkins notion of a meme as a unit of cultural evolution capable of individual learning" [33].) The memetic approach is used to combine a global/local strategy of search in the whole solution space. The *k*-means algorithm was used as a local strategy for improving agents in the MA. Arrival of foreign agents (random generation in evolution process) was used to promote diversity in the population and prevent the population from converging too quickly. The Bayesian Information Criterion (BIC) expressed by formula (5) was used as a fitness function [5, 34]. The evolution process is based on one agent at a time (not of populations) in a specific number of islands and the VSM is used for representing documents in the clustering stage, but in the labeling stage the frequent phrases model is used. Agents-WDC steps can be summarized in Fig 8:

$$BIC = n\,Ln\left(\frac{SSE}{n}\right) + k\,Ln(n),$$
$$SSE = \sum_{j=1}^{k}\sum_{i=1}^{n}\left(P_{i,j}\|x_i - c_j\|\right). \quad (5)$$

Here n is the total number of documents, k is the number of clusters and SSE is the sum of squared error from the similarities of the different clusters; $P_{i,j}$ is 1 if the document x_i belong to cluster j and 0 in other case, and c_j is the centroid of the cluster j.

Initialize algorithm parameters: In this research, the optimization problem lies in minimizing the BIC criteria (Fitness function). Agents-WDC needs the following parameters to be specified: Number of Islands (NI), Population Size (PS), Mutation Rate (MR), Minimum Bandwidth (MinB) and Maximum Bandwidth (MaxB) for mutation operation, Maximum Execution Time (MET) in milliseconds or Maximum Number of Iterations (MNI) to stop the algorithm execution.

Representation and Initialization: In Agents-WDC, each agent has a different number of clusters, a list of centroids, and the objective function value, based on BIC, which depends on the centroids' location in each agent and the number of centroids.

The cluster centers in the agent consist of D x K real numbers, where K is the number of clusters and D is the total number of terms (words in the vocabulary). For example, in three-dimensional data, the agent < [0.3|0.2|0.7], [0.4|0.5|0.1], [0.4|0.1|0.9], [0.0|0.8|0.7], 0.789> encodes centers of four (K value) clusters with a fitness value of 0.789.

1 **Initialize algorithm parameters**.
2 **Repeat** (inner sentences are executed in parallel—each population correspond to an island)
3 **Randomly initialize population** (*PS* agents), which encode cluster centers with different numbers of clusters.
4 **Run** the *k*-means routine for each agent in population.
5 **Calculate fitness** value for each agent in the initial population based on (5).
6 **Repeat**
7 Select pairing parents based on ***roulette wheel.***
8 **Generate one intermediate offspring** by applying genetic operators (***crossover and mutation***) of the paired parents.
9 **Run** the *k*-means routine for the offspring.
10 **Calculate fitness** value for the offspring based on (5).
11 **If** the offspring is invalid (i.e., number of clusters equal to one due to the death units problem in the clustering process) **then** it is replaced with a new agent randomly initialized (***arrival of foreign agents***)
12 **If** the fitness of the offspring is better than the worst agent on population, **then *replace the worst agent*** by the offspring.
13 **Until** MET or MNI is reached.
14 **Select best solution** (agent) in the population.
15 **Until** NI Island finished the process.
16 **Select best solution** (agent) from all islands.
17 **Assign labels to clusters**.
18 **Overlap clusters**.

***k*-means routine** (input: Initial set of centroids)
1 **Repeat**
2 **Re-compute membership** of each document according to current centroids and cosine similarity based on (6).
3 **Update centroids** based on new membership information
4 **Until** no changes in clusters
5 **Return** final set of centroids

Fig 8. Agents-WDC algorithm for web document clustering

Initially, each centroid corresponds to a different document randomly selected in the TDM matrix (Forgy strategy in the *k*-means algorithm [35]).

The initial number of clusters, K value, is randomly calculated from 2 to K_{max} (inclusive), where K is a natural number and K_{max} is the upper limit of the number of clusters and is taken to be $\sqrt{N/2}+1$ (where N is the number of documents in the TDM matrix), which is a rule of thumb used in the clustering literature by many researchers.

Roulette wheel: In step 7, one parent p1 is chosen from the population based on roulette wheel selection [36]. Also, its mate p_2 is chosen by the same process (preventing p_1 equal to p_2).

Crossover and mutation: At step 8 a traditional n-point crossover is used [37]. During crossover, the cluster centers are considered to be indivisible, so crossover points can only lie in between two cluster centers. In this process, just one offspring is generated. After crossover, a low probability of mutation (MR) is applied to the offspring. Uniform mutation between Minimum Bandwidth (MinB) and Maximum Bandwidth (MaxB) (as found in the Harmony Search Algorithm [23]) is applied to the chosen cluster dimension / attribute / term [x = x ± Random (MinB, MaxB)]. When mutation operation generates a value that reaches data boundaries, the mutation value is applied in the opposite way (mirror):

$$Sim(d_i, d_j) = \frac{\sum_{i=1}^{D}\left(W_{i,d_i}W_{i,d_j}\right)}{\sqrt{\sum_{i=1}^{D}\left(W_{i,d_i}\right)^2}\sqrt{\sum_{i=1}^{D}\left(W_{i,d_j}\right)^2}}. \quad (6)$$

Assign labels to clusters: This step corresponds to Step 2 "Frequent Phrase Extraction" in Lingo [11], but in Agents-WDC this method is used for each generated cluster in previous steps. By the above method, some changes were made to the original algorithm. This process works as shown in Fig 9.

Overlap clusters: Finally, each cluster includes documents that fall into other clusters too, if these documents are at a distance of less than or equal to the average distance of the cluster.

01 **Conversion of the representation**: Each document in the current cluster is converted from character-based to word-based representation.

02 **Document concatenation**: All documents in the current cluster are concatenated and a new document with the inverted version of the concatenated documents is created.

03 **Complete phrase discovery**: Right-complete phrases and left-complete phrases are discovered in the current cluster, then the right-complete phrases and left-complete phrases are alphabetically sorted, and then the left- and right-complete phrases are combined into a set of complete phrases.

04 **Final selection**: Terms and phrases whose frequency exceeds the Term Frequency Threshold are selected for the current cluster.

05 **Building of the "Others" label and cluster**: if some documents don't reach the Term Frequency Threshold, then they are sent to the other clusters.

06 **Cluster label induction**: In the current cluster, a Term-document matrix is built. Then, using cosine similarity, the best candidate terms or phrases for the cluster (which optimize SSE) are selected.

Fig 9. Frequent Phrase Algorithm for Labeling

IV. Experiments

Measuring the clustering performance of a document clustering algorithm is a complex issue. There are many different approaches and no standard methodology.

In general, there are two main categories of evaluation: internal quality (based on objective functions without reference to the output, this is the least used) and external quality (which evaluates the output clustering). External quality assessment can be further divided into gold-standard, task-oriented and user evaluation.

In gold-standard evaluation, results of the algorithm are compared with a pre-determined ideal clustering. In task-oriented evaluation, a performance analysis of a particular part of an algorithm is done.

External evaluation using gold-standard evaluation and user evaluation is the most common approach for evaluating the performance of web document clustering algorithms [38]. Thus, in this research this is the approach that has been used.

A. Datasets for Assessment

The Open Directory Project (or DMOZ) is commonly used as a neutral third party classifier, using human editors to classify manually and store thousands of websites. In this research a total of fifty datasets were randomly built. Datasets are available online at www.unicauca.edu.co/~ccobos/wdc/wdc.htm.

On average, datasets have 129.2 documents, 6 topics and 643.9 terms. Fig 10 shows different views of the datasets content and Table 1 shows detailed information from each dataset.

B. Parameters and Measures

Parameter values in Agents-WDC were equal for all datasets. NI equal to 2, PS equal to 5, MR equal to 0.5%, MinB equal to 0,0005, MaxB equal to 0.005, and MNI between 0 and 40 (depending on the experiment). Kmax value was equal to $\sqrt{N/2}+1$, where N is the number of documents.

There are many different methods proposed for measuring the quality of a generated clustering compared to an ideal clustering. Three of the best known are precision, recall and F-measure, commonly used in information retrieval and classification tasks [9].

In this research, the weighted Precision, weighted Recall and weighted F-measure (the harmonic means of precision and recall) measures are used to evaluate the quality of solution.

Given a collection of clusters, $\{C_1, C_2, \ldots C_k\}$, to evaluate its weighted Precision, weighted Recall and weighted F-measure with respect to a collection of ideal clusters $\{C_1^i, C_2^i, \ldots C_h^i\}$, these steps are followed: (a) find for each ideal cluster C_n^i a distinct cluster C_m that best approximates it in the collection being evaluated, and evaluate $P(C, C^i)$, $R(C, C^i)$, and $F(C, C^i)$ as defined by (7) and (8). (b) Calculate the weighted Precision (P), weighted Recall (R) and weighted F-measure (F) based on (9):

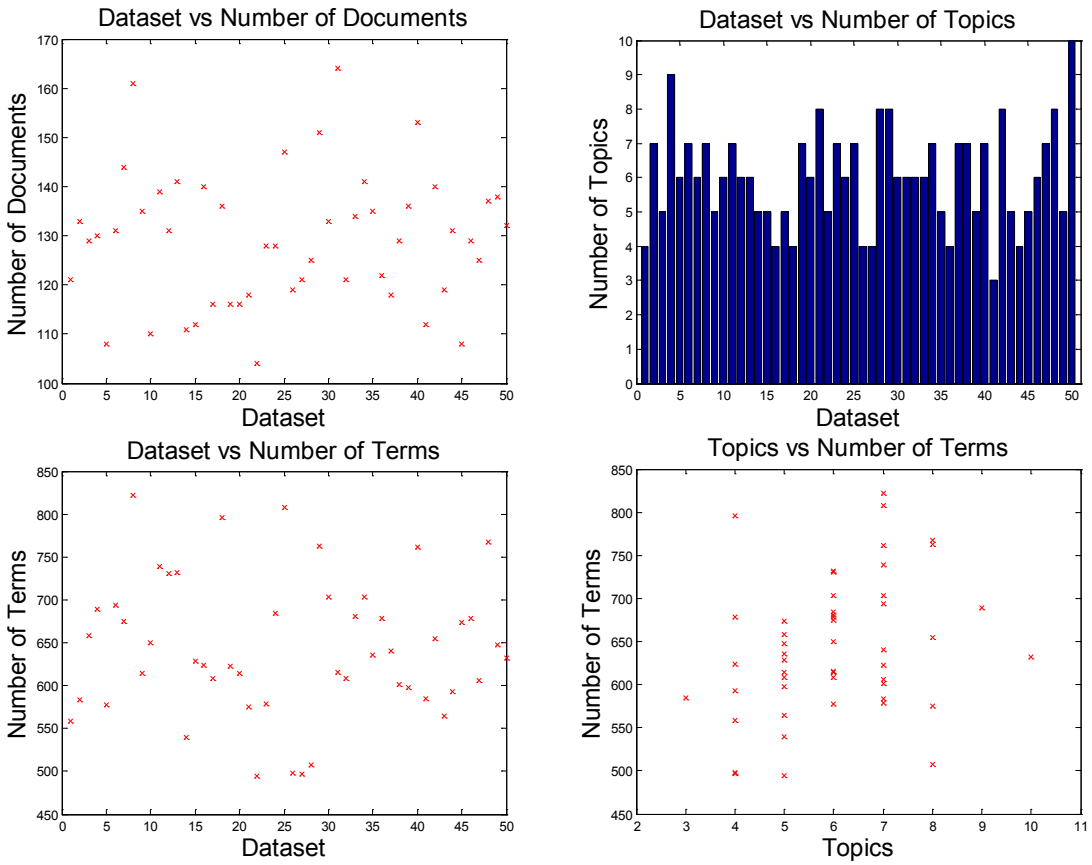


Fig 10. Datasets used for evaluation vs. documents, topics and number of terms

TABLE 1
GENERAL DESCRIPTION OF DATASETS USED FOR EVALUATION
(P STANDS FOR PRECISION AND R FOR RECALL USING SUPPORT VECTOR MACHINE)

Dataset	Documents	Topics	Attributes	P	R	Dataset	Documents	Topics	Attributes	P	R
1	121	4	559	96.865	96.694	26	119	4	498	89.428	88.235
2	133	7	583	90.183	89.474	27	121	4	497	90.062	88.43
3	129	5	658	94.358	93.023	28	125	8	507	90.757	89.6
4	130	9	689	87.601	83.846	29	151	8	763	90.694	89.404
5	108	6	578	84.453	81.481	30	133	6	703	87.352	85.714
6	131	7	694	94.252	93.893	31	164	6	616	96.048	95.732
7	144	6	675	93.984	93.056	32	121	6	609	91.982	90.909
8	161	7	822	92.501	91.304	33	134	6	681	87.574	88.06
9	135	5	614	92.131	91.111	34	141	7	703	91.357	89.362
10	110	6	650	92.629	89.091	35	135	5	636	97.9	97.778
11	139	7	739	94.427	93.525	36	122	4	679	96.006	95.902
12	131	6	731	90.037	89.313	37	118	7	641	85.089	79.661
13	141	6	732	67.174	66.429	38	129	7	601	88.219	86.822
14	111	5	540	95.118	93.694	39	136	5	598	95.18	94.853
15	112	5	629	94.943	94.643	40	153	7	761	89.747	89.542
16	140	4	624	93.786	93.571	41	112	3	585	92.695	91.071
17	116	5	609	92.92	92.241	42	140	8	655	87.875	87.143
18	136	4	796	94.443	94.118	43	119	5	564	94.624	94.118
19	116	7	623	94.31	93.966	44	131	4	593	94.445	93.13
20	116	6	614	88.61	84.483	45	108	5	674	80.426	80.556
21	118	8	575	83.678	78.814	46	129	6	679	91.334	89.147
22	104	5	495	93.477	93.269	47	125	7	606	87.263	86.4
23	128	7	579	92.07	90.625	48	137	8	767	91.094	90.511
24	128	6	684	86.416	85.156	49	138	5	648	89.577	88.406
25	147	7	808	88.835	87.075	50	132	10	632	88.509	76.515

TABLE 2
PRECISION, RECALL AND F-MEASURE IN K-MEANS, BISECTING K-MEANS, STC, LINGO AND AGENTS-WDC

Algorithm	Time	Precision	Recall	F-Measure	Number of Fitness evaluations	Ideal number of topics (average)	Obtained number of clusters (average)
k-means	1.0 ± 0.262	75.08 ± 13.39	55.78	63.59 ± 9.06	–	6.02 ± 1.464	**8.72** ± 1.386
Bisecting k-means	0.53 ± 0.009	70.49 ± 10.22	40.08	49.21 ± 4.75	–	6.02 ± 1.464	11.94 ± 1.23
STC	**0.02** ± 0.003	77.68 ± 9.59	45.18	56.71 ± 8.857	–	6.02 ± 1.464	15.97 ± 0.24
Lingo	0.68 ± 0.177	**80.34** ± 4.419	29.53	43.00 ± 4.761	–	6.02 ± 1.464	34.46 ± 1.784
Agents-WDC	1.55 ± 0.38	79.72 ± 13.95	**59.29**	**67.43** ± 9.979	12 = 7 + PS	6.02 ± 1.464	**8.96** ± 2.185
Agents-WDC	1.78 ± 0.45	**81.46** ± 11.15	**61.17**	**69.29** ± 9.464	14 = 9 + PS	6.02 ± 1.464	**8.80** ± 2.000
Agents-WDC	1.85 ± 0.56	**82.63** ± 9.311	**61.07**	**69.72** ± 8.886	16 = 11 + PS	6.02 ± 1.464	**8.90** ± 1.632
Agents-WDC	5.57 2.073	**88.85** ± 8.731	63.95	73.53± 7.698	45 = 40 + PS	6.02 ± 1.464	9.44 ± 2.21

$$P(C, C^i) = \frac{|C \cap C^i|}{|C|}, \qquad R(C, C^i) = \frac{|C \cap C^i|}{|C^i|}; \tag{7}$$

$$F(C, C^i) = \frac{2P(C, C^i)R(C, C^i)}{P(C, C^i) + R(C, C^i)}; \tag{8}$$

$$P = \frac{1}{T}\sum_{j=1}^{h} |C_j^i| P(C_m, C_j^i); \quad R = \frac{1}{T}\sum_{j=1}^{h} |C_j^i| R(C_m, C_j^i); \quad F = \frac{2PR}{P+R}; \quad T = \sum_{j=1}^{h} |C_j^i|. \tag{9}$$

Here, C is a cluster of documents and cluster C^i is an ideal cluster of documents.

C. Results with Datasets

The algorithm was compared with a version of *k*-means (it executes several solutions of *k*-means and selects the best solutions based on BIC criteria), Bisecting K-means, STC and Lingo (last three algorithms are provided by the free open source Carrot2 Document Clustering at www.carrot2.org and were used with its default values).

Using an on-line scenario (with 2.0 seconds as a maximum time of execution and without query expansion support), algorithms was executed 30 times over each dataset and the averages were calculated to show them as results. These promising results are shown in Table 2. High values of Precision, Recall, and F-Measure are desirable.

In Table 2, the best results are presented when Agents-WDC is executed 11 iterations (approximately 1.85 seconds in a desktop computer with windows vista of 32 bits, 4 GB of RAM and Intel Pentium Dual CPU at 2.16 GHz. Time has a linear correlation with the iterations, equivalent in this setting to $Time = 0{,}1116 * iterations + 0{,}1395$). Also, Agents-WDC reports very competitive results from 7 iterations (1.55 seconds of execution time). Recall and F-Measure is always better with Agents-WDC algorithm.

Lingo and STC reports very good precisions with a low time of execution, but Recall and F-Measure are too far from Agents-WDC results. The recall distance between Agents-WDC and STC is around 15% on Recall and around of 30% against Lingo.

Lingo reported the lowest rate of dispersion in precision, while Agents-WDC reported in 1.85 seconds more than twice that value. Although in Agents-WDC the precision variation decreases over the iterations, this is an issue that should be studied further.

Another important difference between Agents-WDC, Bisecting *k*-means, STC and Lingo is the number of clusters. Agents-WDC always defines a better value of K (number of clusters). In Lingo and STC with an average of 28 and 9 extra clusters respectively, results of precision can be biased. Therefore, the research group plans to use another kind of metrics to compare results of STC and Lingo, for example BCubed Precision and BCubed Recall [39].

In Fig 11, curves of precision, recall and f-measure through different number of generations are shown. All values increasing with the number of generations. Therefore, when users can wait for results, Agents-WDC organized in better way clusters of documents and proved the best option. BIC with cosine similarity is a good option for web document clustering because precision and recall both increase when Agents-WDC optimizes BIC ($Precision = 6.2759 * \ln(Generations) + 65.015$ with $R^2 = 95.43\%$), but in some generations (e.g. 4 to 6 generations) this positive relation fails. Thus, the research group plans to define a better fitness function for evolutionary algorithms in web document clustering based on genetic programming.

Further analysis showed that in general Agents-WDC increases the quality of cluster (based on precision and recall) when it uses more generations regardless of the number of documents, number of topics, or number of attributes in the dataset. Some datasets, though, do not comply with this rule.

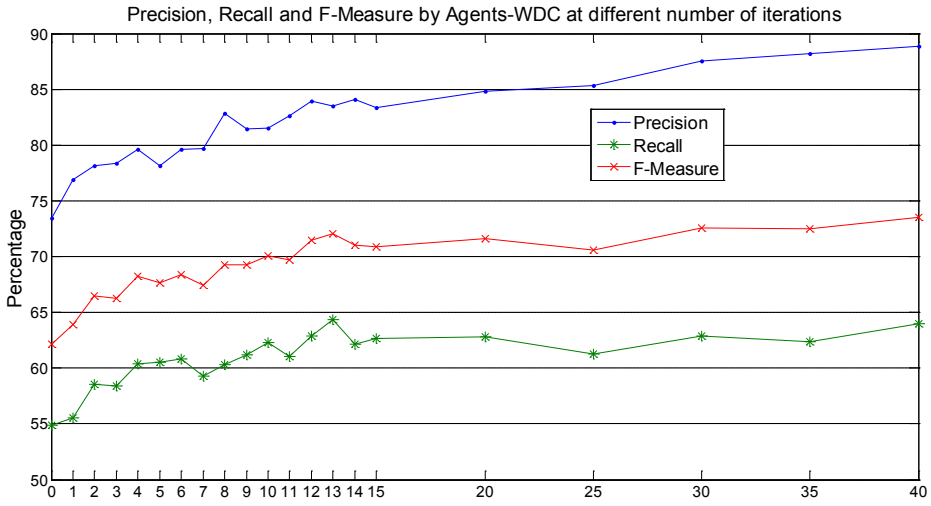


Fig 11. Precision, Recall and F-Measure for Agents-WDC through different periods of time

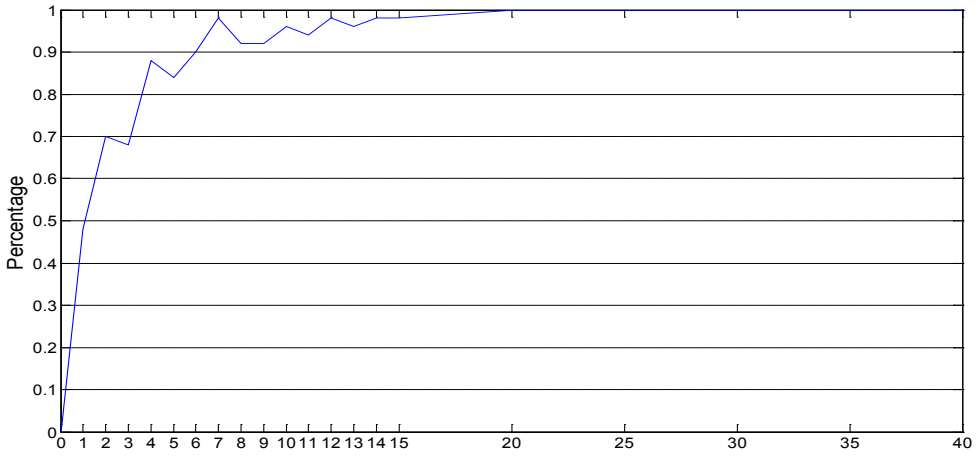


Fig 12. Effectiveness of new solutions generated at different number of iterations

This situation reinforces the need for defining a new objective function as was mentioned above, but also implies the need to analyze the impact of noise on the *k*-means algorithm, and the need to use other local optimization algorithms.

New solutions (agents) generated (using the selection, crossover, mutation and replace operators from Agents-WDC) increase its effectively over iterations. Fig 12 shows a 48% of effectively of the new solution in the first iteration, i.e. new solution is better than other solutions in population. Next, the effectiveness increases to 70% in second iteration, then it increases to 90% in sixth iteration, and finally is around 100% over the twelfth iteration.

Agents-WDC also provided better cluster labels than Bisecting k-means, STC and Lingo. For example, Table 3 shows labels generated by all algorithms for dataset1 with 4 non-overlapping topics. Note that the clusters generated and the order in which they are generated are different between the algorithms.

It is clear that Agents-WDC and STC generate best labels, while Lingo generates longer phrases and Bisecting *k*-means generates a long list of terms for each cluster. Lingo's long labels, while expressive, can be too specific and not always meaningful (e.g. "Shows how to Use"). Lingo only classified 74 out of 121 documents, much fewer than Agents-WDC, STC and Bisecting *k*-means. Agents-WDC favors labels with specific meanings closely related to documents in the cluster.

D. User Evaluation

Based on [40], a user-based evaluation method was used to assess the clustering results produced by the model (on-line scenario with 40 users) when data sources are Google, Yahoo! and Bing. For a set of groups created in response to a single query, the user answered whether or not:

TABLE 3
LABELS GENERATED BY BISECTING K-MEANS, STC, LINGO AND AGENTS-WDC OVER DATASET 1

Real category in DMOZ	Bisecting k-means	STC	Lingo	Agents-WDC (1.2seg)
Top/Business/Textiles_and_Nonwovens/Fibers/Wholesale_and_Distribution (16), Top/Health/Conditions_and_Diseases/Cancer/Breast (22), Top/Computers/Programming/Languages/Regular_Expressions (34), Top/Shopping/Food/Sweeteners/Honey (49)	Cancer, Breast, Male(13), Nonwovens, Polyester, English(13), Regular, Allows, Perform(12), Regular, Show, Windows(12), Beeswax, Candles, Overview(10), Diagnosis, Symptoms, Prevention(10), Raw, Business, Unprocessed(10), New, Royal, Zealand(9), Examines, Mitchell, Scott(6), Regex, JavaScript, Tester(6), Beeswax, Sioux, Clinically-oriented(5), Forest, Ordering, Trees(5), National, Centre, Ovarian(4), Wax, Cooperative, Indiana(4), Links, Representatives, Sites(2)	Regular Expressions(31), Treatment, Diagnosis, Breast Cancer(17), Yarns, Natural and Man-made Fibers(10), Honey(44), Nonwovens, Staple Fiber, Polyester(12), Regular Expression(8), Polyester Staple Fiber, Nonwovens Production, Yarn Spinning(5), Information(19), Natural(17), Beeswax(15), Fibers(13), Products(13), Raw(11), Raw Honey(6), Offers(10), Other Topics(6)	Breast Cancer(9), Beeswax Candles(6), Overview(5), Produces Raw(5), Business(4), English and German(4), Sold(4), Windows(4), Introduction to Regular(3), New Zealand(3), North Dakota(3), Organic(3), Parts(3), Polyester and Polyamide Filaments(3), Power of Regular(3), Shows how to Use(3), Sizes(3), Commercial(2), Farmer Market Schedule(2), Flower(2), Gift Baskets(2), Help(2), International Merchants(2), Jar(2), Male Breast Cancer(2), National Breast and Ovarian Cancer Centre(2), Regex(2), Risk Factors(2), Scott Mitchell Examines(2), Search(2), Short Video and Details of Symptoms(2), Source Code(2), Visual(2), Other Topics(47)	Information Overview (14), Produces Raw Honey (31), Regular Expressions This Article (31), Bee Pollen (12), Details Of Symptoms Causes Diagnosis Treatment Prevention (16), Natural and Man Made Fibers Yarns (17)

– (Q1: concise and meaningful cluster labels) the cluster label for each group is in general representative of the cluster content (much: R3, little: R2, or nothing: R1). Concise and meaningful cluster labels help users to decide which groups should review.
– (Q2: Usefulness of clusters) the cluster description and content is useful (R3), moderately useful (R2) or useless (R1). Usefulness of clusters is a general assessment (quality of labels and content) only for those groups that users decided relevant to query.

Then, for each document in each cluster, the user answered whether or not:

– (Q3) the document (snippet) matches with the cluster (very well matching: R3, moderately matching: R2, or not-matching: R1), A very well matching document would contain exactly the information suggested by the cluster label. A moderately matching document would still be somehow related to the group's topic. A non-matching document, even though it might contain several words from the group's label, would be completely irrelevant to its cluster.
– (Q4) the document relevance (order or rank) in the cluster was adequate (adequate: R3, moderately suitable: R2, or inadequate: R1). The most relevant documents should appear in the top of the list of group outcomes. This makes the user spend less time to solve their information needs.

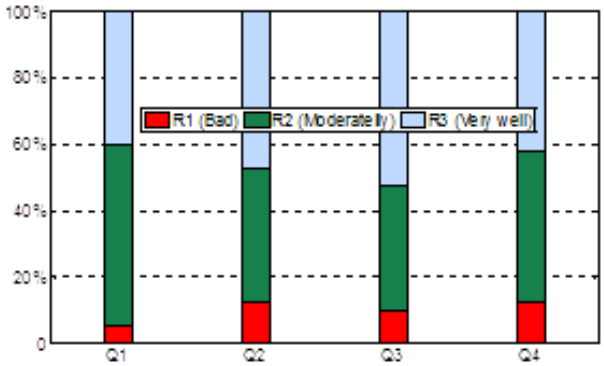


Fig 13. General results for the four questions (TDM-BIC-FPH)

General results of Agents-WDC are shown in Fig 13. Most user responses (90%) are R3 or R2. Therefore, results are very promising and it is necessary to do a set of very controlled experiments with more users, in order to generalize results. In summary, most of the users find that: cluster labels are representative, clusters are useful and documents are well organized in each cluster.

V. CONCLUSIONS AND FUTURE WORK

The proposed personalized web document clustering model allows users to define better queries, based on WordNet (semantic similarity of terms) and a user profile (order based on the new IDF function and correlation of terms). In the description of the model, the query expansion process,

acquisition of search results from traditional web search engines, the preprocessing of input data, a web clustering algorithm based on a memetic approach, and a proposal for cluster labeling are all detailed. All of these processes were easily modeled with agents.

The Clustering and Labeling Agent uses the Agents-WDC algorithm. This algorithm is a web document clustering algorithm based on Memetic Algorithms (global/local search strategy) and the *k*-means algorithm (local solution improvement strategy) with the capacity of automatically defining the number of clusters. Agents-WDC shows promising experimental results in standard datasets. Comparison with *k*-means, Bisecting *k*-means, STC and Lingo show Agents-WDC is a better algorithm for web document clustering in both on-line and off-line scenarios.

The Bayesian Information Criterion (BIC) with cosine similarity is a good option for web document clustering because precision and recall both increase when Agents-WDC algorithm evolve, but in some cases this positive relation fail.

New solutions (agents) generated in Agents-WDC algorithm based on roulette selection, a traditional n-point crossover, uniform mutation, and local improvement with k-means show a high rate of success (around 90% from fifth iteration) in the evolutionary process.

Agents-WDC uses two document representations models, initially it uses vector space model in the clustering process and then it uses full text for the labels creation process. This combination improves quality of cluster labels and the general quality of the clustering process.

There follow some suggestions for future work: applying the model to several datasets (other datasets based on DMOZ, Google results, Yahoo! results, among others) in on-line and off-line scenarios; evaluate Agents-WDC using BCubed Precision and BCubed Recall and compare results with Lingo, STC and other web document clustering algorithms. Define a new fitness function for evolutionary algorithms in web document clustering, using for example genetic programming.

Evaluate TopicSearch alongside other traditional and evolutionary algorithms for web document clustering; making use of WordNet to work with concepts (Concept-Document Matrix) instead of terms (Term-Document Matrix) and comparing the results; evaluating the entire model with a lot of users in different contexts; and evaluating the impact of query expansion over the time.

ACKNOWLEDGMENT

The work in this paper was supported by a Research Grant from the University of Cauca under Project VRI-2560 and the National University of Colombia.

REFERENCES

[1] C. Carpineto, *et al.*, "A survey of Web clustering engines," *ACM Comput. Surv.,* vol. 41, pp. 1-38, 2009.

[2] R. Baeza-Yates, A. and B. Ribeiro-Neto, *Modern Information Retrieval*: Addison-Wesley Longman Publishing Co., Inc., 1999.

[3] C. Carpineto, *et al.*, "Evaluating subtopic retrieval methods: Clustering versus diversification of search results," *Information Processing & Management,* vol. 48, pp. 358-373, 2012.

[4] K. Hammouda, "Web Mining: Clustering Web Documents A Preliminary Review," ed, 2001, pp. 1-13.

[5] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*: Prentice-Hall, Inc., 1988.

[6] M. Steinbach, *et al.*, "A comparison of document clustering techniques," in *KDD workshop on text mining*, Boston, MA, USA., 2000, pp. 1-20.

[7] Y. Li, *et al.*, "Text document clustering based on frequent word meaning sequences," *Data & Knowledge Engineering,* vol. 64, pp. 381-404, 2008.

[8] Z. Oren and E. Oren, "Web document clustering: a feasibility demonstration," presented at the Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval, Melbourne, Australia, 1998.

[9] M. Mahdavi and H. Abolhassani, "Harmony K-means algorithm for document clustering," *Data Mining and Knowledge Discovery,* vol. 18, pp. 370-391, 2009.

[10] P. Berkhin, *et al.*, "A Survey of Clustering Data Mining Techniques," in *Grouping Multidimensional Data*, ed: Springer-Verlag, 2006, pp. 25-71.

[11] S. Osiński and D. Weiss, "A concept-driven algorithm for clustering search results," *Intelligent Systems, IEEE,* vol. 20, pp. 48-54, 2005.

[12] D. Zhang and Y. Dong, "Semantic, Hierarchical, Online Clustering of Web Search Results," in *Advanced Web Technologies and Applications*, ed, 2004, pp. 69-78.

[13] B. Fung, *et al.*, "Hierarchical document clustering using frequent itemsets," in *Proceedings of the SIAM International Conference on Data Mining*, 2003, pp. 59-70.

[14] G. Mecca, *et al.*, "A new algorithm for clustering search results," *Data & Knowledge Engineering,* vol. 62, pp. 504-522, 2007.

[15] F. Beil, *et al.*, "Frequent term-based text clustering," in *KDD '02: International conference on Knowledge discovery and data mining (ACM SIGKDD)*, Edmonton, Alberta, Canada, 2002, pp. 436-442.

[16] L. Jing, "Survey of Text Clustering," ed, 2008.

[17] W. Song, *et al.*, "Genetic algorithm for text clustering using ontology and evaluating the validity of various semantic similarity measures," *Expert Systems with Applications,* vol. 36, pp. 9095-9104, 2009.

[18] L. Xiang-Wei, *et al.*, "The research of text clustering algorithms based on frequent term sets," in *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 2005, pp. 2352-2356 Vol. 4.

[19] A. K. Jain, *et al.*, "Data clustering: a review," *ACM Comput. Surv.,* vol. 31, pp. 264-323, 1999.

[20] S. Osiński and D. Weiss, "Carrot 2: Design of a Flexible and Efficient Web Information Retrieval Framework," in *Advances in Web Intelligence*, ed, 2005, pp. 439-444.

[21] X. Wei, *et al.*, "Document clustering based on non-negative matrix factorization," presented at the Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, Toronto, Canada, 2003.

[22] Z. Zhong-Yuan and J. Zhang, "Survey on the Variations and Applications of Nonnegative Matrix Factorization," in *ISORA'10: The Ninth International Symposium on Operations Research and Its Applications*, Chengdu-Jiuzhaigou, China, 2010, pp. 317–323.

[23] Z. Geem, *et al.*, "A New Heuristic Optimization Algorithm: Harmony Search," *Simulation,* vol. 76, pp. 60-68, 2001.

[24] R. Forsati, *et al.*, "Hybridization of K-Means and Harmony Search Methods for Web Page Clustering," in *WI-IAT '08: IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 2008, pp. 329-335.

[25] M. Mahdavi, *et al.*, "Novel meta-heuristic algorithms for clustering web documents," *Applied Mathematics and Computation,* vol. 201, pp. 441-451, 2008.

[26] W. Song and S. Park, "Genetic Algorithm-Based Text Clustering Technique," in *Advances in Natural Computation*, ed, 2006, pp. 779-782.

[27] C. Cobos, *et al.*, "Web document clustering based on Global-Best Harmony Search, K-means, Frequent Term Sets and Bayesian

Information Criterion," in *2010 IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, Spain, 2010, pp. 4637-4644.

[28] C. Cobos*, et al.*, "Web Document Clustering based on a New Niching Memetic Algorithm, Term-Document Matrix and Bayesian Information Criterion," in *2010 IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, Spain, 2010, pp. 4629-4636.

[29] C. Manning*, et al.* (2008). *Introduction to Information Retrieval*. Available: http://www-csli.stanford.edu/~hinrich/information-retrieval-book.html

[30] L. Yongli*, et al.*, "A Query Expansion Algorithm Based on Phrases Semantic Similarity," presented at the Proceedings of the 2008 International Symposiums on Information Processing, 2008.

[31] S. E. Robertson and K. Sparck-Jones, "Relevance weighting of search terms," in *Document retrieval systems*, ed: Taylor Graham Publishing, 1988, pp. 143-160.

[32] C. Cobos*, et al.*, "Algoritmos de Expansión de Consulta basados en una Nueva Función Discreta de Relevancia," *Revista UIS Ingenierías,* vol. 10, pp. 9-22, Junio 2011.

[33] Q. H. Nguyen*, et al.*, "A study on the design issues of Memetic Algorithm," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 2007, pp. 2390-2397.

[34] A. Webb, *Statistical Pattern Recognition, 2nd Edition*: {John Wiley & Sons}, 2002.

[35] S. J. Redmond and C. Heneghan, "A method for initialising the K-means clustering algorithm using kd-trees," *Pattern Recognition Letters,* vol. 28, pp. 965-973, 2007.

[36] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: The MIT Press, 1999.

[37] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*: Addison-Wesley Longman Publishing Co., Inc., 1989.

[38] T. Matsumoto and E. Hung, "Fuzzy clustering and relevance ranking of web search results with differentiating cluster label generation," in *Fuzzy Systems (FUZZ), 2010 IEEE International Conference on*, 2010, pp. 1-8.

[39] E. Amigó*, et al.*, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Inf. Retr.,* vol. 12, pp. 461-486, 2009.

[40] S. Osiński, "An Algorithm for clustering of web search results," Master, Poznań University of Technology, Poland, 2003.

Recommending Machine Translation Output to Translators by Estimating Translation Effort: A Case Study

Prashant Mathur, Nick Ruiz, and Marcello Federico

***Abstract*—In this paper we use the statistics provided by a field experiment to explore the utility of supplying machine translation suggestions in a computer-assisted translation (CAT) environment. Regression models are trained for each user in order to estimate the time to edit (TTE) for the current translation segment. We use a combination of features from the current segment and aggregated features from formerly translated segments selected with content-based filtering approaches commonly used in recommendation systems. We present and evaluate decision function heuristics to determine if machine translation output will be useful for the translator in the given segment. We find that our regression models do a reasonable job for some users in predicting TTE given only a small number of training examples; although noise in the actual TTE for seemingly similar segments yields large error margins. We propose to include the estimation of TTE in CAT recommendation systems as a well-correlated metric for translation quality.**

***Index Terms*—Machine translation, computer-assisted translation, quality estimation, recommender systems.**

I. Introduction

RECENT advances in Statistical Machine Translation (SMT) have been due to the large availability of parallel corpora. A natural goal is to apply machine translation to the computer-assisted translation (CAT) domain to increase translator productivity. CAT tools typically consist of a Translation Memory (TM) which stores segments that have been translated before by the users. When a translator is translating a new sentence, the sentence is first looked up in the TM and if there is a fuzzy match (a partial match score above a given threshold) the CAT tool suggests the translation in the TM to the translator. In the common scenario where the TM does not provide any suggestions to the translator, the translator must translate the segment from scratch.

The aim of CAT tools is to improve the productivity of translator and to ensure consistency. Integrating SMT system in a CAT tool has been shown to speed up the translation process [1]. Machine translation output provides excellent coverage that can help overcome sparsity in the TM; however, MT output can also add additional noise on the screen that can distract the translator from her task. Ideally, the goal for SMT in the CAT scenario is to provide machine translation suggestions only when their quality in order to guarantee an increase in the user's productivity for a given segment.

The aim of the EU-funded MateCat[1] project is to increase translator productivity by providing self-tuning, user-adaptive and informative MT in the CAT scenario. In this paper we propose the integration of a recommendation system framework using content-based filtering to suggest when MT output should be presented on the translator's screen, based on the difficulty of the current segment and his previous behavior on similar segments. We do so by estimating the amount of effort, as a function of time, to translate the current segment, given a MT suggestion. Given the estimated time to translate a sentence, we attempt to define a decision function to determine if the MT output will increase the productivity of the translator.

This paper is organized as follows. In Section II we describe related work in the field of quality estimation. Section III outlines the conditions of a preliminary field test conducted at the beginning of the MateCat project. Section IV describes the methodology used to estimate the time to edit a sentence (TTE). Our experiment and results using the field test dataset are described in Sections V and VI, respectively. Finally, we discuss the results of the experiment and provide suggestions for future work.

II. Previous Work

Automatic Quality Estimation (QE) for Machine Translation (MT) seeks to predict the usefulness of MT outputs without observing reference translations. QE can be cast as a machine learning problem whose goal is to predict a quality score based on one or more metrics, including human post-editing effort.

A baseline regression system was constructed in [2] for the WMT 2012 Quality Estimation shared task that uses MT and surface-level features derived from a training set to predict a user's perceived level of post-editing effort for newly translated sentences. Levenshtein features are used in [3] which measure the distance between the current sentence with each of the closest entry in the training corpus. Low edit

Manuscript received on December 7, 2012; accepted for publication on January 11, 2013.

Prashant Mathur and Nick Ruiz are with University of Trento and FBK, Italy.

Marcello Federico is with FBK, Italy.

[1]http://www.matecat.com

distances imply that the current sentence is close to the training set and thus its quality is expected to be high.

Sentences in a development set are ranked by their sentence-level BLEU scores [4] and divided the development set into quartiles in [5]. Inspired by *TrustRank* [6], additional regression features are added to measure the distance between the current sentence and the high or low quality quartile sets in the development data. The distance is measured via n-gram matches through a modified BLEU score and is evaluated in both the source and the target language directions.

The prediction of word- and sentence-level MT errors are treated as a sequence labeling task in [7], using alignment, agreement, and POS-based features. Each token is labeled by the type of error occurring (e.g. insertion, deletion, substitution, or shift).

Quality estimation for MT is treated as a binary classification problem for computer-assisted translation in [8]. They predict the usefulness of MT output over translation memory recommendations in terms of the number of words edited to match a reference translation. This metric is known as the translation error rate (TER) [9]. The system also provides confidence scores based on posterior classification probabilities. A MT output is recommended if its corresponding TER score is lower than that of a TM suggestion.

Due to the nature of the field test, we treat the SMT system as a black box and cannot use SMT-based features in our model. However, we consider the reverse translation fuzzy match score as a feature in our model.

III. Field Test

The EU-funded project MateCat was launched in early 2012 with the aim to integrate Statistical Machine Translation systems such as Moses [10] with a state-of-the-art CAT tool to improve translator productivity. The goal of MateCat is to seamlessly integrate a MT engine as a back-end process inside the CAT tool. Translators will receive translations either from TM matches or from the MT engine. One of the aims of the project is to recommend MT outputs if the machine translation requires less post-editing than the translation memory.

A feasibility study was conducted in [1] as a field test which integrated a production-quality MT Engine (namely, Google Translate[2]) in SDL Trados Studio[3]. Twelve professional translators worked on real translation projects covering the information technology and legal domains. Documents were translated both from English to German and English to Italian. The experiment was held in two parts. In the first part, a baseline was established by providing translators with only TM matches for suggestions. We refer to this baseline as TM experiments. In the second part MT outputs were viable alongside the TM matches. We refer to these as MT experiments. To measure the productivity of translators two indicators were used: the *post editing speed* which is the average number of words processed by translator in one hour, and the *post editing effort* which is the average percentage of word changes applied by the translator on each suggestion. The results from the experiment show that providing MT recommendations significantly increased productivity across all users.

[2]http://translate.google.com

[3]http://www.trados.com/en/

IV. Methodology

Modern recommendation systems typically use two mechanisms: content-based and collaborative filtering. In content-based filtering an item is recommended to a user based on its similarity between items she previously observed. The user's judgments on previously seen items are stored in a user profile, which may also contain additional user information. An item profile is constructed based on a set of characteristics or attributes describing it. The user's profile is combined with item profiles to find new items that a user may prefer. In a CAT scenario, "item profiles" of previously translated segments can be aggregated and used to predict a user's judgment on the quality of a machine translation in a future segment.

In collaborative filtering, a given user is compared against a collection of other users with similar profiles to provide recommendations for new items, even if the items recommended are dissimilar to those preferred by the user in the past. Such filtering increases the pool of items that can be recommended to a user. In a CAT scenario, translation recommendations could be provided based on the previous translations of other users. Unfortunately, such an approach is not useful in a professional translation scenario, where translators must maintain consistency within their own projects. Additionally, a careful look at the data provided in the MateCat field test shows that the translators behave quite differently. However, such an approach might be useful for crowd-sourced or community-based translations with a large number of amateur translators.

CAT systems aim at increasing the productivity of the translators by providing translation suggestions, usually in the form of a TM. It is a necessity that the translations recommended from the MT system are good. If the cost of post-editing a translation is higher than translating the segment from scratch then it decreases the productivity of the user. An ideal recommendation system suggests a translation only when the cost of post-editing is low. Thus, we only use content-based filtering, treating each translation segment as an item to be compared against segments previously translated by the same user. We combine features drawn from the item profile of the current segment along with a collection of item profiles of similar segments to predict the time required to translate the segment. We call this the *time to edit* (TTE). TTE is one of several indicators for translation effort. If additionally providing MT outputs does not significantly improve the translation effort over scenarios where only the TM is available, then it is not useful to recommend the MT output to the translator.

In [8], most of the features for translation recommendation come directly from the SMT models (e.g. phrase translation scores, language model scores). These features are combined with system independent features such as language model perplexity scores on target side, fuzzy match scores and lexical translation scores from a word alignment model [11]. However, in this work we are restricted to the features used in the field test. The underlying MT engine and language models used by Google Translate are unavailable for analysis. Instead, we used features in the MateCat field test, such as *word count*, *TTE* on a sentence, *match percent* (percent match between a TM and the current segment), and the *after match* score (how close the suggestion and translation are).

However, there were several limitations in the original field test. The translators worked remotely on a client system at their respective locations and connected to a centralized server which made it hard to capture the user/translator focused features such as keystrokes, the actual time to translate a sentence, and whether the translator actually used the MT output or translated directly from scratch. In the absence of such features we use a number of aggregated features obtained from the similar segments to the current one being translated (see section V-B). We borrow the idea of a pseudo fuzzy match score from [8]. However, instead of Levenshtein distance we use TER. There are two kinds of features, one representing the statistics of current segment (current features) and the one representing the statistics of the similar segments (aggregated features). Both sets of features are taken from the TM and MT outputs.

Table I provides a list of the features considered in this paper. Features such as the source word count (WC_{SR}) and the TER of the reverse-translated MT output against the source text (TER_{SR}) are computed on the current segment. Several other features are computed on the collection of similar TM and MT segments extracted by means of content-based filtering. Aggregated features provide summary statistics on the TTE and time for word (TFW, computed as TTE divided by segment word count), as well as an estimate on the translation error rate (TER_{TG}) and word counts on the MT and post-edited outputs (WC_{MT} and WC_{PE}, respectively). Each summary statistic yields three features, one for the respective TM and MT aggregates and an additional feature for their difference.

TABLE I
FEATURES USED IN THIS EXPERIMENT. CERTAIN FEATURES ARE DERIVED FROM THE CURRENT SEGMENT, WHILE OTHERS ARE COMPUTED AS AN AVERAGE OVER SIMILAR SEGMENTS. WORD COUNT (WC) FOR SOURCE (SR), MT, AND POST-EDITED (PE) SEGMENTS; TTE AND TFW, AND TER ON TARGET (TG) AND SOURCE (SR) SEGMENTS.

Segment	Current	Similar
WC_{SR}	Y	Y
WC_{MT}	N	Y
WC_{PE}	N	Y
TFW	N	Y
TTE	N	Y
TER_{TG}	N	Y
TER_{SR}	Y	N

TABLE II
SEGMENTS PER USER IN THE EN>DE IT DOMAIN CAT SCENARIO. THE DATASET CONSISTS OF 3670 SEGMENTS.

User	TM Segments	MT Segments
User 1	487	486
User 2	498	481
User 3	332	486
User 4	490	410

V. EXPERIMENT

We perform our experiment on the English to German translation of segments in the information technology domain.

A. Preprocessing

As in [1], segments where the processing time per word was less than 0.5 seconds or greater than 30 seconds were removed as outliers. Perfect (100%) TM matches were removed. After filtering, the data set contains a total of 3670 segments, subdivided among four users, as listed in Table II.

Data is split as follows: the first 10% of each split is reserved as a burn-in to accumulate user statistics for extracting similar segments. The remainder of the data is split into 11 folds in a round-robin fashion to minimize the effects of immediately translated segments (i.e., $seg_1 \rightarrow fold_1, seg_2 \rightarrow fold_2, ..., seg_{12} \rightarrow fold_1$, etc.). Ten folds are used for cross-validation purposes to evaluate the utility of the regression model. The folds are later combined and used to evaluate the final held-out set to provide a recommendation to the user.

B. Extracting similar segments

In order to gather statistics on features not observable in the current segment (see Table I), we compute aggregated features from the statistics of similar segments, both in our TM and MT experiments. We rely on the popular cosine similarity metric using unigram features from the source text for identifying similar segments. Given source language bag-of-word features for segments A and B, cosine similarity is defined as:

$$\text{similarity} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}} \quad (1)$$

In order to simulate the computer-assisted translation scenario, we only compute similarity scores on segments that would appear in a user's "translation cache" – e.g., segments that were already translated by the user at a given point in the TM or MT field test experiments. Since the purpose of the experiment is to evaluate the utility of MT suggestions, the cosine similarity is always calculated from a MT segment to any previous segments.

In the cross-validation scenario, each segment in the training fold draws similar segments from the burn-in set and the previous segments in the training set. Formally, for $fold_i$, similar segments are drawn from the previous segments in a candidate pool defined by $\{burn\text{-}in, fold_{-i}\}$ where $-i$ refers to the cross-validated folds not labeled $fold_i$. The test set draws its similar segments from the candidate pool $\{burn\text{-}in, fold_i\}$.

Average similarity scores per user under cross-validation fold 3 are listed in Table III. Given that the average similarity score (sim_{avg}) per user is low, we select candidate segments with a similarity score higher than sim_{avg}, as an arbitrary segment is semantically unrelated to a given segment. From the remaining candidates, we establish a heuristic that selects the candidates within 10% of the segment with the highest similarity (sel_{max}), or those candidates whose score is higher than the average of the selected candidates (sel_{avg}). In other words, we aggregate the similarity scores of segments whose scores are greater than $\max(\text{sel}_{max} - 0.1, \text{sel}_{avg})$ from the candidate pool.

The aggregated features described in the previous section are computed by averaging the feature values of the similar segments within the TM and MT experiments, respectively. In some cases, no candidates are selected for the TM or MT aggregation. In these cases, we substitute with average statistics for the particular user across all segments in the training set.

C. Predicting time to edit

We train linear regression models, combining features of the current segment with statistics on the user's behavior on similar segments and call this the *Aggregated* regression model. Similarly, we build baseline regression models just with features from current segment (e.g. WC_{SR}, TER_{SR}). Using a 10-fold cross validation strategy, we train 10 regression models for each user based on the training set. Each model is designed to predict the TTE for a given segment. We use the linear regression classes provided by the Weka open-source machine learning toolkit [12] to orchestrate our experiment. Our linear regression models use M5 attribute selection [13] which incrementally removes features with the smallest coefficient. A ridge regularization parameter is fixed to 1.0×10^{-8}. The regression results are averaged across each fold and reported in Section VI. For the final MT output recommendation, we train a model using all the training folds and evaluate on our held-out test set.

D. Suggesting MT output

After predicting the TTE values on a held-out test set, we recommend whether or not to present MT output to the user by comparing the TTE of the current segment against the TTE of segments with similar word counts in the MT and TM experiments. Given a segment s and its predicted TTE x, we evaluate the number of standard deviations of x from $\hat{\mu}_{MT}$ and $\hat{\mu}_{TM}$, the bootstrap mean of the TTE values in the MT and TM experiments for segments with a source-side word count in the range $[WC_{SR}(s) - 1, WC_{SR}(s) + 1]$. Thus, the following criterion (inspired by the Z-score) is defined for recommending MT output:

$$f(x) = \begin{cases} 1: & |(x - \hat{\mu}_{MT}) / \hat{\sigma}_{MT}| < |(x - \hat{\mu}_{TM}) / \hat{\sigma}_{TM}| \\ 0: & \text{otherwise} \end{cases} \quad (2)$$

$\hat{\sigma}_*$ is the standard deviation of segments in the corresponding sample. We use the bootstrap mean as a robust mean estimate to account for outliers.

TABLE IV
AVERAGE PERFORMANCE OF AGGREGATED AND BASELINE SYSTEMS AFTER 10-FOLD CROSS-VALIDATION FOR LINEAR REGRESSION MODELS TRAINED FOR EACH USER. AVERAGE NUMBER OF INSTANCES IN EACH FOLD IS GIVEN BY *Instances*. MEAN ABSOLUTE ERROR (MAE) AND ROOT MEAN SQUARE ERROR (RMSE) ARE IN SECONDS.

		Baseline			Aggregated		
User	Instances	Corr.	MAE	RMSE	Corr.	MAE	RMSE
1	38.9	0.6128	31.94	45.70	**0.6300**	31.16	44.87
2	38.6	**0.6371**	25.79	47.35	0.6362	25.82	48.47
3	38.6	**0.5102**	18.39	38.58	0.4672	19.56	39.47
4	33.0	**0.5293**	13.34	25.72	0.4947	14.29	26.70

VI. RESULTS

The results for cross-validation and the final evaluation are reported below.

A. Regression results

1) Cross validated models: Average regression results along with baseline results per user are reported in Table IV. While Users 3 and 4 have a lower correlation coefficient, the Root Mean Square Error (RMSE) remains relatively low with respect to Users 1 and 2. In particular, User 4's regression curve implies that many of the model features do not contribute to the predictive power of the model. In fact, a baseline system using only two features performs better than an aggregated system for all the users except User 1. This is likely due to the fact that there are few samples in our data set available for aggregation. The following features have a significant contribution to the regression model: current WC_{SR}, δWC_{SR} for similar TM and MT segments, and δWC_{TG}.

2) Final models: Regression results of the final test per user are reported in Table V and regression coefficients for each attribute are reported in Table VI. The correlation coefficients are higher than those of the cross-validation experiment.

Here, the aggregated system performs better than baseline only for Users 1 and 3. The majority of errors occur in segments with high word counts, or "outlier" cases where either an identical match appears in the translation memory or the user took an abnormally long amount of time to translate a segment.

TABLE III
SIMILAR SEGMENT EXTRACTION FOR TWO RANDOMLY SELECTED SEGMENTS UNDER CROSS-VALIDATION FOLD 3. CANDIDATES MUST HAVE A SIMILARITY SCORE ABOVE THE USER'S AVERAGE SIMILARITY SCORE. TO BE SELECTED, CANDIDATES MUST BE WITHIN 0.1 OF THE MOST SIMILAR SEGMENT IN THE POOL SEL_{max} OR THE CANDIDATE AVERAGE SEL_{avg} − WHICHEVER IS GREATER.

User	Seg.	Exp.	Candidates	Selected	sel_{avg}	sim_{avg}	sel_{min}	sel_{max}
User 3	11710	MT	47	3	0.327	0.227	0.236	0.530
User 3	11710	TM	65	21	0.307	0.218	0.221	0.447
User 4	13421	MT	49	5	0.347	0.227	0.230	0.542
User 4	13421	TM	261	14	0.369	0.229	0.230	0.607

TABLE VI
COEFFICIENTS FOR THE LINEAR REGRESSION MODEL FEATURES LISTED IN TABLE I. FEATURES FOR THE CURRENT SEGMENT ARE LABELED "CURR". FEATURES COMPUTING THE DIFFERENCE BETWEEN THE MT AND TM AGGREGATIONS ARE LABELED "DIFF".

User	WC_{SR}				WC_{TG}			WC_{PE}		
	Curr	TM	MT	Diff	TM	MT	Diff	TM	MT	Diff
1	5.485	-3.580	-2.232	-2.850	-1.419	3.869	0.000	4.584	-2.908	4.646
2	5.398	8.582	-7.832	11.014	-5.526	3.136	-5.515	-2.570	3.949	-3.382
3	3.516	1.427	-5.051	0.000	-1.114	1.849	0.000	-1.634	3.793	0.000
4	2.233	0.000	0.000	-3.156	0.000	0.000	0.000	0.000	0.000	3.061
User		TFW			TTE			TER_{TG}		TER_{SR}
	Intercept	TM	MT	Diff	TM	MT	Diff	TM	MT	Curr
1	-25.742	0.000	1.574	0.000	-0.134	0.314	0.242	0.158	0.000	0.144
2	-21.001	0.000	2.580	0.000	-0.387	0.412	0.483	0.000	0.000	0.000
3	-16.399	0.000	1.570	1.210	0.000	-0.167	-0.223	0.134	0.000	0.000
4	5.263	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

TABLE V
PERFORMANCE OF BASELINE VERSUS AGGREGATED SYSTEM AFTER FULL TRAINING OF LINEAR REGRESSION MODELS FOR EACH USER, EVALUATED ON THE TEST SET. CORRELATION (CORR.), MEAN ABSOLUTE ERROR (MAE), ROOT MEAN SQUARE ERROR (RMSE).

		Baseline			Aggregated		
User	Instances	Corr.	MAE	RMSE	Corr.	MAE	RMSE
1	40	0.6551	29.54	47.19	**0.6682**	28.16	46.31
2	38	**0.7359**	23.57	33.20	0.7290	23.13	33.81
3	40	0.7001	13.01	17.19	**0.7466**	12.53	16.92
4	34	**0.6702**	11.81	15.81	0.6270	11.95	16.49

a) Important features.: For all users, the average time-for-word (TFW) feature for similar TM segments was not useful. In addition, the average TER for MT segments and the average TER difference were not used. Naturally, the word count of the current segment was proportional to the TTE. Likewise, the average word counts and their computed difference were significant features. The reverse TER score (TER_{SR}) was not significant. All other features were significant.

b) Example predictions.: Table VII shows time-to-edit (TTE) prediction results for User 3. While IDs #15, #18, #19 have a relatively good error bound, outliers such as #32 drive up the RMSE. Incidentally, the MT output provided for the 5-word source segment in #32 was "Die andere MLV ausgewählt wird.", which only requires one word swap operation.

While the MT output aligns closely with the user's draft translation in ID #21, the regression model provided an much higher estimate. In this example, a total of 170 MT candidates with similarity scores above the user average (22.65%) were available, providing a group average of 38.46%. However, only one segment was selected (94.28% similar). No other candidates are within 10% of the highest candidate; thus, the MT statistics were unreliable. We further note that of 1488 training instances, 512 segments had only one similar MT segment in the aggregation; 75 segments had none (using global user averages instead).

B. Recommendation based on time to edit

Given the TTE predictions from our regression models, we provide recommendations on whether to provide MT suggestions to the translator. Figure 1 lists the confusion matrices for each user's regression model and Table VIII lists the precision, recall, and F-measure for each user.

	T_a	F_a
T_e	21	6
F_e	8	5

(a) User 1

	T_a	F_a
T_e	14	8
F_e	6	10

(b) User 2

	T_a	F_a
T_e	25	7
F_e	5	3

(c) User 3

	T_a	F_a
T_e	9	10
F_e	6	9

(d) User 4

Fig. 1. Confusion matrices for each user. T_a and F_a correspond to the actual values. T_e and F_e correspond to the estimated values.

TABLE VIII
PRECISION, RECALL, F-MEASURE FOR EACH USER

User	Precision	Recall	F-Measure
1	77.77	72.41	74.99
2	63.63	70.00	77.13
3	78.25	83.33	80.71
4	47.36	60.00	52.93

Understandably, due to the lack of useful features in User 4's regression model, the overall MT suggestion results are

TABLE VII
SAMPLE TIME-TO-EDIT A SENTENCE (TTE) PREDICTIONS FOR USER 3. "MT?" REFERS TO THE ACTUAL/ESTIMATED RESULT OF THE CLASSIFICATION MODEL AFTER APPLYING THE DECISION FUNCTION DEFINED IN (2).

ID	Segment	Translation	WC	TTE	Est TTE	Error	MT?
15	In the License Allocations tab you can explicitly add a new eQube-BI TcRA context that can connect to the current license server.	In der Registerkarte Lizenzzuweisung können Sie explizit einen neuen eQube - BI TCRA Kontext hinzufügen, der sich mit dem aktuellen Lizenzserver verbinden kann.	22	66	70.12	4.12	1/1
18	You need to assign roles to users so that they can perform certain operations.	Sie müssen Rollen Benutzern zuweisen, sodass sie bestimmte Operationen durchfähren können.	14	12	10.48	-1.52	1/1
19	Add Role	Rolle hinzufügen	2	11	10.99	-0.01	0/0
20	Click Submit, if you want to submit the details you entered.	Klicken Sie auf Senden, wenn Sie die Details übermitteln möchten, die Sie eingegeben haben.	11	20	26.97	6.97	1/0
21	Click the role on the left side of the screen, to which you want to assign operations.	Klicken Sie auf die Rolle auf der linken Seite des Bildschirms, der Sie Operationen zuweisen möchten.	17	17	36.94	19.94	1/1
32	The other MLV gets selected.	Die andere MLV wird ausgewählt.	5	66	18.64	-47.36	0/1

low. For the other three users, we report higher F-measures that suggest a correlation between the estimated and actual TTE scores in terms of our goodness measure. However, what does this say about our goodness measure? The right-most column of Table VII lists the actual vs. estimated predictions for the example segments translated by User 3. ID #20 consists of 11 words having a TTE difference of approximately 7 seconds. Looking more closely, for User 3, $\hat{\mu}_{TM} < \hat{\mu}_{MT}$, which implies that MT is not useful for any segments of this length ($\hat{\mu}_{TM} = 2.47$, $\hat{\sigma}_{TM} = 1.51$, $\hat{\mu}_{MT} = 2.69$, $\hat{\sigma}_{TM} = 3.80$). This underlies the importance of significance testing as one of the missing components in our decision function.

CONCLUSION AND FUTURE WORK

In conclusion, we address the problem of quality estimation for machine translation in the CAT scenario by constructing regression models tailored to each translator in order to estimate the productivity of a user. We estimate user productivity in terms of the time taken to edit a translation (TTE). We combine features from the current segment with aggregated features from similar segments in two field test experiments. We find that a trained regression model predicts TTE reasonably well given a limited data set drawn from the preliminary MateCat field test outlined in [1], with exceptions explained by user inconsistencies and limitations in the data captured.

The estimated TTE values for each segment in our test set is compared against the mean TTE values for similarly long segments in the TM and MT sub-experiments. Segments whose TTE is closer to the MT experiment's mean than its TM counterpart are judged to indicate that suggesting machine translation output will improve user productivity. We evaluate this decision function against the actual TTE values to measure the consistency of the regression model. After careful evaluation, we see that this heuristic is deficient in accurately suggesting MT output to the translator in cases where the population means of similarly long TM and MT segments are close. As such, the choice of a decision function should be revisited.

One potential source of problems in the regression model is that each segment contains a limited number of content words. In practice, the content words are the biggest determiners of coherence in a text. Thus, we propose to add additional features based on the content words to our regression model. Additionally, we propose to add the average similarity scores and the average word length between the current segment and the aggregated TM and MT segments.

ACKNOWLEDGMENTS

This work is partially funded by the European Commission under the FP7 project MateCat, Grant 287688. The authors wish to thank Georgia Koutrika for her valuable suggestions in this experiment.

REFERENCES

[1] M. Federico, A. Cattelan, and M. Trombetti, "Measuring User Productivity in Machine Translation Enhanced Computer Assisted Translation ," in *AMTA 2012*, San Diego, California, October 2012.

[2] L. Specia, M. Turchi, Z. Wang, J. Shawe-Taylor, and C. Saunders, "Improving the confidence of machine translation quality estimates," in *Machine Translation Summit XII*, Ottawa, Canada, 2009.

[3] C. Buck, "Black box features for the WMT 2012 quality estimation shared task," in *Proceedings of the Seventh Workshop on Statistical Machine Translation*. Montreal, Canada: Association for Computational Linguistics, June 2012.

[4] C.-Y. Lin and F. J. Och, "Orange: a method for evaluating automatic evaluation metrics for machine translation," in *Proceedings of Coling 2004*. Geneva, Switzerland: COLING, Aug 23–Aug 27 2004, pp. 501–507.

[5] R. Soricut, N. Bach, and Z. Wang, "The SDL Language Weaver Systems in the WMT12 Quality Estimation Shared Task," in *Proceedings of the Seventh Workshop on Statistical Machine Translation*. Montréal, Canada: Association for Computational Linguistics, June 2012, pp. 145–151. [Online]. Available: http://www.aclweb.org/anthology/W12-3118

[6] R. Soricut and A. Echihabi, "TrustRank: Inducing Trust in Automatic Translations via Ranking," in *ACL*, 2010, pp. 612–621.

[7] N. Bach, F. Huang, and Y. Al-Onaizan, "Goodness: a method for measuring machine translation confidence," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 211–219. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002500

[8] Y. He, Y. Ma, J. van Genabith, and A. Way, "Bridging SMT and TM with Translation Recommendation," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden: Association for Computational Linguistics, July 2010, pp. 622–630. [Online]. Available: http://www.aclweb.org/anthology/P10-1064

[9] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, "A study of translation edit rate with targeted human annotation," in *In Proceedings of Association for Machine Translation in the Americas*, 2006, pp. 223–231.

[10] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst, "Moses: Open source toolkit for statistical machine translation," in *ACL*, 2007.

[11] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer, "The mathematics of statistical machine translation: Parameter estimation," *Computational Linguistics*, vol. 19, no. 2, pp. 263–312, 1993. [Online]. Available: http://aclweb.org/anthology-new/J/J93/J93-2003.pdf

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[13] J. R. Quinlan, "Learning with continuous classes," in *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*. World Scientific, 1992, pp. 343–348.

Scene Boundary Detection from Movie Dialogue: A Genetic Algorithm Approach

Amitava Kundu, Dipankar Das, and Sivaji Bandyopadhyay

***Abstract*—Movie scripts are a rich textual resource that can be tapped for movie content analysis. This article describes a mechanism for fragmenting a sequence of movie script dialogue into scene-wise groups. In other words, it attempts to locate scene transitions using information acquired from a sequence of dialogue units. We collect movie scripts from a web archive. Thereafter, we preprocess them to develop a resource of dialogues. We feed the dialogue sequence from a script to a Genetic Algorithm (GA) framework. The system fragments the sequence into adjacent groups of dialogue units or output 'scenes'. We use SentiWordnet scores and Wordnet distance for dialogue units to optimize this grouping so that adjacent scenes are semantically most dissimilar. Then we compare the resulting fragmented dialogue sequence with the original scene-wise alignment of dialogue in the script.**

***Index Terms*—Dialogue, genetic algorithm, movie script, scene.**

I. Introduction

With loads of digital movies being consumed online, movie search has become an important concern in the information retrieval domain. A lot of studies have reported extraction of semantically meaningful information from movie data [1–4]. Such information is useful in browsing, searching, and genre classification. Feature-length movies usually have extensive lengths. This makes movie information search with sole aid of audio-visual data a daunting task. Therefore, movie search mostly relies on textual metadata associated with movies such as title, keywords, and plot.

In general, a movie includes certain textual information such as subtitles and annotations. In addition, movie scripts provide a wealth of information about the dialogue, characters, story, setting and action in movies. They are instrumental in bringing the filmmaker's imagination to life. Most of the movie scripts conform to a set of semi-regular formatting conventions (www.screenwriting.info). These standard terms and conventions can be reaped in order to extract useful elements from scripts.

Scene boundary detection is a fundamental research area which allows navigating movies with semantic ways. A movie consists of a thousand of shots and hundreds of thousands of frames. A scene is a set of shots and a boundary of one of these shots becomes a scene boundary. A number of studies have attempted scene boundary detection that mainly analyzes visual features and group shots according to the visual shot similarity [5], [6]. Scene boundary detection has also been attempted with the aid of scripts and subtitles. The script-subtitles alignment mechanism has been used in [7–9] for the purpose. It may be mentioned in this context that a script, unlike its subtitle counterpart does not convey any information about the time period in the movie.

Crafted aspects of movie dialogue are useful as they are authored deliberately to convey feelings and perceptions of the characters portrayed [2], [3]. Additionally, the screenplay specifies the emotion of utterances with psychological state descriptors. The current article describes a method of detecting scene boundaries/transitions from movie script dialogues. For the purpose, we have prepared textual resource containing structured information from movie scripts. Scripts available online have been utilized for the purpose. For the experiments, each script has been considered a sequence of dialogue units only. Then an attempt has been made to group the dialogue units scene-wise, as they would have appeared in the movie.

Our method segments the sequential dialogue units based on their changing semantic content and affect. It fragments the sequence of dialogue units in such a fashion that consecutive groups/clusters of dialogue text are most 'dissimilar', semantically and affect-wise. A scriptwriter frames his scenes carefully to narrate his story as per his intentions. We have assumed that each scene he writes is a coherent entity. Since we have chosen to use dialogue only, our notion of a scene for this task has been that a 'scene' is a dialogue sequence which sticks together semantically and sentiment-wise and it is somewhat different in the same sense from its neighboring 'scenes'. We have formulated this task as an optimization problem and adopted a Genetic Algorithm (GA) approach to this head. Genetic Algorithms (GAs) are a class of adaptive, randomized search and optimization techniques guided by principles of natural evolutions and genetics [10], [11]. This being fundamentally an optimized segmentation task, GA seemed to be most appropriate for our purpose.

The rest of the paper is structured as follows. Section II provides the rudiments of movie scripts and a detailed description of resource preparation. Section III describes in

Manuscript received December 15, 2012. Manuscript accepted for publication January 11, 2013.

Amitava Kundu and Sivaji Bandyopadhyay are with the Department of Computer Science & Engineering, Jadavpur University, Kolkata-700032, India (e-mail: amitava.jucse@gmail.com, sivaji_ju_cse@yahoo.com).

Dipankar Das is with the Department of Computer Science & Engineering, National Institue of Technology, Meghalaya, Laitumkhrah, Shillong-793003, Meghalaya, India (email: dipankar.dipnil2005@gmail.com).

detail how Genetic Algorithm has been incorporated in our experiments. Section IV provides the results and observations. Finally, we summarize and present the future directions in Section V.

II. Resource Preparation

Most of the feature-length films are produced with the aid of scripts [12] or screenplays. The script presents a detailed vision of the story, settings, dialogues, situations and actions of a film. It gives the filmmakers, the actors and the crews the necessary assistance in bringing the intended situation to life. Thus it can be looked upon as a rich resource that provides vivid textual descriptions of semantic objects within a film.

The Internet Movie Script Database (IMSDb; www.imsdb.com) is a huge online repository of movie scripts. Hundreds of movie scripts, coming straight from the filmmakers, are available for download. In the present task, scripts were collected from the above mentioned database but the collected scripts were subjected to some necessary preprocessing where the scenes were separately identified and stored in a structured format. The following subsection provides a general overview of the elements in a film script followed by the details of preprocessing steps.

A. Highlights of a Movie Script

As mentioned above, a movie script delineates the story, character, setting, action and dialogue of a film [9], [12]. Additionally, the camera directions and shot boundaries are also included. The script may go through a number of revisions prior to production of the movie. Often, after all the shots have been edited the resulting movie may not conform to all the minute details of the original script.

In general, the actual content of a script follows a semi-regular format as shown in Figure 1, which shows a snippet of script from the movie Godfather-II. In a script, a scene starts with the slug line which establishes the physical context of the action that follows. It indicates whether a scene takes place inside or outside (INT or EXT), the name of the location (e.g. 'THE CHURCH PLAZA'), and can potentially specify the time of day (e.g. DAY or NIGHT). Important people and key objects are usually highlighted by capitalizing their names.

The bulk of a script is comprised of dialogue description. Actions of characters are described along with the dialogues for each character. Dialogues have the character name (usually in all caps) followed by the dialogue text. An optional (V.O.) or (O.S.) may follow the character name indicating that the speaker should be off-screen (V.O. stands for a "Voice Over"). In most scripts, dialogue actions are clearly separated from action and camera directions by enclosing the non-dialogue descriptions in brackets. Usually scarce, parenthetical directives convey phenomena such as inflection or mood. An example of a dialogue unit follows:

GARDNER*: (a little nervous) I've heard a lot about you, Mr. Corleone.*

```
EXT. THE CHURCH PLAZA - NIGHT

The men continue on their night-walk, up to the plaza of the
church.

                              STROLLO
                         (Sicilian)
                ...who harbors the boy Vitone
                Andolini.

The figure of a single man on a mule passes them.

                              MOSCA
                         (Sicilian)
                Let no one give help to the boy
                Vito Andolini...

The man on the mule makes his way out of the village and
disappears into the distance.

We begin to hear, very quietly, the Waltz repeated once again.
```

Fig. 1. Portion of a screenplay of the film Godfather-II

```
DEANNA: Don't follow me!


SCENE: EXT. TAHOE LAWN AND TABLES - MED. SHOT - DAY
CONTEXT: Rushing through the tables, waving an arm jangling with gold jewelry,
CONNIE: Mama...Mama! Here I am!
CONTEXT: She throws her arms around her Mother, who returns the affection somew
MAMA: Constanzia. We expected you last week; we sent the car to pick you up at
CONNIE: I know, it was chaos; but anyway, here I am one week late. (lifting a s
MAMA: (not giving him a chance to greet her) Yes, thank you.
CONNIE: How are the kids?
MAMA: Well, thank you, they asked for you all week.
CONNIE: I got surprises for everybody!
MAMA: (glancing at the wrapping) Bought at the airport.
CONNIE: (gazing about) This is swell. Where's Michael? I've got things to get s
MAMA: You go see your children first, and then you wait to see your brother lik


SCENE: EXT. THE BOATHOUSE - DAY
```

Fig. 2. Snippets of a resource file showing a typical scene from Godfather-II screenplay

```
DEANNA: Don't follow me!

CONNIE: Mama...Mama! Here I am!
MAMA: Constanzia. We expected you last week; we sent the car to pick you up at the airport
CONNIE: I know, it was chaos; but anyway, here I am one week late. (lifting a shiny green
MAMA: (not giving him a chance to greet her) Yes, thank you.
CONNIE: How are the kids?
MAMA: Well, thank you, they asked for you all week.
CONNIE: I got surprises for everybody!
MAMA: (glancing at the wrapping) Bought at the airport.
CONNIE: (gazing about) This is swell. Where's Michael? I've got things to get straight wit
MAMA: You go see your children first, and then you wait to see your brother like everybody

PENTANGELI: Hey, kid! You got any red wine?
```

Fig. 3. Snippets of a resource file showing the dialogues of the scene from Godfather-II shown in Fig. 2.

B. Preprocessing of scripts

As mentioned already, movie scripts are written in a semi-regular format. Variety of formats makes preprocessing a challenging task. To accomplish our experiments, we needed to prepare a structured resource from scripts so that we could evaluate the success of our scene boundary detection algorithm later on. Using knowledge of the usual anatomy of movie scripts outlined in the above subsection, the

information pertaining to different scenes has been identified. In each of the sets of resource files, one for each script, the scenes have been stored in a structured format, as indicated in Figure 2. A scene sets off with the scene heading and the following line describes either a dialogue or an action. Each dialogue turn starts with the name of the speaker in capitals followed by the text. An action starts off with the word "CONTEXT" and describes what is going on in the scene. Each scene is separated from the following and the preceding ones by blank lines. A significant amount of manual labor has been invested in making necessary refinements and corrections before storing the resource in the required structured format.

Additionally, another set of resource files where scene-wise dialogues are stored have been prepared. Here dialogues from different scenes are separated with blank lines. All context information and scene headings have been excluded. These files have been prepared so that a script can later on be considered as a sequence of dialogue units without any knowledge of scene boundary (see Figure 3). However, scenes without dialogue were also abundant. They have been thus ignored altogether.

III. System Framework

In the experimentation, each script has been considered merely as a sequence of dialogue units excluding the context information (and even parenthesized directives).

Our objective was to detect the scene transitions looking at a sequence of dialogue texts. A close manual examination revealed abrupt changes in overall topic and affect across dialogues from consecutive scenes in many cases. This observation motivated us to take an attempt for detecting scene transitions. We have incorporated a genetic algorithm approach for the purpose. Figure 4 and Figure 5 illustrate the scheme and convey the objective.

A. The Genetic Algorithm Approach

Genetic Algorithm (GA) is a search heuristic that mimics the process of natural evolution. GAs belong to the larger class of Evolutionary Algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution: crossover, mutation and selection.

In the present work, the challenge is to frame the task of detecting scene boundaries as an optimization problem. As mentioned above, the changes in topic or overall affect of dialogues have often been observed across consecutive scenes. Therefore, we have made a principal assumption that dialogues from consecutive scenes are "dissimilar" in some sense. The main idea is to segment the sequence of dialogue units into scenes in such a way that makes the consecutive dialogue groups as "dissimilar" as possible. In order to implement GA, we have utilized the Java Genetic Algorithms and Programming Package (JGAP version 3.6.2), an open-source Java based tool.

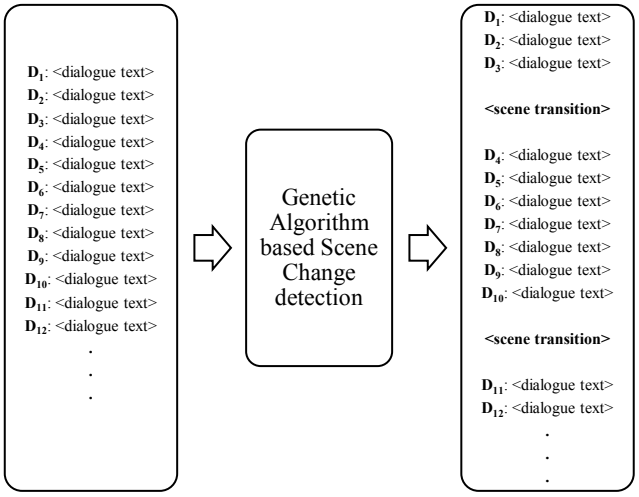


Fig. 4. The basic objective of our scheme. The input is a sequence of dialogue text as spoken in sequence by characters in film; D_1, D_2, D_3…. The system outputs a scene-wise grouping of dialogue units. A group can be seen as set of dialogue units belonging to a particular scene

1) Encoding.

As a genetic algorithm applies to encoded candidate solutions or chromosomes, before one can proceed, an encoding technique of chromosomes has to be chalked out. We have devised a binary encoding technique for the purpose where a chromosome is a string of '0's and '1's. As already mentioned, we have treated a script merely as a sequence of dialogue units. If we consider a script with n dialogue units, in our scheme, each chromosome is a binary string of length n-1. The encoding is such that a '1' at m^{th} allele position of a chromosome indicates a scene transition after the m^{th} dialogue unit while a '0' indicates scene continuation. Since the nth dialogue unit would be the last one in the script, encoding a transition for it in the solution string would be unnecessary. Hence the length of such a string/chromosome is n-1. Figure 6 illustrates the scheme vividly.

2) Crossover, Mutation and Selection.

Since binary encoding has been adopted, usual genetic operators have been conveniently deployed. Single point crossover has been adopted. Mutation involves bit flipping where a '0' is replaced by a '1' and vice-versa. Standard Roulette Wheel Selection has been applied.

3) Population Size and Evolution.

We have incorporated the elitist model of GA wherein the fittest chromosome from each generation is preserved so as to ensure the best solution is not lost. Population size has been kept 50. We run the GA up to 100 generations.

4) Fitness Computation.

The fitness function is at the heart of our genetic algorithm framework. Dialogues across consecutive scenes have been assumed to be different or dissimilar.

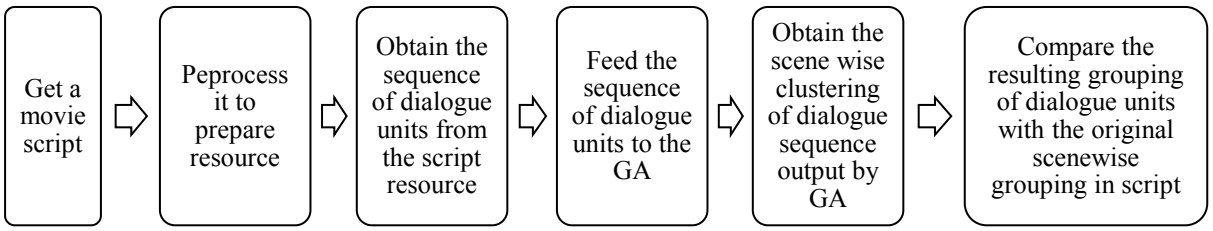


Fig. 5.Workflow of our system

In our scheme, those solutions that encode scene transitions resulting in greater dissimilarity across consecutive scenes have been considered as fitter chromosomes. Thus, given the binary encoding of each chromosome, the resulting scene-segmentation of dialogue units has been easily reconstructed. Thereafter, dissimilarity across consecutive groups of dialogue has been capitalized in computation of fitness. The SentiWordnet 3.0 [13] scores and Wordnet similarity of encoded scenes have also been accounted for.

(a) SentiWornet scores.

For every script, the Stanford CoreNLP[1] (version 1.3.3) has been used to tokenize dialogue unit text, identify the named entities in turns and assign part-of-speech (POS) tags to tokens in a turn. Thereafter SentiWordnet 3.0 has been used to compute the total sentiment scores[2] of each dialogue unit. Sentiment scores of nouns excluding named entities, adjectives, adverbs and verbs only have been considered for this computation.

Consider the dialogue unit given below,

DEANNA: *Relax, Freddie honey. Come dance with me.*

The POS tagged version of the above with sentiment values indicated in parentheses is as follows,

Relax[1]/**VB**(-0.06668) ,/,*Freddie*/**NNP***honey*[2]/**NN**(0.04166) ./. *Come*[3]/**VB**(0.00529) *dance*[4]/**NN**(0.0) *with*/**IN** *me*/**PRP** ./.

Likewise, sum total sentiment score of all dialogues in every scene was computed, as encoded in any given chromosome. If a chromosome encoded scenes (groups of dialogue) S_1, $S_2 \ldots S_n$ with total sentiment values P_1, P_2, …,P_n respectively, its fitness $F_{sentiment}$ is computed as follows,

$$F_{sentiment} = \sum_{i=2}^{n} (P_i - P_{i-1})^2$$

(b) Wordnet distance of keywords.

Each dialogue unit has been tokenized, named entities have been identified and POS tags have been assigned to tokens using Stanford CoreNLP.

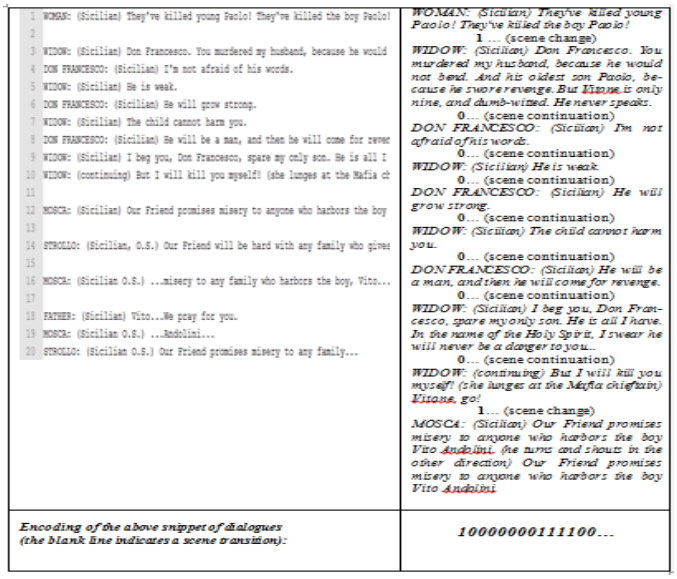


Fig. 6.Snippet of dialogue sequence from Godfather-II and its encoding.

Additionally, each of the dialogue units has been annotated with the CoreNLP parse annotator so as to obtain the Stanford semantic dependency relations [14] of each turn. Object relations have also been identified (if present) in each turn namely *dobj*, *iobj* and *pobj*. Thereafter, the dependent components in those relations have been collected. A dependent in such an object relation has been considered as a potential keyword or topic word of the piece of dialogue. However, a dependent is ignored if it appeared in a list of stop words. An example with the keywords thus identified is given below (numbered ones are the identified words in question)

CONNIE: *I got surprises[1] for everybody[2]!*

Wordnet distance between keywords across consecutive dialogue groups have been considered for fitness computation. The getDistance() method of RiTaWordnet[3] library has been deployed for computation of semantic distance between two words in Wordnet hierarchy.

Say the distance of two word is given by $d(w_1,w_2)$. It is the absolute value of the aforementioned distance metric between two words w_1 and w_2 in the Wordnet ontology.

[1] http://nlp.stanford.edu/software/corenlp.shtml
[2] Online resource: http://sentiwordnet.isti.cnr.it/code/SWN3.java
[3] http://rednoise.org/rita/wordnet/documentation/index.htm

TABLE 1.
PERFORMANCE OF SCENE BOUNDARY DETECTION

Movie	#Dialogue Units in Script	Actual #scene transitions in script dialogue sequence	With $F_{baseline}$		With $F_{sentiment}$		With $F_{wordnet}$		With $F_{combined}$	
			P_{GA}	Transitions correctly detected	P_{GA}	Transitions correctly detected	P_{GA}	Transitions correctly detected	P_{GA}	Transitions correctly detected
Godfather-II	1190	195	31.2%	69.3%	51.3%	46.7%	57.6%	47.1%	55.6%	48.7%
Casino	2478	452	41.7%	64.4%	52.3%	49.1%	51.7%	42.7%	52.7%	49.6%
Citizen Kane	1014	88	37.8%	73.3%	52.1%	45.4%	58.8%	38.7%	55.4%	38.7%
		Average	36.9%	69.0%	51.9%	47.1%	56.0%	42.8%	54.6%	45.7%

It gives the minimum distance between any two senses for the two words in the WordNet tree (result normalized to 0–1). If a chromosome encodes scenes (groups of dialogue) S_1, S_2,…, S_z, with sets of keywords K_1, K_2,…, K_n respectively, then the fitness $F_{wordnet}$ is given by

$$F_{wordnet} = \sum_{i=2}^{n} \sum_{\substack{x \in K_i, \\ y \in K_{i-1}}} d(x, y).$$

(c) Combined Effect.

To examine the combined effect of the above two distances we have also experimented with the following fitness function which mixes the effect of both.

$$F_{combined} = F_{sentiment}\, F_{wordnet}.$$

B. Evaluation of Solution

Once the fittest chromosome across all the generations was obtained, the solution was obtained. We have used the Hamming distance for the evaluation purpose as our solutions have been encoded using binary encoding. Hamming distance of two binary strings is the number of bit positions that differ. Since we had prepared our resource files with the scene-boundary information, we were able to construct the actual solution as per our encoding scene. Then we compared the solution obtained using the GA with the former. If S_{actual} was the binary string representing the actual solution and S_{GA} the solution obtained using GA, we defined the performance metric P_{GA} as follows:

$$P_{GA} = \left(1 - \frac{HammingDistance(S_{actual}, S_{GA})}{\#Dialogueunitsinthescript - 1}\right) \times 100\%$$

$$= \frac{\#Bit\ positions\ that\ match\ between\ S_{GA}\ \&\ S_{actual}}{Size\ of\ encoding\ of\ segmented\ dialogue\ sequence} \times 100\%.$$

The present task is not a supervised learning problem. So accurate spotting of scene transitions is understandably difficult. Thus, besides counting the number of correct transitions located, we have also tried to evaluate the performance differently. The above metric measures the similarity of the dialogue grouping output by the system with the original scene-wise grouping.

We present a simple baseline for comparison. We do not use identities of speakers (their names) of turns anywhere in the experiments. Instead, word overlap of scenes was used in devising a baseline. Say, S_i and S_{i-1} are adjacent dialogue scenes with the sets of words (stop words and named entities excluded) W_i and W_{i-1}. The word overlap can be given by the Jaccard index:

$$J(S_{i-1}, S_i) = \frac{|W_i \cap W_{i-1}|}{|W_i \cup W_{i-1}|}.$$

The closer $J(S_{i-1}, S_i)$ gets to 1 the greater the similarity of adjacent scenes S_i and S_{i-1}. Put in another way, the closer $J'(S_{i-1}, S_i) = 1 - J(S_{i-1}, S_i)$ gets to 1 greater the dissimilarity of the same scenes in terms of common words. Thus the baseline fitness was devised as follows:

$$F_{baseline} = \sum_{i=2}^{n} J'(S_{i-1}, S_i)$$

IV. RESULTS AND OBSERVATIONS

A number of scripts have been chosen and processed for resource preparation. Scripts of three movies, namely *Godfather-II*, *Casino* and *Citizen Kane* have been chosen for the purpose of experimentation. These scripts have been picked up due to the presence of abundant dialogue units (in excess of 1000). Table 1 sums up the performance of the scene transition detection algorithm. It reports the proposed performance metric P_{GA} as well as percentage of transitions correctly detected.

When the encoded version of a solution is considered, the proposed P_{GA} reflects the extent of similarity of the obtained solution with the actual one. In other words it reflects a score for the obtained solution.

The results report an average accuracy of 47.1% in correct detection of scene transitions with the proposed fitness metric $F_{sentiment}$. Use of Wordnet features in fitness computation improves the average value of P_{GA} but results in a fall in average accuracy to 42.8%. The combined sentiment and Wordnet features produce an average accuracy of 45.7%. It seems that SentiWordNet features have been more informative than WordNet based distance.

We have observed a lot of false detections of scene boundaries (detection of a transition where the scene should actually continue), resulting in over-fragmentation of dialogue sequence. The modest values for P_{GA} may be attributed to the same. Moreover, no information about the distribution of number of transitions in scripts was available. Thus, the GA used in our experiments initializes each chromosome randomly without a bias to a '0' (scene continuation) or '1' (scene transition). There has been no bias in mutating a '0' or '1' valued gene.

As far as the baseline is concerned, its average P_{GA} score of 36.9% is outperformed although it gives higher average scene transition detection accuracy (69.0%). We observed that in the baseline system the number of scenes output by the system was way higher than that in the original scripts. For example, in *Citizen Kane* as many as 731 scenes were output while the script describes 88 scenes of dialogue only. In other words, the solution chromosome for all the three movies contain mostly '1's, under the binary encoding scheme. However, P_{GA} dips as expected.

V. Conclusions and Future Directions

In this article, we have reported a mechanism to fragment a sequence of movie script dialogue into scene-wise groups. Wordnet based semantic relatedness and SentiWordnet based affective features have been fed to a genetic algorithm framework to obtain a fragmented, scene-segmented dialogue sequence. The initial results have been encouraging. As a part of future work we are planning to apply our mechanism to easily available movie subtitles. The time information present in subtitles may be capitalized. It may be noted that subtitles hardly contain information about the setting in the movie and conveys mostly the dialogue. The proposed scheme coupled with the time information of subtitles seems to be appropriate for locating scene transitions. Besides, the same algorithm can be suitably modified for identification of emotional episodes in conversation from subtitles or scripts. We are planning to incorporate more textual features (e.g. co-reference resolution) into the proposed framework.

References

[1] J. J. Jung, E. You, and S. B. Park, "Emotion-based character clustering for managing story-based contents: a cinemetric analysis," Multimedia Tools and Applications, pp. 1-17, 2012.

[2] G. I. Lin, and M. A. Walker, "All the world's a stage: Learning character models from film," In Proceedings of the Seventh AI and Interactive Digital Entertainment Conference, AIIDE, vol. 11, 2011.

[3] M. A. Walker, G. I. Lin, and J. E. Sawyer, "An Annotated Corpus of Film Dialogue for Learning and Characterizing Character Style," *Proceedings of LREC*, 2012.

[4] R. E. Banchs, "Movie-DiC: a movie dialogue corpus for research and development," In Proc. of the 50th Annual Meeting of the ACL. 2012, 2012.

[5] M. Cooper, and J. Foote, "Scene boundary detection via video self-similarity analysis," *International Conference on Image Processing*, vol. 3, pp. 378-381, IEEE, 2001.

[6] J. Wang, and T. S. Chua, "A cinematic-based framework for scene boundary detection in video," *The Visual Computer*, vol. 19, no.5, 2003, pp-329-341.

[7] T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar, "Movie/script:Alignment and parsing of video and text transcription," *Proc. 10th European Conf. Computer Vision*, 2008, pp. 158-171.

[8] S. B. Park, H. N. Kim, H. Kim, and G. S. Jo, "Exploiting Script-Subtitles Alignment to Scene Boundary Detection in Movie," *In Multimedia (ISM), 2010 IEEE International Symposium on*, pp. 49-56. IEEE (2010)

[9] R. Turetsky, and N. Dimitrova, "Screenplay alignment for closed system speaker identification and analysis of feature films," *Proc. IEEE Int. Conf. Multimedia and Expo (ICME'04)*, 2004, pp. 1659-1662.

[10] D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addi-son-Wesley, New York, 1989.

[11] L. (Ed.) Davis, "Handbook of Genetic Algorithms," Van Nostrand Reinhold, New York, 1991.

[12] A. Jhala, "Exploiting Structure and Conventions of Movie Scripts for Information Retrieval and Text Mining," *Interactive Storytelling*, 2008, pp. 210-213.

[13] B. Stefano, A. Esuli, and F. Sebastiani, "Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining," In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta, May. European Language Resources Association (ELRA)*, 2010.

[14] M. C. De Marneffe, C. D. Manning, "Stanford typed dependencies manual," nlp. stan-ford. edu/software/dependencies_manual.pdf , 2008.

Efficient Routing of Mobile Agents in a Stochastic Network

Amir Elalouf, Eugene Levner, and T.C.E. Cheng

Abstract—**Mobile agents are autonomous programs that may be dispatched through computer networks. Using a mobile agent is a potentially efficient method to perform transactions and retrieve information in networks. Unknown congestion in a network causes uncertainty in the routing times of mobile agents so the routing of mobile agents cannot rely solely on the average travel time. In this paper we deal with a given stochastic network in which the mobile agent routing time is a random variable. Given pre-specified values R and PR, the objective is to find the path with the minimum expected time under the constraint that the probability that the path time is less than R is at least PR. We show that this problem is NP-hard, and construct an exact pseudo-polynomial algorithm and an ε-approximation algorithm (FPTAS) for the problem.**

Index Terms—**Agent-based architecture, fast routing algorithm, FPTAS, stochastic routing.**

I. Introduction

In the context of distributed computer communication networks, we define an *agent* as "a human or software system that communicates and cooperates with other human or software systems to solve a complex problem that is beyond the capability of each individual system" [1]. This definition is compatible with the definitions given by Jennings and Wooldridge [2], Shen et al. [3–5], and Peng et al. [6]. An *autonomous agent-based system* is a system that is able to function in some environments without the direct intervention of human beings or other agents and that has control over its own actions and internal states. Its major advantage is that it can effectively access distributed resources in a low-bandwidth network. In particular, such a system may be useful in a client/server model, in which a client needs to access a huge database on a server.

This access requires a large amount of data to be transmitted over the network and may significantly waste bandwidth. By sending a mobile program to the server and performing data processing at the server, unnecessary data transmission can be avoided. Even if the client/server connection fails, the mobile program can successfully perform its mission.

Mobile agent-based technologies have been used in distributed computer networks for more than two decades. Establishing the notion of mobile agents in 1994, White [7] describes a computational environment known as "Telescript" in which running programs are able to transport themselves from host to host in a computer network. Tsichritzis [8] introduces the notion of mobile computation by describing a hypothetical computing environment in which all the objects are mobile. Within the scope of this paper, we follow the definitions in [3–11] and define a *mobile agent* as "a software agent that is able to autonomously migrate from one host to another in a computer network."

The latest achievements in multi-agent systems have brought new possibilities for integrated systems management. In typical applications, a mobile agent visits several hosts in a network in order to complete its mission. The hosts provide the agent with information and access to services, as well as a platform for carrying out various actions and for communicating with other agents. The services and information that the agent needs to access are distributed across different sites and are available in different forms and at different levels of accuracy and degrees of reliability. This gives rise to a mobile agent routing problem with uncertain data, in which limited computational resources are available at many possible sites.

A given benefit function determines how much benefit (e.g., information from sites, retrieval data, etc.) each site contributes to an agent's mission. Since many different sites provide information yielding different degrees of benefit, the mobile agent should find a best possible itinerary to visit them under resource constraints. The problem of enhancing the efficiency of mobile agents then reduces to the problem of finding resource-constrained extremal paths in a graph. The agent's routing problem consists in finding an information- or resource-constrained route that provides the best agent performance.

In this study we deal with mobile agent routing in a stochastic network. An agent takes an instruction to move from one location to another until it reaches its destination, where each action/move involves uncertainty. The common objective function for an agent is the minimum expected

Manuscript received on June 1, 2012; accepted for publication on August 23, 2012.

A. Elalouf is with Bar-Ilan University, Ramat Gan, Israel (e-mail: amir.elalouf@biu.ac.il).

E. Levner is with Ashkelon Academic College, Ashkelon, Israel (e-mail: elevner@acad.ash-college.ac.il).

T.C.E. Cheng is with Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: edwin.cheng@inet.polyu.edu.hk).

performance time or the minimum expected cost. In contrast to many earlier known agent routing problems (see, e.g. [12–15]), we study uncertainty by explicitly taking into account variance in agents' routes and probabilistic path characteristics.

We do this by incorporating a corresponding non-linear constraint into the problem formulation. To the best of our knowledge, there is no work in the literature with a focus on the design of efficient (polynomial-time) solution methods for the constrained stochastic agent-routing problem. Aiming to fill this research gap, we develop a fast ε-approximation algorithm for solving the considered problem. Another contribution of this paper is that, whereas many previous works (e.g., [12–14]) have considered acyclic networks, we allow the network to contain cycles, which makes the problem much more practical. While the simple deterministic shortest path problem can be solved in polynomial time, the considered stochastic agent-routing problem turns out to be NP-hard.

In the next section we provide a brief overview of other works related to our study. In Section III we formulate the problem. In Section IV we present the exact dynamic programming (DP) solution algorithm. In Section V we construct a new fully polynomial time approximation scheme (FPTAS) algorithm. Section VI concludes the paper.

II. Related Works

The following basic mobile agent-routing problems have been studied in the literature:

Problem P1. To maximize the total benefit generated from agent travel, subject to the condition that the total travel time (sometimes called "delay") does not exceed a given threshold. Such a problem has been studied by Camponogara and Shima [12] and by Elalouf and Levner [1].

Problem P2. To minimize the agent's total expected travel time to complete the task under the constraint that the total travel cost does not exceed a given budget limit. Such a problem has been investigated by, e.g., Brewington et al. [15], Hassin [14], Goel et al. [16], and Xue et al. [17], among many others.

For the agent routing task, a computational scheme considering multiple objectives has been pursued by Wu et al. [18], who combine three objectives (communication cost, path loss, and detected signal energy level) into a single function and optimize it using a genetic algorithm that outperforms local heuristics.

To evaluate the effectiveness of multi-objective algorithms against a single-objective approach, Rajoopalan [19] applies a more general weighted genetic algorithm (WGA) iterated with different weights in order to obtain different non-dominated routing solutions.

Osman et al. [20] analyze an execution model for agent routing to develop a pragmatic framework for fault tolerance in agent systems. This framework adopts a communication-pair, independent-check pointing strategy.

In this paper we consider the mobile agent framework described by Rech et al. [21] and by Camponogara and Shima [12]. Specifically, we develop a graph-theoretic model for computing the agent's itinerary under resource constraints, and on the basis of this model we design exact DP and approximation solution algorithms.

In what follows, we suggest a general three-stage technique, which follows and extends an earlier computational scheme suggested by Gens and Levner [22, 23] and by Elalouf and Levner [13] for the Knapsack and routing problems, respectively.

The new technique essentially improves on the algorithms proposed by Camponogara and Shima [12] and Hassin [14] for the deterministic constrained routing problems P1 and P2, and also provides a new way to obtain a fast solution for the stochastic routing problem.

III. Problem Formulation

The problem framework is based on a computational network composed of a graph (possibly cyclic), $G = (N, A)$, with a set N of nodes, a set A of arcs, a start node $s = 1$, and a destination node $t = n$, where $|N| = n$ and $|A| = h$. The term t_{ij}, denoting the time to traverse arc (i, j) in G, is a normal random variable characterized by two parameters: the expected time m_{ij} and the variance v_{ij}. The parameters m_{ij} are assumed to be integers. A path p is called *feasible* if the probability that the path time is less than R is at least P_R, where R and P_R are given values. The problem is to find a feasible path with the minimum expected time.

Problem input: $G(N, A)$: a given graph.

For any arc $(i, j) \in A$, two parameters are given: m_{ij}, the expected time; v_{ij}, the variance.

$M(p)$ denotes the expected time to traverse path p; $M(p) = \sum_{(i,j)\in p} m_{ij}$.

$V(p)$ denotes the variance of the time it takes to traverse path p. We assume that all the times t_{ij} are independent random variables, and therefore $V(p) = \sum_{(i,j)\in p} v_{ij}$.

In a mathematical form, the problem is to find a path p such that

$$\min_p \left(M(p)\right)$$

$$s.t$$

$$M(p) + \phi^{-1}(P_R)\sqrt{V(p)} \le R$$

Note that ϕ^{-1} is the inverse form of the standard normal distribution. The meaning of the constraint is evident, i.e., if the constraint is satisfied, the probability that the traverse time will not exceed R is at least P_R.

IV. Exact Solution Algorithm: Dynamic Programming

This section introduces an exact DP solution algorithm. Since m_{ij} are assumed to be integers, DP is a pseudo-polynomial solution algorithm. Its complexity is estimated below.

Let us associate with each path p a pair (M, V), where $M = M(p)$ is the expected time to traverse path p, and, correspondingly, $V = V(p)$ is the variance of the time to traverse p. We deal with sets $S(k)$ of pairs (M, V), arranged in increasing order of the M-values, so that every pair in $S(k)$ corresponds to a path from node s to a node k. In order to restore the path corresponding to a pair (M, V), we define for each pair a predecessor pair and use standard backtracking.

If there are two pairs in $S(k)$, $(M1, V1)$ and $(M2, V2)$ such that $M1 \le M2$ and $V1 \le V2$, then the pair $(M2, V2)$ is called *dominated* and may be discarded. Let UB be an upper bound on the total expected time for the optimal path. For instance, UB can be set to $\sum_{(i,j)\in A} m_{ij}$. The polynomial time DP solution algorithm is as follows:

Algorithm 1. Exact pseudo-polynomial DP solution

1. Input: $G(N, A)$, $|N| = n$, $|A| = h$, $\{(m(i,j), v(i,j) \mid (i,j) \in A\}$, R
2. Output: A constrained path with minimum expected time
3. **Step 1.** [Initialization]
4. Set $S(1) = \{(0, 0)\}$, $S(k) \leftarrow \varnothing$ for $k = 2, \ldots, n$
5. **Step 2.** [Generate $S(2)$ to $S(n)$]
6. Repeat n-1 times
7. for *each arc* $(u, k) \in A$ (*leading from node u to node k*)
8. $W \leftarrow \varnothing$
9. for each pair $(M, V) \in S(u)$ do
10. if $M + m(u,k) + \phi^{-1}(P_R)\sqrt{V + v(u,k)} \le R$ then $W \leftarrow W \cup \{(M+m(u, k), V+v(u, k))\}$
11. endfor
12. $S(k) \leftarrow$ merge($S(k)$, W); during merging eliminate the dominated pairs
13. endfor
14. End Repeat
15. **Step 3**. [Determine optimal solution]
16. find min M in $S(n)$, denote it by *ans*
17. Return *ans* as the optimal time; use backtracking to find optimal path.

Proposition 1.The complexity of the DP solution algorithm (Algorithm 1) is O($hnUB$).

Proof: Since the times are integers and we discard dominated pairs, there are at most UB pairs in sets W and $S(k)$. Furthermore, constructing W in lines 9–11 requires $O(UB)$ elementary operations, because W is constructed from a single $S(k)$. Merging the sorted sets W and $S(k)$ in line 12, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is at most UB).

In Step 2, lines 5–14, we have two nested loops, where the first one begins at line 6 and the second at line 7. These two loops go over all the arcs n–1 times, so in total we have O(hn) iterations of lines 11–13. Thus, the total complexity of Algorithm 1 is O($hnUB$). □

V. Fully Polynomial Time Approximation Scheme

A. General Description of the FPTAS

Our approach to constructing an FPTAS follows the so-called interval partitioning computational scheme. The interval partitioning technique was originally proposed by Sahni [24] for the Knapsack problem and was later improved by Gens and Levner [21], Levner et al. [25], and Elalouf et al. [1]. We suggest a scheme that consists of three main stages:

Stage A: Find a preliminary lower bound LB and an upper bound UB on the optimal path's expected time such that $UB/LB \le n$.

Stage B: Find improved lower and upper bounds such that $UB/LB \le 2$.

Stage C: Partition the interval $[LB,UB]$ into n/ε equal subintervals, delete sufficiently close solutions in the subintervals (taking only one "representative" from each subinterval), and then find an ε-approximation solution using full enumeration of the "representatives".

This technique is similar to that presented by Elalouf et al. [1]. Note, however, that the type of problem treated in the present paper is more practical than that in [1]. First, the problem considered here is of a stochastic nature, so it is described by a non-linear constraint. Second, its underlying graph G is allowed to have cycles. As a result, the algorithm proposed herein has a different complexity compared with that in [1].

B. Stage A: Finding Preliminary Lower and Upper Bounds

We use the following greedy technique: Let $A = \{a_1, a_2, \ldots, a_h\}$ be the set of arcs in $G(N, A)$. Denote graph $G'(N', A')$ with the same set of nodes, i.e., $N' = N$, and the set of arcs $A' \subseteq A$. To define A', we use the notation x_{ai}, a binary variable. If $x_{ai} = 1$ then $a_i \in A'$; otherwise $a_i \notin A'$. We order the arcs in G in non-decreasing order of their expected times, i.e., $m_{[a1]} \le m_{[a2]} \le \cdots \le m_{[ah]}$, and initialize $x_{ai} = 0$ for any $i = 1, \ldots, h$ (i.e., we initialize G' as a graph with no arcs).

Then we sequentially set $x_{[a1]} = 1, x_{[a2]} = 1, \cdots$ and add each arc to the graph until we obtain a path from the source to the destination that satisfies the constraint.

If all $x_{ai} = 1$ but we cannot find such a path, there is no feasible solution for the problem considered. Let x_k be the last variable that is set to 1 in the above procedure. Then we set $m_0 = m_{ak}$. Obviously, the optimal total travel time (denoted by OPT) must lie between m_0 and nm_0. When OPT equals zero, the above greedy procedure in Stage A finds the exact

optimal solution (i.e., a path of zero duration) and Stages B and C are not required.

Proposition 2. The complexity of the FPTAS in Stage A is $O(n^2 \log h)$.

Proof. Sorting the arcs described above is done in $O(h \log h)$. Each check of whether the graph G has a feasible path on a selected set of arcs requires $O(n^2)$ time [26]. The total number of checks is $O(\log h)$ if we use a binary search in the interval $[1, h]$. Thus, the complexity of Stage A is $O(n^2 \log h)$. □

C. Stage B: Finding Improved Bounds

This stage has two building blocks: a test procedure denoted Test(w, ε), and a narrowing procedure denoted BOUNDS, which uses Test(w, ε) as a sub-procedure. The procedure is similar to the testing method described in [1] and [27], with some minor changes that take the stochastic nature of the problem into account.

Test Procedure (Test(w, ε))

Test(w, ε) is a parametric dynamic-programming type algorithm that has the following property: Given positive parameters w and ε, it reports either that the minimum possible expected travel time is $M^* \le w$ or that that $M^* \ge w(1-\varepsilon)$.

Test(w, ε) will be repeatedly applied as a sub-procedure in the algorithm BOUNDS below to narrow the gap between UB and LB until $UB/LB \le 2$.

Associate a pair (M, V) with each path p, where $M = M(p)$ is the path's expected travel time, and, correspondingly, $V = V(p)$ is the variance of the path time. We deal with sets $S(k)$ of pairs (M, V) arranged in increasing order of the M-values so that every pair in $S(k)$ corresponds to a path from the start node s to a node k. As in the DP solution algorithm above, we discard all the dominated pairs in all sets $S(k)$.

If $M2 - M1 \le \delta$, then the pair (M, V) is called δ-*close*. We discard δ-close pairs from set $S(k)$ according to the following procedure:

(a) Let w be a given parameter satisfying $LB \le w \le UB$. For each $S(k)$, partition the interval $[0,w]$ into $\lceil n/\varepsilon \rceil$ equal subintervals of size no greater than $\delta = \varepsilon w/n$.

(b) If, for a given subinterval, there are multiple pairs from S(k) for which the value of M falls into the subinterval, discard all such δ-close pairs, leaving only one representative pair in the subinterval, namely, the pair with the smallest (in this subinterval) V-coordinate.

(c) Any pair (M, V) with $M > w$ (called w-redundant) must be discarded.

The algorithm for Test(w, ε) is as follows:

Algorithm 2. Testing Procedure (Test(w, ε))

1. Input: $G(N, A)$, $|N| = n$, $|A| = h$, $\{(m(i,j), v(i,j) \mid (i,j) \in A\}$, R
2. Input ε, w
3. $\Delta \leftarrow \varepsilon w/n$
4. **Step 1.** [Initialization]
5. Set $S(1) = \{(0, 0)\}$, $S(k) \leftarrow \varnothing$ for $k = 2, \ldots, n$
6. **Step 2.** [Generate $S(1)$ to $S(n)$]
7. Repeat n-1 times
8. for each arc $(u, k) \in A$ (leading from node u to node k)
9. $W \leftarrow \varnothing$
10. for each pair $(M, V) \in S(u)$ do
11. if $M + m(u,k) + \phi^{-1}(P_R)\sqrt{V + v(u,k)} \le R$ then $W \leftarrow W \cup \{(M+m(u, k), V+v(u, k))\}$
12. endfor
13. $S(k) \leftarrow$ merge($S(k)$, W); during merging eliminate the dominated pairs and the δ-close pairs
14. endfor
15. End Repeat
16. **Step 3.** Find a pair (M, V) in $S(n)$, such that $M \le w$.
17. If such a path is found in $S(n)$, return $M^* \le w$.
18. If such a path cannot be found in $S(n)$ return $M^* \ge w(1-\varepsilon)$

Proposition 3. The complexity of Test(w, ε) is $O(hn^2/\varepsilon)$,

Proof. Since the subinterval length is $\delta = \varepsilon w/n$, we have $O(n/\varepsilon)$ subintervals in the interval $[0, w]$. Therefore there are $O(n/\varepsilon)$ representative pairs in sets W and $S(k)$. Further, constructing each W in lines 10-12 requires $O(n/\varepsilon)$ elementary operations. Merging the sorted sets W and $S(k)$ in line 13, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is $O(n/\varepsilon)$). Step 2 (starting in line 6) goes over all the arcs n-1 times, so in total we have $O(nh)$ iterations of lines 10-12. Thus, the total complexity of Algorithm 2 is $O(hn^2/\varepsilon)$. □

The Narrowing Procedure BOUNDS

The narrowing procedure presented in this section (BOUNDS) is adapted from the procedure suggested by Ergun et al. [27] for solving the restricted shortest path. Specifically, when we run Test(w, ε), we choose ε as a function of UB/LB, updating its value from iteration to iteration. To distinguish the allowable error (ε) in the FPTAS from the iteratively changing error in the testing procedure, we denote the latter as θ. The algorithm proceeds as follows:

Algorithm 3. BOUNDS

1. Input: LB and UB such that $UB/LB \le n$.
2. Output: LB and UB such that $UB/LB \le 2$
3. If $UB/LB \le 2$, Goto 10
4. Set $\theta \leftarrow \sqrt{UB/LB} - 1$
5. Set $w \leftarrow \sqrt{LB \cdot UB/(1-\theta)}$
6. Run Test(w, θ)
7. If Test(w, θ) returns that $M^* \le w$ then set $UB \leftarrow w$
8. else set $UB \leftarrow w(1-\theta)$
9. Go to line 3
10. Return the improved LB and UB
11. End

The complexity of BOUNDS is O(hn^2). The proof is along the same line as that of Lemma 5 in [27].

D. Stage C: The ε-Approximation Algorithm (AA)

We start Stage C with *LB* and *UB* values satisfying $UB/LB \le 2$, and obtain an ε-approximation path.

Associate with each path *p* a pair (*M*, *V*), where, as above, $M = M(p)$ is the path expected time, and, correspondingly, $V = V(p)$ is the path variance. We deal with sets $S(k)$ of pairs (*M*, *V*) arranged in increasing order of the *M*-values so that every pair in $S(k)$ corresponds to a path from the start node *s* to a node *k*. As in DP, we delete all the dominated pairs in all the $S(k)$ sets. In addition to deleting the dominated pairs, we delete δ-close pairs as follows:

(a) In each $S(k)$, partition the interval [0, *UB*] into $\lceil (UB/LB)(n/\varepsilon) \rceil$ equal subintervals of size no greater than $\delta = \varepsilon LB/n$;

(b) If, for a given subinterval, there are multiple pairs from $S(k)$ for which the value of *M* falls into the subinterval, discard all such δ-close pairs, leaving only one representative pair in the subinterval, namely, the pair with the smallest (in this subinterval) *V*-coordinate.

(c) A pair (*M*, *V*) with $M > UB$ may be discarded.

The corresponding algorithm proceeds as follows:

Algorithm 4. ε-approximation algorithm (AA (*LB*, *UB*, ε))

1. Input: $G(N, A)$, $|N| = n$, $|A| = h$, $\{(m(i,j), v(i,j) \mid (i,j) \in A\}$, *R*
2. Input *UB*, *LB*, ε
3. $\Delta \leftarrow \varepsilon LB/n$
4. Output: ε-approximation path such that path expected time is at most (1+ ε)OPT
5. **Step 1.** [Initialization]
6. Set $S(1) = \{(0, 0)\}$, $S(k) \leftarrow \varnothing$ for $k = 2, \ldots, n$
7. **Step 2.** [Generate $S(2)$ to $S(n)$]
8. Repeat *n*-1 times
9. for each arc $(u, k) \in A$ (leading from node *u* to node *k*)
10. $W \leftarrow \varnothing$
11. for each pair $(M, V) \in S(u)$ do
12. if $M + m(u,k) + \phi^{-1}(P_R)\sqrt{V + v(u,k)} \le R$ then $W \leftarrow W \cup \{(M+m(u, k), V+v(u, k))\}$
13. endfor
14. $S(k) \leftarrow$ merge($S(k)$, *W*); during merging eliminate the dominated pairs and the δ-close pairs
15. endfor
16. End Repeat
17. **Step 3**. [Determine approximate solution]
18. find min *M* in $S(n)$, denote it by *ans*
19. Return *ans* as the ε-approximation expected time, use backtracking to find the path
20. The path's expected time is at most (1+ε)OPT.

Theorem 1. The complexity of AA(*LB*, *UB*, ε) is O(hn^2/ε). The complexity of the entire three-stage FPTAS is O(hn^2/ε).

Proof: Since the subinterval length is $\delta = \varepsilon LB/n$, we have $O(n(UB/LB)(1/\varepsilon))$ subintervals in interval [0, *UB*], and since $UB/LB \le 2$, there are O(n/ε) subintervals in the interval [*LB*, *UB*]. Therefore, there are O(n/ε) representative pairs in any set *W*, *T*, and $S(k)$.

Constructing each *W* in lines 11–13 requires O(n/ε) elementary operations because *W* is constructed from a single $S(k)$. Merging the sorted sets *W* and *T* in line 14, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is O(n/ε)). In Step 2 we have O(nh) iterations of lines 11–13. Thus, the total complexity of Algorithm 4 is O(hn^2/ε). Since Step C dominates Steps A and B of the algorithm, the complexity of the entire approximation algorithm is O(hn^2/ε). □

VI. Concluding Remarks

The main contribution of this work is a novel routing scheme for mobile agents in a wireless stochastic network that optimizes agent performance and reduces possible delays. An auxiliary dynamic programming algorithm running in pseudo-polynomial time is proposed for developing a fast routing strategy.

Notably, algorithm complexity is thoroughly analyzed. The mathematical model and algorithms presented in this paper can serve as a prototype for future commercial protocols for mobile agent routing over stochastic networks.

Future research should focus on developing more realistic models and solution algorithms that incorporate a broader variety of the practical characteristics of real-world computer and communication networks.

References

[1] A. Elalouf, E. Levner, and T. C. E. Cheng, "Efficient routing of mobile agents for agent-based integrated enterprise management: A general acceleration technique," *Lecture Notes in Business Information Processing*, vol. 88, pp. 1–20, 2011.

[2] N. R. Jennings and M. J. Wooldridge, "Applications of Intelligent Agents," in *Agent Technology: Foundations, Applications, and Markets*, N. R. Jennings, M. J. Wooldridge, Eds., Heidelberg: Springer, 1998, pp. 3–28.

[3] W. Shen, D. H. Norrie, and J.-P. Barthes, *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*. London: Taylor and Francis, 2001.

[4] W. Shen, D. Xue, and D. H. Norrie, "An agent-based manufacturing enterprise infrastructure for distributed integrated intelligent manufacturing systems," in *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agents*, London, UK, 1997, pp. 1–16.

[5] W. Shen, "Distributed manufacturing scheduling using intelligent agents," *IEEE Intelligent Systems*, vol. 17, pp. 88–94, 2002.

[6] Y. Peng, T. Finin, Y. Labrou, B. Chu, J. Long, X. Tolone, and A. Boughannam, "A multi-agent system for enterprise integration," in *Proc. of PAAM'98*, London, UK, 1998, pp. 155–169.

[7] J. E. White, *Telescript Technology: The Foundation for the Electronic Marketplace*, White Paper, Mountain View, CA, USA: General Magic, Inc., 1994.

[8] D. Tsichritzis, *Objectworld, Office Automation*. Heidelberg: Springer-Verlag, 1985.

[9] W. Shen, Q. Hao, H. J. Yoon, and D. H. Norrie, "Applications of agent-based systems in intelligent manufacturing: An updated review," *Advanced Engineering Informatics*, vol. 20, pp. 415–431, 2006.

[10] T. Papaioannou, *Using Mobile Agents to Improve the Alignment between Manufacturing and Its IT Support Systems, Robotics and Autonomous Systems*. Amsterdam: Elsevier, 1999.

[11] W. Shen, F. Maturana, and D. H. Norrie, "MetaMorph II: An agent-based architecture for distributed intelligent design and manufacturing," *Journal of Intelligent Manufacturing*, vol. 11, pp. 237–251, 2000.

[12] E. Camponogara and R. B. Shima, "Mobile agent routing with time constraints: A resource constrained longest-path approach," *Journal of Universal Computer Science*, vol. 16, pp. 372–401, 2010.

[13] A. Elalouf and E. Levner, "General techniques for accelerating FPTAS for the routing and knapsack problems," in *Abstract Book, Annual Meeting 2011 of Operations Research Society of Israel (ORSIS 2011)*, Akko, Israel, 2011, p. 14.

[14] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol. 17, 36–42, 1992.

[15] B. Brewington, R. Gray, K. Moizumi, D. Kotz, G. Cybenko, and D. Rus, "Mobile agents in distributed information retrieval," in *Intelligent Information Agents*, M. Klusch, Ed., Heidelberg: Springer Verlag, 1999, pp. 355–395.

[16] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis, "Efficient computation of delay-sensitive routes from one source to all destinations," in *IEEE Infocom'2001*, Washington, DC: IEEE Press, 2001, pp. 854–858.

[17] G. Xue, A. Sen, W. Zhang, J. Tang, and K. Thulasiraman, "Finding a path subject to many additive QoS constraints," *IEEE Transactions on Networking*, vol. 15, pp. 201–211, 2007.

[18] Q. Wu, N. S. V. Rao, J. Barhen, S. S. Iyengar, V. K. Vaishnavi, H. Qi, and K. Chakrabarty, "On computing mobile agent routes for data fusion in distributed sensor networks," *IEEE Trans. Knowledge and Data Engineering*, vol. 16, pp. 740–753, June 2004.

[19] R. Rajagopalan, C. K. Mohan, P. Varshney, and K. Mehrotra, "Multi-objective mobile agent routing in wireless sensor networks," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on 2–5 Sept. 2005*, vol. 2, 2005, pp. 1730–1737.

[20] T. Osman, W. Wagealla, and A. Bargiela, "An approach to rollback recovery of collaborating mobile agents," IEEE Trans. Systems, Man and Cybernetics, Part C, vol. 34, pp. 48–57, Feb 2004.

[21] L. Rech, R. S. Oliveira, and C. B. Montez, "Dynamic determination of the itinerary of mobile agents with timing constraints," in *Proc. IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, Compiegne, France, 2005, pp. 45–50.

[22] G. V. Gens and E. V. Levner, "Fast approximation algorithms for job sequencing with deadlines," *Discrete Applied Mathematics*, vol. 3, pp. 313–318, 1981.

[23] G. V. Gens and E. V. Levner, "Fast approximation algorithms for knapsack type problems," in *Lecture Notes in Control and Information Sciences*, vol. 23, Berlin: Springer Verlag, 1980.

[24] S. Sahni, "Algorithms for scheduling independent tasks," *Journal of the ACM*, vol. 23, pp .116–127, 1976.

[25] E. Levner, A. Elalouf, and T. C. E. Cheng, "An improved FPTAS for mobile agent routing with time constraints," *Journal of Universal Computer Science*, vol. 17, 1854–1862, 2011.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

[27] F. Ergun, R. Sinha, and L. Zhang, "An improved FPTAS for restricted shortest path," *Information Processing Letters*, vol. 83, pp. 287–291, 2002.

Differential Evolution for the Control Gain's Optimal Tuning of a Four-bar Mechanism

María Bárbara Calva-Yáñez, Paola Andrea Niño-Suárez, Miguel Gabriel Villarreal-Cervantes, Gabriel Sepúlveda-Cervantes, and Edgar Alfredo Portilla-Flores

***Abstract*—In this paper the variation of the velocity error of a four-bar mechanism with spring and damping forces is reduced by solving a dynamic optimization problem using a differential evolution algorithm with a constraint handling mechanism. The optimal design of the velocity control for the mechanism is formulated as a dynamic optimization problem. Moreover, in order to compare the results of the differential evolution algorithm, a simulation experiment of the proposed control strategy was carried out. The simulation results and discussion are presented in order to evaluate the performance of both approaches in the control of the mechanism.**

***Index Terms*—Velocity control, differential evolution algorithm, four-bar mechanism, dynamic optimization.**

I. Introduction

THE four-bar mechanism (FBM) is extensively used in several engineering applications [1], [2], [3]. This is due to the topological simplicity, functional versatility and because this mechanism can generate a cyclic trajectory (path generation). Hence, the four-bar mechanism has been widely studied in the last decades. The path generation of the four-bar mechanism is achieved by using analytical, numerical and graphical methods [4]–[6]. Nevertheless, the statement of optimization problems to increase the number of precision points and the tracking precision are been used in the path generation of the FBM [7], [8].

In the analysis and design of the FBM the main assumption considers that the angular velocity of the actuator is constant. Nevertheless, it is not always fulfilled, if an electric motor drives the crank. For example, when the crank rotates, the center of mass of the FBM may move. The change of the inertia of the FBM yield an external load to the motor such that the angular velocity of the crank is not constant. Hence, it is important to select the appropriate control system that guarantee an uniform and efficient regulation of the angular velocity.

Manuscript received on February 25, 2013; accepted for publication on May 23, 2013.

M. B. Calva-Yáñez, M. G. Villarreal-Cervantes, G. Sepúlveda-Cervantes, and E. A. Portilla-Flores are with the Instituto Politécnico Nacional, CIDETEC, Mechatronic Section, Postgraduate Department, Juan de Dios Bátiz s/n, C.P. 07700 D.F., Mexico (e-mail: b_calva@hotmail.com; {mvillarrealc, gsepulveda, aportilla}@ipn.mx).

P. A. Niño-Suárez is with the Instituto Politécnico Nacional, ESIME-AZC, SEPI Section, Las Granjas 682, C.P. 02250 D.F., Mexico (e-mail: pninos@ipn.mx).

There exist several advanced control strategy that may guarantee robustness in the angular velocity, such as robust control [9], adaptive control [10], etc. Nevertheless, from an industrial point of view, PID controllers can provide a good performance if the gains are correctly tuned in.

In this paper, the modified PID controller presented in [11] is used to regulate the angular velocity of a four-bar mechanism with spring and damping forces (FBM-SDF). The optimal PID control gains is found by considering a dynamic optimization problem and by using a constraint handling mechanism in the differential evolution algorithm to solve it. The effectiveness of the algorithm is shown in simulation results.

The paper is organized as follows: Section II presents the coupled dynamics of the four-bar mechanism with DC motor. Section III presents the control strategy for the system. In Section IV, the dynamic optimization problem is stated for finding the optimal control gains. The constraint handling differential evolution algorithm is show in Section V. The simulation results and discussion are given in Section VI and finally, the conclusions are drawn in Section VII.

II. Dynamic Model

The four-bar mechanism with spring and damping forces (FBM-SDF) has one degree of freedom (*dof*) in the crank (link L_2). This *dof* is actuated by a DC motor. The schematic representation of the mechanism is shown in Fig. 1. The mass, the inertia, the length, the mass center length and the mass center angle of the i-th link are represented by $m_i, J_i, L_i, r_i, \phi_i$, respectively. The angle of the i-th link with respect to the base frame (X-Y) is named as θ_i. The stiffness constant of the spring and the damping coefficient of the damper are represented by k and C.

The kinematic analysis [5] of the FBM-SDF is required to obtain the angular velocity $\dot{\theta}_i \ \forall i = 2, 3, 4$ and the linear velocity v_{ix}, v_{iy} of the mass center of the i-th link with respect to the inertial frame. The angular and the linear velocity is described in (1)-(3).

$$\dot{\theta}_i = \gamma_i \dot{\theta}_2 \tag{1}$$

$$v_{ix} = \alpha_i \dot{\theta}_2 \tag{2}$$

$$v_{iy} = \beta_i \dot{\theta}_2 \tag{3}$$

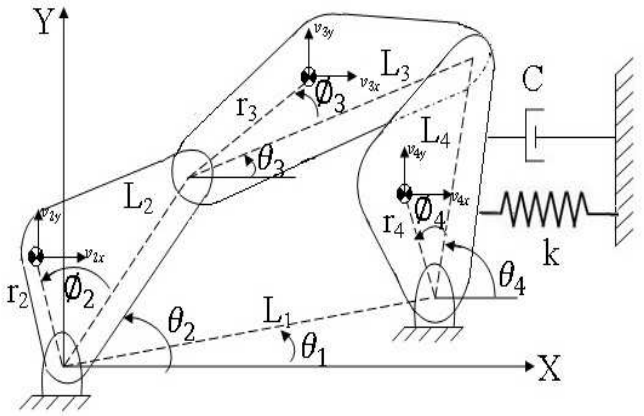


Fig. 1. Four-bar mechanism with spring and damping forces

where:

$$\alpha_2 = -r_2 \sin(\theta_2 + \phi_2) \tag{4}$$

$$\alpha_3 = -L_2 \sin\theta_2 - r_3\gamma_3 \sin(\theta_3 + \phi_3) \tag{5}$$

$$\alpha_4 = -r_4\gamma_4 \sin(\theta_4 + \phi_4) \tag{6}$$

$$\beta_2 = r_2 \cos(\theta_2 + \phi_2) \tag{7}$$

$$\beta_3 = L_2 \cos\theta_2 - r_3\gamma_3 \cos(\theta_3 + \phi_3) \tag{8}$$

$$\beta_4 = -r_4\gamma_4 \cos(\theta_4 + \phi_4) \tag{9}$$

$$\gamma_2 = 1 \tag{10}$$

$$\gamma_3 = \frac{L_2 \sin(\theta_4 - \theta_2)}{L_3 \sin(\theta_3 - \theta_4)} \tag{11}$$

$$\gamma_4 = \frac{L_2 \sin(\theta_3 - \theta_2)}{L_3 \sin(\theta_3 - \theta_4)} \tag{12}$$

Defining the Lagrangian function $\tilde{L}$ (13), where K and P is the kinetic and potential energy, respectively.

$$\tilde{L} = K - P \tag{13}$$

where:

$$K = \sum_{i=2}^{4} \left(\frac{1}{2} m_i \left(v_{ix}^2 + v_{iy}^2 \right) + \frac{1}{2} J_i \dot{\theta}_i^2 \right) = \frac{1}{2} A\left(\theta_2\right) \dot{\theta}_2^2 \tag{14}$$

$$P = \frac{1}{2} k \left(\theta_4 - \theta_{4,0}\right)^2 \tag{15}$$

$$A\left(\theta_2\right) = \sum_{i=2}^{4} \left(m_i \left(\alpha_i^2 + \beta_i^2 \right) + \gamma_i^2 J_i \right) \tag{16}$$

Using θ_2 as the generalized coordinate and following the methodology described in [11], the Euler-Lagrange formulation [12] which described the dynamic model of the FBM-SDF is given by (17), where D is the Rayleigh's dissipation function and $\theta_{4,0}$ is the angular position of the link 4 when the spring is in equilibrium.

$$T = \frac{d}{dt}\left(\frac{\partial \tilde{L}}{\partial \dot{\theta}_2}\right) - \frac{\partial \tilde{L}}{\partial \theta_2} + \frac{\partial D}{\partial \dot{\theta}_2} \tag{17}$$

where:

$$D = \frac{1}{2} C \dot{\theta}_4^{\,2} \tag{18}$$

The total and partial derivatives of (17) is given by (19).

$$T = A\left(\theta_2\right) \ddot{\theta}_2 + \frac{1}{2} \frac{dA\left(\theta_2\right)}{d\theta_2} \dot{\theta}_2^2 + k\gamma_4 \left(\theta_4 - \theta_{4,0}\right) + C\gamma_4^2 \dot{\theta}_2 \tag{19}$$

where:

$$A\left(\theta_2\right) = C_0 + C_1\gamma_3^2 + C_2\gamma_4^2 + C_3\gamma_3 \cos\left(\theta_2 - \theta_3 - \phi_3\right) \tag{20}$$

$$\begin{aligned} \frac{dA\left(\theta_2\right)}{d\theta_2} &= 2C_1\gamma_3 \frac{d\gamma_3}{d\theta_2} + 2C_2\gamma_4 \frac{d\gamma_4}{d\theta_2} \\ &+ C_3 \frac{d\gamma_3}{d\theta_2} \cos\left(\theta_2 - \theta_3 - \phi_3\right) \\ &- C_3\gamma_3 \left(1 - \gamma_3\right) \sin\left(\theta_2 - \theta_3 - \phi_3\right) \end{aligned} \tag{21}$$

$$C_0 = J_2 + m_2 r_2^2 + m_3 L_2^2 \tag{22}$$

$$C_1 = J_3 + m_3 r_3^2 \tag{23}$$

$$C_2 = J_4 + m_4 r_4^2 \tag{24}$$

$$C_4 = 2m_3 L_2 r_3 \tag{25}$$

$$\frac{d\gamma_3}{d\theta_2} = \frac{L_2 \left(D_1 + D_2\right)}{L_3 \sin^2\left(\theta_3 - \theta_4\right)} \tag{26}$$

$$\frac{d\gamma_4}{d\theta_2} = \frac{L_2 \left(D_3 + D_4\right)}{L_4 \sin^2\left(\theta_3 - \theta_4\right)} \tag{27}$$

$$D_1 = \left(\gamma_4 - 1\right) \sin\left(\theta_3 - \theta_4\right) \cos\left(\theta_4 - \theta_2\right) \tag{28}$$

$$D_2 = \left(\gamma_4 - \gamma_3\right) \sin\left(\theta_4 - \theta_2\right) \cos\left(\theta_3 - \theta_4\right) \tag{29}$$

$$D_3 = \left(\gamma_3 - 1\right) \sin\left(\theta_3 - \theta_4\right) \cos\left(\theta_3 - \theta_2\right) \tag{30}$$

$$D_4 = \left(\gamma_4 - \gamma_3\right) \sin\left(\theta_3 - \theta_2\right) \cos\left(\theta_3 - \theta_4\right) \tag{31}$$

In order to model the full dynamics of the FBM-SDF, the dynamic of the actuator [13] must be included. A schematic diagram of the DC motor is represented in Fig. 2, where L and R represent the inductance and the armature resistance, $i(t)$ and $V_{in}(t)$ are the current and voltage input, respectively. J and B is the inertia moment and the friction coefficient of the output shaft. T_L, T_m and V_b is the load torque, the magnetic motor torque and the Back electromotive force of the motor, respectively. The motor constant is represented by K_f and the constant of the back electromotive force is represented by K_b.

The dynamic model of the DC motor [14] consists on modeling the electrical and mechanical parts. Using Kirchhoff's second law, the closed loop circuit of Fig. 2 can be written as (32).

$$L\frac{di\left(t\right)}{dt} + Ri\left(t\right) = V_{in}\left(t\right) - K_b\dot{\theta}_1 \tag{32}$$

By using the Newton's second law in the mechanical part of the DC motor, the equation (33) is obtanied, where T_a and

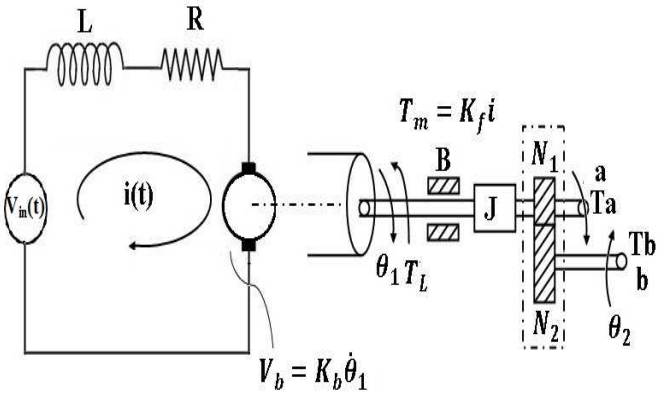


Fig. 2. Schematic diagram of a DC motor

T_b is the output torque of the shaft a and b, respectively (see Fig. 2).

$$T_m - B\dot{\theta}_1 - T_a - T_L = J\ddot{\theta}_1 \tag{33}$$

The mechanical transmission among the two gears in the shafts is expressed in (34), where r_i and $N_i \ \forall \ i = 1, 2$ is the radius and the number of teeth of the gears.

$$\frac{T_b}{T_a} = \frac{\dot{\theta}_1}{\dot{\theta}_2} = \frac{r_2}{r_1} = \frac{N_2}{N_1} = n \tag{34}$$

Substituting T_a from (33) to (34), the torque applied to the mechanical system is written as (35).

$$T_b = n\left(T_m - T_L - B\dot{\theta}_1 - J\ddot{\theta}_1\right) \tag{35}$$

Using the relation $\dot{\theta}_1 = n\dot{\theta}_2$ in (34), $T_m = K_f i$ and $T_L = 0$, the dynamic equation of the DC motor is given by (36)-(37).

$$T_b = nK_f i\,(t) - n^2 B\dot{\theta}_2 - n^2 J\ddot{\theta}_2 \tag{36}$$

$$L\frac{di\,(t)}{dt} + Ri\,(t) = V_{in}\,(t) - nk_b\dot{\theta}_2 \tag{37}$$

Hence, the coupled dynamics of the DC motor with the FBM-SDF is given by combining (36), (37) and (19). Let the state variable vector $x = [x_1, x_2, x_3]^T = [\theta_2, \dot{\theta}_2, i]^T$ and the input vector $u = V_{in}$, the coupled dynamics in a state space representation of the DC motor with the FBM-SDF is given by (38).

$$\begin{aligned}\dot{x} &= f(\vec{x}, u(t), t) \\ &= \begin{bmatrix} x_2 \\ A_0\left[A_1 x_2^2 + A_2 x_2 + nK_f x_3 + A_3\right] \\ \frac{1}{L}\left(u\,(t) - nK_b x_2 - Rx_3\right) \end{bmatrix}\end{aligned} \tag{38}$$

where:

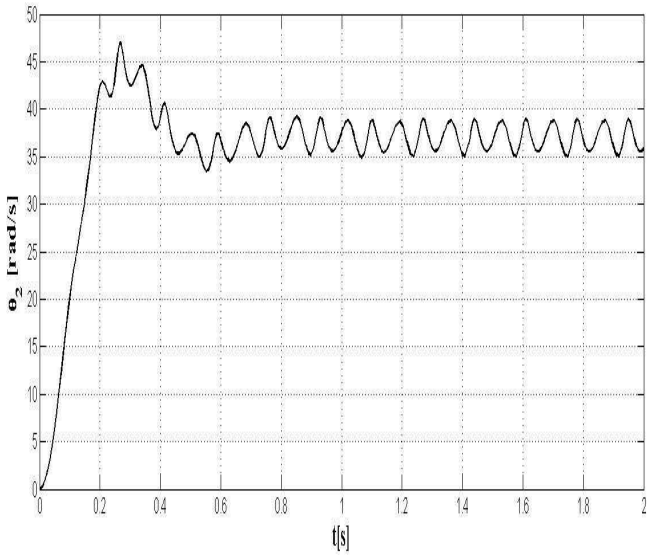


Fig. 3. Input angular velocity $\dot{\theta}_2$ of the FBM-SDF without a control strategy

$$A_0 = \frac{1}{A\,(x_1) + n^2 J1} \tag{39}$$

$$A_1 = -\frac{1}{2}\frac{dA\,(x_1)}{dx_1} \tag{40}$$

$$A_2 = -\left(C\gamma_2^4 + n^2 B\right) \tag{41}$$

$$A_3 = -k\gamma_4\,(\theta_4 - \theta_{4,0}) \tag{42}$$

III. Control Strategy

In the synthesis of mechanism, the main assumption is to consider the input velocity as a constant. Nevertheless, this can no be ensured without a closed loop control system. In Fig. 3 shows the behavior of the input angular velocity $\dot{\theta}_2$ when a constant voltage of 30 $Volts$ is applied. It is observed that the input angular velocity is not constant. This is true because the four-bar mechanism presents dead-centre positions and it adds uncertain loads in the crank.

Based on the work of Tao and Sadler [11], the proposed control strategy is used in this paper. This controller is stated as in (43), where K_p, K_I and K_D is the proportional (P), integral (I) and derivative (D) gains, respectively. The velocity error and its derivative are represented by $e(t) = \dot{\theta}_2^d - \dot{\theta}_2$ and $\dot{e}(t) = -\ddot{\theta}_2$, where $\dot{\theta}_2^d$ is the constant desired velocity.

$$u\,(t) = K_p e\,(t)\int_0^t \dot{\theta}_2^d dt + K_I \int_0^t e\,(t)\,dt + K_D\dot{e}\,(t) \tag{43}$$

IV. Dynamic Optimization Problem to Find the Optimum Controller Gains for Constant Input Velocity of the FBM-SDF

The dynamic optimization problem [15] consist on finding the optimum design variables $\vec{p} \in R^3$ such that minimize the

objective function (44) subject to the closed-loop system of the FNM-SDF (45) with the initial state vector x_0, inequalities constraints (48) and bounds in the design variable (49).

$$\min_{\overrightarrow{p}\,\epsilon R^3} F(\overrightarrow{p}) \tag{44}$$

subject to:

$$\dot{x} = f(\overrightarrow{x}, u(\vec{p}, t), t) \tag{45}$$

$$u(t) = K_p e(t) \int_0^t \dot{\theta}_2^d dt + K_I \int_0^t e(t)\, dt + K_D \dot{e}(t) \tag{46}$$

$$\overrightarrow{x}(0) = x_0 \tag{47}$$

$$g(\overrightarrow{x}) \leq 0 \tag{48}$$

$$p_{i,\min} \leq \overrightarrow{p} \leq p_{i,\max} \tag{49}$$

In the next subsections, variables, functions and all parts that involve the dynamic optimization problem (DOP) are described.

A. Design Variable Vector

The design variable vector $\vec{p} = [K_p, K_D, K_I]^T \in R^3$ includes the gains of the modified PID controller.

B. Objective Function

The variation of the input velocity of the crank is chosen as the objective function in the optimization problem. This is an important issue due to a bad selection of the PID gains, the input velocity of the crank could be considerably affected. The objective function is written as in (50), where $Max()$ and $Min()$ is the maximum and minimum value of the input velocity presented in the time interval $[0, t_f]$.

$$F(\vec{p}) = |Max(x_2(t)) - Min(x_2(t))|; t \in [0, t_f] \tag{50}$$

C. Constraints

The first constraint (45) is the solution of the differential equation of the dynamic model of the FBM-SDF choosing x_0 as the initial condition. This constraints provide the dynamic behavior of the system in the optimization problem.

The inequality constraints consist on establishing that the rise time t_r of the angular velocity of the crank $\dot{\theta}_2(t)$ is less than $0.1s$ and the overshoot does not exceed of 1.7% of the desired angular velocity $\dot{\theta}_2^d$. These constraints is stated as in (51) and (52), respectively.

$$g_1 : t_r \leq 0.1s \tag{51}$$

$$g_2 : \dot{\theta}_2(t_r) \leq \dot{\theta}_2^d + 0.017\dot{\theta}_2^d \tag{52}$$

The bounds in the design variable vector are defined by $\vec{p}_{i,min}$ and $\vec{p}_{i,max}$ $\forall\ i = 1, 2, 3$.

```
1   BEGIN
2     G = 0
3     Create a random population x⃗^i_G ∀i = 1, ..., NP
6     Evaluate F(x⃗^i_G), g(x⃗^i_G), ∀i = 1, ..., NP
7     Do
8       For i = 1 to NP Do
9       Select randomly {r1 ≠ r2 ≠ r3} ∈ x⃗_G.
10      j_rand =randint(1, D)
11        For j = 1 to D Do
12          If (rand_j[0, 1) < CR or j = j_rand) Then
13            u^i_{j,G+1} = x^{r1}_{j,G} + F(x^{r2}_{j,G} − x^{r3}_{j,G})
14          Else
15            u^i_{j,G+1} = x^i_{j,G}
16          End If
17        End For
18        Evaluate F(u⃗^i_{G+1}), g(u⃗^i_{G+1})
19        If (g(u⃗^i_{G+1}) = 0 and g(x⃗^i_G)=0) Then
20          If (F(u⃗^i_{G+1}) < F(x⃗^i_G)) Then
21            x⃗^i_{G+1} = u⃗^i_{G+1}
22          Else
23            x⃗^i_{G+1} = x⃗^i_G
24          End If
25        Else If (g(u⃗^i_{G+1}) < g(x⃗^i_G)) Then
26          x⃗^i_{G+1} = u⃗^i_{G+1}
27        Else
28          x⃗^i_{G+1} = x⃗^i_G; End If
29        End If
30      End For
31      G = G + 1
32    While (G ≤ G_Max)
```

Fig. 4. CHDE algorithm

V. Differential Evolution Algorithm

In the last decades, the use of heuristic techniques have been used in engineering problems [16], [17]–[19]. This is due to the increment of the technological advances and because problems are non-convex, discontinuous and/or present discrete variables that make it difficult (or imposible) to solve them by traditional optimization techniques such as nonlinear programming techniques.

In this work, the differential evolution (DE) algorithm [20] with a constraint-handling mechanism [17] is used to solve the dynamic optimization problem. The constraint-handling differential evolution (CHDE) algorithm is shown in Fig. 4. The constraint handling mechanism is included in the selection operation between the trial vector $\vec{u}^i_{G+1}$ and the target vector $\vec{x}^i_G$ in order to remain one of them in the population for the next generation. This mechanism consists on passing the best individual between them for the next generation (elitism). The best individual is the individual without constraint violation and with less or equal objective function value or when both individuals are unfeasible, the best individual is the one with less constraint violation (see line 19 and 25 of Fig. 4).

For more details of the algorithm consult [20] and [17].

VI. Simulation Results

The simulation results consist on using the CHDE algorithm in the dynamic optimization stated above. Four parameters

in the CHDE algorithm must be chosen. In this case, the population size NP consists of 100 individuals. The scaling factor F and the crossover constant CR are randomly generated in the interval $F \in [0.3, 0.9]$ at each generation, and in the interval $CR \in [0.8, 1)$ at each optimization process. The stop criterion is when the number of generations is fulfilled $G_{Max} = 200$.

The CHDE algorithm is programmed in Matlab Release 7.9 on a Windows platform. Computational experiments were performed on a PC with a 1.83 GHz Core 2 Duo processor and 2 GB of RAM. Ten independent runs of the CHDE algorithm are performed.

On the other hand, in order to solve the dynamic optimization problem (44)-(49), the closed-loop system (45) must be solved numerically. Hence, the Runge-Kutta method (RKM) is used to solved it, with initial condition chosen as $x_0 = [0, 0, 0]^T$, with a desired velocity selected as $\dot{\theta}_2^d = 30$ rad/s and with the kinematic and dynamic parameters of the coupled dynamics proposed as in Table I.

The bound of the design variable vector is defined as $\vec{p}_{i,min} = 0.1$, $\vec{p}_{i,max} = 50 \ \forall \ i = 1, 2, 3$.

All runs of the algorithm converge to the optimum design variable vector $\vec{p}^* = [50, 16.1881, 1.4394]^T$ with a performance function value of $F(\vec{p}^*) = 0.2389$. This means that local solutions are not found by the algorithm and the found solution can be considered as the global one. The mean of the time spends to converge the algorithm is ten minutes.

However, in order to compare the behavior of the system performance with the optimum design variable vector $\vec{p}^*$, the behavior of the system performance with PID gains obtained by a trial and error procedure is carried out. Such tuning is called experimental tuning in this paper. The experimental tuning considers the bounds $\vec{p}_{i,min}$ and $\vec{p}_{i,max}$.

In general, the design of a PID controller of linear system is broadly studied [21]. Nevertheless, the design of a PID controller of non-linear systems is not a trivial task. Tuning of a PID controller by using bifurcation theory is used for non-linear system [22]. From the feedback control strategy proposed in the closed loop system, the choice of the controller gains is realized so that ensures the desired convergence. The closed loop stability of the proposed strategy is stated by considering the convergence of the tracking errors.

The experimental tuning procedure is done by keeping in mind that the higher the proportional gain the lower the speed fluctuation and the steady-state error. On the other hand, excessively high proportional gains may lead to a large amount of overshoot if the derivative gain is not large enough. Additionally, increasing the derivative gain will decrease the overshoot, but the system response will be slower during the start-up period. The found gains need to fulfill the estimated performance, overshoot$\leq 1.5\%$, steady-error$\leq 1.0\%$ and rise time≤ 0.1 second. The resulting design variable vector with the experimental tuning is $\vec{p}^*_{et} = [45.55, 5.25, 1]^T$ with a performance function value of $F(\vec{p}^*_{et}) = 0.2702$.

TABLE I
PARAMETERS OF THE FBM-SSDF AND THE DC MOTOR

FBM-SDF's parameters		
$L_1 = 0.5593[m]$	$J_2 = 0.00071\,[kg\,m^2]$	$m_2 = 1.362\,[kg]$
$L_2 = 0.102[m]$	$J_3 = 0.0173\,[kg\,m^2]$	$m_3 = 1.362\,[kg]$
$L_3 = 0.610[m]$	$J_4 = 0.00509\,[kg\,m^2]$	$m_4 = 0.2041\,[kg]$
$L_4 = 0.406[m]$	$\phi_2 = \phi_3 = \phi_4 = 0\,[rad]$	
$r_2 = 0\,[m]$	$r_3 = 0.305\,[m]$	$r_4 = 0.203\,[m]$
Motor's parameters		
$R = 0.4\,[\Omega]$	$L = 0.05[H]$	$K_f = 0.678\,[Nm/A]$
$K_b = 0.678[V\ s]$	$B = 0.226[Nms]$	$J = 0.056[kgm^2]$

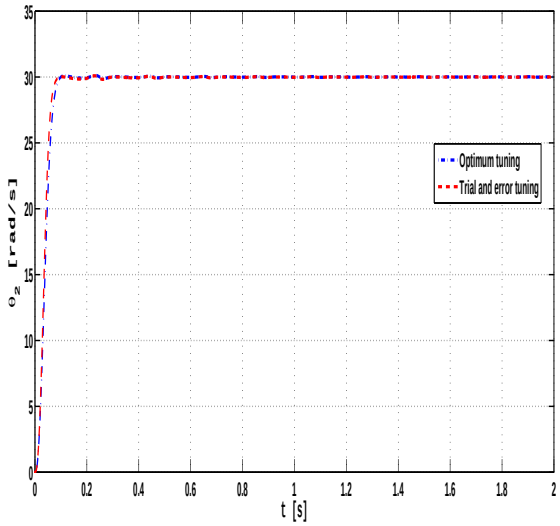


Fig. 5. Angular velocity of the crank with both tuning approaches: the optimum and experimental tuning

It is important to remark that in the experimental tuning procedure, several possible solutions were obtained,but they were not feasible from the optimization problem point of view. After several trials, we finally find the vector $\vec{p}^*_{et}$ which fulfill the constraints in the optimization problem.

In Fig. 5, the angular velocity of the crank with both tuning approaches is shown. It is observed that in the optimum tunning, the angular velocity presents a deviation of 0.79% from the desired angular velocity. Also, the angular velocity deviation on the second case was 0.9%. Finally, the rise time by each one of the approaches were $0.1s$ and less of $0.1s$, respectively. This indicates that the constraints in the dynamic optimization problem are satisfied.

On the other hand, the behavior of the control signal with the optimum design variable vector and with the experimental tuning is shown in Fig. 6. As it is observed, the control signal of the optimum approach has a lower overshoot than the second approach to reach the reference value of 30 rad/s. This implies greater energy consumption by using the gains of the experimental tuning. Also, both approaches produce a control signal which compensates the nonlinear loads in order to reduce the angular velocity variation.

In Fig. 7, a zoom of the angular velocity of the crank

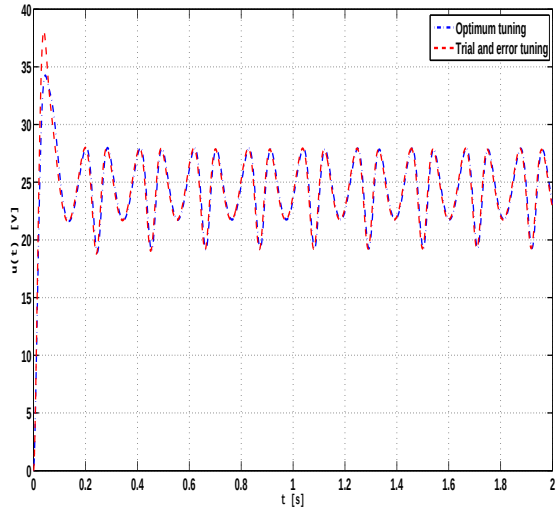


Fig. 6. Control signal dynamic behavior with both approaches

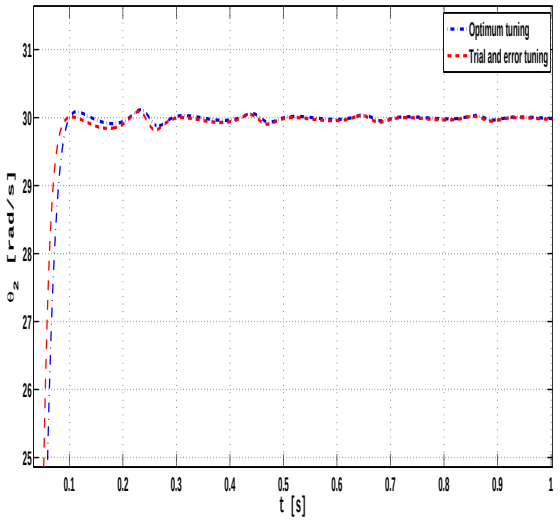


Fig. 7. Angular velocity of the crank between the time period of $0.05s$ and $1s$ with both approaches

with both approaches in the time period between $0.05s$ and $1s$ is shown. It is observed that in the experimental tuning procedure, the rise time of the angular velocity is less than the optimum approach. However, the steady state behavior of the optimum approach is most softly than the experimental tuning procedure.

It is important to comment the although both approaches produce good results, the best of them is the optimal one. In addition, the CHDE algorithm is successfully applied to tuning the PID controller without requiring a priori knowledge of the system and in the experimental tuning procedure is necessary this knowledge.

VII. Conclusion

In this paper, the optimal gains of a PID controller for a four-bar mechanism with spring and damping forces is found by using a differential evolution algorithm with a constraint handling mechanism. In order to compare the performance of the system with the optimum control, an alternative experimental tuning procedure is carried out. The variation of the crank's velocity error for a four-bar mechanism with spring and damping forces is reduced by using both approaches. In addition, the rise time and the overshoot of the velocity signal are limited to be in an closed interval. However, simulation results of the closed-loop system show that the found optimal gains provide a better performance than the gains obtained by experimental tuning procedure.

Finally, the main advantage of using the differential evolution algorithm with a constraint handling mechanism for finding the optimum PID gains is that it does not require a priori knowledge of the system and it is easy to program it. Therefore the CHDE algorithm is becoming more used to solve this kind of nonlinear and discontinuous problems.

Further research involves the redesign of the structural and controller parameters considering the dynamic model and by using alternative evolutive algorithms.

Acknowledgements

The first author acknowledges support from CONACYT through a scholarship to pursue graduate studies at Instituto Politécnico Nacional. Authors acknowledge support from CONACyT through grant No. 182298 and support from COFAA and SIP of the Intituto Politécnico Nacional through grant No. 20131053 and 20131350.

References

[1] R. Madangopal, Z. A. Khan, and S. K. Agrawal, "Biologically inspired design of small flapping wing air vehicles using four-bar mechanisms and quasi-steady aerodynamics," *Journal of Mechanical Design*, vol. 127, pp. 809–815, 2005.

[2] M. E. Alfaro, D. I. Bolnick, and P. C. Wainwright, "Evolutionary dynamics of complex biomechanical systems: an example using the four-bar mechanism," *Evolution, International Journal of Organic Evolution*, vol. 58, no. 3, pp. 495–503, 2004.

[3] M. W. Westneat, "Feeding mechanics of teleost fishes (labridae; perciformes): A test of four-bar linkage models," *Journal of Morphology*, vol. 205, no. 3, pp. 269–295, 1990.

[4] V. Parlakta, E. Söylemez, and E. Tanik, "On the synthesis of a geared four-bar mechanism," *Mechanism and Machine Theory*, vol. 45, no. 8, pp. 1142–1152, 2010.

[5] J. E. Shigley and J. J. J. Uicker, *Theory of machines and mechanism*. Mc. Graw Hill, 1995.

[6] J. Hrones and G. Nelson, *Analysis of Four Bar Linkage*. MIT Press and Wiley, 1951.

[7] M. Khorshidi, M. Soheilypour, M. Peyro, A. Atai, and M. S. Panahi, "Optimal design of four-bar mechanisms using a hybrid multi-objective ga with adaptive local search," *Mechanism and Machine Theory*, vol. 46, no. 10, pp. 1453–1465, 2011.

[8] N. Nariman-Zadeh, M. Felezi, A. Jamali, and M. Ganji, "Pareto optimal synthesis of four-bar mechanisms for path generation," *Mechanism and Machine Theory*, vol. 44, no. 1, pp. 180–191, 2009.

[9] R. A. Freeman and P. V. Kokotovic, *Robust Nonlinear Control Design: State-Space and Lyapunov Techniques*. Birkhäuser, 2008.

[10] J.-J. Slotine and W. Li, *Applied Nonlinear Control*. Prentice Hall, 1991.

[11] J. Tao and J. P. Sadler, "Constant speed control of a motor driven mechanism system," *Mechanism and Machine Theory*, vol. 30, no. 5, pp. 737–748, 1995.
[12] D. T. Greenwood, *Classical Dynamics*. Dover Publications, INC, 1997.
[13] G. Tao and P. V. Kokotovic., *Adaptive control of systems with actuator and sensor nonlinearities*. John Wiley and Sons, 1996.
[14] J. Chiasson, *Modeling and High Performance Control of Electric Machines*. Wiley-IEEE Press, 2005.
[15] W. Hong, S. Wang, P. Li, G. Wozny, and L. T. Biegler, "A quasi-secuential approach to large-scale dynamic optimization problems," *American Institute of Chemical Engineers*, vol. 52, no. 1, pp. 255–268, 2005.
[16] M. G. Villarreal-Cervantes, C. A. Cruz-Villar, J. Alvarez-Gallegos, and E. A. Portilla-Flores, "Differential evolution techniques for the structure-control design of a five-bar parallel robot," *Engineering Optimization*, vol. 42, no. 6, pp. 535–565, 2010.
[17] E. A. Portilla-Flores, E. Mezura-Montes, J. Alvarez-Gallegos, C. A. Coello-Coellod, C. A. Cruz-Villar, and M. G. Villarreal-Cervantes, "Parametric reconfiguration improvement in non-iterative concurrent mechatronic design using an evolutionary-based approach," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 5, pp. 757–771, 2011.
[18] M. G. Villarreal-Cervantes, C. A. Cruz-Villar, J. Álvarez Gallegos, and E. A. Portilla-Flores, "Kinematic dexterity maximization of an omnidirectional wheeled mobile robot: A comparison of metaheuristic and sqp algorithms," *International Journal of Advanced Robotic Systems*, vol. 9, no. 161, pp. 1–12, 2012.
[19] M. G. Villarreal-Cervantes, C. A. Cruz-Villar, J. Alvarez-Gallegos, and E. A. Portilla-Flores, "Robust structure-control design approach for mechatronic systems," *IEEE/ASME Transactions on Mechatronics*, 2013 On line version DOI: 10.1109/TMECH.2012.2208196.
[20] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: A practical approach to global optimization*. Springer, December 2005.
[21] I. Gyongy and D. Clarke, "On the automatic tuning and adaptation of pid controllers," *Control Engineering Practice*, vol. 14, no. 1, pp. 149–163, 2006.
[22] M. P. Polo, P. Albertos, and J. Ángel Berná Galiano, "Tuning of a pid controlled gyro by using the bifurcation theory," *Systems & Control Letters*, vol. 57, no. 1, pp. 10–17, 2008.

Patrones de implementación para incluir comportamientos proactivos

Mailyn Moreno, Alternán Carrasco, Alejandro Rosete y Martha D. Delgado

***Resumen*—La programación orientada a objeto enfrenta retos como es el desarrollo de software en ambientes distribuidos. En esta línea ha surgido el paradigma de agentes. Un agente exhibe comportamientos que lo diferencia de un objeto, como la autonomía y la proactividad. La proactividad permite desarrollar sistemas dirigidos por metas, en los que no es necesaria una petición para que se inicie un trabajo. Incorporar proactividad a un software es hoy una necesidad, existe una gran dependencia de los sistemas computarizados y es mayor la delegación de tareas en ellos. Los patrones se han utilizado con éxito en la reducción de tiempo de desarrollo y el número de errores en el desarrollo de software, además de ser una guía para resolver un problema típico. En este trabajo se presentan dos patrones de implementación para incorporar proactividad en un software y facilitar el trabajo con los agentes. Se incluye un caso de estudio del uso de los patrones propuestos en un observatorio tecnológico.**

***Palabras claves*—Agente, patrones, patrón de implementación, proactividad.**

Implementation Patterns to include Proactive Behaviors

***Abstract*—Object oriented programming is facing challenges such as the development of software in distributed environments. Along this line has emerged the paradigm of agents. An agent shows behaviors, such as autonomy and proactivity, that differentiates it from an object. Proactivity allows developing goal-directed systems, in which a request is not necessary to start a task. Adding proactivity to a software is nowadays essential, there is a big dependence on computer systems and it is greater the delegation of tasks to them. The patterns have been used successfully in reducing development time and the number of errors in software, besides of being a guide to solve a typical problem. In this paper, we present two implementation patterns to add proactivity to software and to make it easier to work with agents. A case study about the development of a technology observatory using both patterns is also included.**

***Index Terms*—Agent, patterns, implementation pattern, proactivity.**

Manuscrito recibido el 18 de marzo de 2013; aceptado para la publicación el 23 de mayo de 2013.

Mailyn Moreno, Alejandro Rosete y Martha D. Delgado pertenecen a la Facultad de Ingeniería Informática, Instituto Superior Politécnico "José Antonio Echeverría", La Habana, Cuba (e-mail: {my, rosete, marta}@ceis.cujae.edu.cu).

Alternán Carrasco pertenece al Complejo de Investigaciones Tecnológicas Integradas, La Habana, Cuba (e-mail: acarrasco@udio.cujae.edu.cu).

I. Introducción

El paradigma de agentes constituye una tecnología prominente y atractiva en la informática actual. Los agentes y los sistemas multiagente están contribuyendo actualmente en dominios diversos, tales como recuperación de datos, interfaces de usuario, comercio electrónico, robótica, colaboración por computadora, juegos de computadora, educación y entrenamiento, entre otras [1]. Además los agentes están emergiendo como una nueva manera de pensamiento, como un paradigma conceptual para analizar problemas y diseñar sistemas, y ocuparse de la complejidad, distribución e interactividad; mientras que proporcionan una nueva perspectiva en la computación y la inteligencia [1]. Los agentes son entidades que poseen propiedades como la autonomía, la proactividad, la habilidad social, entre otras [2]. Existen agentes orientados a metas, que le dan solución a diferentes problemas. Entre las propiedades más significativas que diferencian a los agentes de los objetos está la proactividad, es decir, los agentes no sólo actúan en respuesta a su ambiente sino que son capaces de tener comportamiento orientado a metas [3].

En la actualidad, los software mayormente se construyen bajo el paradigma de orientación a objetos, ya que este paradigma ha alcanzado un gran auge [4]. En nuestros días hay una tendencia elevada de utilizar enfoques orientados a objetos en todos los sistemas que se construyen, debido a las facilidades que brinda para la reutilización del código [4]. Los patrones de diseño son una muestra indiscutible de la fortaleza que tiene la orientación a objetos [5].

Los patrones son una solución a problemas típicos y con su empleo se puede hacer un desarrollo de software más rápido y con mayor calidad [6].

En el desarrollo de un software orientado a objetos no se tiene en cuenta los comportamientos proactivos que se puedan incluir, los que pueden ser beneficiosos a los usuarios finales. Esto se debe a la naturaleza propia del objeto que exhibe un comportamiento mediante la invocación de un método [7].

La proactividad es una característica muy beneficiosa para el software actualmente, se desea que los programas trabajen por las personas con sólo saber sus intereses. Con la proactividad se pueden obtener asistentes personales en las computadoras que ayuden en la búsqueda de información [8] y a la hora de la toma de decisiones [9]. En la vigilancia tecnológica la proactividad es muy provechosa [10]. Según Henderson-Seller es posible obtener un sistema híbrido

agente + objetos, donde se tenga en un software orientado a objeto con características de los agentes [3].

Se han desarrollado trabajos con patrones para la orientación a agentes con son [11], [12] pero estos no se enfocan en la implementación, o en problemas medulares como la proactividad.

En este trabajo se hace una propuesta de patrones de implementación que utilizan como base los patrones de diseño de la orientación a objetos y las recomendaciones de trabajos de patrones para la orientación a agentes para incorporar proactividad a un software. Se aprovecha las ventajas que provee la filosofía y las plataforma de desarrollo de agentes para incluir este comportamiento. Se desarrolla un caso de estudio sobre un observatorio tecnológico para aplicar los patrones propuestos.

II. Ingeniería de software

La ingeniería de software es el uso de los principios de ingeniería robustos, dirigidos a obtener software económico de gran fiabilidad, además de ser capaces de trabajar sobre las máquinas reales con las que se cuentan. En el desarrollo de un software es fundamental o casi imprescindible utilizar los principios de la ingeniería de software. La misma comprende todos los flujos de trabajo dentro el desarrollo de un software, el análisis, el diseño del sistema, la implementación, las pruebas, el control de versiones entre otros [13].

La ingeniería de software ha evolucionado por diferentes etapas para llegar a lo que existe hoy en día. Por ejemplo pasó por el enfoque estructurado, luego llegó el enfoque orientado a objetos. El paradigma orientado a objetos fue un cambio en la forma de pensar acerca del proceso de descomposición de problemas [7]. Un objeto encapsula estados (valores de datos) y comportamientos (operaciones). En la programación orientada a objetos la acción se inicia mediante la trasmisión de un mensaje al objeto. Un objeto exhibirá su comportamiento mediante la invocación de un método como respuesta a un mensaje [7].

El enfoque orientado a objetos está lejos de ser perfecto para el desarrollo de un software, pero para la mayoría de los desarrolladores es lo mejor que existe para el desarrollo de los mismo [4]. En nuestros días hay una tendencia elevada de utilizar enfoques orientados a objetos en todos los sistemas que se construyen, debido a las facilidades que brinda para la reutilización del código [14].

En el desarrollo del software orientado objeto ha tomado auge la utilización del Proceso Unificado de Desarrollo (*RUP*, *Rational Unified Process*) [4], que es una propuesta de proceso para el desarrollo de software orientado a objetos que utiliza *UML* (*Unified Modelling Language*) [15], [16] como lenguaje que permite el modelado de sistemas con tecnología orientada a objetos.

RUP es un proceso de desarrollo dirigido por casos de uso. Según [4] "Un caso de uso especifica una secuencia de acciones, incluyendo variantes, que el sistema puede llevar a cabo, y que producen un resultado observable de valor para un actor concreto". La comunicación para iniciar un caso de uso es a través de un mensaje o petición de un actor. Esto implica que los mismos no son autoiniciables, no tienen la iniciativa de hacer algo sin una petición u orden. Este comportamiento es intrínseco en la orientación a objetos porque los objetos trabajan para dar respuesta a un mensaje.

A. Tendencias de la Computación

La historia de la computación en la actualidad se ha caracterizado por cinco importantes y continuas tendencias [1]:

1. Ubicuidad

La ubicuidad es una consecuencia de la reducción constante en el costo de la computación, posibilitando introducir el poder de procesamiento en los lugares y con dispositivos que no han sido rentables hasta ahora.

2. Interconexión

Hace dos décadas los sistemas de computadoras eran entidades generalmente aisladas, solo se comunicaban con los operadores humanos. Los sistemas de computadora hoy en día se conectan a una red en grandes sistemas distribuidos. Internet es el claro ejemplo en que se evidencia la dificultad de encontrar computadoras que no tengan la capacidad y necesidad de acceder a Internet.

3. Inteligencia

Esta tendencia está dirigida hacia sistemas cada vez más complejos y sofisticados. Es por ello que la complejidad de las tareas que es capaz el ser humano de automatizar y la delegación en las computadoras ha crecido regularmente.

4. Delegación

La delegación implica que se le dé el control a sistemas informáticos de tareas cada vez más numerosas e importantes. Se observa con regularidad que se delegan tareas a los sistemas de computadoras como pilotear aeronaves.

5. Orientación a humano

Esta última tendencia trata acerca del trabajo constante de aumentar el grado de abstracción de las metáforas que se usan para entender y usar las computadoras. Estas se acercan cada vez más a la forma humana de actuar, que reflejen la forma en que el humano entiende el mundo. Esta tendencia es evidente en todas las formas en que se interactúa con las computadoras.

B. Retos

Ante las nuevas tendencias, la orientación a objetos y *RUP* tratan de adaptarse a los requisitos de los sistemas distribuidos abiertos. Uno de los autores del conocido *RUP*, Grady Booch, ha planteado la necesidad de nuevas técnicas para descomponer (dividir en pedazos más pequeños que puedan tratarse independientemente), abstraer (posibilidad de modelar concentrándose en determinados aspectos y obviando otros detalles de menor importancia), organizar jerárquicamente (posibilidad de identificar organizaciones, gestionar la relaciones entre los componentes de la misma solución que

incluso permitan su tratamiento de grupo como un todo según convenga y ver cómo lograr que entre todos se haga la tarea) [17].

En esta misma línea de desarrollo de software para ambientes distribuidos cada día toma más fuerza un paradigma que muchos consideran como el próximo paso de avance en la tecnología de desarrollo de software: los agentes [17].

Construir software que resuelvan problemas de negocios actuales no es una tarea fácil. Al incrementarse las aplicaciones sofisticadas demandadas por diversos tipos de negocios y competir con un ventaja en el mercado las tecnologías orientadas a objetos pueden ser complementada por las tecnologías orientas a agentes [3].

III. AGENTES

Aunque no hay total unificación en cuanto a qué es un agente, un intento de unificar los esfuerzos para el desarrollo de esta tecnología puede encontrarse en FIPA (*Foundation for Intelligent Phisical Agents*) [18] que los define como una entidad de software con un grupo de propiedades entre las que se destacan ser capaz de actuar en un ambiente, comunicarse directamente con otros agentes, estar condicionado por un conjunto de tendencias u objetivos, manejar recursos propios, ser capaz de percibir su ambiente y tomar de él una representación parcial, ser una entidad que posee habilidades y ofrece servicios, que puede reproducirse, etc. [1].

De forma general, varios autores reconocen en los agentes diversas propiedades, entre las que se destacan el ser autónomos, reactivos, proactivos y tener habilidad social [17], [19], [20].

Los agentes brindan una vía efectiva para descomponer los sistemas complejos, son una vía natural de modelarlos y su abstracción para tratar las relaciones organizacionales es apropiada para estos sistemas [2].

Franklin, después de estudiar doce definiciones, llegó a la conclusión que los agentes tienen entre sus propiedades principales la autonomía, estar orientados a metas, ser colaborativos, flexibles, autoiniciables, con continuidad temporal, comunicativos, adaptativos y móviles. Agrega que un agente autónomo es un sistema situado en un ambiente que percibe el ambiente y actúa sobre él, en el tiempo, según su agenda propia y de esta manera produce efectos en lo que él mismo podrá sentir en el futuro [21].

Russel y Norvig, tienen una visión más flexible de los agentes, como una herramienta para analizar sistemas y no como una característica abstracta que divida al mundo en agentes y no-agentes [22].

De forma general las anteriores definiciones son válidas, con distintos grados de amplitud y reflejando aspectos diferentes, aunque ninguna entra en contradicción con las otras. Se puede decir que las propiedades fundamentales de los agentes son: autonomía, reactividad, proactividad y habilidad social. Las mismas se pueden resumir como sigue [1].

1. Autonomía

Actúan totalmente independientes y pueden decidir su propio comportamiento, particularmente como responder a un mensaje enviado por otro agente.

2. Reactividad

Perciben del entorno y responden a los cambios de éste.

3. Proactividad

No sólo actúan en respuesta a su ambiente, sino que son capaces de tener comportamiento orientado a metas y objetivos. Pueden actuar sin que exista una orden externa, tomando la iniciativa.

4. Habilidad *Social*

Tienen la capacidad de interactuar con otros agentes mediante algún mecanismo de comunicación. Esto le permite lograr metas que por sí solos no puede lograr.

Lo novedoso de los agentes es que pueden ser proactivos, tienen un alto grado de autonomía y están situados en un entorno con el que interactúan. Esto se hace especialmente cierto en áreas como ambientes inteligentes, negocio electrónico, servicios Web, bioinformática, entre otras. Estas áreas demandan software que sean robustos, que puedan operar con diferentes tipos de ambientes, que puedan evolucionar en el tiempo para responder a los cambios de los requisitos, entre otras características.

La mayor diferencia del enfoque orientado a agentes con el enfoque orientado a objetos, es que los agentes pueden tener autonomía, mostrar comportamientos proactivos que no se puedan predecir completamente desde el inicio [23].

Uno de los retos que enfrenta la orientación a objetos además, es sencillamente que no permite capturar varios aspectos de los sistemas de agentes. Es difícil capturar en un modelo de objetos nociones de los agentes como acciones que se hacen proactivamente o reacciones dinámicas a un cambio de su entorno.

Es la proactividad una de las características más distintivas de los agentes [1], [24]. La proactividad es un comportamiento dirigido por metas. El agente trabaja para alcanzar una meta. El comportamiento proactivo permite que se le pase las metas al software y este trabaje para cumplirlas cuando tenga las condiciones para hacerlo.

Los agentes al tener habilidad social normalmente no se les encuentra solos en un sistemas, sino que un sistema puede estar compuesto por más de un agente. Los Sistemas Multiagente (SMA) están compuestos por agentes que tienen conocimiento sobre su entorno, que cumplen con objetivos y metas determinadas por sus responsabilidades. Estos agentes no son independientes aunque sí autónomos en mayor o menor medida. Son entidades de un todo, donde pueden interactuar entre ellos informando y consultando a otros agentes teniendo en cuenta lo que realiza cada uno de ellos, llegando a ser capaces de conocer el papel que tienen todos dentro del sistema según la capacidad que cada uno tenga de actuar y percibir [1].

Un aspecto clave para el desarrollo de los SMA ha sido la especificación de los lenguajes de comunicación de agentes (ACL por sus siglas en inglés) [25]. Un ejemplo de ACL es FIPA-ACL [26], [27],donde se define una biblioteca con una lista de actos comunicativos estándares, cada uno descrito con los parámetros y sus significados, junto a especificaciones en una lógica de precondiciones y efectos racionales. Además, los mensajes pueden ser descritos según un acto comunicativo, una acción ejecutada en un contexto determinado que implica un grupo de consecuencias, que permite a los agentes entender la intención del mensaje recibido, en la forma de compromisos, derechos y comportamientos.

Para el desarrollo de sistemas multiagente existen varias plataformas de desarrollo. JADE[1] está entre las más conocida y utilizada de las plataformas por las facilidades que brinda, entre las que están permitir el desarrollo de aplicaciones de agentes en el cumplimiento con las especificaciones FIPA para sistemas multiagente [28].

La plataforma de agentes JADE trata de mantener en alto el funcionamiento de un sistema de agentes distribuidos con el lenguaje Java. De acuerdo con el enfoque de sistemas multi-agente, una aplicación sobre la base de la plataforma JADE se compone de un conjunto de agentes cooperantes que se pueden comunicar entre sí a través del intercambio de mensajes. Cada agente está inmerso en un ambiente sobre el que puede actuar y en los cuales los acontecimientos pueden ser percibidos. El ambiente puede evolucionar de forma dinámica y los agentes aparecen y desaparecen en el sistema de acuerdo a las necesidades y los requisitos de las aplicaciones. JADE proporciona los servicios básicos necesarios para la distribución de aplicaciones en el ambiente permitiendo a cada agente descubrir a otros dinámicamente y comunicarse con ellos [29].

Para desarrollar un sistema multiagente los desarrolladores necesitan implementar algunas funcionalidades que son de vital importancia para la ejecución del sistema. Entre las características más comunes están las siguientes:

1. Ejecución y control de la plataforma que contiene a los agentes.
2. Gestionar el ciclo de vida de los agentes.
3. Comunicar los agentes que viven dentro del sistema.
   - 3.1. Enviar y recibir mensajes desde y hacia otros agentes.
   - 3.2. Procesar el mensaje y tomar acciones dependiendo de su contenido.
4. Comunicarse con los agentes coordinadores de la plataforma.
   - 4.1. Ver el estado de algún módulo del sistema.
   - 4.2. Buscar un agente para ver su estado.

[1] Java Agent DEvelopment Framework, http://jade.tilab.com

IV. PATRONES

El desarrollo de software basado patrones y modelos está rehaciendo el mundo de los desarrolladores de software [6].

De acuerdo con el diccionario de inglés Oxford[2] un patrón "es una forma lógica o regular", "un modelo", "el diseño o las instrucciones para hacer algo" o "un ejemplo excelente". Todos estos significados se aplican en el desarrollo de software, según [30] la tercera definición es la más acertada.

Christopher Alexander dice que, "Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, entonces describen el núcleo de la solución para ese problema, de manera tal que usted pueda utilizar esta solución un millón de veces, sin tener que hacerlo dos veces de la misma forma" [31]. Aunque es una definición para la arquitectura y la construcción, se puede utilizar para el desarrollo de software. Los patrones, según la disciplina de la Ingeniería de Software en que se manifiesta el problema que resuelven, pueden ser de diseño, de implementación, etc. [5], [14].

"Los patrones de diseños son descripciones de las comunicaciones entre objetos y clases que son personalizables para resolver un problema general de diseño en un contexto particular" [5].

Los patrones de implementación son un módulo de software único en un lenguaje de programación en particular. Una característica crucial es que son fácilmente reconocibles por el software, lo que facilita la automatización [14], [32].

Los patrones de implementación comparten muchos beneficios con los patrones de diseño por ejemplo establecen un vocabulario. Como los patrones de diseño, los patrones de implementación proveen un medio para organizar y trasmitir una base de conocimiento.

Según Beck "Los patrones de implementación proveen un catálogo de problemas comunes en programación y la forma en que […] se pueden resolver estos problemas" [14].

Los patrones de implementación permiten que el trabajo de los programadores sea más efectivo a medida que gastan menos tiempo en partes mundanas y repetitivas de su trabajo y le dedican más tiempo a resolver problemas verdaderamente únicos [14].

En general, un patrón tiene cuatro elementos esenciales [5]:

1. Nombre del patrón

El nombre del patrón es un indicador que se usa para describir un problema, sus soluciones y consecuencias en pocas palabras.

2. El problema

El problema describe cuándo aplicar el patrón, explicando el problema y su contexto.

3. La solución

La solución se compone de los elementos que resulten el problema, sus relaciones, responsabilidades y las colaboraciones.

[2] Oxford: Oxford English Dictionary, http://www.oed.com

4. Las consecuencias

Las consecuencias son los resultados y los cambios resultantes de aplicar el patrón lo que incluye su impacto sobre la flexibilidad de un sistema, la extensibilidad o la portabilidad.

Los patrones como idea y principio se pueden utilizar tanto en la orientación a objetos [5] y en la orientación a agentes [11], [12], [33]. En la orientación a objetos los patrones más conocidos y utilizados son los patrones de diseño, ya que los mismos se pueden llevar hasta la implementación.

A. Patrones de Diseño en la orientación a objetos

Los patrones de diseño pueden ser de diferentes tipos dentro de los cuales se encuentran los creacionales, los estructurales y los de comportamiento [5], [6].

1. Patrones de creación:

Ayudan a hacer un sistema independientemente de cómo son creados, compuestos y representados sus objetos. Un patrón creacional de clase utiliza la herencia para cambiar la clase que es instanciada, mientras que un patrón creacional de objeto delegará la particularización a otro objeto. Dentro de estos se encuentran: *abstract factory, builder, factory method, prototype y singleton*.

2. Patrones estructurales

Los patrones estructurales se relacionan con el modo en que las clases y objetos son compuestas para formar estructuras más grandes. Las clases de patrones estructurales usan, por ejemplo, la herencia para componer o implementar interfaces. Por ejemplo considere como la herencia múltiple mezcla dos o más clases en una; el resultado es una clase que combine las propiedades de su clase padre. Este patrón es particularmente útil para hacer bibliotecas de clases independientes desarrolladas para trabajar juntas. Algunos de estos patrones estructurales son: *adapter, bridge, composite, decorator, facade, flyweight y proxy*.

3. Patrones de comportamiento

Los patrones de comportamiento tienen relación con los algoritmos y la asignación de responsabilidades entre objetos. Este no solo describe patrones de objetos o clases, sino también la comunicación entre ellos. Estos patrones caracterizan flujos de control que son difíciles de seguir en tiempo de ejecución. Cambian su enfoque de flujo de control para dejar que se concentre en la manera en que los objetos son interconectados. Dentro de estos se localizan los siguientes patrones: *chain of Responsibility, command, interpreter, iterator, mediator, memento, observer, state, strategy, template method y visitor*.

En este punto se quiere hacer énfasis en el patrón *Observer*, conocido también por Dependencia o Publicación-Suscripción. Este patrón define una dependencia de uno a muchos entre objetos, de modo que cuando un objeto cambia de estado, todas sus dependencias son notificadas y actualizadas automáticamente.

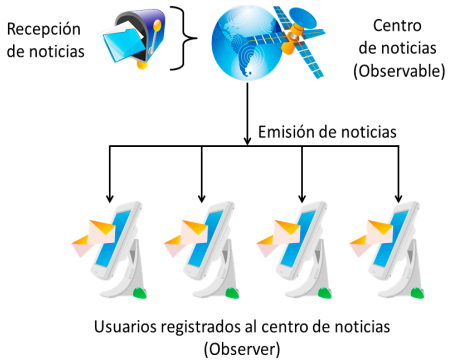


Fig. 1. Ejemplo del patrón *Observer*.

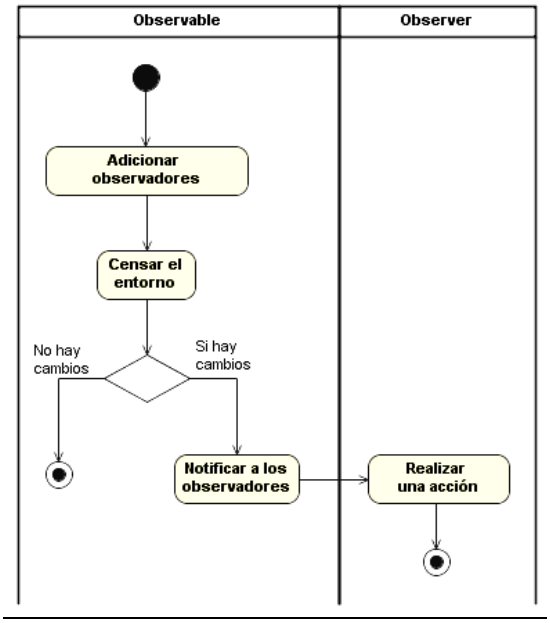


Fig. 2. Diagrama de actividad del patrón *Observer*.

La figura 1 muestra un caso típico del patrón *Observer*. Se trata de un centro de noticias al que están inscritos usuarios con sus preferencias. Al recibir noticias nuevas, el centro distribuye las mismas según la preferencia de los usuarios inscritos.

En la figura 2 se muestra el flujo de trabajo de los componentes en el sistema. Esta figura 2 muestra el ciclo de comportamiento del patrón *Observer*. Se inicia con la adición de observadores, luego la instancia *Observable* monitorea el entorno para ver si han ocurrido cambios. Si existe algo que deba ser notificado a los observadores, envía un mensaje con los datos necesarios para que las instancias de observadores actúen en consecuencia. Este proceso se repite periódicamente en el tiempo.

B. Patrones en la orientación a agentes

En la orientación a agentes se han propuesto patrones de diseño para resolver varios problemas propios de los sistemas multiagente.

Uno de los primeros trabajo es propuesto en [34] y está enfocado en agentes móviles con Aglets[3]. Ese trabajo incluye tres clasificaciones muy orientadas a agentes móviles. Están los patrones de viaje (*traveling*) que están relacionados con el reenvio y enrutamientos de los agentes móviles, patrones de tareas (*task*) para estructurar el trabajo con los agentes y patrones de interacción (*interaction*) para localizar y facilitar las interacciones entre los agentes. Los patrones están desarrollados en Java y enfocados en el diseño. Utilizan diagramas de clases y de interacción para exponerlos y explicarlos.

En ese trabajo se presentan dos aplicaciones basadas en agentes (*File Searcher* y *Enhanced File Searcher*), donde se emplean combinación de patrones. Esas aplicaciones se utilizan para la implementación de agentes móviles que buscan ficheros con cierta cadena en el nombre y que pueden viajar por varios servidores para hacer la búsqueda. En ambos casos se basan en una filosofía reactiva donde los agentes buscan lo que le pide un "*master*" y lo devuelven, pero no conservan ninguna memoria de esa búsqueda. Tampoco se modela una solución a la gestión de cambios en los ficheros almacenados en los servidores sin necesidad de volver a enviar la búsqueda [34].

En [11] y [35] se enfatiza en la necesidad de los patrones de diseño en orientación a agentes, como forma de recolectar y formalizar experiencias para soluciones basadas en este paradigma. En ese trabajo se definen 4 clases de patrones: metapatrones, patrones metafóricos, patrones arquitecturales y antipatrones. Siguiendo esta clasificación se desarrolla una propuesta de 11 patrones. Según Sauvage muchos patrones orientados a agentes son realmente patrones orientados a objetos, ya que no van a aspectos singulares de la orientación a agentes, como la autonomía, las interacciones, entre otras [11]. Además expresa que muchos patrones en la orientación a agentes se enfocan en el diseño obviando la importancia de tener patrones orientados a agentes en varías dimensiones como el análisis o la implementación.

En [36] se presenta un esquema de clasificación bidimensional de los patrones. En su clasificación según el aspecto de diseño (clasificación horizontal), están los estructurales, de comportamiento, sociales y de movilidad. Según el nivel de diseño (clasificación vertical), están los patrones de análisis de roles, patrones de diseño de agentes, patrones de diseño de sistema, patrones de la arquitectura del agente y los patrones de implementación del agente. Un mérito importante de ese trabajo es que su clasificación es amplia y cubre varios niveles de abstracción. Aunque los patrones se presentan en términos de los conceptos de la metodología ROADMAP [37] lo hace de una forma abarcadora y general. Se exponen algunos ejemplos de patrones de agentes y sus clasificaciones. Se enfatiza en que esta clasificación se enfoca más en las nociones del paradigma de agentes, no usando los de orientación a objetos. Entre los campos que sugieren para describir los patrones están: el aspecto de diseño (clasificación horizontal) y el nivel de diseño (clasificación vertical) [36].

Existen otros trabajos, tal es el caso del repositorio de patrones propuesto por el grupo de desarrollo de la metodología PASSI que propone un conjunto de patrones, algunos ejemplos son [33], [38]:

- Patrones multiagente que están relacionados con la colaboración entre dos o más agentes
- Patrones para un solo agente donde se propone una solución para la estructura interna de un agente junto con sus planes de realización de un servicio específico
- Patrones de comportamiento que proponen una solución para agregar una capacidad específica al agente
- Patrones de especificación de acciones que agregan una funcionalidad simple al agente.

Todos estos patrones son para desarrollar un sistema multi-agente más robusto.

Sabatucci en el trabajo [12] se enfoca en patrones de diseño y defiende que un aspecto importante de los patrones no es usarlo solos, sino hacer una combinación de varios. En ese trabajo se hace la formalización de los patrones con un lenguaje que favorece la combinación. Los patrones que propone están integrados a PASSI.

En ninguno de los patrones que se describen en los trabajos anteriormente mencionados sobre patrones para la orientación a agentes se hace énfasis en la proactividad o ambientes a observar, sino en otras propiedades como la cooperación, la comunicación, la estructura organizacional de los agentes u otras. La mayoría de estos patrones se enfocan en el diseño y no en la implementación. En esta dirección, no se conoce de ningún trabajo enfocado en simplificar el trabajo con JADE [39], encapsulando la solución de problemas comunes en la construcción de un SMA.

Particularmente estos dos aspectos (la proactividad, y la simplificación la configuración de JADE) son dos problemas comunes en muchas soluciones basadas en SMA, para las cuales no se conocen que hayan patrones definidos.

V. Patrones de implementación para incluir proactividad

En esta sección se proponen dos patrones de implementación. El patrón *Implementation_JADE* se enfoca en simplificar la configuración de JADE (para crear y manejar agente) y el patrón *Proactive Observer_JADE* se enfoca en la incorporación de proactividad. Estos patrones siguen las recomendaciones de [11] y [36] de que los patrones en la orientación a agentes se enfoquen a las propiedades singulares de los agentes.

Como un patrón de implementación debe estar hecho en un lenguaje de programación específico, se utiliza el lenguaje Java, que es el utilizado por la plataforma JADE. Teniendo en cuenta la consolidación alcanzada por JADE como plataforma

[3] Aglets, http://www.research.ibm.com/trl/aglets

de software libre para el despliegue de un sistema multi-agente y que está basada en el estándar FIPA, se decidió utilizar dicha plataforma para la propuesta de los patrones.

El patrón *Implementation_JADE* respeta la idea de Beck [14] de que los patrones de implementación traten de que los programadores se enfoquen en lo que es realmente singular de cada problema, ya que encapsula parte de la complejidad del trabajo con JADE. Esto se muestra en el ejemplo que se presenta en sección siguiente. El patrón *Implementation_JADE* desarrollado se usa luego en varios lugares y simplificó el trabajo evitando "trabajo mundano y repetitivo" [14] y enfocando el esfuerzo en "problemas realmente únicos" [14].

El patrón *Proactive Observer_JADE* respeta la sugerencia de Sabatucci [12] de enfatizar en la composición de patrones para crear nuevos patrones. Este patrón tiene relación con el patrón *Ecological Recogniser* mencionado en [36]. Ese patrón trata de inferir las intenciones de los agentes y se enfoca en el descubrimiento. En el caso del *Proactive Observer_JADE* las intenciones se conocen y se relacionan con un ambiente que se observa. Esto no está concebido en *Ecological Recogniser* en la forma de estar enfocado en la observación y la decisión cuando la intención es conocida

En la presentación que sigue de ambos patrones se incluyen los campos clasificación horizontal y clasificación vertical sugerida en [36].

A. Descripción de los patrones de implementación en JADE

1. Patrón Implementation_JADE

Este patrón simplifica el uso y configuración de los aspectos principales para el trabajo en la plataforma JADE. A continuación se detallan los elementos esenciales del patrón:

Nombre del patrón: *Implementation_JADE*.

Problema: Específicamente, el patrón que aquí se describe se debe utilizar cuando se quiera implementar las características más comunes y que son de vital importancia para la ejecución del sistema de un sistema multi-agentes mencionadas en la sección III.

Solución: Este patrón utiliza la plataforma JADE para el trabajo con los agentes, sirviendo como intermediario a las funcionalidades que implementa JADE.

El patrón *Implementation_JADE* contiene 7 grupos de operaciones:

1. Inicialización de la plataforma de agentes JADE.
   - 1.1. Configurar algunos parámetros de funcionamiento de la plataforma.
   - 1.2. Crear los contenedores (son necesarios para colocar los agentes dentro).
   - 1.3. Ejecutar los agentes en los contenedores correspondientes.
2. Unión a una plataforma ya existente.
   - 2.1. En ocasiones es necesario tener a los agentes en localidades físicas diferentes, por lo que hay que ejecutar contenedores y agentes en una plataforma que ya existía con anterioridad.
3. Volver a conectar a un agente que ha perdido a la plataforma que lo maneja.
   - 3.1. En el escenario en que un agente esté de forma física en una localidad diferente, puede ser posible que la plataforma colapse por alguna razón y sin embargo, que sea necesario que los agentes sigan trabajando de forma independiente.
   - 3.2. Luego cuando la plataforma vuelva a funcionar, los agentes pueden reincorporarse a su plataforma correspondiente y socializar los resultados que obtuvieron mientras trabajaban solos.
4. Implementar un comportamiento cíclico para poder procesar los mensajes que recibe un agente determinado.
   - 4.1. El desarrollador puede decidir qué tipo de mensaje son aceptados.
   - 4.2. Permite que el desarrollador procese el mensaje como desee.
5. Todo el trabajo relacionado con enviar mensajes hacia los agentes.
   - 5.1. El desarrollador puede enviar mensajes a un agente con una gran variedad de parámetros, que van desde los más básicos a los más complejos.
   - 5.2. La mayoría de las veces solo se necesita enviar un mensaje sencillo a un agente conocido, pero en otras ocasiones, el procesamiento es mayor.
6. Trabajo con el agente AMS (*Agent Management Service*)
   - 6.1. El AMS tiene un registro de los agentes y sus atributos.
   - 6.2. Controla el buen funcionamiento de la plataforma.
   - 6.3. Incluir maneras de interactuar con el AMS para conocer datos sobre los agentes de la plataforma. Ejemplo: saber dónde están los agentes para comunicarse con ellos o el estado de un contenedor determinado.
7. Trabajo con el agente DF (*Directory Facilitator*)
   - 7.1. El control que tiene el DF sobre los agentes es comparado con el de las páginas amarillas.
   - 7.2. Con el DF se puede encontrar un agente que cumpla con un atributo determinado siempre y cuando ese agente se haya registrado con el DF.
   - 7.3. Del mismo modo que con el AMS, aquí están las posibles maneras de comunicarse con el DF para obtener los datos que el desarrollador necesita.

Para lograr estas operaciones mencionadas anteriormente se definieron un grupo de clases con funcionalidades generales.

Init_Platform inicializa la plataforma JADE con las configuraciones que esta permite, como por ejemplo, el puerto de conexión. Se encarga de crear los contenedores en los que se almacenarán los agentes e inicializa los agentes.

Join_Platform crea los contenedores y agentes externos, que son subscritos a una plataforma que ha sido inicializada.

Work_DF contiene todo el trabajo que se realiza en coordinación con el DF (*Directory Facilitator*), que es similar a las páginas amarillas de la guía telefónica. Registra en el directorio del DF el servicio que un agente ofrece, para ofrecer la posibilidad de buscar agentes en esos registros.

Work_ACL engloba el trabajo que se realiza con el uso de los mensajes *ACL* (*Agent Communication Language*). Configura los mensajes con los parámetros que son introducidos como: tipo de mensaje, el contenido del mensaje, la identificación de los agentes involucrados en la comunicación, el objeto que se quiere enviar, etc.

Work_AMS contiene todo el trabajo que se realiza en coordinación con el AMS (Agent Management System). Devuelve la descripción del agente que cumple con la condición que el usuario desee.

Behaviour_Receive_Msg implementa un *Cyclic Behaviour*[4], para obtener y procesar los mensajes ACL que le llegan al agente al que pertenece. Brinda la posibilidad de extender el método "*processMessage*", que se ejecuta cuando se obtiene un mensaje.

El diagrama de clase donde se exponen las principales clases del patrón Implementation_JADE están en la figura 3 junto con las clases del otro patrón de implementación que aquí se propone.

Consecuencias: Este patrón encapsula la capa de abstracción que facilita la configuración del patrón *Proactive Observer_JADE*.

Su objetivo es simplificar el desarrollo de un conjunto de agentes, al tiempo que garantiza el cumplimiento de los estándares a través de un amplio conjunto de servicios del sistema y los agentes con JADE. Permite que los desarrolladores puedan hacer uso de la tecnología de agentes de una forma más sencilla

Clasificación horizontal: Estructural y social.

Clasificación vertical: Implementación del agente.

2. Patrón Proactive Observer_JADE

Este patrón utiliza los principios de patrón de comportamiento *Observer* [5] y los combina con el patrón *Implementation_JADE*.

Nombre del patrón: *Proactive Observer_JADE.*

Problema: Se utiliza el patrón en cualquiera de las siguientes situaciones:

Cuando una abstracción tiene dos aspectos, uno en función de las demás.

Cuando hay un cambio en una entidad, y es necesario cambiar a los demás y no se sabe cuántas entidades más habría que cambiar.

Cuando un entidad debe ser capaz de notificar a otras entidades sin hacer suposiciones acerca de quiénes son estas entidades.

[4] Comportamiento implementado en JADE para efectuar una acción indefinidamente.

Solución: Para implementar el patrón se necesitan de dos entidades: "*Observable*" y "*Observer*".

Entre los métodos más relevantes de la entidad "*Observable*" están los siguientes:

1. Censar el ambiente cada determinado tiempo para detectar algún cambio.
2. Gestionar una lista con los observadores que se subscriban.
3. Notificar a los observadores con los datos encontrados en el cambio ocurrido.

La entidad "*Observer*" tiene que tener métodos como:

1. Implementar el proceso de subscripción a la lista del "*Observable*".
2. Actualizar su estado interno.

Esta permite realizar una acción cuando se le notifique del cambio.

Para lograr una implementación genérica del patrón con agentes, se deben crear fundamentalmente dos entidades: *Observer* y *Observable*. Los agentes que cumplirán con estos roles tendrán la capacidad de comunicarse y actuar autónomamente, pudiendo hasta cambiar su comportamiento dependiendo de las situaciones a las que se enfrenten.

El Agente Observable debe tener una lista interna de los agentes Observer para poder alertarlos de los cambios que detecta. Además debe esperar los mensajes de subscripción de los agentes Observer para poder sumarlos a la lista mencionada y enviar la respuesta de la subscripción. Por último debe tener la capacidad de enviar un mensaje a los Observers con los datos necesarios del cambio encontrado.

El agente Observer tiene que conocer los Observables existentes en el entorno para luego decidir a cual subscribirse. Además debe actualizar su estado cuando le llegue un mensaje con los datos que describen el cambio ocurrido y actuar en consecuencia.

De forma general se necesita que estos agentes se ejecuten en una infraestructura que gestione los mensajes y el ciclo de vida de los agentes.

Como se describió anteriormente en el patrón *Implementation_JADE* se implementaron un grupo de clases para facilitar el trabajo con la plataforma de agentes JADE. Sobre la base de estas clases se realizaron otras que ejecutan las funcionalidades del patrón *Proactive Observer_JADE* para incluir proactividad. De esta forma, los siguientes pasos deben ejecutarse en el momento de comenzar a utilizar el patrón *Proactive Observer_JADE* creado:

1. Iniciar la plataforma a través de la clase de apoyo *Init_Paltform*.
2. Crear los contenedores que contienen a los agentes.
3. Añadir los agentes necesarios a los contenedores.
4. El usuario programador debe implementar las acciones del *Observable* y los *Observers*. En este caso la clase creada para este fin es *Agent_Actions*.

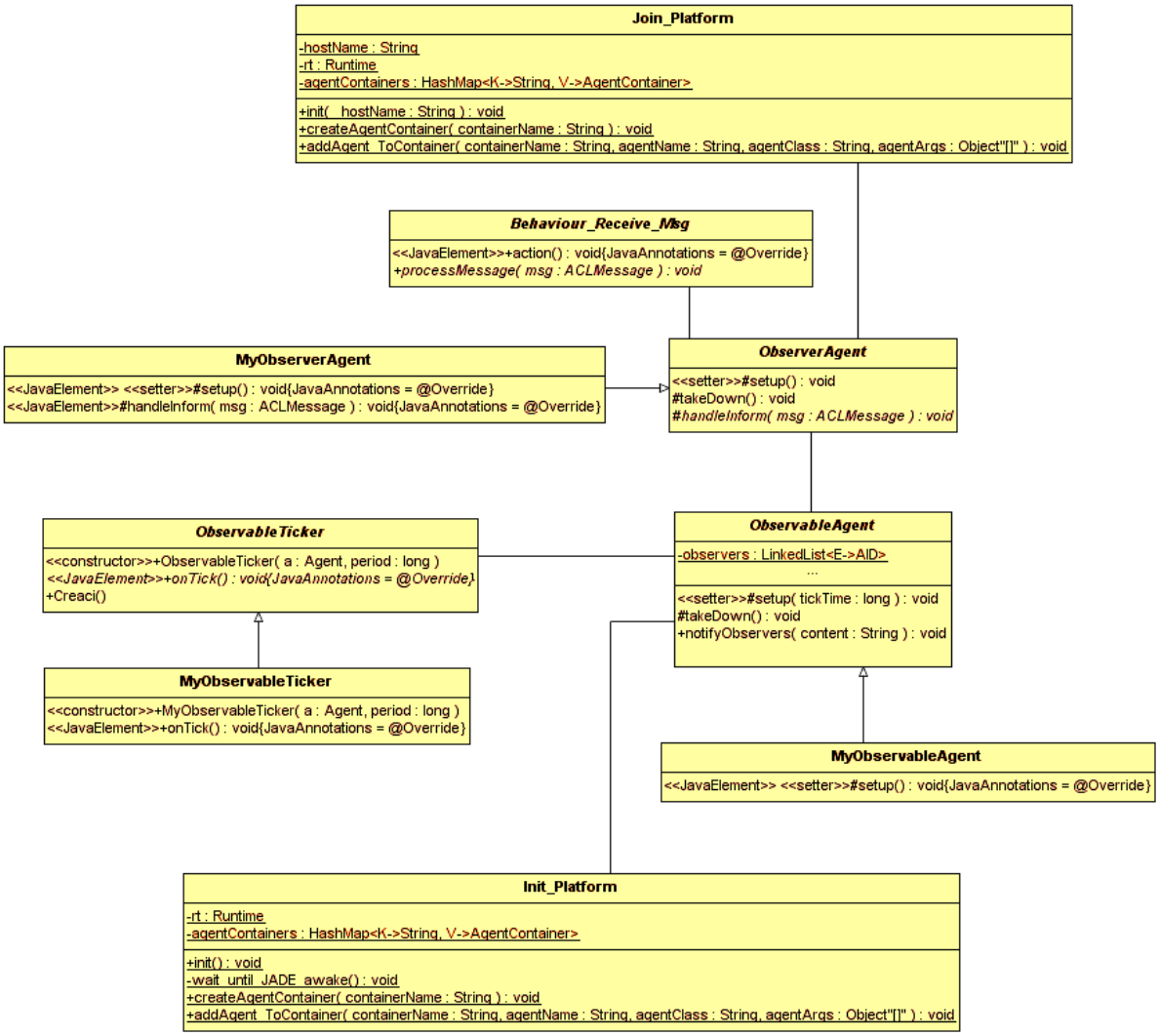


Fig. 3. Diagrama de clases de los patrones *Implementation_JADE* y *Proactive_Observer_JADE*.

Se implementó el patrón *Proactive_Observer_JADE* utilizando agentes, con las clases que proporciona JADE, logrando que el patrón funcione en un ambiente distribuido. A continuación se explican las clases que le dan las funcionalidades al patrón.

Agent_Actions tiene un método *ObserverAction* que se ejecuta cuando al agente *Observer* le llega un mensaje y un método *ObservableAction* que censa el ambiente cada determinado tiempo. La clase está diseñada para que el usuario coloque el código de las acciones que desea realizar en cada caso.

ObservableAgent extiende de *Agent*, escucha los mensajes del tipo *Subscribe* [27] que le llegan de un *Observer*. Cuando un mensaje de este tipo es recibido, se decide si adicionarlo a la lista de *Observers* o no. Si hay algún cambio se notifica a los *Observers* que se encuentren en la lista. La observación del ambiente se realiza usando la implementación del *ObservableTicker*.

ObserverAgent extiende *Agent*, al ejecutarse se subscribe a un Observable y espera el mensaje de respuesta de si fue subscrito o no. Espera por la notificación del *Observable* y realiza alguna acción.

ObservableTicker extiende el funcionamiento de *"TickerBehaviour"* de JADE. Brinda la posibilidad de extender el método *"onTick"*. Este método se encarga de cada cierto tiempo verificar si hubo algún cambio en el entorno.

Las clases que heredan de estas tres últimas, llamadas con el sufijo *My*, son aquellas que el programador debe implementar para que realicen las operaciones que se necesiten. Por ejemplo, como revisar el ambiente y qué hacer cuando al *Observer* le llega la notificación de que *Observable* encontró algo monitoreando el ambiente.

La figura 3 muestra el diagrama de las clases principales del patrón *Implementation_JADE* y del patrón *Proactive_Observer_JADE*. La figura 4 describe el modelo en capas que muestra cómo se relacionan las clases.

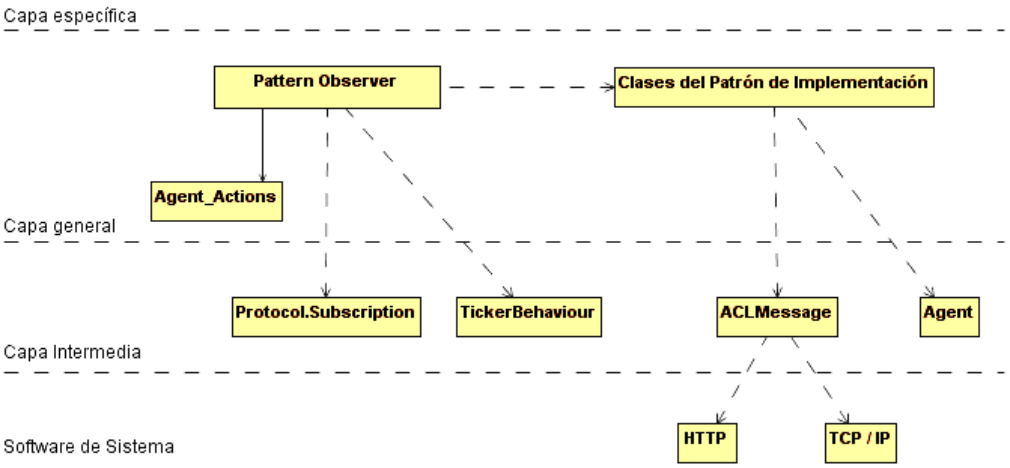


Fig. 4. Diagrama en capas de la relación entre las clases.

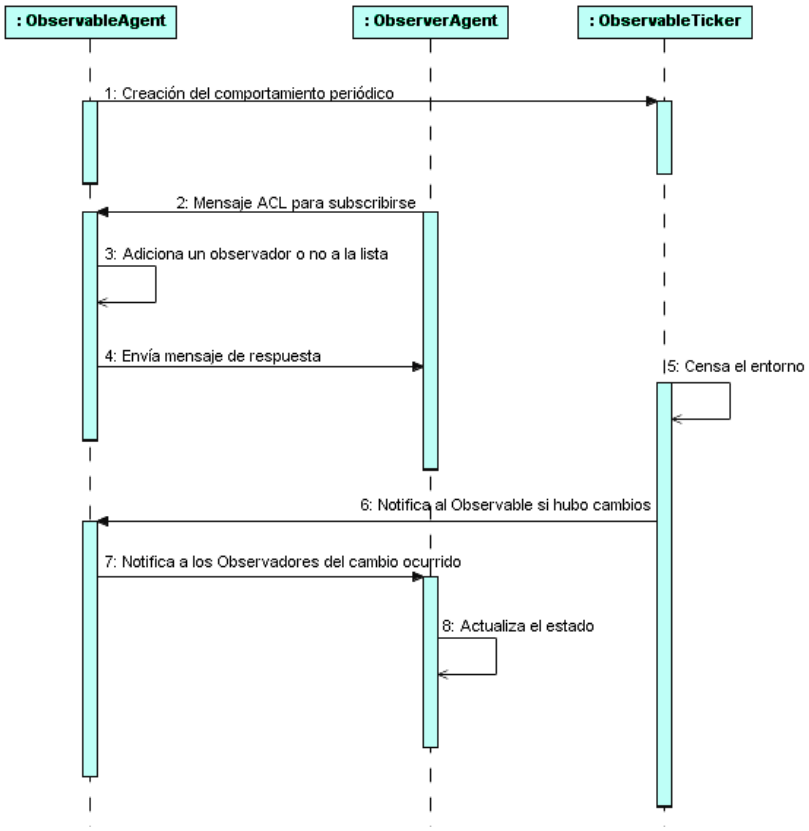


Fig. 5. Diagrama de secuencia del funcionamiento de *Proactive Observer_JADE*

El funcionamiento de las entidades que se ejecutan en el patrón *Proactive_Observer_JADE* puede entenderse mejor en el diagrama de secuencia de la figura 5. Se indican las diferentes llamadas a las funciones de los agentes para que se vea la interacción entre ellos.

Las clases representadas en este diagrama interactúan de manera que *ObservableAgent* llama al constructor de *ObservableTicker*, para que así se pueda censar el ambiente cada cierto tiempo. *ObserverAgent* envía un mensaje a *ObservableAgent* para subscribirse a él y espera la respuesta del mismo.

ObservableTicker llama al método "*onTick*" para censar el entorno y ver si ha ocurrido algún cambio. Si hay cambios entonces notifica a *ObservableAgent* para que envíe un mensaje a los observadores de su lista, los que a su vez actualizan su estado.

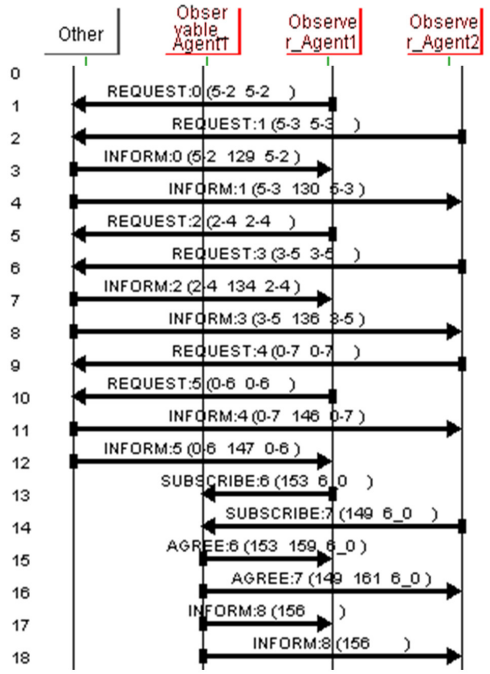


Fig. 6. Vista del despliegue del Proactive_Observer_JADE

Entre las funcionalidades más importantes del *Proactive_Observer_JADE*, que utiliza como base el Observable, es la gestión de los *Observers*, avisándoles de los cambios encontrados. A su vez, los *Observers* deben subscribirse a un *Observable* cuando se inician.

De acuerdo a estas características se implementó un agente *Observable* que al iniciar espera un mensaje de subscripción de los sucesores y ejecuta un comportamiento "*Ticker*" que censa el ambiente para informar de algún cambio ocurrido. Cuando inicia el *Observer*, él conoce quien debe ser su predecesor, y le envía un mensaje de subscripción. El ciclo de ejecución del patrón sigue los pasos explicados anteriormente y se muestran en la figura 6.

Los cuadrados de color rojo representan los agentes que se están ejecutando en la plataforma JADE. Las flechas son los mensajes que son enviados entre ellos en el transcurso del tiempo. Además los mensajes de un mismo color significan que uno es respuesta del otro. Los primeros dos mensajes son parte del funcionamiento de JADE, y como se puede observar las flechas van desde el **ams** al **df** y a otro agente**.** Esto significa que el primero está reportando alguna información interna de JADE. Para un mejor entendimiento, se detallarán solo los mensajes relacionados con el agente **Observer 1**. Pero cabe destacar que los demás *Observer* también realizan las mismas actividades.

En la figura 6 se muestran los mensajes numerados, a continuación se explican algunos de ellos según esa numeración:

3: primer mensaje relacionado con el patrón, donde el **Observer 1** se registra al directorio del **df**. Luego el **df** informa de que el registro fue realizado.

7: respuesta recibida del mensaje 3.

10: enviado desde el **Observer 1** hacia el **ams** para pedir la identificación del agente **Observable** al cuál quiere subscribirse.

14: respuesta recibida del mensaje 10.

19: el **Observer 1** se subscribe a **Observable.**

21: el **Observer 1** recibe la respuesta de la subscripción.

27: **Observable** le informa a **Observer 1** del cambio encontrado.

Consecuencias: A partir del uso del patrón se garantiza que las entidades *Observer* reciban una notificación encontrada por la entidad *Observable* y puedan tomar decisiones o realizar una acción. Con el uso del patrón se logra además el funcionamiento de los agentes con un despliegue distribuido.

Clasificación horizontal: Estructural, de comportamiento y social.

Clasificación vertical: Implementación del agente.

VI. Aplicación de los patrones en un caso de estudio

La proactividad se puede utilizar en cualquier sistema para aumentar sus prestaciones de cara al usuario. Algunos agentes pueden verse como un "objeto astuto" o un "objeto que puede decir no". Viéndolo así un sistema híbrido agente+objetos es completamente viable [3].

Para lograr incorporar comportamientos proactivos en un software orientado a objetos, lograr una hibridación y manejar la proactividad en software orientados a objetos se pueden utilizar los dos patrones de implementación propuestos.

Un caso a tener en cuenta la proactividad es en la construcción de un observatorio tecnológico. Un observatorio es una herramienta para realizar vigilancia tecnológica, que reconoce cambios en el dominio de información que procesa, gestiona y observa, por lo tanto, teniendo en cuenta comportamientos previos, puede avisar con antelación de ciertas variaciones o diferencias en parámetros que evalúa, generando un conocimiento con un alto nivel de importancia al ser actual y novedoso, que puede ser utilizado por los receptores que tengan interés en esa información [40].

Una situación que aún no tiene una respuesta acertada, es que muchos OT operan gracias a las personas que trabajan dándole soporte, buscando, procesando, resumiendo, colocando noticias en los sitios web e informando a los clientes de sus descubrimientos. El desarrollo y buen funcionamiento de un OT enfrenta no sólo el problema relacionado con el número y nivel académico del personal que lo integra [40]. También es necesario que los OT tengan la capacidad de ser proactivos en cuanto a la búsqueda de información, de estar orientados a metas a partir de las necesidades de sus usuarios. Los observatorios deben utilizar un método claro, riguroso y neutro de alerta temprana para sus usuarios [10].

Descripción general del Observatorio Tecnológico: El sistema se divide en una capa cliente y una capa que representa al observatorio como se ve en la figura 7. El sistema tiene agentes personales, estos son una frontera entre

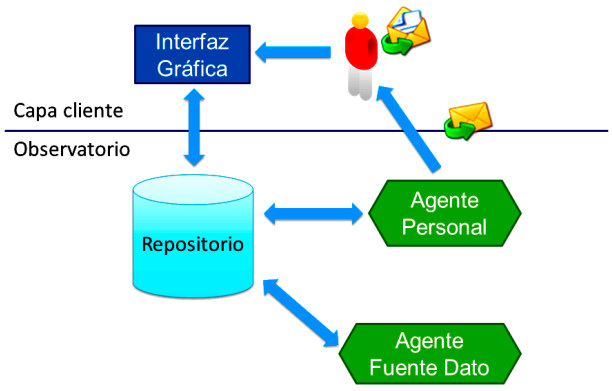


Fig. 7. Vista esquemática del observatorio.

el usuario y el observatorio, son los encargados de representar al usuario en todo momento. El Agente Personal (AP) se dedica a gestionar la información que el usuario necesita y lo hace a partir de sus intereses. También tiene un repositorio, al cual se subscriben los AP, de esta manera, publicando la documentación perteneciente a las línea de investigación de sus usuarios.

Otros pueden acceder a ella cuando sea necesario. Además tiene agentes Fuentes de Datos que están alerta a los pedidos de descarga y búsqueda de los AP. Si alguien necesita una información en específico ésta es pedida a su Agente Personal, que busca en las fuentes de datos disponibles y envía la respuesta al usuario.

Problemas: En este sistema se presentan dos problemas fundamentales que son tratados por los patrones descritos en este documento.

1. El primero está relacionado con la dificultad del trabajo con JADE de forma general, ya sea la gestión de los agentes, los contenedores, la plataforma, envío y recepción de mensajes ACL, etc.

2. Otro problema se encuentra cuando un especialista necesita una información que es relevante para su trabajo y debe esperar a que en la planificación de su Agente Personal, se realice la búsqueda de recursos. Esto conlleva a que el especialista debe esperar un tiempo para recibir resultados de la búsqueda.

Se quiere que de forma periódica el AP haga un reconocimiento del entorno para encontrar información nueva y relevante para su usuario. Primero, debe comunicar con otros AP y si estos no dan una respuesta satisfactoria debe pasar a tramitar sus pedidos con el Agente Fuente de Datos. Cuando se encuentra algo nuevo el AP debe enviar un correo electrónico a su usuario con los resultados.

Soluciones: En la implementación del sistema se utilizó el patrón *Implementation_JADE* y el patrón *Proactive_Observer_JADE*. A continuación se explica cómo fueron usados cada uno.

1. El sistema del Observatorio utiliza las clases de apoyo para ejecutar la Plataforma JADE, brindadas por el patrón Implementation_JADE, para inicializar los contenedores y los agentes, para comunicarse con los agentes del servidor JADE (ams y df), además envía y recibe los mensajes ACL entre agentes. El programador realiza todas estas funciones de forma sencilla y flexible con el mínimo esfuerzo posible.

2. Se implementó un agente **FuentedeDatos_Agent** que extiende las funcionalidades de la clase ObservableAgent, que por las características de este sistema la función que efectúa es gestionar un sitio (repositorio). También se implementaron tres agentes **Personal_Agent** que extienden la clase ObserverAgent, que en este sistema atiende a los especialistas. Ambas clases se vinculan por medio del patrón Proactive_Observer_JADE.

De forma natural el agente personal solicita la búsqueda de documentos a los agentes fuentes de datos del sistema, de forma que los últimos realizan búsquedas en sus correspondientes sitios y devuelven una lista de aquellos documentos que cumplan con las palabras pedidas por los usuarios.

Al utilizar el patrón *Proactive_Observer_JADE* una persona puede desear subscribirse a un sitio, consiguiendo que el agente Fuente de Dato que gestiona ese sitio pueda enviar resultados en el momento en que los encuentra al Agente Personal que atiende a ese usuario. Este último entonces envía un correo con los resultados encontrados antes de tener que realizar una búsqueda que viene condicionada por la planificación en su ciclo.

Los tres Agentes Personales asumen el rol de *Observer* y el Agente Fuente de Datos el de *Observable*. En la figura 8 se muestra como los Agentes Personales se subscriben al Agente Fuente de Datos de la misma forma en que fue explicado el patrón *Proactive_Observer_JADE* anteriormente. En los mensajes del 31–33 están los envíos de informaciones encontradas para estos tres especialistas, en el momento en que se encuentran los documentos para ellos.

En la figura 9 se muestra un ejemplo del correo enviado por el Agente Personal al especialista.

Todo la información relevante a un usurario se obtiene de forma proactiva, con solo decir sus intereses. El agente personal que representa al usuario con sus intereses, utilizando el patrón *Proactive_Observer_JADE*, se mantiene buscando cada cambio, en los sitios que se escogen. Cuando hay un cambio, el usuario recibe un correo con lo nuevo encontrado en los sitios o lo que ha socializado otro agente personal.

VII. Conclusiones

En este trabajo se propuso dos patrones de implementación utilizando como base el patrón de diseño de la orientación a objeto *Observer* y siguiendo la filosofía de agentes. Para desarrollar el patrón *Implemetation_JADE* se tomó como base la plataforma de desarrollo de agentes JADE y en el mismo se

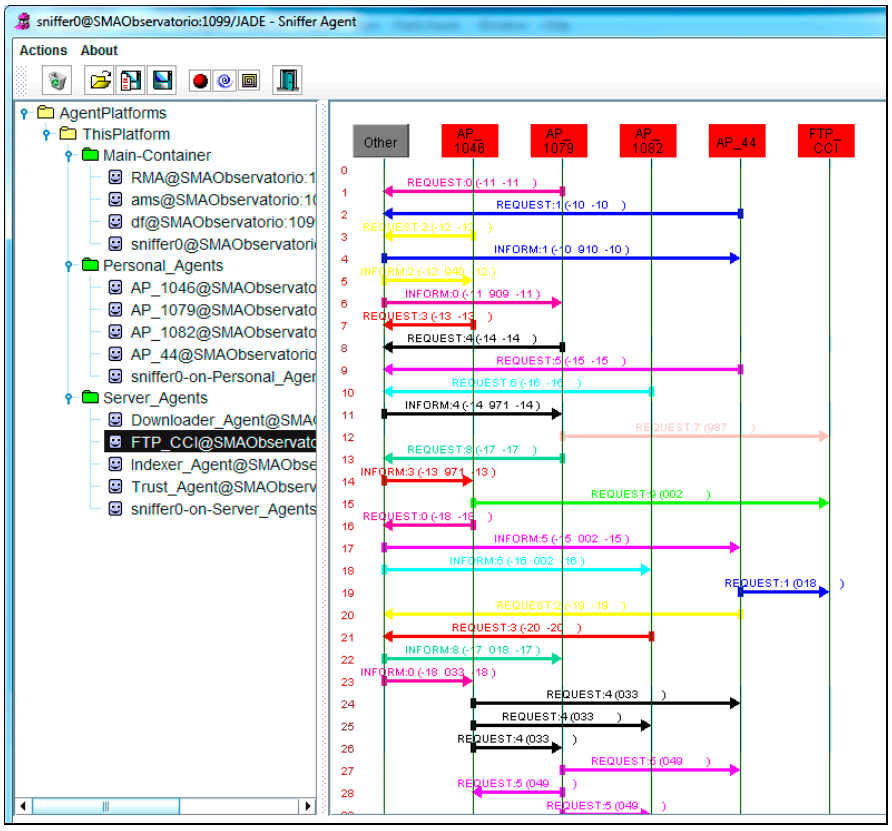


Fig. 8. La pantalla de ciclo de vida de los patrones Implementation_JADE y el patrón Proactive_Observer_JADE en el Observatorio

Las palabras buscadas fueron: web service, team foundation server
El Observatorio ha encontrado 52 recursos y 0 URL que pueden ser de su interés.

A continuación se muestra la cantidad de recursos encontrados de cada fuente.

FTP_CCI	49
FTP_TELECO	3

Recursos organizados por orden de relevancia.

Lista de recursos encontrados:
Nombre: Professional Team Foundation Server 2010.pdf
Relevance: 93.04
Term - Frequency:
web service - 19
team foundation server - 2120
Page count: 722
Author: Ed Blankenship, Martin Woodward, Grant Holliday & Brian Keller
Creation date: 2011/03/15 08:52:55
Modification date: 2011/05/18 22:57:22
Title: Professional Team Foundation Server 2010

Fig. 9.Correo electrónico de resultado de la ejecución de los patrones *Implementation_JADE* y el patrón *Proactive_Observer_JADE* en el Observatorio

da una capa de abstracción para el trabajo con las funcionalidades de agente de una forma sencilla. Este patrón sirvió como núcleo para el patrón *Proactive_Observer_JADE* permite incluir entidades que a partir de una meta y cambios en el ambiente que revisan se realice una acción proactiva.

Ambos patrones se utilizaron en un caso de estudio relacionado con problemas en un observatorio tecnológico. Al aplicar los patrones en el caso de estudio se pudo comprobar que se pudo agregar de forma satisfactoria un comportamiento proactivo beneficioso para el usuario. La inclusión de características proactivas en un Observatorio Tecnológico mejora el rendimiento del mismo, ya que el sistema es capaz de adelantarse a las solicitudes de información de los usuarios. Los patrones propuestos presentan una alta reutilización para los programadores que deseen utilizarlos, debido a la facilidad del lenguaje Java con el que fueron desarrollados. Con los mismos se puede incorporar proactividad en un sistema y manejar de una forma sencilla los agentes.

Referencias

[1] M. Wooldridge, "An Introduction to MultiAgent Systems," 2nd ed. John Wiley & Sons, 2009.

[2] N.R. Jennings. (2000). "On agent-based software engineering," *Artificial Intelligence*, 117(2), pp. 277-296.

[3] B. Henderson-Sellers and P. Giorgini, "Agent-Oriented Methodologies," 1st ed. Hershey: Idea Group Inc, 2005.

[4] I. Jacobson, G. Booch and J. Rumbaugh, "The Unified Software Development Process," reprint ed. Prentice Hall, 2012.

[5] E. Gamma, Design Patterns: "Elements of Reusable Object-oriented Software," ed. Pearson Education, 2004.

[6] A. Shalloway and J.J. Trott, "Design Patterns Explained: A New Perspective on Object-Oriented Design," ed. Addison-Wesley, 2002.

[7] T. Budd, "An introduction to object-oriented programming," 3rd ed. Addison-Wesley, 2002.

[8] C. Ruey Shun and C. Duen Kai. (2008). "Apply ontology and agent technology to construct virtual observatory," *Expert Systems with Applications*, 34(3), pp. 2019–2028.

[9] A. Adla. (2006). "A Cooperative Intelligent Decision Support System for Contingency Management," *Journal of Computer Science*, 2(10).

[10] L. Rey Vázquez. (2009). "Informe APEI sobre vigilancia tecnológica", Asociación Profesional de Especialistas en Información. [Online]. Available: http://eprints.rclis.org/17578.

[11] S. Sauvage, "Agent Oriented Design Patterns: A Case Study," in *Proc. of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, Vol. 3, 2004, pp. 1496–1497.

[12] L. Sabatucci, M. Cossentino and S. Gaglio, "A Semantic Description For Agent Design Patterns," in *Proceedings of the Sixth International Workshop "From Agent Theory to Agent Implementation" (AT2AI-6) at The Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2008)*, 2008, pp. May 13.

[13] R.S. Pressman, "Software engineering: a practitioner's approach", 7th ed. McGraw-Hill Higher Education, 2010.

[14] K. Beck, "Implementation patterns," 1st ed. Addison-Wesley, 2008.

[15] M. Fowler, "UML distilled", 3rd ed. Addison-Wesley, 2004.

[16] J. Rumbaugh, I. Jacobson and G. Booch, "The Unified Modeling Language Reference Manual," 2nd reprint ed. Addison-Wesley, 2010.

[17] N.R. Jennings, "An agent-based approach for building complex software systems," *Comm. of the ACM*, 44(4), 2001, pp. 35–41.

[18] FIPA, "FIPA Agent Management Specification," Foundation for Intelligent Physical Agents, 2003. [Online]. Available: http://www.fipa.org/specs/fipa00023/XC00023H.html.

[19] F. Zambonelli and A. Omicini, "Challenges and Research Directions in Agent-Oriented Software Engineering," *Autonomous Agents and Multi-Agent Systems*, 9(3), 2004, pp. 253–283.

[20] F. Dignum *et al*., "Open Agent Systems," in *Agent-Oriented Software Engineering VIII*, Springer Berlin Heidelberg, 2008, pp. 73–87.

[21] S. Franklin and A. Graesser, "Is it an Agent, or Just a Program?: A Taxonomy for Autonomous Agents," in *Proceedings of the Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages*, 1997, pp. 21–35.

[22] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," 3rd, illustrated ed. Prentice Hall, 2010.

[23] B. Henderson-Sellers, "From Object-Oriented to Agent-Oriented Software Engineering Methodologies," in *Software Engineering for Multi-Agent Systems III*, Springer Berlin Heidelberg, 2005, pp. 1–18.

[24] S.A. O'Malley and S.A. DeLoach, "Determining When to Use an Agent-Oriented Software Engineering Paradigm," in *Agent-Oriented Software Engineering II*, Springer, 2002, pp. 188–205.

[25] E. German and L. Sheremetov, "An Agent Framework for Processing FIPA-ACL Messages Based on Interaction Models," in *Agent-Oriented Software Engineering VII*, Springer, 2008, pp. 88–102.

[26] FIPA, "FIPA Communicative Act Library Specification," Foundation for Intelligent Physical Agents, 2003. [Online]. Available: http://www.fipa.org/specs/fipa00037/SC00037J.html.

[27] FIPA, FIPA ACL Message Structure Specification. Foundation for Intelligent Physical Agents, 2003. [Online]. Available: http://www.fipa.org/specs/fipa00061/SC00061G.html.

[28] F. Bellifemine *et al*., "Jade—A Java Agent Development Framework," in *Multi-Agent Programming*, Springer US, 2005, pp. 125–147.

[29] F.L. Bellifemine, G. Caire and D. Greenwood, "Developing Multi-Agent Systems with JADE," ed. Wiley, 2007.

[30] P. Evitts, "A UML pattern language," ed. Macmillan Technical Publishing, 2000.

[31] C. Alexander, S. Ishikawa and M. Silverstein, "A Pattern Language: Towns, Buildings, Construction," 21th ed. New York: Oxford University Press, 1977.

[32] J. Gil and I. Maman, "Implementation Patterns. Department of Computer Science Technion-Israel Institute of Technology", 2004. [Online]. Available: http://www.cs.technion.ac.il/~imaman/stuff/ip-ecoop05.pdf.

[33] M. Cossentino, L. Sabatucci and A. Chella, "Patterns Reuse in the PASSI Methodology," in *Engineering Societies in the Agents World IV*, Springer Berlin Heidelberg, 2004, pp. 294–310.

[34] Y. Aridor and D.B. Lange, "Agent design patterns: elements of agent application design," in *Proceedings of the Second international conference on Autonomous agents*, 1998, pp. 108–115.

[35] S. Sauvage, "Design Patterns for Multiagent Systems Design," in *MICAI 2004: Advances in Artificial Intelligence*, Springer Berlin Heidelberg, 2004, pp. 352–361.

[36] A. Oluyomi, S. Karunasekera and L. Sterling, "An Agent Design Pattern Classification Scheme: Capturing the Notions of Agency in Agent Design Patterns," in *Proceedings of the 1th Asia-Pacific Software Engineering Conference*, 2004, pp. 456–463.

[37] F. Bergenti, M.-P. Gleizes and F. Zambonelli, Methodologies and Software Engineering for Agent Systems: The Agent-Oriented Software Engineering Handbook, ed. Springer, 2004.

[38] A. Chella, M. Cossentino and L. Sabatucci. "Tools and patterns in designing multi-agent systems with PASSI," *WSEAS Transactions on Communications*, 3(1), 2004, pp. 352-358.

[39] F. Bellifemine *et al*., "JADE-A Java Agent Development Framework," in *Multi-Agent Programming Languages, Platforms and Applications*, Springer, 2005, pp. 125–147.

[40] I. de la Vega, "Tipología de Observatorios de Ciencia y Tecnología," Los casos de América Latina y Europa. *Revista Española De Documentación Científica*, 2007, 30(4), pp. 545–552.

Redes neuronales dinámicas aplicadas a la recomendación musical optimizada

Laura Elena Gómez Sánchez, Humberto Sossa Azuela, Ricardo Barrón,
Francisco Cuevas y Julio F. Jimenez Vielma

***Resumen*—En este trabajo se presenta un método basado en la operación de las llamadas redes neuronales dinámicas (RND), para la recomendación musical optimizada. Las redes son entrenadas con las señales de cada melodía, y no con descriptores tradicionales. La propuesta fue probada con una base de datos compuesta por 1,000 melodías, a diferentes frecuencias de muestreo.**

***Palabras clave*—Recuperación de información musical, red neuronal dinámica, descriptor musical.**

Dynamic Neural Networks Applied to Optimized Music Recommendation

Abstract—A method based on the operation of so called dynamic neural networks (DNN) for music recommendation is described. DNNs are trained with the signals of each melody and not with traditional descriptors. The method has been tested with a database composed of 1.200 melodies, at different sampling frequencies.

***Index Terms*—Music information retrieval, dynamic neural networks, musical descriptor.**

I. Introducción

La Recuperación de Información Musical (MIR por sus siglas en inglés) ha sido definida por Stephen Downie como la "investigación multidisciplinaria que se esfuerza por desarrollar sistemas innovadores de búsqueda basados en el contenido, interfaces novedosas, y mecanismos para que el vasto mundo de la música esté al alcance de todos". Dado al gran interés en esta área de investigación, y a los costos elevados de las bases de datos de música, la mayoría de los investigadores se han visto en la necesidad de crear y utilizar sus propias bases de datos que no se encuentran en la literatura, esto debido a los derechos de autor.

El área de recuperación de información musical se organiza de acuerdo a los casos según cada tipo de consulta, de acuerdo a la forma de comparar la entrada con la salida. Las consultas y la salida puede ser información textual (metadatos), fragmentos de música, grabaciones, partituras o características de la música. Ésta se divide en tres áreas principales de estudio:

- **Análisis simbólico**. Se refiere a la recuperación de información musical a través de partituras digitales [2], [3], [4], [5] y [6].
- **Metadatos**. Tiene que ver con la recuperación de información musical usando metadatos [7], [8], [9] y [10].
- **Análisis de señales acústicas**. Tiene que ver con la recuperación de información musical mediante señales sonoras musicales [11], [12], [13], [14], [15], [16] y [17].

El contorno melódico se utiliza para representar las melodías, característica principal que se utiliza en [18] y [19], la melodía se transforma en una secuencia U, D, R que representan la nota superior, inferior o igual a la nota anterior, respectivamente. Este enfoque simplifica demasiado la melodía que no puede discriminar correctamente entre otras melodías, sobre todo cuando se tiene una gran cantidad de datos. En [20] no solo usa el contorno melódico, también agrega el uso del intervalo del tono y el ritmo. Posteriormente en [21] se introducen cuatro tipos básicos de segmento (A, B, C, D) para el modelo del contorno musical. En ese mismo año en [22] se utiliza un nuevo indicador entre la consulta y las canciones que se proponen.

Varias técnicas de búsqueda basadas en metadatos se han realizado con el modelo de espacio vectorial, el modelo Booleano, indexación, invertir el archivo de índice, medida del coseno, etc. [23], [24] y [25]. En el área de recuperación de información, existen técnicas de indexación y agrupamiento aplicadas al manejo de recuperación musical [26].

CompariSong, en primer lugar, convierte el archivo de audio en segmentos, 10 segmentos por segundo se extraen en base a la frecuencia fundamental y la energía, la serie de tiempo se convierte en letras, para encontrar la correspondencia entre la consulta y la base de datos se utiliza la distancia Levenshtein [27].

Manuscrito recibido el 28 de mayo de 2012; aceptado para la publicación el 5 de junio del 2012.

Laura Elena Gómez Sánchez, Humberto Sossa Azuela, Ricardo Barrón y Julio F. Jimenez Vielma pertenecen al Centro de Investigación en Computación-IPN, Unidad Profesional Adolfo-López Mateos, Av. Juan de Dios Batiz s/n y M. Othon de Mendizábal, Zacatenco, México, DF. 07738, México (e-mail: lenis45@hotmail.com, hsossa@cic.ipn. mx, rbarron@cic.ipn.mx, jfvielma@cio.mx).

Francisco Cuevas pertenece al Centro de Investigaciones en Óptica A.C. Loma del Bosque #115, Col. Lomas del Campestre C.P. 37150, León, Gto. México (e-mail: fjcuevas@cio.mx).

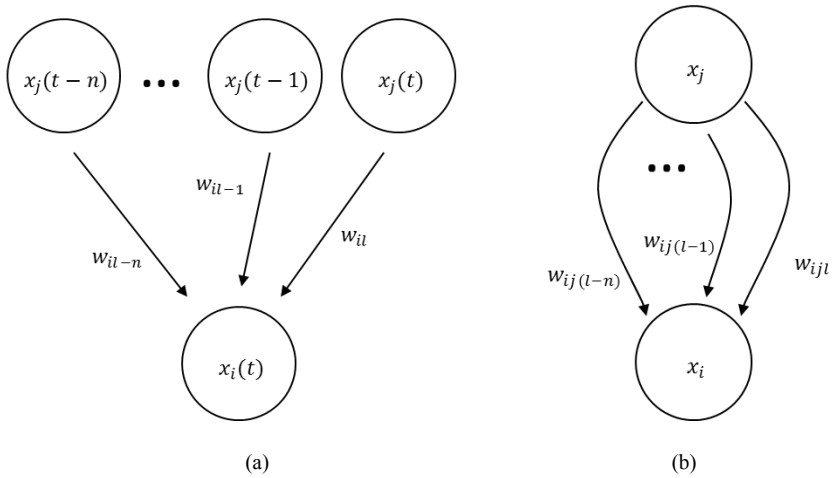


Fig. 1. Conexión de dos neuronas con retardo temporal

La implementación de redes neuronales, tales como las redes de Hopfield, Jordan y Elman, han sido ampliamente utilizadas, al igual que las técnicas de minería de datos dado que el manejo de metadatos es menor.

Existe un sistema híbrido de búsqueda por metadatos y consulta por tarareo, en el cual se realiza un tipo de filtro a través de una búsqueda racional de metadatos y la segunda mediante el tarareo para realizar un control de archivos por consulta.

Midomi es un sitio web comercial desarrollado por la corporación **Melodis** en 2006 [29]. Midomi trabaja en consultas mediante tarareo, canto y silbido, también proporciona una búsqueda avanzada sobre el género y lenguaje, tiene una base de datos musical de más de dos millones de melodías.

SoundHound es otro proyecto de la misma corporación especialmente diseñado para teléfonos móviles e iPods. Funciona igual que Midomi, por medio de tarareo, canto y silbido. Actualmente está diseñado para iPhone, iPod touch, iPod y teléfonos móviles android [30].

Musipedia es un proyecto de código abierto haciendo hincapié en la recuperación de melodías de manera muy diversa [31]. Este sistema proporciona facilidad de búsqueda de melodías en tres diferentes categorías, mediante transcripción melódica, transcripción del contorno melódico y la base rítmica. Soporta consultas por tarareo, silbido, canto, contornos y por golpeteo.

En este artículo se describe un método para la recomendación musical basada en el uso de redes neuronales dinámicas (RND). El método trabaja con diferentes frecuencias de muestreo en formato WAV. La base de datos cuenta con dos conjuntos de melodías, uno para entrenamiento y el otro para las consultas. Una vez que el conjunto de melodías es usado para entrenar las RND, se utilizan los pesos sinápticos de las redes como descriptores para la recomendación musical.

El resto del trabajo se organiza como sigue. En la sección 2 se describe como las redes neuronales de retardo temporal (TDNN por sus siglas en inglés) pueden ser usadas para la recomendación musical. En la sección 3 se detalla el trabajo realizado. En la sección 4 se muestran los experimentos y resultados obtenidos, respectivamente. Finalmente, en la sección 5 se dan las conclusiones de este trabajo.

I. Recuperación de información musical usando TDNN

A. Redes neuronales de retardo temporal (TDNN)

La arquitectura TDNN fue desarrollada en [32]. Esta arquitectura se diseñó originalmente para el procesamiento de patrones de secuencias de voz en series de tiempo con desplazamientos.

Cuando se usan redes multicapa para el tratamiento de secuencias, se suele aplicar una idea muy simple: la entrada de la red se compone no sólo del valor de la secuencia en un determinado instante, sino por valores en instantes anteriores. Es como alimentar la red con una ventana temporal.

La idea de introducir el estado de una variable en diversos instantes en la red no sólo se puede aplicar a la entrada, sino también a las activaciones de las neuronas. Una red donde las activaciones de algunas neuronas son simplemente una copia de las activaciones de otra en instantes anteriores de la denominada Red Neuronal con Retardo Temporal o Time-Delay Neural Network (TDNN) [33] y [34].

Las neuronas con las que se trabaja en redes multicapa con retardo temporal responden a la ecuación:

$$x_i = f_i\left(\sum_j w_{ij}, x_j\right). \quad (1)$$

Como se observa, no existe una dependencia temporal, y la propagación o cálculo de las activaciones se realiza desde la capa superior a la inferior como en cualquier red multicapa.

En estas redes un paso de tiempo hay que entenderlo como iteración. La conexión entre la neurona *j* y la neurona *i*, con retardos temporales, se realizará como:

$$x_i = f_i\left(\sum_l w_{il}, x_l\right), \tag{2}$$

$$x_l = x_j(t - t_l), \tag{3}$$

donde t significa iteración y *tl* es el retardo temporal. Las neuronas *xl* son simplemente copias de la activación de *xj* en instantes o iteraciones anteriores. Se puede dar otra interpretación que consiste en asignar a los pesos distintas velocidades de conexión, siendo unos más lentos que otros, con lo cual en vez de tener una capa con varias neuronas conteniendo copias de las activaciones de la *j* tendríamos sólo la neurona *j* pero conectada a la *i* con varios pesos de distinta velocidad. La ecuación anterior se transformaría en:

$$x_i = f_i(\sum_j \sum_k w_{ijk}, x_j). \tag{4}$$

donde w_{ijk} correspondería al peso que conecta la neurona *j* con la *i* con retardo o velocidad *k*. En la figura 1 se visualiza la conexión entre dos neuronas para las dos interpretaciones. Esta última interpretación tiene un a gran importancia ya que es bien sabido que existen retrasos temporales significativos en los axones y sinapsis de las redes de neuronas biológicas.

B. Algoritmo Levenberg-Marquardt

Este algoritmo fue diseñado para encontrar las raíces de funciones formadas por la suma de los cuadrados de funciones no lineales, siendo el aprendizaje de redes neuronales, una aplicación especial de este algoritmo. El algoritmo de Levenberg Marquardt es una variación del método de iterativo de Newton para encontrar las raíces de una función.

El algoritmo de Levenberg Marquardt puede aplicarse en cualquier problema donde se necesite encontrar los valores de las raíces de una función; en el caso de las redes neuronales artificiales, la función es el error cuadrático medio de las salidas de la red y las raíces de esta función son los valores correctos de los pesos sinápticos.

En la ecuación (5), se presenta como se localiza un valor mínimo ($x_{\min}$) de una función de una variable $f(x)$, utilizando la primera y segunda derivada de acuerdo al método de Newton:

$$x_{\min}(t+1) = x_{\min}(t) - \frac{f'(x_{\min}(t))}{f''(x_{\min}(t))}. \tag{5}$$

Con base en esta ecuación se puede inferir la ecuación (6), donde se minimice el error global E_p en el espacio de los pesos sinápticos representado por la matriz *W*:

$$W(t+1) = W(t) - \frac{E_p'}{E_p''}. \tag{6}$$

La segunda derivada del error global (E_p'') corresponde a la matriz Hessiana *H* y la primera derivada (E_p') la conocemos como el vector gradiente *G*. El vector gradiente y la matriz Hessiana de la función de error los podemos calcular utilizando la regla de la cadena. Así, el vector gradiente se compone por las derivadas parciales del error con respecto a cada uno de los pesos w_i de la red, el elemento (*i*,*j*) de la matriz Hessiana se calcula con las segundas derivadas parciales del error con respecto a los pesos w_i y w_j.

Debido a la carga computacional que implica calcular de manera exacta la matriz *H*, se hace una estimación de la misma [35]. Debido a esto, en (6) se introduce un mecanismo de control para evitar los problemas que se puedan tener en la actualización de pesos de la red, dando origen a (7):

$$W(t+1) = W(t) - (H + \lambda I)^{-1} G. \tag{7}$$

El mecanismo de control para garantizar la convergencia del algoritmo consiste en introducir un factor λI. En primer lugar se prueba la ecuación del método de Newton. Si al evaluarla, el algoritmo no converge (el valor del error comienza a crecer), se elimina este valor y se incrementa el valor de λ en (7), con el fin de minimizar el efecto de la matriz *H* en la actualización de pesos. Si λ es muy grande, el efecto de la matriz *H* prácticamente desaparece y la actualización de pesos se hace esencialmente con el algoritmo de gradiente descendente. Si el algoritmo tiene una clara tendencia hacia la convergencia se disminuye el valor de λ con el fin de aumentar el efecto de la matriz *H*. De esta manera se garantiza que el algoritmo se comporta con un predominio del Método de Newton.

El método Levenberg Marquardt mezcla sutilmente el método de Newton y el método Gradiente Descendente en una única ecuación para estimar la actualización de los pesos de la red neuronal.

C. Audio digital

El audio digital es la representación de señales sonoras mediante un conjunto de datos binarios. Un sistema completo de audio digital comienza habitualmente con un transceptor (micrófono) que convierte la onda de presión que representa el sonido a una señal eléctrica analógica.

Esta señal analógica atraviesa un sistema de procesado analógico de señal, en el que se puede realizar limitaciones en frecuencia, ecualización, amplificación y otros procesos como el de compansión. La ecualización tiene como objetivo contrarrestar la particular respuesta en frecuencia del transceptor utilizado de forma que la señal analógica se asemeje mucho más a la señal audio originario.

Tras el procesado analógico la señal se muestrea, se cuantifica y se codifica. El muestreo toma un número discreto de valores de la señal analógica por segundo (tasa de muestreo) y la cuantificación asigna valores analógicos discretos a esas muestras, lo que supone una pérdida de

información (la señal ya no es la misma que la original). La codificación asigna una secuencia de bits a cada valor analógico discreto. La longitud de la secuencia de bits es función del número de niveles analógicos empleados en la cuantificación. La tasa de muestreo y el número de bits por muestra son dos de los parámetros fundamentales a elegir cuando se quiere procesar digitalmente una determinada señal de audio.

Los formatos de audio digital tratan de representar ese conjunto de muestras digitales (o una modificación) de las mismas de forma eficiente, tal que se optimice en función de la aplicación, o bien el volumen de los datos a almacenar o bien la capacidad de procesamiento necesaria para obtener las muestras de partida.

En este sentido hay un formato de audio muy extendido que no se considera de audio digital: el formato MIDI. MIDI no parte de muestras digitales del sonido, sino que almacena la descripción musical del sonido, siendo una representación de la partitura de los mismos.

El sistema de audio digital suele terminar con el proceso inverso al descrito. De la representación digital almacenada se obtienen el conjunto de muestras que representan. Estas muestras pasan por un proceso de conversión digital analógica proporcionando una señal analógica que tras un procesado (filtrado, amplificación, ecualización, etc.) inciden sobre el transceptor de salida (altavoz) que convierte la señal eléctrica a una onda de presión que representa el sonido.

Los parámetros básicos para describir la secuencia de muestras que representa el sonido son:

- El número de canales: 1 para mono, 2 para estéreo, 4 para el sonido cuadrafónico, etc.
- Frecuencia de muestreo: El número de muestras tomadas por Segundo en cada canal.
- Número de bits por muestra: Habitualmente 8 ó 16 bits.

Como regla general, las muestras de audio multicanal suelen organizarse en tramas. Una trama es una secuencia de tantas muestras como canales, correspondiendo cada una a un canal. En este sentido el número de muestras por segundo coincide con el número de tramas por segundo. En estéreo, el canal izquierdo suele ser el primero.

La calidad del audio digital depende fuertemente de los parámetros con los que esa señal de sonido ha sido adquirida, pero no son los únicos parámetros importantes para determinar la calidad. Una forma de estimar la calidad del sonido digital es analizar la señal diferencia entre el sonido original y el sonido reproducido a partir de su representación digital.

De acuerdo a lo anterior se puede hablar de una relación señal a ruido. Para los sistemas de audio que realicen compresiones digitales tipo *lossless*, esta medida va a estar determinada por el número de bits por muestra y la tasa de muestreo.

El número de bits por muestra determina un número de niveles de cuantificación y éstos una relación señal a ruido de pico de portadora que depende de forma cuadrática del número de bits por muestra para el caso de la cuantificación uniforme. La tasa de muestreo establece una cota superior para las componentes espectrales que pueden representarse, pudiendo aparecer distorsión lineal en la señal de salida y *aliasing* (o solapamiento de espectros) si el filtrado de la señal no es el adecuado. Para los sistemas digitales con otro tipo de compresión la relación señal a ruido puede indicar valores muy pequeños aunque las señales sean idénticas para el oído humano.

La frecuencia de muestreo es un número que indica la cantidad de muestras que se toman en determinado intervalo de tiempo, la resolución o profundidad del sonido es un número que indica cuántos bits (dígitos binarios, ceros y unos) se utilizan para representar cada muestra. Tanto la frecuencia como la resolución están directamente relacionadas con la calidad del sonido digital almacenado. Mientras mayores sean estos indicadores, más parecida será la calidad del sonido digitalizado con respecto al real. El estándar definido cuando se crearon los discos compactos de audio especifica que el sonido digita almacenado en ellos debe poseer una frecuencia de 44 100 KHz y 16 bits estéreo. Esto significa que se deben tomar unas 44 100 muestras por segundo, cada una se representará con 16 bits, y en dos canales independientes (sonido estéreo).

II. Trabajo realizado en MIR

De acuerdo a la literatura el análisis espectral de la señal en el dominio de la frecuencia ha brindado mejores resultados que las técnicas enfocadas al análisis de la misma en el dominio del tiempo. Algunas de las características como el tono, duración, ritmo, correlación cruzada, FFT entre otros están ampliamente ligados a la firma digital. Sin embargo, utilizar directamente la información contenida en la melodía sin hacer uso de firmas digitales o de funciones tradicionales, se evita el pre procesamiento de la información que estas requieren. En este trabajo se utiliza la melodía original para realizar la recuperación de información musical. No se ha realizado ningún cambio o tratamiento previo a las melodías para tratar de adaptarlas a un modelo tradicional, se introducen directamente a la TDNN, dicho procesamiento se muestra en la figura 2.

Tenemos una base de datos de melodías de los Beatles en formato WAV, se cuenta con un conjunto de entrenamiento y otro de prueba. Cada melodía es entrenada en una red neuronal de retardo temporal (TDNN_(*i*), donde *i* es la melodía, al terminar el entrenamiento se obtiene una matriz de pesos *WNN*_(*i*).

De cada melodía que se almacena en la base de datos, se obtiene un vector de datos que puede ser de longitud variable. El número de retardos es igual al número de neuronas en la

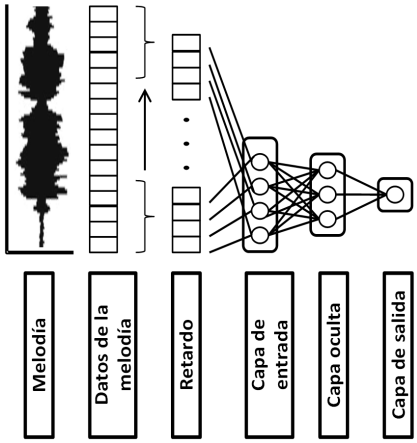


Fig. 2. Estructura de una Red Neuronal con Retardo Temporal

capa de entrada para el primer bloque de datos, en otras palabras se hace un ventaneo del vector. Cada una de las ventanas es la siguiente entrada de la red. Dicho proceso se aplica en toda la melodía.

La matriz obtenida es nuestro descriptor musical, descartando por completo cualquier descriptor tradicional. Esto se puede observar en la figura 3.

Para la recuperación de una melodía se introduce un segmento consulta, dicho segmento entra a una red neuronal de retardo temporal con los pesos sinápticos previamente entrenados, obteniendo los errores de recuperación por cada red (Re_(i)).

Este error se genera a partir de la comparación del segmento consulta en relación con la señal estimada o la predicción de la señal obtenida desde la red. Estos errores son almacenados en un vector, finalmente se aplica $\arg\min()$ retornando un índice (n^*), indicando que red neuronal tuvo el menor error. Este procedimiento se ilustra en la figura 4.

El error de recuperación está dado por:

$$\text{Re\_}i = \frac{\sum_{j=1}^{N}\left(x_j - y_j\right)^2}{w}. \qquad (8)$$

donde x_i son las matrices de pesos previamente entrenados, y_i es el segmento a reconocer, y w es el número de ventanas en las que el segmento fue dividido.

III. Experimentos y resultados

La topología de la TDNN seleccionada consistió de tres capas: una de entrada (número de neuronas es igual al número de retardos de la red), una oculta y una capa de salida (que corresponde a la predicción, ya que es una neurona). Este modelo de red fue programado en Matlab y los datos de entrada no han sido normalizados.

El algoritmo utilizado para el entrenamiento es el backpropagation. Este procedimiento ajusta iterativamente todos los pesos de la red con el fin de disminuir el error obtenido en la unidad de salida, utilizando el algoritmo Levengerg-Marquardt como función de activación.

Los pesos de conexión están inicializados aleatoriamente a [0:3]. Por razones de la velocidad de convergencia de todas las muestras entrenadas que se presentan una vez que los pesos se actualizan.

Se usan archivos en formato WAV de 16 bits (en modo estéreo, para entrenamiento o recuperación solo se utiliza un canal para evitar el *over-processing*, recordando que la información de un canal es copia fiel del otro. La base de datos para esta experimentación es de 1000 melodías (Beatles y Elvis Presley).

La configuración usada para el entrenamiento de las melodías fue de 5 y 10 neuronas en la capa oculta (debido a que en pruebas anteriores son las que mejores resultados han dado) y las iteraciones fueron de 15, 25 y 35, con un retardo de 10, recordando que el tamaño del retardo es igual al número de neuronas en la capa de entrada, obteniendo una predicción de datos en la salida. Al terminar el entrenamiento de cada melodía se obtiene su matriz de pesos, las cuales se utilizarán como descriptores.

Se realizaron pruebas con diferentes tipos de frecuencia de muestreo en formato WAV, utilizando la base de datos mencionada anteriormente. Para estos experimentos se cuenta con un conjunto de entrenamiento y un conjunto diferente para las consultas, cabe destacar que algunas melodías cuentan con 2 o 3 versiones diferentes, lo que permite analizar el desempeño de nuestra propuesta. La tabla I resume las características principales de los datos utilizados.

Como se puede observar la ventana de consulta tiene diferentes valores, ya que a menor frecuencia se cuenta con menos información para realizar un reconocimiento con un segmento pequeño de consulta. Las tablas II y III muestran el rendimiento obtenido en cada configuración utilizada. La tabla II muestra el tamaño necesario de la ventana consulta para una recuperación perfecta. La tabla III muestra el tamaño mínimo para obtener una recomendación de 10 melodías para el usuario. El porcentaje de recuperación por recomendación de melodías puede observarse en la Tabla IV.

IV. Conclusiones

En este trabajo se describió como las llamadas redes neuronales con retrasos pueden ser usadas para describir el contenido musical de melodías para su posterior recuperación, basándose en partes de dichas melodías. Mediante las TDNN se logró resolver el problema planteado sin necesidad de realizar un pre-procesamiento, evitando así el utilizar algún descriptor tradicional o firma digital de la melodía.

A diferencia de otras técnicas de MIR, la melodía original se puede considerar como una serie temporal que se introduce directamente en la TDNN, la salida de la red codifica una descripción de la melodía en la matriz de pesos. Por ejemplo, si se entrena una TDNN con una melodía de 7 938 000 de

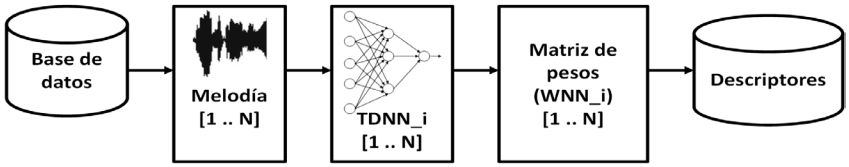


Fig. 3. Estructura de entrenamiento de las melodías con TDNN

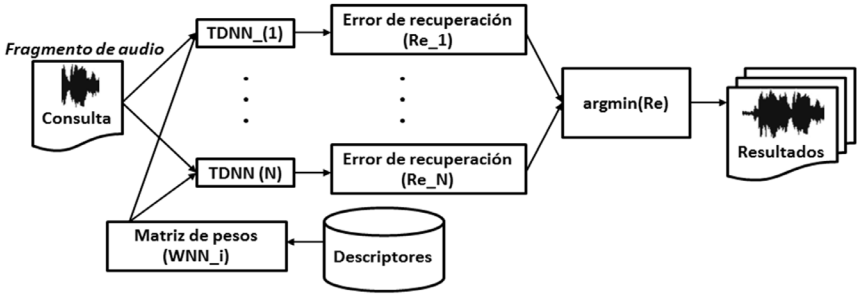


Fig. 4. Procedimiento de recuperación de una melodía usando el modelo propuesto

TABLA I
RASGOS CONSIDERADOS PARA EL AUDIO DIGITAL (WAV)

Tasa de muestreo (KHz)	Rango de la ventana de consulta (Datos)
22 050	25 000 – 60 000
24 000	20 000 – 54 000
32 000	15 000 – 45 000
44 100	10 000 – 40 000

TABLA II
TABLA DE RECUPERACIÓN PERFECTA

Tasa de muestreo (KHz):		22 050	24 000	32 000	44 100
Neuronas en la capa oculta	Iteraciones	Rango de la ventana de consulta (Datos)			
5	15	58 000	54 000	32 000	32 000
	25	41 000	40 000	35 000	29 000
	35	48 000	43 000	38 000	18 000
10	15	59 000	41 000	35 000	40 000
	25	43 000	57 000	39 000	31 000
	35	57 000	46 000	45 000	25 000

TABLA III
TABLA DE RECOMENDACIÓN DE 10 MELODÍAS

Tasa de muestreo (KHz):		22 050	24 000	32 000	44 100
Neuronas en la capa oculta	Iteraciones	Rango de la ventana de consulta (Datos)			
5	15	42 000	38 000	27 000	25 000
	25	31 000	29 000	24 000	21 000
	35	35 000	31 000	31 000	15 000
10	15	44 000	36 000	25 000	32 000
	25	36 000	52 000	28 000	24 000
	35	51 000	42 000	37 000	19 000

TABLA IV
TABLA DE PROCENTAJES DE RECUPERACIÓN POR RECOMENDACIÓN

Tasa de muestreo (KHz):		22 050	24 000	32 000	44 100
Neuronas en la capa oculta	Iteraciones	Porcentajes %			
5	15	72	74	92	89
	25	79	79	85	92
	35	75	75	83	94
10	15	80	80	84	82
	25	77	77	87	86
	35	74	74	81	87

cuadros (aproximadamente 3 minutos de una melodía) con calidad de audio a 44 100 KHz para una recuperación perfecta, solo se necesitan máximo 40 000 cuadros, lo cual refleja menos del 1% del total de la melodía.

Con los resultados obtenidos en esta experimentación se ha observado que el sistema funciona muy bien incluso al trabajar con diferentes frecuencias de muestreo, los mejores porcentajes se obtuvieron con frecuencias de 32 000 y 44 100 KHz, debido a que se tiene una mejor calidad de audio, sin embargo las frecuencias restantes logran realizar una buena recomendación musical. En experimentaciones previas se usaba el conjunto de entrenamiento tanto para el entrenamiento como para la consulta.

En el presente trabajo se analizó el desempeño del método propuesto al usar versiones diferentes de las melodías aprendidas para la recuperación.

REFERENCIAS

[1] F. Wiering, "Can humans benefit from music information retrieval?," AMR'06, *Proc. of the 4th international conference on Adaptive Multimedia Retrieval: User, Context and Feedback*, Springer, 2007, p. 82–94.

[2] K. Lemstrom, G.A. Wiggins and D. Meredith, "A three-layer approach for music retrieval in large databases," *2nd International Symposium on Music Information Retrieval, Bloomington*, USA, 2001, p. 13–14.

[3] H. Hoashi and K. Matsumoto, "Personalization of user profiles for content-based music retrieval based on relevance feedback," *Proceeding of the eleventh ACM international conference on Multimedia*, New York, USA, 2003, p. 110–119.

[4] H. Zhuge, "An inexact model matching approach and its applications," *Journal of Systems and Software*, 67 (3), 2003, p. 201–212.

[5] E. Hwang and S. Rho, "FMF (fast melody finder): A web-based music retrieval system", *Lecture Notes in Computer Science* vol. 277. Springer, 2004, p. 179–192.

[6] E. Hwang and S. Rho, FMF: "Query adaptive melody retrieval system," *Journal of Systems and Software*, 79 (1), 2006, p. 43–56.

[7] N. Ali, M. Mshtaq, "Hybrid query by humming and metadata search system (HQMS) analysis over diverse features," *International Journal of Advanced Computer Science and Applications*, Vol. 2, No. 9, 2011, p. 58.

[8] P. W. Bertin-Mahieux, B. Whitman, P. Lamere, "The Million Song Dataset," *12th Conference of International Society for Music Information Retrieval (ISMIR 2011)*, 2011.

[9] C. McKay, D. Bainbridge, D., "A Musical Web Mining and Audio Feature Extraction Extension to the Greenstone Digital Library Software," *12th Conference of International Society for Music Information Retrieval (ISMIR 2011)*, 2011.

[10] M. Weigl And C. Guastavino, "User Studies in the Music Information Retrieval Literature," *12th Conference of International Society for Music Information Retrieval (ISMIR 2011)*, 2011.

[11] M. Ryynanen, A. Klapuri, "Transcription of the singing melody in polyphonic music", *ISMIR 2006*.

[12] F. Ren, D. B. Bracewell, "Advanced information retrieval," *Journal Electronic Notes in Theoretical Computer Science (ENTCS)* 225, 2009, p. 303–317.

[13] K. Dressler, "Audio melody extraction", late breaking at ISMIR 2010, *Proceedings International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, Netherlands, 2010.

[14] P. V. Kranenburg, J. Garbers, A. Volk, F. Wiering, L. P. Grijp, R. C. Veltkamp, "Collaboration perspectives for folk song research and music information retrieval: The indispensable role of computational musicology," *Journal of Interdisciplinary Music Studies*, 4 (1), 2010, p. 17–43.

[15] J. Salamon, E. Gómez, "Melody extraction from polyphonic music audio," *Music Information Retrieval Evaluation eXchange (MIREX)*, Utrecht, The Netherlands, 2010.

[16] J. Salamon, E. Gómez, "Melody Extraction from Polyphonic Music: MIREX 2011", in *Music Information Retrieval Evaluation eXchange (MIREX) 2011*, extended abstract, Miami, USA, 2011.

[17] J. Salamon, E. Gómez, "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics," *IEEE Transactions on Audio, Speech and Language Processing*, 20(6), 2012, p. 1759–1770.

[18] A. Ghias, "Query By Humming-Musical Information Retrieval in an Audio Database," *Proc. of ACM Multimedia 95*, 1995, p 231–236.

[19] S. Blackburn, D. De Roure, "A Tool for Content Based Navigation of Music," *Proc. ACM Multimedia 98*, 1998, p. 361–368.

[20] R. J. McNab, "Towards the Digital Music Library: Tune Retrieval from Acoustic Input," *Proc. of Digital Libraries*, 1996, p. 11–18.

[21] A. L. P. Chen, M. Chang, J. Chen, "Query by Music Segments: An Efficient Approach for Song Retrieval," *Proc. of IEEE International Conference on Multimedia and Expo*, 2000.

[22] C. Francu and C.G. Nevill-Manning, "Distance metrics and indexing strategies for a digital library of popular music." *2000 IEEE International Conference on Multimedia and Expo, 2000, ICME 2000*, Vol. 2, IEEE, 2000, p. 889–892.

[23] N.C. Maddage, H. Li, and M.S. Kankanhalli, "Music structure based vector space retrieval," *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2006, p. 67–74.

[24] N. Kosugi, Y. Nishihara, S. Kon'ya, M. Yamamuro, and K. Kushima, "Music retrieval by humming-using similarity retrieval over high dimensional feature vector space," *1999 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, IEEE, 1999, p. 404–407.

[25] L. Chen and B.G. Hu, "An implementation of web based query by humming system. *2007 IEEE International Conference on Multimedia and Expo*, IEEE, 2007, p. 1467–1470.

[26] L. Lu, H. You, and H.J. Zhang, "A new approach to query by humming in music retrieval," *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2001.

[27] G. Dzhambazov, "Comparisong: Audio comparison engine," *International Book Series* Number 11, 2009, p 33.

[28] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney, "Content-based music information retrieval: current directions and future challenges," *Proceedings of the IEEE*, 96(4), 2008, p. 668–696.

[29] Midomi, http://www.midomi.com.

[30] Soundhound. http://www.soundhound.com.

[31] Musipedia. musipedia. URL: http://www.musipedia.org/.

[32] A. Weibel, T. Hanazawa, G. Hinton, K. Shikano and K. Lang, "Phenomena Recognition Using Time-delay Neural Networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3), 1989, 328–339.

[33] J. B. Hampshire & A. H. Waibel, "A Novel Objetive Function for Improved Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Transactions on Neural Networks*, 1, 1990, p. 216–228.

[34] K. Lang and G. Hinton, "A Time-Delay Neural Network Arquitecture for Speech Recognition," Carnegie Mellon University, *Tech. Reprt. CMU-CS-88-152*, 1988.

[35] T. Masters, "Advanced Algorithms for Neural Network: A C++ Sourcebook," John Wiley & Sons Inc, 1995.

Journal Information and Instructions for Authors

I. Journal Information

Polibits is a half-yearly open-access research journal published since 1989 by the *Centro de Innovación y Desarrollo Tecnológico en Cómputo* (CIDETEC: Center of Innovation and Technological Development in Computing) of the *Instituto Politécnico Nacional* (IPN: National Polytechnic Institute), Mexico City, Mexico.

The journal has double-blind review procedure. It publishes papers in English and Spanish (with abstract in English). Publication has no cost for the authors.

A. Main Topics of Interest

The journal publishes research papers in all areas of computer science and computer engineering, with emphasis on applied research. The main topics of interest include, but are not limited to, the following:

- Artificial Intelligence
- Natural Language Processing
- Fuzzy Logic
- Computer Vision
- Multiagent Systems
- Bioinformatics
- Neural Networks
- Evolutionary Algorithms
- Knowledge Representation
- Expert Systems
- Intelligent Interfaces
- Multimedia and Virtual Reality
- Machine Learning
- Pattern Recognition
- Intelligent Tutoring Systems
- Semantic Web
- Robotics
- Geo-processing
- Database Systems
- Data Mining
- Software Engineering
- Web Design
- Compilers
- Formal Languages
- Operating Systems
- Distributed Systems
- Parallelism
- Real Time Systems
- Algorithm Theory
- Scientific Computing
- High-Performance Computing
- Networks and Connectivity
- Cryptography
- Informatics Security
- Digital Systems Design
- Digital Signal Processing
- Control Systems
- Virtual Instrumentation
- Computer Architectures

B. Indexing

The journal is listed in the list of excellence of the CONACYT (Mexican Ministry of Science) and indexed in the following international indices: LatIndex, SciELO, Periódica, and e-revistas.

There are currently only two Mexican computer science journals recognized by the CONACYT in its list of excellence, *Polibits* being one of them.

II. Instructions for Authors

A. Submission

Papers ready for peer review are received through the Web submission system on www.easychair.org/conferences/?conf=polibits1; see also updated information on the web page of the journal, www.cidetec.ipn.mx/polibits.

The papers can be written in English or Spanish. In case of Spanish, author names, abstract, and keywords must be provided in both Spanish and English; in recent issues of the journal you can find examples of how they are fomatted.

Only full papers are reviewed; abstracts are not considered as submissions. The review procedure is double-blind. Therefore, papers should be submitted without names and affiliations of the authors and without any other data that reveal the authors' identity.

For review, a PDF file is to be submitted. In case of acceptance, the authors will need to upload the source code of the paper, either Microsoft Word or TeX with all supplementary files necessary for compilation. Upon acceptance notification the authors receive further instructions on uploading the camera-ready source files.

Papers can be submitted at any moment; if accepted, the paper will be scheduled for inclusion in one of forthcoming issues, according to availability and the size of backlog. While we make every reasonable effort for fast review and publication, we cannot guarantee any specific time for this.

B. Format

The papers should be submitted in the format of the IEEE Transactions 8x11 2-column format, see http://www.ieee.org/publications_standards/publications/authors/author_templates.html. (while the journal uses this format for submissions, it is in no way affiliated with, or endorsed by, IEEE). The actual publication format differs from the one mentioned above; the papers will be adjusted by the editorial team.

There is no specific page limit: we welcome both short and long papers, provided that the quality and novelty of the paper adequately justifies its length. Usually the papers are between 10 and 20 pages; much shorter papers often do not offer sufficient detail to justify publication.

The editors keep the right to copyedit or modify the format and style of the final version of the paper if necessary.