ONTOLOGY INTEGRATION APPROACHES AND ITS IMPACT ON TEXT CATEGORIZATION

Hamid Machhour[1] and Ismail Kassou[2]

[1]ENSIAS, Mohammed V Souissi University, Rabat, Morocco
hamid.machhour@gmail.com
[2] ENSIAS, Mohammed V Souissi University, Rabat, Morocco
kassou@ensias.ma

ABSTRACT

This article will introduce some approaches for improving text categorization models by integrating previously imported ontologies. From the Reuters Corpus Volume I (RCV1) dataset, some categories very similar in content and related to telecommunications, Internet and computer areas were selected for models experiments. Several domain ontologies, covering these areas were built and integrated to categorization models for their improvements.

KEYWORDS

Text categorization, ontology integration, document annotation.

1. INTRODUCTION

With the evolution of web 2 and social networks, the need for fast and efficient tools in text analysis and categorization become progressively more important and the ontology based approach float back in top of plain text processing [1-6]. This paper will introduce an application of ontology integration approaches to text categorization on information technology (IT) domain.

Text categorization aims to assign predefined categories to documents based on their content. Thus, it begins with a set of training documents where each document is already tagged by a category. The categorization consists of determining a model able to assign the correct category for a new document. Several statistical techniques and supervised classification [7-11] were applied in text categorization. In the second part of this work we present a brief description for experimented text categorization models: Naïve Bayes, N-grams, TF-IDF and K-Nearest Neighbours. Next, we discuss the built domain ontologies and the principle of its integration to improve models. Finally, we conclude by presenting the results of our experiments on a subset of RCV1.

2. EXPERIMENTED MODELS

There are several techniques for text categorization. We present in the following a brief description for experimented text categorization algorithms: Naïve Bayes, N-grams based, TF-IDF and K-Nearest Neighbours.

2.1. Naïve Bayes

The model Naive Bayes is constructed using learning documents to estimate the probability that a new document V be classified in a category c_j. We use Bayes's theorem to estimate the probabilities [15]:

DOI : 10.5121/ijdkp.2013.3303

$$p(c_j/V) = p(V/c_j)/p(V), p(V) = \sum_{c \epsilon C} p(V/c)p(c), p(V/c) = \prod_i p(x_i/c) \quad (1)$$

The quantity $p(c)$ refers to the probability that an arbitrary document is classified in the category c before even that its features are known. It is generally the same for every category. The training model step tries to calculate the probability $p(x_i/c)$ of every feature i in every category c. The "naïve" formula (1) is correct only if we suppose that the features given the category c are independent. Although the conditional independence of document features supposition, the Naïve Bayes model is surprisingly efficient.

2.2. N-gram based

This method uses a different way to divide the text into n-grams. An n-gram is a sequence of n consecutive characters [16]. For a given document, all n-grams that can be generated is the family of character sequences obtained by moving, by step of one character, a window of n characters on the text body. Thus, each document or category can be represented by a profile composed by the most frequent n-grams [16]. Alternatively, each document is represented by a vector with each component represent the weight of the corresponding n-gram [17]. This technique generates a large number of components compared to text analyzing based on word separators, but it is very tolerant to spelling mistakes and it is perfect for all sequences types on discrete alphabets like DNA sequences. For a sequence of n-grams sc, the assigned category to sc is the one with $p(c_i/sc)$ is maximal. The Bayes algorithm is used to determine this conditional probability: $p(c_i/sc) \approx p(sc/c_i) \times p(sc)$

2.3. TF-IDF

This algorithm uses the words vector to describe categories and documents. The principle here is closer to Rocchio algorithm [18], such that the set of words extracted from documents of a category c are accumulated into a single vector. The vector created is then normalized according to the equation:

$$x_i = \sqrt{f_i} \times idf(x_i) = \sqrt{f_i} \times log(K/n_i)$$

With x_i is the vector component representing the word in a category c and f_i is the word frequency in c. The square root is used here to reduce the effect of high frequencies, n_i present the number of categories in which the word i appears and K the number of categories. Comparing distances between the vector of a document d and each category's vector provides the best category can be affected to d.

2.4. K-nearest neighbours

Basic approach. The approach k-nearest neighbors (KNN) is the best known categorization technique has proven its effectiveness against the textual data processing [13]. Instead of building explicit models, the classification of each target document consists of watching from documents already classified, the category of k documents that are most similar to the target. The choice of the value of k is the difficulty of this approach.

To decide if a test document d belongs to a category c, the similarity between d and each document d_j of the training set is determined. A smaller distance (greater similarity) between two documents indicates that they are more similar. Then, the k most similar records to d are selected. As each document d_j is already labeled with a category, the category with the largest proportion of k documents is assigned to d. There are several similarity measures that can be used in KNN. In [12] some similarity measures were evaluated and the performance of KNN differs depending on the used measure.

KNN based on Lucene score. The idea is to use the score of the search engine Lucene as a similarity measure. Initially, the training documents are converted into a Lucene index in which each document's category is stored. Next, the text of a new document is converted into a Boolean query composed of the most relevant words and then sent to the search engine. Response list represents the potential documents sorted by descending order of their scores. The most popular category of k response documents is assigned to the new document. According to the Lucene API documentation[1], the score represents the similarity between the document and the sent query. In [12] this categorization model was detailed and experienced, it presented better performance than other more complex methods. In the following of this paper we use "KNN-LB" notation to indicate the KNN categorization based on Lucene score as similarity measure.

3. ONTOLOGY INTEGRATION APPROACHES

3.1. Introduction

In information science, ontology is often defined as the formal representation of knowledge in a domain with a set of concepts linked by semantic relations. Several research studies work on ontologies, and construction methods [19, 20, 21] are the largest interest of researchers in this field. However, these methods remain mostly theoretical and not generalized. Thousands of ontologies are made publicly available with a wide variation on its qualities. This provides the appearance of ontology evaluation techniques to ensure the quality of content and construction methods. Ontology evaluation can be on several criteria [22-24] such as the ontology structure, its coverage of a particular area and its wealth, performance of tasks which is designed for and its alignment level with other ontologies.

Another interesting point on which ontology evaluation can be approved is its use as a black-box. This generally applies to ontologies that are totally embedded in an application performing specific tasks [25]. An example of such application could be a categorization model that uses specific domain ontology for categorization of documents. The approach we present in this paper is around this point. It is a simple approach for ontologies integration in categorization models. Then according to the results, the system could be considered as an ontologies evaluator.

3.2. Used ontologies

Ontology construction studies the development process, the life cycle and the meth-ods and methodologies for building ontologies. This work focuses on ontologies usage as a black-box rather than their construction. Used ontologies consist of concepts linked by specific relations. A concept has a similar structure to the representation proposed by w3c[2] to characterize concepts (eg, Figure 1).

Concept	
Id :	String
Label :	String
Comment :	String
subClassOf :	List<Concept>
sameAs :	List<Concept>
seeAlso :	List<Concept>

Figure 1. An ontology concept schema

[1] http://lucene.apache.org/java/2_4_0/api/org/apache/lucene/search/Similarity.html
[2] http://www.w3.org/2004/02/skos/core/

A concept is characterized by a unique identifier in the ontology (Id), a name (label), a definition (Comment), the list of super concepts (subClassOf), the list of semantically close concepts (sameAs) and list of concepts that appear explicitly after "See" or "See also" in the definition of the concept (seeAlso).

We constructed four different ontologies based on glossaries and dictionaries related to telecommunications, internet and computer areas. These glossaries covering the categories selected from RCV1 collection for experimentations. These constructed ontologies are publicly available on the link: http://code.google.com/p/scsccorpus/downloads/list

Ontology 1 (ONT1). This ontology was created based on the book [26]. This is a glossary of terms and abbreviations in the field of telecommunications. The built ontology contains 15552 concepts and 16172 relations with in 1683 relations for type sameAs, 14327 seeAlso relations and 162 subClassOf relations.

Ontology 2 (ONT2). This ontology is created based on the 20th edition of the dictionary named "Newton's Telecom Dictionary" [27]. This ontology covers the areas of telecommunications, computer and Internet. It contains about 21997 concepts and 22440 relations between concepts.

Ontology 3 (ONT3). This ontology is based on a dictionary named "Internetworking Terms" published by the famous translator Babylon3. The final ontology contains about 464 concepts and 2512 relations.

Ontology 4 (ONT4). Another Babylon dictionary, published under the name "Glossary of Internet and PC Terminology" is used here to build an ontology covering Internet and computer areas. This ontology contains 652 concepts and 1859 relations between concepts.

The following table (Table 1) summarizes the number of concepts and relations contained in the created ontologies.

Table 1. Built ontologies

Ontology	Concepts	Relations			
		Total	sameAs	seeAlso	subClassOf
ONT1	15552	16172	**1683**	14327	162
ONT2	21997	22440	22	15373	7045
ONT3	464	2512	312	1439	761
ONT4	652	1859	600	1253	6

3.3. Documents annotation

The annotation is the describing of the web document content by semantic patterns written in languages comprehensible by human and computer. These languages include XML, RDF or OWL. For the moment, the majority of web documents contain plain text without annotation.

We use the ‘annotation’ notion because the used approach is very similar and it is one of the fully automated annotation techniques. The approach allows linking the plain text of a document to the concepts of a predefined ontology. It begins with finding ontology concepts homograph to each word of document. Next, a score calculated by the formula (4), is affected to each concept homograph based on its context (2) and the context (3) of the word υ.

[3] http://dictionary.babylon.com/index.html/

Definition 1. A concept o of an ontology Θ is homograph to a word υ if υ is the label of o, or if υ is one of the labels of its concepts synonyms. In our case the synonymy is represented in Θ by the relation type sameAs.

Definition 2. The context Γ(o) of a concept o is composed by these super $\Phi(o,\tau)$ and sub-concepts $\varphi(o,\tau)$ using one or more relations τ defined in Θ, while the context Γ(υ) of a word υ is composed by extracted words of document. Lets ϑ the word set of a document:

$$\Gamma(o) = \cup_{\tau\in\Theta}\{\Phi(o,\tau) \cup \varphi(o,\tau)\} \quad (2)$$

$$\Gamma(\upsilon) = \vartheta - \{\upsilon\} \quad (3)$$

$$\chi(o,\upsilon) = |\Gamma(\upsilon) \cap \Gamma(o)|/|\Gamma(o)| \quad (4)$$

Finally, only concepts homograph with the score (3) is maximal will be added to the list of concepts annotating the document. If a concept is already exists in the list its occurrence is incremented. By maximizing the score (3), this annotation approach overcomes the problem of concepts ambiguity. Whatever the used text pre-processing level, ambiguity occurs when several concepts are homographs to the same word in the document. In the following the annotation algorithm:

Input
ϑ**:** List of words of a document d.
Θ: Domain ontology
Output
ϑ′**:** (⊂ Θ) list of concepts annotating d**.**

```
Algorithm  Annotate (ϑ, Θ, ϑ′){
  For each word υ ∈ ϑ{
    // Find the concepts homographs to υ
    Array H[0..n] ←homograph(υ,Θ)
    Array χ[0..n]:Scores of concepts of H
    sm ← 0: Max score
    For i←0 to n {
      // calculate the score χ
      χ[i] = |Γ(υ)∩Γ(H[i])|/|Γ(H[i])|
      sm ←(χ[i] > sm)? χ[i]:sm
    }
    // add to ϑ′ the concepts of H whose χ is maximum and not null.
    If(sm>0){
      For i←0 to n {
        If(χ[i] = sm){
          Update(ϑ′, H[i])
        }
      }
    }
  }
}
```

3.4. Integration strategies

When indexing a document, relevant words are linked to the ontology concepts by applying the algorithm presented above. Thereby, for a document's vector v from the a dataset Δ, a vector v′ is created in the space represented only by concepts of an ontology Θ:

$$\phi: \Delta \rightarrow \Theta, \phi(v) = v'$$

Ontology integration consists of how to mix v and v′ on a modeling task. We can differentiate four strategies for integrating ontology in the categorization model:

Strategy 1: concepts adding. During the model training and testing phases, v is extended by new entries of the vector v′.

Strategy 2: concepts replacement. This strategy is similar to the previous one, but we replace here the entries of the vector document v by their concepts homographs from v′ linked to the document. Unlinked words are kept in v.

Strategy 3: ontology concepts only. Here the vector document v is replaced by the vector v′.

In [28], the authors was experimented this three strategies in the text clustering. They have integrated WorldNet ontology on a tested segmentation models and the first strategy (strategy 1) showed better performance compared to other strategies.

Strategy 4: Combining models. The principal her is based on combining scores of two independent categorization models. Let M a set of categorization models based on vectors v, and M′ the set of those based on vectors v′. Categorizing a document d, represented by it vector v_d and v'_d, by combining two models m ϵ M and m′ ϵ M′ follow four steps:

1. Classify the vector v_d by applying the model m
2. Classify the vector v'_d by applying the model m'
3. For each category c_i
   a. Get and normalize scores $s_m(c_i)$ and $s_{m'}(c_i)$ computed respectively in step 1and 2
   b. Compute its final score $s(c_i)$using the formula:
   $$s(c_i) = (1 - x) \times s_m(c_i) + x \times s_{m'}(c_i) \; with \; 0 < x < 1$$
   x indicates the proportion of the combination of m' in m.
4. The category with the best score $s(c_i)$ is affected to d.

4. METHODOLOGY

4.1. Dataset preparation

We used the corpus RCV1 as it was developed by [29] and a version can be down-loaded from [30]. It contains 23149 training documents and 781265 test documents, for a total of 804414 documents already presented in the form of bag of words after an automatic Natural Language Processing (NLP). We have selected from this corpus a subset consist of categories very close in content and related to telecommunications, Internet and computer areas. The following table (Table 2) summarizes these categories:

Table 2. Used subset of RCV1

Topic	code	Train	Test	Total
Computer systems and software	I33020	3729	1786	5515
Datacommunications and networking	I33030	1199	567	1766
Telecommunications equipment	I34400	2158	1004	3162
Telecommunications	I79020	4726	2026	6752

The column named Topic represents categories and the next column indicates their identifiers in RCV1 dataset. To reduce noise and get more performances of categorization, these categories contain only documents linked at less to one or more concepts of used ontologies. This is obtained by applying the annotation algorithm presented previously in section 3.3. The three last

columns correspond to the total of documents obtained for each category and its train and test subsets. Used dataset is available at: http://scsccorpus.googlecode.com/files/ANNOTATED.rar

4.2. Documents indexing

To experiment categorization models NB, N-Gram, TF-IDF and KNN, we used the Lingpipe[4] API for the extraction of word vectors related to the documents in the used corpus. Also, Lucene indexing is used to experiment the KNN-LB model. The following table (Table 3) presents the tested models, the type of indexing and used Java classes.

Table 3. Tested models and associated indexing types

Model	Indexation	Java classes	API
NB	Words frequency	NaiveBayesClassifier, LMClassifier<LanguageModel, MultivariateDistribution>	Lingpipe
N-Gram	N-grams frequency	DynamicLMClassifier <NGramBoundaryLM> LMClassifier<LanguageModel, MultivariateDistribution>	Lingpipe
TF-IDF	Words tf-idf	TfIdfClassifierTrainer <CharSequence> Classifier <CharSequence, ScoredClassification>	Lingpipe
KNN	Words tf-idf	KnnClassifier <CharSequence> Classifier <CharSequence, ScoredClassification>	Lingpipe
KNN-LB	Terms tf-idf	IndexWriter, Document, Field, StandardAnalyzer, Searcher, Query, Hits	Lucene

NaiveBayesClassifier class is used to create the NB model during learning. Then the model is loaded by *LMClassifier* class for test phase. The latter class is also used for testing the N-Gram model while *DynamicLMClassifier* class is used to create it.

KNN model with the cosine distance and the TF-IDF model are loaded and tested by the *Classifier* class. *KnnClassifier* class is used to generate the KNN model while *TfIdfClassifierTrainer* class is used for the TF-IDF model.

For KNN-LB, whose similarity is based on Lucene score, the creation and testing of the model are made differently. Such that the generated model is a Lucene search engine index. This index is created based on training documents and by using Java classes: *IndexWriter*, *Document*, *Field* and *StandardAnalyzer*. The test of the model is realized using *Searcher*, *Query* and *Hits* classes. It sends the content of test document in a query to the created index and retrieves the k closest documents in response.

4.3. Importing an ontology

Ontologies are stored in RDF[5] format. This format is based on XML and defined as the standard for data exchange on the web. The Jena API, version 2.6.4, is used to import and analyze the RDF ontology, and then the concepts and relations are stored in a relational database using Java DB API. This is done in order to simplify the search for concepts and annotating documents in the followed steps. The relational schema of the database is shown in Figure 2.

[4] http://alias-i.com/lingpipe/
[5] http://www.w3.org/RDF/

4.4. Documents annotation

Annotation algorithm was implemented on java classes. We have used all defined relations in the ontology to determine super and sub-concepts when calculating the context of a concept. The annotation result is stored in the same database containing imported ontologies (Figure 2).

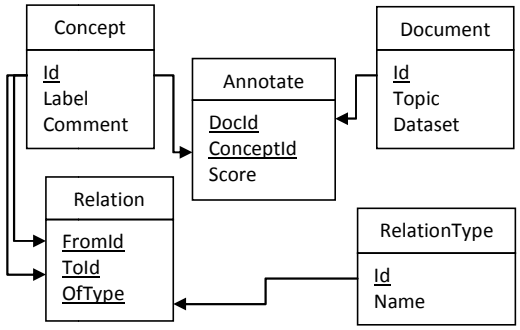

Figure 2. Relational schema of the database storing imported ontologies and annotation results.

Entity named (*Concept*) represents the concepts of the imported ontology, relations between concepts (e.g. *subCassOf*, *sameAs*, *seeAlso* ...) are represented by the entity named (*Relation*). When annotating a document, its identifier and category are stored in the entity named (*Document*), while the identifiers of concepts, result of annotating process are stored in the entity named (*Annotate*).

5. EXPERIMENTS AND RESULTS

In this work we are limited our experiments to strategies 1, 3 and 4. First, we tested models presented above in table 3 without application of strategies introduced in section 3.4. Next, we applied strategy 1 consists of adding concepts to documents vectors, then strategy 3 using annotation results only and finally combining the categorization models following strategy 4.

5.1. Strategy 1 and 3

The following figure (Figure 3) shows the values of the micro-F1 corresponding to the different models tested on the following three cases:

- Case 1 (Only): experiment without the use of ontology.
- Case 2 (Str1): use of ontology ONT1 following strategy 1.
- Case 3 (Str3): use of ontology ONT1 following strategy 3.

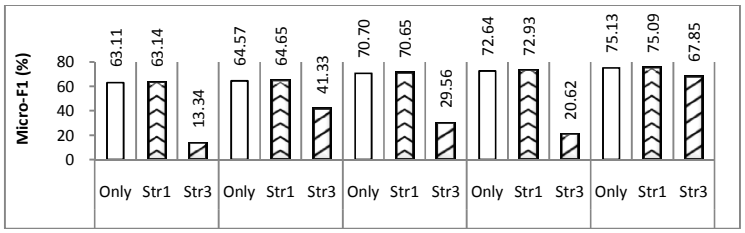

Figure 3. Micro-F1 measures of models according to cases: Only, Strategy 1 and Strategy 3

KNN-LB with K=15 presented a higher value (75.13%) of the micro-F1 compared to other models in case 1 (Only) where we have not used ontology. TF-IDF, KNN and NB showed a slight improvement by integrating ontology ONT1 according to strategy 1. These low improvements are justified by the high level of text pre-processing used on works of [30] and a lot of stemmed words can't be matched on the ontologies when searching for homographs in annotation process.

The categorization following strategy 3 shows low F1 values for the most of tested models compared to the categorization without ontology integration and to strategy 1. This can be justified since, according to Strategy 3, we lose too much information when representing documents only by the relevant concepts of the integrated ontology. An exception can be noticed in the case of KNN-LB, as the performance of this model have not been overly influenced by the information losing when applying strategy 3, and the value of F1 (67.85%) is concurrent with those of other models even in cases 1 and 2. The same behaviour discussed above was observed also when using the three other ontologies ONT2, ONT3 and ONT4 but with a different obtained values of the micro-F1 measure.

5.2. Strategy 4

Combining the categorization models following strategy 4, is influenced by the gap between the performances of participant models. The principal of combination was introduced in section 3.4 above. Several tests were conducted to find the value of $x \in]0,1[$ maximizing F1 value of the composed model. Each test concerned the value of x incremented by a step of 0.05. The following figure (Figure 4) shows results of an example of experiments carried out for N-Gram (n=6) which was combined KNN-LB (k=15).

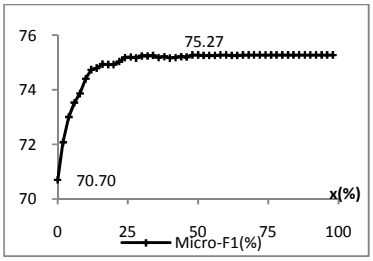

Figure 4. Evolution of Micro-F1 of the N-gram (n=6) model improved by combination with KNN-LB (k=15). For x=0.48 (48%), micro-F1 of the composed model has reached its maximum (75.27%).

We tested all possible combinations between the models TF-IDF, NB, N-Gram, KNN (k=10) and KNN-LB (k=15) without integration of ontologies. The following figure (Figure 5) shows the maximum value of the micro-F1 for each combination.

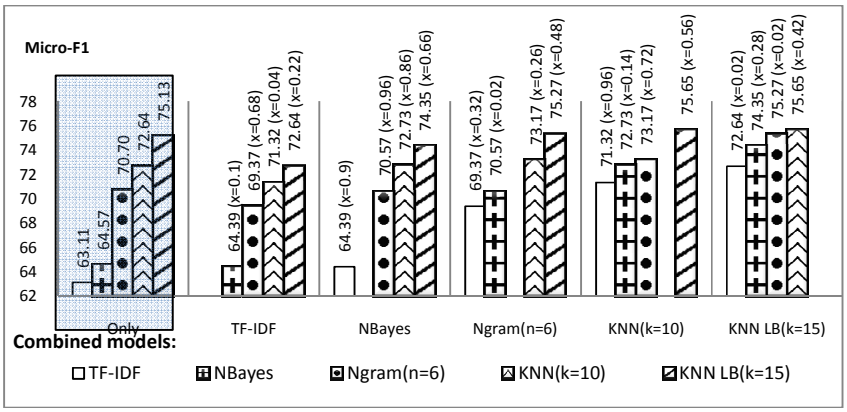

Figure 5. F1 maximum of each combination between the five models TF-IDF, NB, N-Gram(n=6), KNN(k=10) and KNN-LB(k=15) following strategy 4.

We observe that if the micro-F1 of the combined model is superior, the composed model's performance will often be better than the base model. We can take the example of TF-IDF with

the value of the micro-F1 (63.11%), when other models with a higher micro-F1 are combined to TF-IDF, there is always x ϵ]0,1[maximizing micro-F1 of the composed model with a higher value than 63.11%.

The converse of this observation is not always correct; this means that when combining a model with lower micro-F1 than the base model, the performance of the composed model can be improved. Taking the example of N-Gram (70.70%) combined to KNN (k=10), at x=0.26 the micro-F1 (73.17%) of composed model is higher than that of the base model KNN (72.64%). However, in most cases where the difference between the values of the micro-F1 is more important, the micro-F1 of the composed model degrades and becomes lower than that of the basic model that we want to improve by this combination.

In order to improve experienced categorization models by ontology integration following the strategy 4 and for all reasons discussed above, we chose to use the KNN-LB with k=15 as the combined categorization model with the better micro-F1 measure (see Figure 3). Thus, the difference between the performance of the combined model and those that we want to improve is optimal.

The following figure (Figure 6) summarize the experimentation results for categorization model TF-IDF, NB, N-Gram (n=6), KNN (k=10) and KNN-LB (k=15). For each model we represent it F1 value when used alone (Only), then when integrating ontologies following the strategy 1 (Strategy1) and finally when it combined with KNN-LB (k=15) model following the strategy 4.

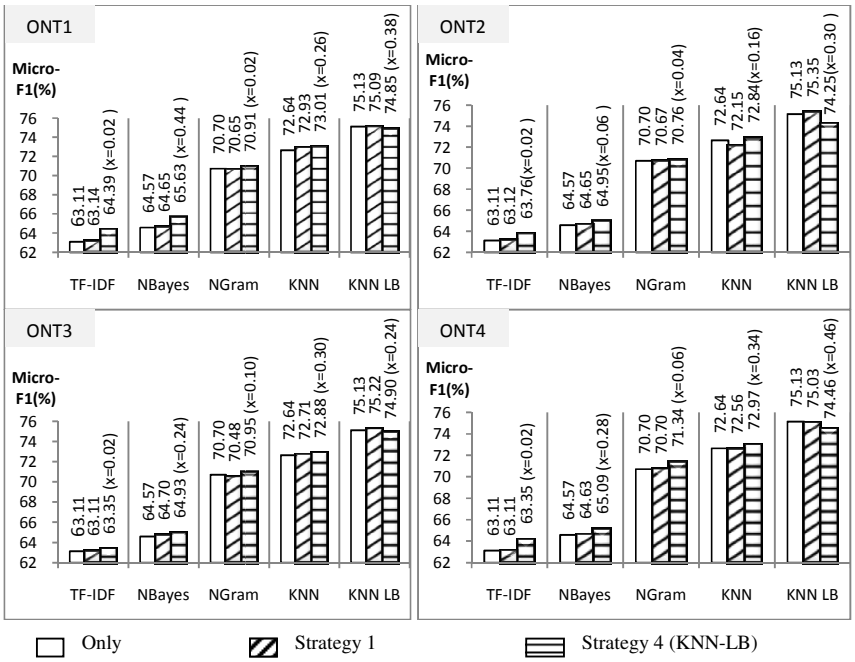

Figure 6. Micro-F1 values of categorization models after integration of ontologies according to the strategy 1 and the strategy 4 with KNN-LB as the combined model.

We note that for values of x less than 0.5 and different depending on models, we were able to achieve improved values of the micro-F1 for tested models under strategy 4. Improvement is observed also under the strategy 1 for each case of integration of the four ontologies. With the exception of KNN-LB model, the values according to Strategy 4 exceeded those obtained in the case of the use of strategy 1.

6. CONCLUSION AND DISCUSSION

In this paper we have presented an experimentation of three approaches for improving text document categorization models by integrating ontologies. Some categories from Reuters Corpus Volume I (RCV1) were chosen to test these approaches. These categories are related to telecommunications, Internet and computer areas. Four domain ontologies covering these areas were built and integrated for improvement of models. In test and train phases, documents are annotated by concepts of ontologies in a fully automatic algorithm. This later need to be compared with others similar algorithms and that will be the subject of our future work.

Integration strategies which consist in adding concept of ontologies to the vector-document and combining categorization models, showed a low improvement in performance of tested models. These low improvements are justified by the high level of text pre-processing presented on dataset and a lot of stemmed words can't be matched when searching for homographs in the annotation process. Another gain of this work was the construction of large ontologies on IT domains based on known glossaries and dictionaries. These ontologies are presented in standard structure and easily exploitable in a portable RDF format.

However, it is very early to speak about efficiency and performance of these approaches because it is clear that many important tests and justifications still to be presented, like a comparative study with others similar approaches and applying this approaches to other domains with validated ontologies.

REFERENCES

[1] S.-J. Yen, Y.-C. Wu, J.-C. Yang et al., "A support vector machine-based context-ranking model for question answering," Information Sciences, vol. 224, pp. 77–87, 2013.

[2] D. Sánchez, and M. Batet, "A semantic similarity method based on information content exploiting multiple ontologies," Expert Systems with Applications, vol. 40, no. 4, pp. 1393–1399, 2013.

[3] D. Milne, and I. H. Witten, "An open-source toolkit for mining Wikipedia," Artificial Intelligence, vol. 194, pp. 222–239, 2013.

[4] E. Kontopoulosa, C. Berberidisa, T. Dergiadesa et al., "Ontology-based sentiment analysis of twitter posts," Expert Systems with Applications, vol. In Press, Uncorrected Proof, 18 January 2013, 2013.

[5] C. H. Li, J. C. Yang, and S. C. Park, "Text categorization algorithms using semantic approaches, corpus-based thesaurus and WordNet," Expert Systems with Applications, vol. 39, no. 1, pp. 765–772, 2012.

[6] H. Kim, and Su-ShingChen, "Associative Naive Bayes classifier:Automated linking of gene ontology to medline documents," Pattern Recognition, vol. 42, pp. 1777-1785, 2009.

[7] W. B. Cavnar, "Using an n-gram-based document representation with a vector processing retrieval model." pp. 269-277, 1994.

[8] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features." pp. 137-142, 1998.

[9] Y. Yang, and X. Liu, "A re-examination of text categorization methods." pp. 42-49, 1996.

[10] Y. Yang, "An evaluation of statistical approaches to text categorization," Information Retrieval, vol. 1, no. 2, pp. 69-90, 1999.

[11] Y. Yiming, and P. Jan O., "A comparative study on feature selection in text categorization." pp. 412-420, 1997

[12] Machhour, H., Kassou, I.: Text categorization using the kNN classifier based on Lucene score. Internal report (2012)

[13] Yang, Y.: An evaluation of statistical approaches to text categorization. Information Retrieval 1, 69-90 (1999)

[14] Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Tenth European Conference on Machine Learning (ECML). pp. 42-137, (1998)

[15] Mitchell, T. M.: Machine Learning. McGraw Hill (ed.), New York (1996)

[16] Cavnar, W.,Trenkl, J.: N-Gram Based Text Categorization. In: Document Analysis and Information Retrieval. Las Vegas (1994)

[17] Cavnar, W. B.: Using an n-gram-based document representation with a vector processing retrieval model. In: Third Text REtrieval Conference (TREC-3). pp. 77-269, (1994)
[18] Rocchio, J. J.: Relevance Feedback in Information Retrieval. Vol. G. Salton, Prentice- Hall Inc, (1971)
[19] Euzenat, J.,Shvaiko, P.: Ontology Matching. 9th (ed.), Vol. pp. Springer, (2007)
[20] Gómez-Pérez, A., Fernández-López, M.,Corcho, O.: Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web. 2nd, Springer, (2007)
[21] Stumme, G.,Maedche, A.: FCA-MERGE: bottom-up merging of ontologies. In: 17th international joint conference on Artificial intelligence. Vol. 1, pp. 30-225, Morgan Kaufmann Publishers Inc, (2001)
[22] Euzenat, J., Meilicke, C., Stuckenschmidt, H., Shvaiko, P.,Trojahn, C.: Ontology Alignment Evaluation Initiative: Six Years of Experience. Journal on Data Semantics XV 6720, 92-158 (2011)
[23] Gómez-Pérez, A.: Evaluation of Ontologies. Intelligent Systems 16, 391-409 (2001)
[24] Noy, N. F.,Musen, M. A.: Evaluating Ontology-Mapping Tools: Requirements and Experience. In: OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge. Angele J.,Sure Y. (ed.), Siguenza, Spain (2002)
[25] Gangemi, A., M. Pisanelli, D.,Steve, G.: An overview of the ONIONS project: applying ontologies to the integration of medical terminologies. Data & Knowledge Engineering - Special issue on formal ontology and conceptual modeling 31, 183-220 (1999)
[26] Arabi, A. A.: Comprehensive Glossary of Telecom Abbreviations and Acronyms. (2007)
[27] Newton, H.: Newton's Telecom Dictionary. 20th (ed.). (2004)
[28] Hotho, A., Staab, S.,Stumme, G.: Wordnet improves Text Document Clustering. In: Semantic Web Workshop of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval. Toronto Canada (2003)
[29] Lewis, D. D., Yang, Y., Rose, G. T.,Li, F.: RCV1: A New Benchmark Collection for Text Categorization Research. The Journal of Machine Learning Research 5, pp. 97-361(2004)
[30] Lewis, D. D.: The LYRL2004 Distribution of the RCV1-v2 Text Categorization. (2004)

Authors

Mr. Hamid Machhour is a Phd Student in Department of Computer Science and Decision Support, ENSIAS, Mohammed V Souissi University, Rabat, Morocco. His current research interests include knowledge management, strategic intelligence, text mining applications, web document categorization, and focused crawling algorithms.

Pr. Ismail Kassou is working as a Professor, Deputy Director in charge of cooperation and partnership in ENSIAS and Head of Doctoral Studies Centre ST2I, Mohammed V Souissi University, Rabat, Morocco. His research interests include knowledge management, text mining, web mining, search engines, and strategic intelligence.