2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

# Ensemble Learning for Keyword Extraction from Event Descriptions

Pedro Geadas and Ana Alves and Bernardete Ribeiro

*Abstract*— **Automatic keyword extraction (AKE) from textual sources took a valuable step towards harnessing the problem of efficient scanning of large document collections. Particularly in the context of urban mobility, where the most relevant events in the city are advertised on-line, it becomes difficult to know exactly what is happening in a place. In this paper we tackle this problem by extracting a set of keywords from different kinds of textual sources, focusing on the urban events context. We propose an ensemble of automatic keyword extraction systems KEA (Keyphrase Extraction Algorithm) and KUSCO (Knowledge Unsupervised Search for instantiating Concepts on lightweight Ontologies) and Conditional Random Fields (CRF). Unlike KEA and KUSCO which are well-known tools for automatic keyword extraction, CRF needs further preprocessing. Therefore, a tool for handling AKE from the documents using CRF is developed. The architecture for the AKE ensemble system is designed and efficient integration of component applications is achieved. Finally, we empirically show that our AKE ensemble system significantly succeeds on baseline sources and urban events collections.**

## I. Introduction

Nowadays the most relevant events in the city are advertised on-line. However, it often becomes difficult to know exactly what is happening in a place: information is spread across too many websites, many times not easily understandable. The result of the World Wide Web (WWW) exponential growth is an huge amount of data chaotically organized, which turns out tasks like accessing, searching and keeping information difficult. With so many data drifting in the Web, most of the times neither labeled nor categorized, finding the desired information is generally time wasting. Automatic extraction and summarization methods play an essential role to tackle this problem. Therefore, the goal of automatic extraction is to apply the power and speed of computation to the problems of access and discoverability, adding value to information organization and retrieval without the significant costs and drawbacks associated with human indexers.

In this paper we propose an ensemble of different classifiers for effectively extracting a set of keywords from small textual sources and show that the proposed ensemble application achieves better performance than the individual component systems (up to a certain extent concerning the differences between the individual classifiers' reliability), obtaining better results than those reported in the last Keyphrase Extraction Contest (SemEval 2010, http://semeval2.fbk.eu/semeval2.php?location=Rankings/ranking5.html).

The proposed approach comprises two supervised, KEA and CRF, and one unsupervised, KUSCO, machine learning keyword extraction methodologies.

The empirical tests were carried out in *Hulth's dataset* of scientific journal *paper abstracts*, in *Krap's dataset abstracts* from *Computer Science* domain. Furthermore, for further validating our approach, two collections of documents regarding music personalities' descriptions extracted from Wikipedia and descriptions about events in general, like theatre plays and music concerts, retrieved from YourSingapore web-pages (http://www.yoursingapore.com/) were considered.

This paper is organized as follows. In the Section 2, we present the background on the automatic keyword extraction regarding related work and specific applications. We describe the ensemble learning methodologies, in particular we look at both the keyword extraction component classifiers and the ways to combine them. The proposed approach is presented in Section 3. Section 4 deals with the experimental setup and in Section 5 results are presented and discussed showing the validity of our approach. Finally, in Section 6 we make conclusions and point some lines of future work.

## II. AKE - Automatic Keyword Extraction

AKE is the problem of automatically identifying the relevant words lying within a document and has been researched for more than half a century [1]. Such keywords may constitute useful entries for building an automatic index for a document collection, can be used to classify a text, or may serve as a concise summary for a given document [2]. While some of those methods rely merely on *statistical* and *linguistic knowledge*, recent works are more focused on *Machine Learning* techniques.

In this line, a wide range of methods have successfully been proposed for tasks of *AKE* [3] achieving better results than solely use statistics or linguistic knowledge about documents.

### A. Related Work

Examples of statistical methods include word frequency [1], word co-occurrence [4] and the *TF*IDF (Term Frequency - Inverse Document Frequency)* term weighting model [5]. Such methods have proved to be insufficient to overcome such problems by their own, thus another line of automatic extraction methods considered the linguistic features of words. Hulth [6] examines different methods of

Pedro Geadas is with CISUC-Center of Informatics and Systems of the University of Coimbra, Ana Alves is with CISUC and IPC-Polytechnic Institute of Coimbra, Bernardete Ribeiro is with CISUC and Department of Informatics Engineering, University of Coimbra (email: pmrg@student.dei.uc.pt, {ana,bribeiro}@dei.uc.pt).This work was supported by the Crowds project-PTDC/EIA-EIA/115014/2009 and InfoCrowds project-PTDC/ECM-TRA/1898/2012.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

incorporating this knowledge into AKE and considers syntactic features such as part-of-speech (PoS) tags to the classifier looking only at noun phrases to be candidate phrases. In turn, [7] showed that by using lexical resources (EDR - electronic dictionary and Princeton University's WordNet) in such a task results in slightly higher performances than by just resorting to a purely statistically based method.

Supervised algorithms found in [8] classified words as positive or negative examples of keywords, first applying the *C4.5* decision tree induction algorithm and later a custom-developed algorithm, *GenEx*. The authors conclude that by incorporating specialized procedural domain knowledge keyphrases could be better generated. Perhaps one of the main contributions to the field is KEA that was proposed in [9], using the *TF*IDF* score of a phrase as well as fined tuned features to build a Naive Bayes classifier. An improvement over KEA, called KEA++, has been proposed in [10] and also take advantage of semantic information on terms and phrases gleaned from a domain-specific thesaurus. Other approach based on KEA, but relying on *bagged decision trees* instead of Naive Bayes for classification was proposed in [11] and was called *Maui*. In [12] and [3] neural networks have been proposed and their results demonstrate that their method outperforms KEA. In [13] various classification models are compared in the task of extracting meaningful keywords from extremely short texts like those we find today on social networking services on the Web. They used a set of features to train those models (TFIDF, linguistic information, relative position, length of social snippet, document frequency and capitalization) and the best results reported used Gradient Boosting Machine (GMB).

CRF, today considered the state-of-the-art sequence labeling method, was first proposed by Lafferty et al. [14] back in 2001 and since then many published works explored this technique. Peng and McCallum [15] showed that CRF outperforms other methods at the task of extracting structured information, such as the information related to the authors and citations from a collection of research papers. In [16] CRF was successful proposed for the task of keyword extraction from Chinese scientific papers. Recently, in [17] AKE based on a combination of CRFs and a specific document structure was also presented, while the authors argue the results improved dramatically over the existing ones.

With respect to the unsupervised approach they usually consist of ranking each of the candidate keywords using multiple features and heuristics and selecting the top rated ones [18],[4]. In [2] TextRank, a graph-based ranking model based on the co-occurrence relation between words was presented, although other works graph mining based were also published [19],[20],[21]. Recently, focusing on keyword extraction from small textual sources such as event and product descriptions (often holding between 30 and 60 words), a novel unsupervised keyword extraction approach was proposed in [22], called Informativeness-based Keyword Extraction, that uses clustering and three levels of word evaluation (corpus, cluster and document level) to address the challenges of short documents.

### B. Existing Applications and Tools

We have analysed some of the existing solutions and test their operation which are briefly described in the next sections.

*1) KEA:* Considered one of the main contributions to the field of keyword extraction, KEA (*Keyphrase Extraction Algorithm*) [9] can be basically separated into *three phases*, illustrated in Figure 1.

First, because not all phrases in a document are equally likely to be keyphrases *a priori*, KEA performs some text preprocessing tasks in order to identify *candidate phrases*. The process starts by splitting the input text according to phrase boundaries (punctuation marks, dashes, brackets, and numbers) and unwanted characters are removed. KEA takes then all the sub-sequences of these initial phrases (up to length three by default, but it can be changed), as *candidate phrases* and eliminates the phrases that begin or end with a *stop-word* or phrases that are a *proper noun*. The stop-word list used by KEA contains 425 words in nine syntactic classes (conjunctions, articles, particles, prepositions, pronouns, anomalous verbs, adjectives, and adverbs); finally, these candidate phrases are then case-folded and stemmed.

Second, two specific *attributes* are used to discriminate between keyphrases and not-keyphrases: the *TF*IDF* score of a phrase, and the *distance* into the document of the phrase's first appearance (the number of words that precede the first occurrence of the term, divided by the number of words in the document). This corresponds to the *feature calculation* phase.

The third phase concerns KEA extraction phase. KEA computes the TF*IDF scores and distance values for all phrases in the *new document*, using the procedure described above, taking the discretization obtained from the training documents. The naive Bayes model is then applied to each phrase, computing the estimated probability of it being a keyphrase. The result is a list of phrases ranked according to their associated probabilities. Assuming that the user wants to extract $r$ keyphrases, KEA then outputs the $r$ highest ranked phrases [9].

*2) KUSCO:* KUSCO (*Knowledge Unsupervised Search for instantiating Concepts on lightweight Ontologies* [23]) is a system that indexes a set of *concepts* with given Points of Interest (POIs) and Events, *semantically enriching* them. Generally, KUSCO retrieves information on the Web about POIs and Events and extract the most relevant terms from theses texts. After extracted, those terms are contextualized and enriched with semantic information. Since KUSCO is constituted of different modules, we will focus on the Meaning Extraction module where *term extraction* from textual descriptions is performed. In Figure 2, one can see a more detailed visualization of KUSCO's Meaning Extraction module.

For each text describing a POI or an event, the Meaning Extraction module on KUSCO executes a sequence of Natural Language Processing steps to automatically extract the

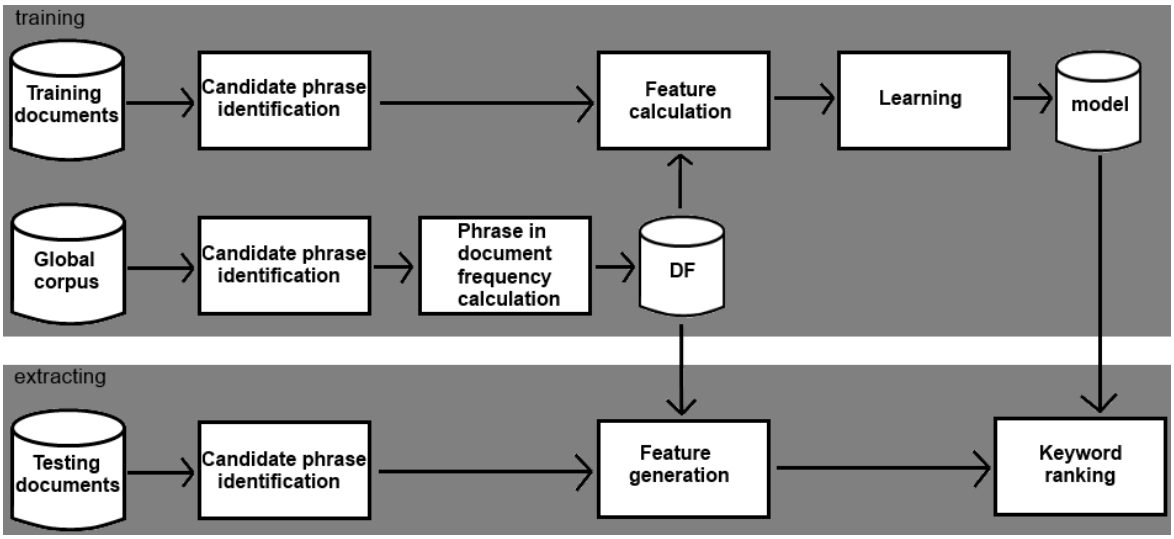


Fig. 1. KEA training and extraction processes ([9]).

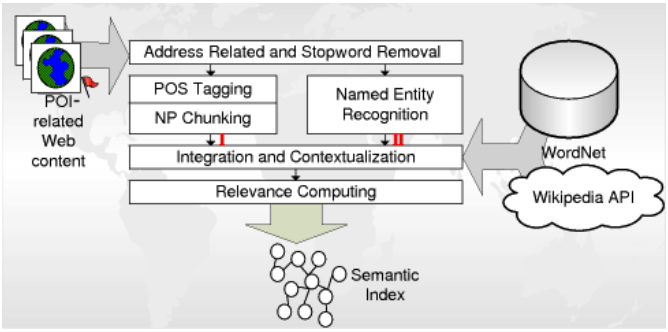


Fig. 2. KUSCO's Meaning Extraction module ([23]).

relevant related terms. Each text is broken up into paragraphs, paragraphs into sentences, and sentences into words. Words in a sentence are then tagged by the Brill's Part-of-Speech (POS) tagger [24] which labels each word as a noun, verb, adjective, etc. A Noun Phrase chunker [25] is then applied in order to identify every group of words with a head noun which functions together just as a single term. At the same time, the original text is also processed by a Named Entity recognizer [26] to identify proper names in the text. As figure 2 shows, noun phrases (flow I, which applies POS tagging and NP chunking) are represented by common nouns while the entities (flow II, which applies NER) are represented by proper nouns. Each term in both groups is represented using single or a compound noun, and it is contextualized in lexical resources (WordNet and Wikipedia) which guide the extraction process by validating common-sense terms and which are also used to infer the meaning of each term. These terms are called concepts only after they are contextualized, and their relevance is computed through an extended version of TF*IDF that considers the semantics of each term.

*3) CRF - Conditional Random Fields:* The linear-chain CRF has been applied in natural language processing, including named-entity recognition (NER), feature induction for NER, identifying protein names in biology abstracts, segmenting addresses in Web pages, information integration word alignment in machine translation, citation extraction from research papers, word segmentation and many others [27]. Unlike the majority of methods that do not use most of the features existing in a document, CRF can utilize most of those features sufficiently and effectively for efficient keyword extraction. Experimental results indicate that the CRF model can enhance keyword extraction and it outperforms the other machine learning methods [16].

In short, CRF is an *undirected graphical model* that encodes a *conditional probability distribution* with a given set of features.For the given observation sequential data $X(X_1X_2,\ldots,X_n)$, and their corresponding status labels $Y(Y_1Y_2,\ldots,Y_n)$, a linear chain structure CRF defines the conditional probability as follows:

$$P(Y|X) = \frac{1}{Z_x}\exp\sum_i\sum_j \lambda_j f_j(y_{i-1}, y_i, X, i) \qquad (1)$$

where $Z_x$ is a normalization and it makes the probability of all state sequences sum equal to 1, $f_j(y_{i-1}, y_i, X, i)$ is a feature function and $\lambda_j$ is a learned weight associated with feature $f_j$. The interested reader can find more information in the literature ([14]).

### C. Ensemble Learning Approaches

Four approaches are devised for building classifier ensembles each one focusing a different level of action. Approach A concerns the different ways of combining the results from the classifiers. *Majority voting* in particular its weighted version are the most widespread choices when the individual classifiers give label outputs [28]. Many ensemble paradigms employ the same classification model, for example, a decision tree or a neural network, but there is no evidence that this strategy is better than using different models (Approach B). At feature level (Approach C) different feature subsets

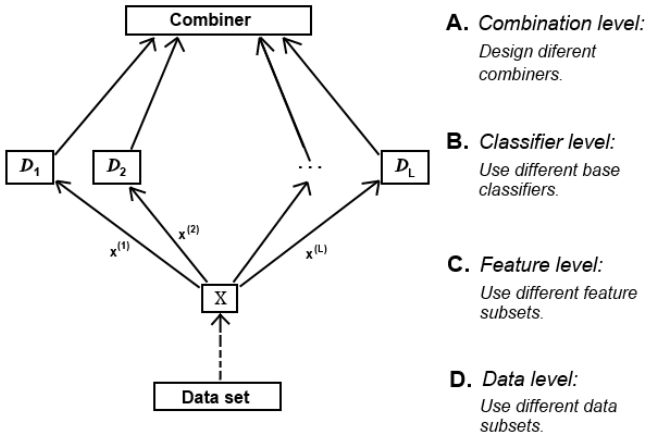


Fig. 3. Approaches to build classifier ensembles ([28]).

can be used for the classifiers, either if they use the same classification model or not. Finally, the data sets can be modified so that each classifier in the ensemble is trained on its own data set (Approach D).

Hulth [29] presented an algorithm for AKE combining statistical and linguistic methods, showing that the number of incorrect assigned keywords could be highly reduced, by combining then the predictions of several classifiers. Using an ensemble of Neural Networks, Wang et.al [3] depicted a method where the keyphrase extraction is viewed as a crisp binary classification task, training the neural network ensemble to classify whether a phrase is keyphrase or not. To discriminate between positive and negative examples, the following features (or attributes) of a phrase in a given document are adopted: its *term frequency*, whether they *appear in the title, abstract or headings* (subheadings), and its *frequency* appearing in the paragraphs of the given document, i.e., the distribution of a phrase in a given document.

Later, Zhang [30] combined several statistical machine learning models to extract keywords from Chinese documents. The method selects keywords through voting, from the multiples models created, and the experimental results show that the proposed ensemble learning method outperforms other methods also investigated, according to $F_1$ measurement and the extraction model using weighted votes outperforms the model that does not use weights.

## III. Proposed approach

The architecture of the proposed system is represented in Figure 4. It depicts the different components of the application and reveals the system's processing flow, since the moment an input text is passed to the application until the moment that it produces the desired output, i.e, a set of keywords for each of the unlabeled files. An explanation of the main stages identified is given below as well.

### A. Preprocessing

Preprocessing tasks are usually a prerequisite to text classification. The objective of this stage is cleaning the input text as much as possible by eliminating unnecessary words and characters (KUSCO, KEA) or, just structuring the text correctly (CRF), for further classification.

KEA and KUSCO already perform preprocessing tasks internally, according to their needs. CRF requires extra/different preprocessing operations than those needed by the other two applications. In order to identify the most of the features present in the text, a slight modification in the text is needed. Two major aspects were then considered, during this phase:

***Structural issues:***

1) Separating each phrase of the text, one per line;
2) Keeping punctuations present in each phrase (except quote marks, which actually produced better results when removed);
3) Keeping stop-words present in each phrase;
4) Each token of each phrase separated by a space.

***Feature and keyword's automatic tagging:***

1) Tagging the true keywords of each document within brackets (E.g.: [ correctly tagged keyword ] );
2) Identifying a set of features present in the text, namely those described in Table I.

### B. Keyword Classifiers

After the preprocessing phase, there are two different phases where within the system: the training phase, where a set of manually tagged texts is used to create a model for each classifier; and the extracting phase, where each classifier is used to test new texts. Ideally, these models should complement one another, each being specialized in a part of the domain where the others do not perform so well (just as human executives seek advisers whose skills complement each other). To achieve this, we use two already existing applications, combined with a third system (CRF) implemented on this purpose. If we look in Figure 3, this corresponds to the classifier level (Approach B). Furthermore, the features taken in consideration by each system used were different, so we are also introducing some diversity at the feature level (Approach C).

The CRF implementation used in this work was developed under MALLET[1] framework, acronym for MAchine Learning for LanguagE Toolkit. Table I portrays the list of features used to train the CRF model.

From the applications used here, only KEA allowed to define the specific number of keywords to be extracted. In the other hand, KUSCO and CRF extract as much keywords as they can. So, instead of defining a specific number, the approach taken was to set a limit of keywords that can be extracted. Best results were achieved when setting a maximum of thirty keywords per application.

While KEA and KUSCO have been presented earlier, a more detailed view of the CRF operation is also depicted in Figure 4.

[1]MALLET (http://mallet.cs.umass.edu/) is a Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications to text.

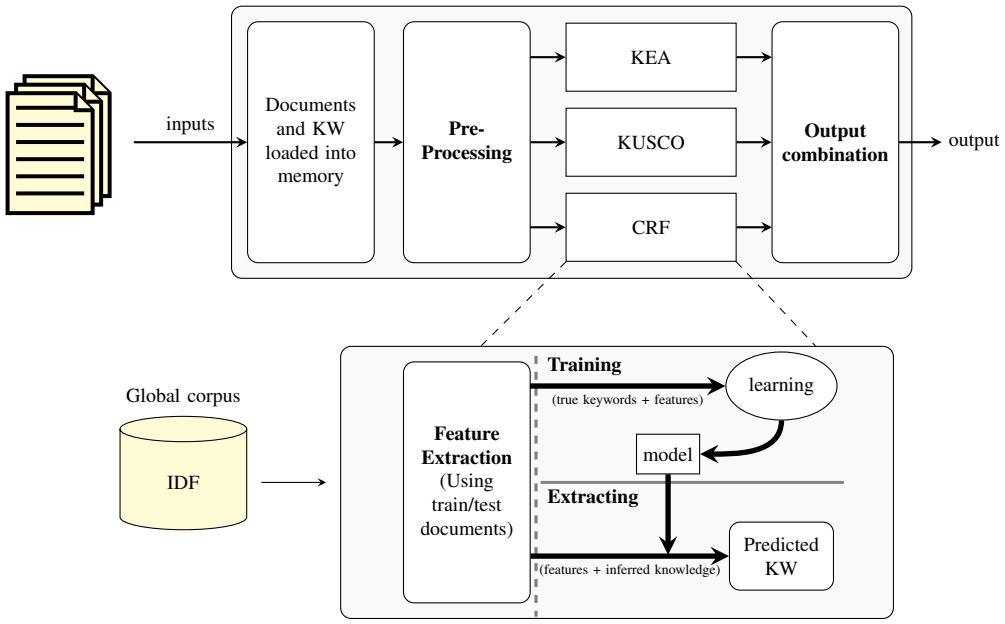


Fig. 4. Proposed system's architecture.

### C. Combining results

The ensemble level exploits ways of combining individual classifiers outputs, so here we analyse two versions of the widely used method of majority voting: simple and weighted majority voting.

*1) Simple Majority Voting:* To better understand how the voting procedure takes place, assume that the label outputs of the classifiers are given as $c$-dimensional binary vectors, $[d_{i,1}, \ldots, d_{i,c}]^T \in \{0,1\}^c, i = 1, \ldots, L$, where $d_{i,1} = 1$ if $D_i$ labels $x$ in $\omega_j$, and 0 otherwise. The *plurality vote* will result in an ensemble decision for class $\omega_k$ if

$$\sum_{i=1}^{L} d_{i,k} = \max_{j=1}^{c} \sum_{i=1}^{L} d_{i,j} \tag{2}$$

The plurality vote of Equation 2, is called in a wide sense *the majority vote*, and is the most often used rule from the majority vote group. Various studies are devoted to the majority vote for classifier combination [28].

*2) Weighted Majority Voting:* If classifiers in the ensemble are not of identical accuracy, then it is reasonable to attempt to give to the more competent classifiers more strength in making the final decision. The label outputs can be represented as degrees of support for the classes in the following way:

$$d_{i,j} = \begin{cases} 1, & \text{if } D_i \text{ labels } x \text{ in } \omega_j, \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The discriminant function for class $\omega_j$ obtained through weighted voting is

$$g_j(x) = \sum_{i=1}^{L} b_i d_{i,j}, \tag{4}$$

where $b_i$ is a coefficient for classifier $D_i$, which corresponds to the weight of that classifier. Thus the value of the discriminant Function 4 will be the sum of the coefficients for these members of the ensemble whose output for $x$ is $\omega_j$ [28].

## IV. Experimental Setup

### A. Datasets

In order to test the system performance, four different datasets of English texts were used and are described in the next paragraphs.

The first dataset from now on addressed as *Hulth's dataset*, consists of 2000 scientific journal *paper abstracts* with their corresponding title, from the Inspec (www.iee.org/publish/inspec/) database. Hulth's documents were obtained from *Computers and Control* and *Information Technology* and have been widely used in previous related works (e.g. [2], [6]).

The second one, from now on designated by *Krap's dataset*, consists of 2304 *full papers* from *Computer Science* domain. Each document has clearly indicated its title, abstract, body and references [31]. Nevertheless, only the abstracts were used here.

Finally, for validating the results obtained from the previous scientific datasets, other two collections of documents were used: the first one composed by 420 descriptions about *events* in general, like theatre plays and music concerts, retrieved from YourSingapore web-pages (www.yoursingapore.com/); the other comprising 112 music personalities' descriptions extracted from Wikipedia, labeled by twenty volunteers. Unfortunately, we only had one volunteer to do the manual labeling of the event descriptions dataset.

| | Features | Explanation | Range |
|---|---|---|---|
| 1 | Word | current token | - |
| 2 | PoS | Part-of-Speech of a token | {DT, VB, NN, (...)} |
| 3 | First Position | if a token is the first token in a sentence | {0, 1} |
| 4 | CAPITAL-IZED | if a token is capitalized | {0, 1} |
| 5 | Initial CAP | if a token begins with a capital | {0, 1} |
| 6 | Mixed CAPS | if a token contains both lower and upper cases | {0, 1} |
| 7 | Contains Digits | if a token contains digits | {0, 1} |
| 8 | All Digits | if a token is a number | {0, 1} |
| 9 | Hyphenated | if a token contains hyphens | {0, 1} |
| 10 | Dollar Sign | if a token contains the $ sign | {0, 1} |
| 11 | Ends In Dot | if a token ends with a dot | {0, 1} |
| 12 | Lonely Initial | if the token is an initial (e.g.: P.) | {0, 1} |
| 13 | Single Char | if the token is a single char (letter, number, symbol | {0, 1} |
| 14 | End Punctuation | if the token is sentence end punctuation | {0, 1} |
| 15 | Apostrophe | if the token contains an apostrophe ( ' ) | {0, 1} |
| 16 | Line Number | the line number of the current sentence | {1, 2 ...N} |
| 17 | TF | Term Frequency of the term in the document | [0, 1] |
| 18 | IDF | Inverse Document Frequency of the term in Wikipedia global corpus | $\mathbb{R}$ |
| 19 | TF*IDF | the Term-Frequency * Inverse Document Frequency of a term in the document | $\mathbb{R}$ |
| 20 | Windowed Features | the PoS and Word features of the *first*, *first and second* and *first, second and third* tokens before and after the current token | - |

TABLE I

LIST OF FEATURES USED TO TRAIN CRF MODEL IN THIS WORK.

### B. Tests Performed

From the set of tests here described, some aspects are worth noting. Tests ranging from 1 to 5 were applied over abstracts from Hulth's dataset and only Test 5 was replicated on abstracts from Krap's dataset. For Krap's dataset, tests 1 to 3 and 4 were not performed because as we extracted the title and abstract from the full documents, all the possible miss-indentations, miss-structuring and missing keywords were immediately corrected and the labeling method already used the stems rather than the full keywords. Tests 6 and 7 tried to verify if the type and number of the gold standard keywords may influence the learning of the classifiers, filtering some of the documents out of the dataset.

*1) Original Dataset:* This test can be seen as the baseline test. It was performed knowing in advance that only about 76% of the keywords were in fact present in the abstracts on the Hulth's dataset. This is explained by the author due to the fact that volunteers had access to full articles and not only to abstract in order to manually identify correct keywords [29]. Beyond that, this first test was also conducted without any kind of extra preprocessing being applied to the text files, i.e, documents are delivered to each of the applications without suffering any modifications.

*2) Removing unseen Keywords:* The second test differs from the first in one aspect: keywords that did not exist in abstracts were removed from the respective file's true keywords, so at this point, it is guaranteed that 100% of the keywords can be in fact found in the document they pertain to. We did this using the keywords stems to find occurrences of each keyword. This test was made in order to really perceive the true extraction power of each classifier, without that starting external limitation.

*3) Document structuring using OpenNLP Sentence Splitter:* Some features that we use to train our CRF model II, like PoS tagging or Windowed Features, are most likely to suffer from badly structured sentences in documents (e.g. line breaks in the middle of sentences).In order to attest that, we used the OpenNLP (http://opennlp.apache.org/) Sentence Splitter to pre-process documents, structuring them so that each of them contained only one sentence per line (that are to be tagged later). Despite no further enhancement being expected in KEA and KUSCO, once they do not have sentence structure in account, we expect improvements coming from the CRF, since it uses many of the features from the text.

*4) Stem-based keyword labeling method:* Until here, we automatically tagged true keywords exactly as they appeared in their respective .key files. This test verifies if the method of labeling the true keywords (exact-keyword or stemmed-keyword) has impact in the learning of the classifiers.

*5) The new Porter Stemmer:* A closer look to the stemmer used (the English Porter Stemmer) showed that it was not performing as expected for some apparently basic cases. We found out that a new version of the stemmer, but with some bugs corrected, was available in the community and replaced the old one with this (http://snowball.tartarus.org/algorithms/english/stemmer.html).

*6) Document filtering - digits in true keywords:* In this test we removed the documents whose keywords contained digits. We decided to do so because they were a minority (about 10 % of the total for Hulth's dataset, less than 10% for the others) and we wanted to understand if this type of keywords has impact in the classifiers' performance.

*7) Document filtering - number of true keywords:* Here, additionally to the constraint added in the last test, only documents having between *five* and *ten* keywords were used. We chose to do this because while some documents had only one assigned keyword others had more than fifteen, i.e., the difference between minimum and maximum number of keywords was to big and some training groups had considerably different average of keywords per file among them, making it harder to interpret the respective results.

## V. EXPERIMENTAL RESULTS

For evaluating our system we used *ten fold cross validation*. The results of the tests made are shown in Figures 5, 6 and 7, where the micro-averaged scores of each application

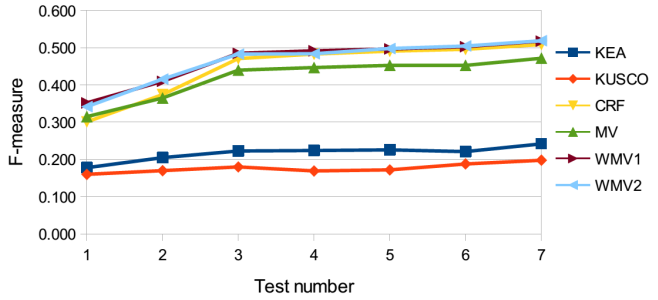


Fig. 5. Hulth's dataset results.

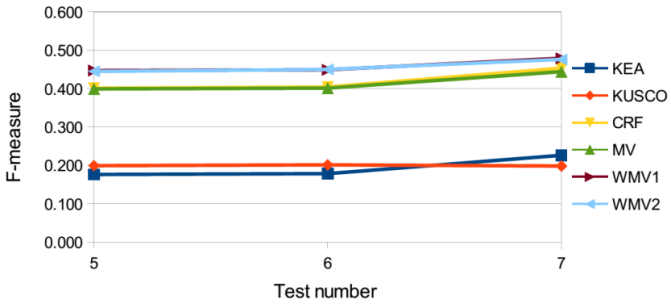


Fig. 6. Krap's dataset results.

are depicted for each dataset. Figure 7 compares the best results obtained with each dataset.

The weight of each classifier in the final ensemble can be learned from their individual performance. Considering the number of keywords found by a classifier compared to the number of true keywords present in each text in a given labeled dataset, we used several Regression algorithms in Weka framework[2], having the best result with Pace Regression algorithm, with a precision of $0.65 \pm 0.13$. Empirically we found that CRF when has great performance it does not depend on the contribution of other classifiers. In the opposite side, when CRF is not so well, the ensemble achieve better results. In Table II we present two different weight configurations that led to the best results, when *Weighted*

[2]Weka is a collection of machine learning algorithms for data mining tasks, available at http://www.cs.waikato.ac.nz/ml/weka/.

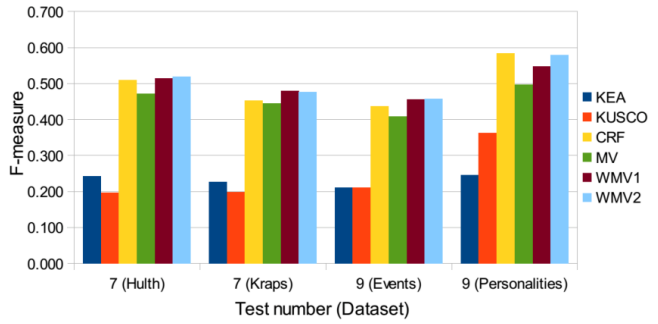


Fig. 7. Results validation.

| Weighted version | CRF weight | KUSCO weight | KEA weight |
|---|---|---|---|
| WMV 1 (when CRF $F_1 < 0.48$) | 54% | 36% | 10% |
| WMV 2 (when CRF $F_1 > 0.48$) | 62% | 30% | 8% |

TABLE II
MODEL WEIGHTS PRODUCING BEST RESULTS.

*Majority Voting* was used, depending on how well CRF performed.

Nevertheless, attentive reader will notice that despite KEA is achieving better $F_1$ scores than KUSCO in almost all tests shown, it is always given less vote weight. This can be explained due to the type of the extracted keywords: KUSCO guarantees that each term it extracts is unique while KEA does not guarantee that. In fact, many of the terms extracted by KEA contain each other (e.g.: [extracted example keyword], [extracted keyword], [example keyword]), which gives an undesired emphasis to the *same* keyword when the voting phase occurs. To avoid this, the vote weight of KEA had to be lower than that of KUSCO.

During the experimentation we noticed that CRF achieved higher Precision (the keywords actually found) rather than Recall (the number of keywords found). This means that keywords extracted by CRF are usually correct but are generally not many, so the improvement that we see in the final results (for the ensemble) come from compensating this lack of keywords that CRF can extract for some documents with the other applications.

Document structure was another factor influencing the results obtained. As one can see, after applying the OpenNLP Sentence Splitter to split the sentences of each document correctly, great improvement was observed. This happened because features as PoS and Windowed Features used to train CRF became more effective, since they depend a lot about the structure of the sentence analysed.

Yet concerning structural issues, the number of true keywords present in the files seems to affect the performance of the applications, as well as the type (precisely in this case, if they contained digits) of keywords given as gold standard. Thus, removing unseen keywords and those that contain numbers resulted in better performances observed.

It can also be stated from the results obtained, since Test 2, that the *Simple Majority Voting* no longer improves the results of the best individual application (CRF). This can be explained due to the huge differences in the classifiers' reliability, which differs much from one application to other. Nevertheless, CRF takes advantage of the other classifiers in the ensemble like KUSCO when the latter can cover a great diversity of simple and compound keywords since it uses WordNet and Wikipedia for keyword verification. This is done without the need of training neither labeled examples.

Another objective for this work was validating the results obtained with scientific datasets, with the non-scientific ones. From the results concerning those of the Event descriptions,

a performance improvement is still visible, similarly to what happened with the scientific datasets. Nevertheless, for the second non-scientific dataset here tested, that consisting in Personalities descriptions, the results using the ensemble did not improved those obtained by the CRF itself, despite the different configuration used for the Weighted Majority Vote. This can be explained because the superb performance that CRF system achieved with this dataset (depicted in figure 7) and indicates that above a certain limit, no further gains can be achieved: the difference between the classifiers' performances is to large, causing any keyword coming from the lower classifiers (KUSCO and KEA) being not good enough to improve the performance of the higher one (CRF).

The results presented here also seem to indicate something that had already been stated before: CRF is highly dependent on the document structure and type. A closer look to each of these datasets, show that all the documents pertaining to the Personalities dataset are very similar among each other, which seems to be the reason why CRF achieves even better performance in this case.

## VI. Conclusions and Future Work

The work presented in this paper exploits AKE from textual sources in general, since it was successfully applied to scientific and non-scientific domains.

The proposed approach builds a consensus-based machine learning methodology (both supervised and unsupervised). Moreover, using an ensemble of several applications to improve the performance revealed to be very effective over single classifiers results. For combining the outputs of the individual models, two methods of majority voting are used: simple majority and weighted majority. While the former gives equal weight to all predictive models, the latter gives more weight to those who present better predictive performance. However, one factor that can limit the enhancement seems to be the difference in each application's performance: the more one of the applications outperforms the others, the smaller are the gains.

This work has shown that by combining models of different existing applications, instead of using a more traditional method to generate different models, is also a viable method to create an ensemble application and the empirical results here obtained, which improved those of each the individual systems, attest that.

In future work it will be worth improving the CRF itself, by adding new features that were not present yet but might be considered important.

## VII. Acknowledgments

This project involved collaboration with the Massachusetts Institute of Technology (MIT) and was funded by the Portuguese Science Technology Foundation (FCT). Special thanks to Filipe Rodrigues for providing the original CRF implementation, to Anette Hulth for pointing us toward valuable scientific datasets and to Su Nam Kim for maintaining the public on-line repository which holds these.

## References

[1] H. P. Luhn. The automatic creation of literature abstracts. *IBM J. Res. Dev.*, 2(2):159–165, Apr. 1958.

[2] R. Mihalcea and P. Tarau. Textrank: Bringing order into text. In *EMNLP*, pages 404–411, 2004.

[3] J. Wang, H. Peng, J.-s. Hu, and J. Zhang. Ensemble learning for keyphrases extraction from scientific document. In ISNN'06, pgs 1267–1272, Berlin. Springer-Verlag.

[4] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.

[5] S. Robertson. Understanding inverse document frequency: On theoretical arguments for idf. *Journal of Documentation*, 60:2004, 2004.

[6] A. Hulth. Improved automatic keyword extraction given more linguistic knowledge. *Empirical Methods in NLP*, pgs 216–223, 2003.

[7] L. van der Plas, V. Pallotta, M. Rajman, and H. Ghorbel. Automatic keyword extraction from spoken text. a comparison of two lexical resources: the edr and wordnet. *CoRR*, cs.CL/0410062, 2004.

[8] P. D. Turney. Learning algorithms for keyphrase extraction. *INFORMATION RETRIEVAL*, 2:303–336, 1999.

[9] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. KEA: practical automatic keyphrase extraction. In DL '99, pgs 254–255, ACM.

[10] O. Medelyan and I. H. Witten. Thesaurus based automatic keyphrase indexing. In *JCDL '06*, pages 296–297, New York, NY, USA, 2006.

[11] O. Medelyan, E. Frank, and I. H. Witten. Human-competitive tagging using automatic keyphrase extraction. *EMNLP*, pgs 1318–1327, 2009.

[12] K. Sarkar, M. Nasipuri, and S. Ghose. A new approach to keyphrase extraction using neural networks. *CoRR*, abs/1004.3274, 2010.

[13] Z. Li, D. Zhou, Y.-F. Juan, and J. Han. Keyword extraction for social snippets. In WWW'10, pgs 1143–1144, 2010.

[14] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML '01*, pages 282–289, San Francisco, CA, USA, 2001.

[15] F. Peng and A. McCallum. Information extraction from research papers using crfs. *Inf. Proc. Manag.*, 42(4)963–979, 2006.

[16] C. Zhang. Automatic keyword extraction from documents using conditional random fields, 2008.

[17] H.-W. X. Feng Yu and D.-Q. Zheng. Key-phrase extraction based on a combination of crf model with document structure. *8th Int. Conference on Computational Intelligence and Security*, 0:406–410, 2012.

[18] J. Wan and J. Xiao. Single document keyphrase extraction using neighborhood knowledge. In AAAI'08, pgs 855–860. AAAI Press.

[19] M. P. Grineva, M. N. Grinev, and D. Lizorkin. Extracting key terms from noisy and multitheme documents. In *WWW*, pgs 661–670, 2009.

[20] R. Ortiz, D. Pinto, M. Tovar, and H. Jiménez-Salazar. Buap: An unsupervised approach to automatic keyphrase extraction from scientific articles. In, SemEval '10, pgs 174–177, USA, 2010. ACL.

[21] N. C. Stuart Rose, Dave Engel and W. Cowley. *Automatic keyword extraction from individual documents*. John Wiley & Sons, Ltd, 2010.

[22] M. Timonen, T. Toivanen, Y. Teng, C. Chen, and L. He. Informativeness-based keyword extraction from short documents. In *KDIR*, pgs 411–421, 2012.

[23] A. Alves, B. Antunes, F. C. Pereira, and C. Bento. Semantic enrichment of places: Ontology learning from web. *Int. J. Know.-Based Intell. Eng. Syst.*, 13(1):19–30, 2009.

[24] E. Brill. Some advances in transformation-based part of speech tagging. In *Nat. Conf. on Artificial Intelligence*, pages 722–727, 1994.

[25] L. Ramshaw and M. Marcus. Text Chunking using Transformation-Based Learning. In *Proc. of WVLC-1995*, Cambridge, USA, 1995.

[26] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL '05*, pages 363–370, 2005.

[27] C. Sutton and A. McCallum. An introduction to crfs, 2010.

[28] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[29] A. Hulth. *Combining Machine Learning and Natural Language Processing for Automatic Keyword Extraction*. Report series, 2004.

[30] C. Zhang. Combining statistical machine learning models to extract keywords from chinese documents. In *Advanced Data Mining and Applications*, 5678, pgs 745–754, 2009.

[31] M. Krapivin, A. Autayeu, and M. Marchese. Large dataset for keyphrases extraction. Tech. Report DISI-09-055, Trento, Italy, 2008.

[32] I. Oelze. Automatic keyword extraction for database search, 2009.