Ontology-based Flexible Topic Classification of Crowdsourcing Textual Resources

Stefan Daniel Dumitrescu, Stefan Trausan-Matu, Mihaela Brut, Florence Sèdes

Ontology-based flexible topic classification of crowdsourcing textual resources

Stefan Daniel Dumitrescu
Romanian Academy Research Institute for Artificial Intelligence
13 Calea 13 Septembrie
Bucharest, Romania
sdumitrescu@racai.ro

Stefan Trausan-Matu
Politehnica University of Bucharest
313 Splaiul Independentei
Bucharest, Romania
stefan.trausan@cs.pub.ro

Mihaela Brut
Thales Services
1 Avenue Augustin Fresnel
Palaiseau, France
mihaela.brut@thalesgroup.com

Florence Sedes
Research Inst. In Computer Science
118 Route de Narbonne
Toulouse, France
sedes@irit.fr

ABSTRACT
The paper presents a solution to the problem of capitalizing in different contexts and by different stakeholders the time-stamped new documents produced by social Web sites (including news, blog entries, and uploaded documents). The solution core includes an ontology-based method to express the interest topics and to automatically classify them. For such textual content obtained in real-time, we propose an unsupervised text classification system based on general YAGO ontology, graph algorithms and a custom scoring method. The system shows good performance using only ontology information and the ontology structure itself. We compare our system against a SVM-based (Support Vector Machine) classic text classification approach. For determining the relevance of a specific document for a specific topic, our approach develops and compares the ontology sub graphs corresponding to the query and to the document. It leads to a high flexibility in terms of capitalizing the already classified documents when refining and changing the interest topic: a graph-based matching of the already obtained ontology-based document representation against the new query representation is enough to assess the document relevance.

Categories and Subject Descriptors
I.2.4 [Knowledge Representation Formalisms and Methods]: Semantic networks, ontologies; I.2.7 [Natural Language Processing]: Text analysis; I.7 [Document and text processing]: Document preparation, Index generation.

MEDES'13, October 28-31, 2013, Neumünster Abbey, Luxembourg.

General Terms
Algorithms, Experimentation, Theory.

Keywords
Ontology, text classification, concept, knowledge engineering, graph algorithm

1. INTRODUCTION
Information extracted and inferred from social networks is deemed of utmost relevance as a valuable input for multiple stakeholders acting at a city level. Journalists, economic analysts, sociologists, police, city municipality want to be informed or to easily locate by themselves the latest news concerning a specific topic of interest for the current moment. Moreover, it is very important to scan the social websites and assess in real-time the population feedback and reactions to a public/private institution initiative. Actually, the ontology-based semantic description of texts provides a very good basis for a more efficient and pertinent exploitation of them for different functionalities such as information retrieval or various functionalities tailored to the current user needs. This article presents a solution for automatically classifying this type of content according to the YAGO general ontology and for determining documents' relevance for different topics that dynamically become of interest for different stakeholders.

Our solution approach is to delimitate the extended set of interest topics corresponding to the user query (through a query enrichment process), and to consider only the ontology concepts associated with these topics among the document annotations, as well as their maximum two-degree related concepts. Thus, the document annotation process is alleviated while non-obvious semantic relations between topics and documents are detected.

The proposed method is based on an implemented system that adopts YAGO as a support ontology in order to extend an initial query topic and to perform unsupervised text classification given an initial topic list. It uses the ontology as a knowledge source on

which it applies graph algorithms to detect and create a partial sub-graph illustrating the relations between the concepts that characterize each document. Thus, our solution avoids the use of machine learning algorithms in the main processing phase, while only employing such algorithms like a Maximum Entropy model for sentence identification and token splitting in the document pre-processing phase.

On the other hand, once analyzed, the textual resources keep their acquired YAGO ontology-based representation. When different stakeholders express new interest topics, these topics are as well processed and enhanced based on the YAGO ontology, while documents annotations are only performed based on the new topics. The relevant documents are detected further to a graph-matching algorithm.

We also engage in a discussion on the benefits and problems of using ontologies for such a task.

2. LITERATURE REVIEW

2.1 Watching and Gathering Data from Web

With the increase of the Web content volume and types, the methods and tools for gathering data from Web are more and more efficient. The Web crawlers, among which we could mention the open-source Heritrix or DataSparkSearch[1], are focused on specific issues and efficient due to the multiple policies they are able to handle, such as for page selection and re-visit, for politeness (in order to avoid the Web sites overloading), for crawling parallelization (in the case of distributed Web sites). The Social Web contributes as well to the Web data gathering alleviation due to their techniques to notify about the Web content evolution. The site updates and new articles are signaled through, for example, the RSS feeds. The specific content that becomes popular is highlighted, for example, through the featured articles or through articles signaled as most popular. The private or public institutions are notified about the users' reactions to their published initiative (through comments, responses, scoring).

Crowdsourcing is a powerful method to gather useful information by directly involving voluntary users. As an example in the context of smart city innovation, crowdsourcing methods for idea generation and idea selection are investigated and applied [12]. The effort to actively engage people in systems as participants is fully rewarded by crowdsourcing contributions capitalizing their knowledge and expertise [13]. As an example, the campaign "Investigate your MP"[2] organized by the newspaper The Guardian gathered multiple tens of thousands of people to review the 458.832 pages of the report on the scandal about excessive expense submissions. Concerning the topic on how to make smarter a city, ideas from people are gathered through an impressive number of Web sites[3], whose focused crawling could be very interesting for specific city stakeholders.

The main issue to handle such data published on the Web is not its gathering, but its automatic processing according to some pre-defined (and still evolving) interest topics.

[1] https://webarchive.jira.com/wiki/display/Heritrix/Heritrix, http://www.dataparksearch.org/

[2] http://www.guardian.co.uk/politics/mps-expenses

[3] a few examples: http://www.smartcities.info, http://www.citymayors.com, http://www.smart-cities.eu, http://www.smartcitiesineurope.com, etc.

Big Data Analytics is the emerging technology to support the storage and processing of large volumes of information and to make the information available when needed to the systems demanding it. In this context, Hadoop technology is specialized for massive processing in a distributed infrastructure and HBase Database for the storage of large amount of data. Additionally, STORM supports data processing in real time with any programming language and the easy integration with different database management systems. Existing SmartCities solutions, such as the IBM, Intelligent Operations Centre for Smarter Cities, are more transversal solutions (emergency management, public safety, social services, monitoring transportation or water networks) that provide executive dashboards for city leaders, but do not cover the previous processing demands.

Nevertheless, semantic-oriented data analysis functionality is not supported by the current Big Data Analytics solutions, whereas they could substantially capitalize in the context of smart cities, as illustrated above. In order to expose the text analysis context of our solution, we further present an overview of the topic classification approaches.

2.2 Topic Classification Approaches

The domain of text classification is, at present, dominated by machine learning methods, statistical methods, with knowledge engineering methods trailing behind [1]. While a large variety of approaches could be noticed, the best performing systems consistently use algorithms like SVM (Support Vector Machine) to achieve consistent and good results. Machine learning algorithms like SVM, Naïve Bayes, Maximum Entropy are relatively simple to understand and use, and unlike knowledge engineering methods, they do not require large knowledge-bases to be pre-defined manually by engineers. Also, this kind of systems is not domain related, unlike most engineering approaches that are focused on restricted domains or even sub-domains (as it happens, for example, in the medical domain where compact parts of the consecrated ontologies are adopted for certain medical specializations). The good functioning of these algorithms usually requires the "translation" of the documents into feature vectors. Common feature vector construction involves term frequency, document frequency, term frequency and inverse document frequency combined, information gain, term strength, and chi-statistic [2], [3].

Latent Semantic Indexing (LSI) has been used in conjunction with WordNet or other domain ontologies to reduce the dimensionality of feature vectors [4][5]. The main idea of LSI is that there is a semantic structure between the words in a document that can be discovered and used to group similar documents into similar space structures using statistical analysis. Using LSI means that after document preprocessing, the document vector is obtained (in the form of $d=\{(keyword_i, weight_i) | i=1..n\}$, its dimensionality is reduced using LSI and then it is compared to every category vector topic. The category vector that is closest to the document vector is the topic assigned to that document. [6] showed an slight increase in performance when using LSI and an ontology as opposed to simply using a Naïve Bayes classifier (or equivalent) and an ontology. In [7] we developed a text classification method where the LSI technique was combined with a WordNet-based text analysis. However, while LSI is effective in mitigating word similarities, it is quite difficult to maintain such a system when the document size varies and any modification of the initial set of documents requires that the entire semantic space to be reconstructed [8].

Concerning the ontology-based approaches to text classification, it can be noticed that most often domain ontologies are used [9]. Domain ontologies are usually small and contain very specific facts about a domain, like certain group of illnesses for the medical domain, names and hierarchy of wines for the oenological domain or car parts for the automotive domain. When applied to a collection of texts from a certain area, a domain ontology focusing on that area will be much more effective than a general ontology. However, for diverse collections of documents, the use of domain ontologies is no longer possible.

For the purpose of this paper we will use the YAGO ontology [10]. YAGO is a general, light-weight and extensible ontology with high coverage and quality. YAGO was built as a very large, accurate (95+ accuracy) and simple to use ontology for machines including WordNet entities and hierarchy, and information extracted from Wikipedia like named entities (people, organizations, geographic locations, books, songs, products, etc.), and also relations among these entities. In the YAGO model all objects are entities and any two such entities can stand in a relation. YAGO uses two sources of information: WordNet and Wikipedia. From WordNet it borrows the hypernym hierarchy, while from Wikipedia it borrows entities and uses them as arguments to the relations implemented in YAGO. Each WordNet synset is translated in a YAGO class. Thus, WordNet defines the upper hierarchy, while Wikipedia contributes to the lower, most specific branches. These are also linked up using the subClassOf relation. WordNet synsets have words with similar meaning. In summary, YAGO stores more than 2 million entities with 20 million facts about them.

Our solution is destined to multiple stakeholders acting at the city level (e.g. journalists, economic analysts, sociologists, police, city municipality representatives), providing them access to a commonly developed knowledge base where the emerging social Web resources are indexed based on the YAGO ontology.

The first provided support concern the interest topic formulation, since for a specific stakeholder it is very difficult to exhaustively express it. For an initial topic (e.g. "pesticides in the baby food"), a list of YAGO concepts accompanied by their corresponding weight is provided, while a supplementary manual topic refinement of the topic is facilitated, based on the YAGO structure: either to enrich or reduce the topic, either to reconsider the weights associated with the composing concepts.

3. SYSTEM IMPLEMENTATION

This section discusses system architecture and implementation. The system can be logically divided into three major modules: Processor, Analysis and Evaluator.

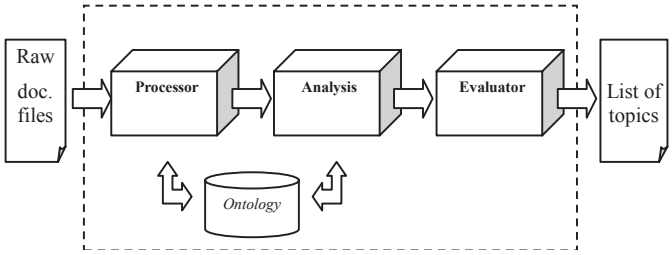

Figure 1. System architecture

A quick overview of the way the system works: First, at initialization phase, the topic list is constructed. Then a document is fed to the Processor where it is parsed and words are extracted from it, along other useful information, as word frequency and word type, form, etc. The words are then associated to classes or entities from the ontology. Next, in the Analysis module, each word's associated class is found in the ontology, and then a score is added to each topic's overall score, depending on the influence of the word to that respective topic. After all words have been processed, the topics are sorted by their descending scores in the Evaluator module. The topic with the biggest score is the document's proposed topic. Below, we present each module, starting with the initial topic list creation.

3.1 The topic list creation

The data set for each topic is created at program initialization, before analyzing any document. For the current system we had a collection of 848 news articles that are placed into 50 topics. For each topic an array of 'concepts' is created. A concept is in itself an array of similar words, much like the senses in a WordNet synset. However, we do not store simple words in the concepts arrays, but YAGO classes corresponding to the words. We assign each class in a concept array a weight, and to each concept in a topic another weight. This allows us a fine-grained control over word/concept influence.

For example, let us consider the topic 'Pesticides in baby food'. We create the concepts of 'pesticide', 'baby' and 'food', assigning each a weight of 1.0, meaning full importance. The concept 'baby' has, for example, the classes of 'wordnet_baby_109827683' with a weight of 1.0 and 'wordnet_child_109918554' with a weight of 0.7, meaning we prefer baby over child if possible, but child should also be considered relevant if found in a document.

The topic list was created semi-automatically, meaning that we wrote a side-program that took each topic, extracted words from it and located their probable classes; our work was only to remove or to add some words to better define a topic and adjusted weights in three degrees of separation (1.0, 0.7 and 0.4 – heuristically chosen).

This is the only part of the entire system that depends on human intervention. For each word in each topic the list of possible concepts was found automatically. We could have used some flavor of Word Sense Disambiguation [14] (even basic Lesk could work to a degree) to differentiate the weights between each word's possible classes, but due to time constraints as well as not wanting to introduce a new error source (WSD is far from perfect) we chose to select the weights ourselves (not perfect either, but closer) and better evaluate the ontology-based algorithm by itself.

3.2 Processor Module

This module takes as input a raw text document and outputs a list of words, each of the words having a set of probable classes in the ontology.

The document is first split into sentences using a Maximum Entropy (ME) model with very good results. Then, for each sentence, it is split into individual words (tokens), using another ME model. The tokens are then analyzed and their part of speech is determined, their form (singular or plural – if the word is in its plural form, it is inflected to its singular form) and whether the token is a Named Entity or a simple common word. All this information is needed to determine which words are searched for in the ontology (we restricted to only common nouns and Named

Entities), and what classes are assigned to each of them. Also, if a word is found more than 1 time then a) its frequency count is increased, and b) it is considered already 'disambiguated' – we consider that a repeated word always means the same thing.

For example, the word 'governments' is searched for in the ontology and it yields the following classes: 'wordnet_government_108050678', 'wordnet_government_101124794', 'wordnet_government_105663671' and 'wordnet_politics_106148148'. The search in the ontology is performed using the `means` relation. We ask YAGO to tell us all the classes it knows where that class means 'government'. Thus, for each word we obtain a set of classes that we keep.

Each document is defined as a set of words, each with their determined classes. This output is further fed to the Analysis Module.

3.3 Analysis Module

The module takes as input the list of words with their associated classes and as output it assigns a score to each topic concept. For each class in the set of each word, it first finds it in the ontology. Then, it performs a graph walk starting with this class as a center, limited to two levels of depth. Every node it reaches is analyzed. If the node belongs to one of the concepts in a topic, that concept's score is increased accordingly. The weight of the concept is increased by *I*:

$$I = (1/2)^{d_{wc}} * lg^2(1 + freq_w)$$

This is a heuristically chosen formula, where d_{wc} is the distance from the word *w* being analyzed to the matched concept *c*. We restrict the distance to maximum 2, so the distance modifier will either be 1 (when the distance is zero – meaning the word *w* is exactly the concept *c*), 0.5 (direct link between *w* and *c* – distance 1) or 0.25 (for distance 2). Further distances in the ontology usually yield very poor semantic connections, so we argue that a maximum distance of 2 is sufficient. The distance modifier is then multiplied by the squared logarithm of the frequency of the word *w*. We have used a logarithm to dampen the influence effect of exceedingly repeated words.

For example, we analyze the word 'vegetable', with two YAGO classes, one of which is 'wordnet_vegetable_107707451'. We start the graph search from this class, up to a distance of maximum 2. The system's console displays:

```
Entity: wordnet_vegetable_107707451

          Match 0.69 (d:0) e:
wordnet_vegetable_107707451 TOP[71 Vegetables,
Fruit and Cancer] > CON[vegetable]

          Match 0.69 (d:0) e:
wordnet_vegetable_107707451 TOP[90 Vegetable
Exporters] > CON[vegetable]

          Match 0.1725 (d:2) e:
wordnet_food_107555863 TOP[41 Pesticides in Baby
Food] > CON[food]
```

meaning than it has found three matches, two of them being an identical match (distance 0) and thus increasing the score by 0.69 of the 'vegetable' concepts of topics 71 and 90. It also increases the score of concept 'food' (topic 41) by only 0.1725 because class 'wordnet_food_107555863' is at distance 2 from the starting point of 'wordnet_vegetable_107707451'.

As such, for every word in the list it performs the graph walk. For every class in the ontology it encounters, if that class belongs to a concept of a topic, it increases the concept's score accordingly.

In the example below, the score composition of concept 'food' from topic 41 'Pesticides in Baby Food' is shown, as each matching concept adds a small increase *I* to the final score of 8.5574:

```
concept [food/1.0] 8.5574:
wordnet_food_107555863/1.0:8.5574

 >> Add 1.0*3.76=3.76 to 3.76 from
wordnet_food_107555863/42
 >> Add 0.25*0.69=0.1725 to 3.9324 from
wordnet_game_107650449/1
 >> Add 0.5*2.89=1.445 to 5.3774 from
wordnet_meat_107649854/17
 >> Add 0.5*1.1=0.55 to 5.9274 from
wordnet_fish_107775375/2
 >> Add 0.5*0.69=0.345 to 6.2724 from
wordnet_cheese_107850329/1
 >> Add 0.5*1.95=0.975 to 7.2474 from
wordnet_seafood_107776866/6
 >> Add 0.25*2.48=0.62 to 7.8674 from
wordnet_shellfish_107783210/11
 >> Add 0.25*0.69=0.1725 to 8.04 from
wordnet_beef_107663592/1
 >> Add 0.25*0.69=0.1725 to 8.2124 from
wordnet_delicatessen_107594406/1
 >> Add 0.25*0.69=0.1725 to 8.384 from
wordnet_vegetable_107707451/1
 >> Add 0.25*0.69=0.1725 to 8.5574 from
wordnet_pork_107668702/1
```

It should be noted that in the examples above we give only WordNet related classes (e.g. class wordnet_beef_107663592) from YAGO because the relations between them are easier to understand and there are more links between them than between named entities (e.g. class France, class Tunnel_Channel, etc). However, YAGO's imported WordNet hierarchy contains only around 65.000 classes from the more than 1 million entities known. Named entities contribute just as much as (and in some cases even more than) the common entities that our system uses.

3.4 Evaluator Module

The evaluator module takes as input the topic lists with their scores, and evaluates them. The output is a sorted list of relevant topics.

The scoring method we used here is to add the scores of each topic's individual concepts, taking into consideration the number of non-zero concepts: first, we calculate the general score for each topic that is composed of the simple sum of the scores of its concepts. Then we evaluate the first 3-5 best topics, and we calculate the average score difference between each topic, which will call the error margin. Then, for that document, if the general simple score of the best topic is greater that the score of the second topic plus the error margin, we assume that the first topic is the correct topic. If the second topic is within the error margin of the first, we count for each topic the number of concepts that have a score greater than 0. The topic we believe is correct becomes the topic that has the highest 'coverage'. For example, if we have two topics, even if the first topic has a higher score than the second, if the second topic (being within the error margin) has a greater percent of concepts with a score above zero, we will choose the second topic as correct (as a concrete example, if topic 1 has 4 out of 5 concepts greater than zero, and topic 2 has 4 out of 4 concepts greater than zero, then we choose topic 2, because topic 1 has only 4/5 = 0.8 coverage while topic 2 has 4/4 = 1.0).

Even though this is a heuristically chosen metric, it provides better results than just simple concept addition. The 'coverage'

idea was introduced to simulate the human tendency to assign a topic to a document if that document has more words that are found in the topic rather than just a few common words with the topic but repeated several times.

For the purposes of this article, the scoring method provides relatively good performance. However, for larger collections of data (above 10.000 documents) we believe that a linear formula with adjustable parameters will provide an even better topic differentiating metric. The adjustable parameters could be determined in a training phase to estimate either a better document-individual error margin, or even a global margin.

4. EVALUATION

Before evaluating the results, a quick description of the data collection on which we tested is needed. We used the LA94 TREC Information-Retrieval Text Research Collection, representing a sampling of news articles published by the Los Angeles Times in 1994. The collection includes 828 such articles, which are classified over 50 topics. The articles propose news on different topics such as entertainment, movies, television, music, politics, business, health, technology, etc.

In order to compare our results against a standard method of text classification used today, we have implemented a SVM based system for text classification. The system is built in Java and uses core functionality from WEKA [11]. Each document is parsed, and a feature vector is extracted. The vector is further elaborated upon, eliminating stop-words, using lowercase tokens, setting a minimum term frequency for allowed terms, pruning periodically, using a stemmer, and finally applying a TF*IDF transform. The SVM is then trained on the document collection, and evaluated using a random-seed, 10-fold cross validation. We have tried to build the evaluator system as best as possible using the latest feature vector techniques and the best classifier for this job, the SVM.

Table 1. Comparison between the proposed system and a standard SVM state-of-the-art method

System	Performance (correctly classified documents)
Our ontology-centric system	529 / 828 (63.88%)
SVM comparison system	661 / 828 (79.83%)

The SVM comparison system at this moment performs better, by a margin of around 16%. However, our proposed system achieves a respectable performance of 63.88% using only the ontology as a source of information. We designed this system as proof-of-concept, to test the possibility of using ontologies as the core of a text classification system, and to see the performance degree of such an approach.

While the system proves effective even at this stage, we believe that its performance can be improved, especially concerning the topic list creation. Performance is affected by the description of each the topic (meaning the 'concepts' of each topic).

Table 2 shows some of the performance gains after manually tweaking some of the topic's concepts. For example, while adding context to topic 50, performance is very slightly improved by almost 1%, while for topic 80 the correct topic classification rate is doubled to 34%. By concept tweaking we mean editing individual concepts. For example, for topic 80 we had the initial concepts of 'hunger' and 'strike'. After adding context, meaning concepts 'government', 'demonstration' and 'cause' (each with a slightly lower weight than the initial two concepts), the detection rate greatly increased.

Table 2. A short comparison between overall system performance grouped by topic, before and after topic tweaking, for 5 out of the 50 total topics

Topic id	Topic	# of docs	Performance before tweaking	Performance after tweaking
50	Revolt_in_ Chiapas	105	98 / 105 (93.3%)	**99** / 105 (**94.28**%)
43	El_Nino_and _the_Weather	11	4 / 11 (36.36%)	**6** / 11 (**54.54**%)
80	Hunger_Strikes	56	9 / 56 (16.07%)	**18** / 32 (**34.61**%)
70	Death_of_ Kim_Il_Sung	33	**28** / 33 (**84.84**%)	22 / 33 (66.66%)
58	Euthanasia	49	14 / 49 (28.57%)	**31** / 49 (**63.26**%)

However, after also tweaking topic 58, the performance negatively affected topic 70's recognition rate. This is due to the adding to topic 58 of the concept 'kill' which was already present in more topics, including topic 70. This means that the word *kill* will now score for topic 58 also. The multiplicity of the same concept in many topics, while unavoidable, does negatively impact performance. It should also be noted after tweaking, each topic has grown from 2-3 concepts to an average of 4 concepts with only few topics having more than 6 concepts.

Another valuable insight from this before / after comparison is related to the ontology information content. We have found out that in some places there are lacks in information in the ontology, while in other places there is an abundance of it. For example, we had trouble finding YAGO classes to describe the concepts for topic 'Solar Energy': while we have 'wordnet_energy_ 111452218' for the 'energy' concept, for the 'solar' concept there is no *solar* class, just classes like 'wordnet_solar_cell_ 104257986', 'wordnet_solar_dish_104258138' or 'wordnet_ solar_house_104258438'. While there is the direct class of 'wordnet_solar_energy_111509697' which we have also used, because it is multi-word, for us to match it we need to have the expression in the text *solar energy*. This means that in documents where the word *solar* is used, if it is not followed by *cell*, *dish*, *house* or *energy*, it will not be counted.

Another topic list related point is that the list needs to be created partially by hand. While usually this is not desired due to the human intervention that is required, we argue that the number of possible topics for any classification is manageable, ranging from a few tens to usually no more than a few hundreds, a relatively simple task for even one person.

Another perspective work aspect concern the fact that we currently consider the document as a bag of words. Improvement can be achieved if words would be analyzed in context – we could consider sentences and relations between words in a sentence and even more across sentence boundaries. For example, for the sentence '…genetically modified *apples* may lead to illness …' the noun *apples* would contribute more to topic 'Vegetables, Fruit and Cancer' due to the close proximity to *illness* that itself is

closely related to topic concept 'cancer', than it contributes to topic 'Pesticides in Baby Food', because there is no pesticide or baby related word in *apples*'s proximity. This would be implemented in the Analysis module, introducing a proximity factor to the score awarded to a concept based on the closeness to other related words for each topic. Another improvement that can be somewhat easily employed is to consider word modifiers. For example, for the sentence fragment '… thus avoiding war in the region.' , *war* should contribute to the topic 'Peace-Keeping Forces in Bosnia' because 'avoiding war' implies the 'peace' concept in this topic. A list of antonyms could be used. For example, the negation '*not dry*' directly implies *wet*, which would score for rain-related topics due to rain-water-wet conceptual ontology links. Also, a list of negative verbs could be used, such as for the fragment '... to *avoid war ...*', *avoid* could trigger *war*'s antonym - *peace*.

Furthermore, we could use the ontology as not only a semantic similarity map, but also use the relations themselves as useful information. That would mean identifying subject – object entities and then match the verb that links them to a specific relation in the ontology. This would provide a stronger link between concepts, and an algorithm could judge whether to take into account certain entities or not based on the relations between them. However, the task of Relation Detection and Identification is in itself a difficult problem, requiring a dedicated system of its own. At the time of writing we could not find such a system easily implementable on our YAGO-centric structure.

5. CONCLUSIONS

Our solution proposes a different approach in exploiting the Web and social Web data in the context of smart cities. While technologies exist for data crawling and crowdsourcing, as well as for big data analytics, a semantic-oriented analysis of these data could significantly contribute to knowledge management. We believe that ontologies, especially general ontologies represent a powerful yet somewhat underused tool for the text classification problem in the context of smart cities. The structure of the ontology itself contains information that can be used in the form of concept closeness, synonymy, hypernymy, relation types, etc. As time passes, general ontologies will become larger, thus providing better results even using the same algorithms.

However, the use of ontologies does impose some limitations For example, information density in an ontology varies greatly, meaning some concepts will be defined in more detail than others, that in turn leading to uneven topic recognition accuracy. This problem is usually addressed by using domain ontologies. However, for example, for news articles a domain ontology is mostly useless considering the method we have applied in this article, where we do not use the ontology as a simple hierarchical taxonomy, but as a concept semantic-similarity map.

We propose a system that achieves a good performance rating using only an ontology as its information source, and graph algorithms with a custom scoring method. The system uses the links available in the ontology to assign a score to the semantic similarity between concepts. Future work on the subject will include implementing some of the suggestions in the evaluation section, as well as implementing supervised and unsupervised algorithms to generate the topic list's concepts instead of manually tweaking them, and evaluate performance between this system's versions.

We've chosen to compare our solution with the SVM approach since it is proved by the state of the art as the most efficient method for text classification; however, two issues diminish its efficiency in the case of the presented "smart city" context: firstly, SVM approach provide better results when classifying a fixed document corpus (where the space is determined), but has performance issues when new documents, new topics and thus new keywords are added in real-time (retraining takes time; even online training for machine-learning algorithms like SVM still have problems working in real-time with large scale data, an issue that does not exists in unsupervised approaches); secondly, the SVM approach is not "deterministic", in the sense that for the same input data, after several training iterations (for incoming new data), the same output will be presented (the same classification category for example), while in the semantic Web age the "control" of semantics is an emerging and important issue to be handled;

An YAGO-based solution for text classification enable not only to handle the text semantics when new documents, new topics and thus new keywords appear continuously. In addition, the proposed graph-based method enables to handle the frequent changing of this topics and keywords' space dimension. While at present machine learning algorithms take center stage in the performance, we believe that knowledge engineering methods will slowly gain momentum, and aided by machine learning techniques as well as larger knowledge repositories in the form of ontologies, will provide the better results in this field.

6. REFERENCES

[1] N. Nedjah, Felipe M. G. França, Alberto Ferreira de Souza (Eds.), Intelligent Text Categorization and Clustering, in Studies in Computational Intelligence nr. 64, Springer, (2009)

[2] Ikonomakis, M., Kotsiantis, S., Tampakas, V.: Text Classification Using Machine Learning Techniques. WSEAS Transactions on Computers 4(8), 966–974 (2005)

[3] Sebastiani, F.: Machine Learning in Automated Text Categorization. ACM Computing Surveys 34(1), 1–47 (2002)

[4] Moravec, P., Kolovrat M., Snasel, V. LSI vs. wordnet ontology in dimension reduction for information retrieval, In : V. Snasel, J. Pokorny, K. Richta (eds.), Dateso, 18--26 (2004)

[5] L. Lv, Y. S. Liu and Y. Liu, Realizing English text classification with semantic set index method, Journal of Beijing University of Posts and Telecommunications, vol.29, no.2, pp.22-25, 2006.

[6] Yang, X.-Q., Sun, N., Sun, T.-L., Cao, X.-Y., Zheng, X.-J., The Application of Latent Semantic Indexing and Ontology in Text Classification. International Journal of Innovative Computing, Information and Control ICIC International. Vol. 5, 12(A), 4491--4499. (2009)

[7] A Semantic-Oriented Approach for Organizing and Developing Annotation for E-learning. In: IEEE Transactions on Learning Technologies. IEEE Computer Society (2010)

[8] Lee, Y.-H., Tsao, W.-J., Chu, T.-H.: Use of Ontology to Support Concept-Based Text Categorization, In: Weinhardt, C., Luckner, S., Stöber, J., Designing E-Business Systems. Markets, Services, and Networks, Lecture Notes in Business Information Processing, Volume 22, Part 6, 201—213 (2009)

[9] Gu, H., Zhou, K. Text Classification Based on Domain Ontology. Journal of Communication and Computer, Volume 3, No.5 (Serial No.18), ISSN1548-7709, USA (2006)

[10] Suchanek, F.-M., Kasneci, G., Weikum, G. YAGO - A Core of Semantic Knowledge. In 16th international World Wide Web conference (WWW 2007), 2007.

[11] Hall, M. Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H.: The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1. (2009);

[12] Schuurman, Dimitri et al . Smart Ideas for Smart Cities: Investigating Crowdsourcing for Generating and Selecting Ideas for ICT Innovation in a City Context. J. theor. appl. electron. commer. res., Talca, v. 7, n. 3, dec. (2012);

[13] Erickson, Thomas, Geocentric Crowdsourcing and Smarter Cities: Enabling Urban Intelligence in Cities and Regions, UbiComp'10, September 26–29, 2010, Copenhagen, Denmark. ACM 978-1-60558-843-8/10/09.

[14] Navigli, R. (2009). Word sense disambiguation: A survey. ACM Computer Survey , 41 (2).