International Journal of Computational Linguistics and Applications vol. 7, no. 2, 2016, pp. 29–44
Received 07/02/2016, accepted 07/03/2016, final 19/09/2016
ISSN 0976-0962, http://ijcla.bahripublications.com

Identifying Multiple Topics in Texts

MOHAMED MOUINE[1], DIANA INKPEN[1],
PIERRE-OLIVIER CHARLEBOIS[2] AND TRI HO[2]

[1] *University of Ottawa, Canada*
[2] *ReDock Inc., Canada*

ABSTRACT

In this paper, we present an innovative method for multi-label text classification. Our method uses Lucene to index texts and then assigns one or more classes to a new text based on its similarity relative to an annotated corpus. For finer granularity, we split the text into phrases, and then we focus on the noun phrases. Instead of classifying the entire text, we classify each noun phrase. The result of classifying the text is then assembled as the set of classes allocated to its noun phrases.

KEYWORDS: *multi-label text classification, indexing, recommender systems, noun phrase extraction*

1 INTRODUCTION

Automatic text classification is concerned with learning from a corpus of labelled documents. Often, each document is assigned to a single class from a set of disjoint classes. If we have only two classes, the problem is called binary classification. If we have a set C on n classes, the problem is called n-ary classification. Less often, we encounter problems that need to assign more than one class to each document; this is called a multi-class or multi-label classification problem. In multi-label classification, each document is assigned to a set of classes $E \subseteq C$.

According to [1], we can group the existing methods for multi-label classification into two main categories: problem transformation methods and algorithm adaptation methods. Problem transformation methods transform the multi-label classification problem into one or more single-label classification problems, while adaptation methods extend specific learning algorithms in order to handle multi-label data directly. The method that we propose in this paper can be placed in the group of adaptation methods.

We present in this paper three versions of our method of multi-label text classification. Our method takes as input a corpus of manually annotated texts. Experts analysed each text and assigned classes according to the topics discussed. To classify a new text, we calculate the similarity of it with all the texts of the corpus. We retain only the five most similar texts. In the first version, we use a discriminative algorithm to hold classes that have the highest probability to be attributed to the new text. We consider this version as a baseline. This version is preferred in areas where the number of topics covered in the text is relatively low (2 to 3 maximum). In the second version, we keep the same features as in the first version, but we refine the model.

Based on the annotated corpus, we prepare a statistical model of class co-occurrence. This model allows us, knowing a class, to calculate the probability that another class can co-occur with it in the same text. In the third version, we take into consideration complex texts that may have a high number of topics (up to ten topics). The results of the first and second version may be biased in such a case. We divide the problem then into several problems. We split the text into several phrases (noun phrases). We classify these noun phrases. Finally, the union of all the classes of the noun phrases is used for the final text classification.

In the next section, we give an overview of the related work. Then we present the problem and why we need such a method. In section 5, we explain the idea of our approach. Then, in Section 6 we show the first version of our method, analysis of the results, and the types of text for which this version has good results. We refine these results in the second version of our method (section 7). We use a statistical model to predict the other classes that have a high probability of being among the results. For a more complex text, that needs to be attributed to a large number of classes, we present a third version of our method (section 8). The latter versions led to the best results for the corpus used in this work.

2 Related Work

The problem of classification has been widely studied in data mining, machine learning, databases, and information retrieval communities with many applications in a diversity of domains, such as targeted marketing, medical diagnosis, newsgroups filtering, and document organization. In [2], we can find a survey of a wide variety of text classification algorithms. Web documents and electronic documents in general contain a lot of text written in natural language. Manually analysing a massive amount of data within a short amount of time is impossible. Automatic text classification allows us to save valuable time, which explains its use in several areas.

One of the first areas in which text classification was used is spam filtering [3] [4], and it was expanded to classify the e-mails by subject or by type (promotions, social networks, etc.). A variety of supervised methods can be used for document organization in many domains such as large digital libraries of documents, web collections, scientific literature, or even social feeds. Hierarchically organized document collections can be particularly useful for browsing and retrieval [5]. Another important domain that of is customer reviews, which are often short text documents that can be mined to determine user opinions and other useful information from the reviews [6].

Most of the news services today are electronic in nature, since a large volume of news articles are created every single day by many organizations. In such cases, it is difficult to organize the news articles manually. Therefore, automated methods can be very useful for news categorization in a variety of web portals [7]. This application is also referred to as text filtering. Our current study fits into this type of this application.

There are several methods and algorithms used in the literature for text classification. We cannot recommend any of them as the best, because this depends on the application, the parameters of the algorithms, the type of expected results, the kinds of documents, etc.

One of the oldest methods used in the classification in general and applied to text classification is decision trees [8]. This method searches for a hierarchical division of the underlying data space, based on different characteristics/features extracted from texts. The hierarchical division of the data space is designed to create class scores that are asymmetrical in terms of the distribution of their class. Decision trees examine each feature in order to build a tree that split the data into classes starting with the features that have the highest discrimination power (according to an entropy-based measure named InfoGain). Here, we also mention the widely used Support Vector Machines (SVM) classifier which partitions the data space between two classes and determines the optimal boundaries between them. Another classifier that was applied to text data is the Neural Network (NN) classifier [9], which adapts to the training data based on the word features. Both SVM and NN are discriminative classifiers, as opposed to the generative classifiers, such as Bayesian classifiers. The latter build probabilistic classifiers based on modelling of the underlying characteristics of words features into different classes. The idea is then to classify text based on the posterior probability of documents belonging to different classes based on the word occurrences in documents. Almost all classifiers can be adapted to the case of text data. Some of the other classifiers frequently used include k Nearest Neighbour classifiers, rule-based classifiers, and genetic algorithms.

Other works that are closely-related to our work was described in [10], which treats the topic of multi-label classification and in [11], which proposed binary classifiers that index the texts using

Lucene[3]. [12] used genetic programs to construct Lucene search queries. In these works, the text is analyzed as a single unit. We show in this paper that the multi-label classification obtains better results if we split the text into several subdivisions and classify them separately.

3 Problem Statement

The manual analysis of a text is time consuming. A machine cannot completely replace a human being in this task. However, the machine can be very useful help to help the human to make a decision. An automatic system can be used to save a lot of time when dealing with large amounts of texts, in order to perform summarization, information extraction, information retrieval, or classification.

Let's take a concrete example that we study in this paper. An expert must analyse one or more Request For Proposals (RPF). Each RFP contains between 30 and 300 pages. A decision must be taken at the end of this analysis. The expert needs to decide whether or not to draft a response for a RFP. To make such a decision, he or she needs to check the availability of resources that could accomplish the work requested in the proposal. In other words, the expert analyses the CVs (curriculum vitaes) of available employees and the project descriptions that are created by internal and external consultants, in order to compare them to what is required. Many of the RFPs that are received by organizations are long and complicated to analyse. Therefore, their analysis is a task that consumes a lot of time and resources within these organizations. Due to the length and complexity of the processes, human decision in a similar situation may not be consistent (may be biased). It is difficult for a human expert to analyse a long document, extract all the requirements and find the necessary resources (from hundreds of CVs and project descriptions) in a precise and efficient way. In order to help the human experts, our system performs several operations on the RFPs. To start with, each document uploaded in the

[3] Lucene is Java library for indexing and search.

system is segmented into several parts. For simplicity and because of the structure of the documents, each paragraph is considered a segment. Experts according to the topics covered in these segments classified the old documents that have been uploaded into the system and segmented over the years. Figure 1 shows an example of a segment with the topics that have been assigned by the experts.

We need our system to analyse each segment of the RFP to extract its meaning and identify the topics discussed is the segment. The expert thus will have an overall idea of the content of the document. The expert can decide whether or not to continue to analyse this RFP in more detail. If the decision is to continue, extracting the requirements may prove to be easier by using the topics identified by the system. Moreover, in order to make the decision whether to draft a response or not, the expert has to know if the company has the resources requested. The goal of our system is to analyse the CVs and descriptions of projects in the same way as the RFP, in order to make it easier to match the extracted topics / classes directly among the RFP and the available resources.

4 DATA

Many companies make bids to answer RFPs. An RFP can contain several fields. Typically, these companies are specialized in specific domains. Our corpus was provided by a company specialized in the domain of information technology (IT). Our method can be applied to any field. However, the data used for training our models must be of the same domain as the text that we want to classify. Words can have different meanings depending on the field and thus have a different importance in each domain. We collected 1350 segments annotated by human experts. According to the topics discussed, experts assigns labels to text. Each segment is assigned to multiple classes. The number of classes is between 3 and 10 (see Figure 1 for an example).

We need to index a part of the corpus for training purposes. Our corpus consists of 1350 segments. We used 950 from them as training data. We set several parameters during the validation phase. For

Text segment:

Commitment is required among leaders to ensure that the leadership of the transformation is effective in driving delivery of the required business outcomes (including culture) and addressing the change imperatives. More specifically, that their behaviour is consistent with the objectives of the transformation including:

- Developing and communicating the vision clearly and in a compelling fashion.
- Demonstrating unequivocal and united commitment to realizing the vision and new ways of working.
- Demonstrating that they personally have changed and are adopting the new ways of working and leading.
- Enabling change by making timely and appropriate decisions which elicit action in the transformation.

Topics:

Leadership Transformation, Business Outcomes, Business Outcomes Delivery, Developing Vision, Communicating Vision, Demonstrating Commitment, Enabling Change.

Fig. 1. Example of text with the topics specified by the user

this phase, we reserved 250 segments as validation data that we use to choose the best models (parameter values). The rest of the data, 150 segments, are used as test data, in order test whether the results are good on new text segments.

5 APPROACH

Every segment is composed of many words and phrases. Based on their positions and the number of occurrences, the system creates standard vector space models. Each dimension corresponds to a separate term. It gives more importance to words that often appear (term frequency) in the segment, but are relatively rare in the entire corpus (inverse document frequency). The segments and queries are represented as vectors. If a term appears in a segment, its value in the vector is non-zero. The vector is represented as: $V = [w1, w2, ..., wn]$, where w is the weight of each term. Since the system can compare different segments to find out the closest matches.

To classify a new segment, the system creates a vector representing this segment and computes the similarity of this vector to all the vectors of the vector space. In other words, the new segment is considered as a query and the system computes a similarity (cosine similarity) with all segments of our corpus.

After retrieving the list of all closely matched segments to a given segment, the system will extract all the classes that are bound to those segments in the training data. These classes will be run through a discriminative algorithm (see Algorithm 1) to find out what are the most common classes to choose as the classes to be assigned to the new segment.

The role of the algorithm that we propose and that we call the discriminative algorithm is to find the classes that are most likely to be assigned to a new segment that we want to classify. The most similar segments to the new segment are very likely to address the same topics covered in the new segment. Nevertheless, as a segment address several topics, its segments can be address, in addition to the topics we are interested in, other topics that are not covered in the new segment. We compute the similarities of a new text to the segments in the training data and then take the classes only from the top most similar segments. In order to minimize the error rate, our algorithm considers only the topics that occur most often in the considered segments.

- Each segment from the result has a similarity score to the new segment. This score is applied for all classes that belong to each segment.
- All of the classes' labels will be put into a matrix using the normalized form of the label as the key. The computed value for each key is
  calculated based on the addition of all the scores of the same classes in all segments.
- These values are ranked: the higher one is the best.
- The system can extract a number of classes out of this list or all of them.

Algorithm 1: The discriminative algorithm

6 DOCUMENT CLASSIFICATION WITH LUCENE

In general, a classifier takes some input (represented as a set of features) and returns a classification of the input over a finite number of discrete categories. The goal of text classification is to assign each text to some categories. Given a set of texts that contain discrete category labels, we need a classifier that can assign these labels to new documents based on the similarity of the new documents to the labelled documents (training data).

We use Lucene to index the training data, and then we use this index to do a first-best classification of the test data. In the section 6.1, we will see how we train a classifier using the idex produced by Lucene and in the section 6.2, we will explain how we classify a new segment.

6.1 *Building the Index*

We build a Lucene index over all the documents in the training data. Each segment is processed into a Lucene Document that has two fields: content and topics. Take the example of Figure 1, the first part called ”text” would be in the field ”content” and the second part called ”topics” would be in the field ”topics” of the Lucene document. In this way, we can find the classes (topics) of each indexed segment in Lucene.

We need to convert the text data in fundamental units called tokens. An analyser (preprocessor) is used for this task. We describe the analyser in more detail below. During this analysis, the text undergoes several steps: extraction of words, suppression of the most common words (stopwords) and punctuation, reduction of the words to their base forms (morphological analysis), and lower casing. This text analysis is performed before indexing and query processing. It converts the text data into tokens and these tokens are added as terms to the Lucene index.

MORPHOLOGICAL ANALYSIS A stemmer is a simple algorithm that removes the most common morphological and inflexional endings from words, in order to do a term normalisation process, so that we can recognize different variants of each term. One of the most used stemmers is the Porter stemmer [13]. The same author later developed Snowball [14], to improve on the previous version by eliminating some of the inaccurate stems produced by the Porter stemmer and to extend it to support 13 other languages. Another process of grouping inflected forms of a word is a lemmatisation. This process is closely related to stemming. The lemmatiser produce a base form of a word. The difference is that a stemmer operates on a single word to produce a part of a word as the stem without knowledge of the context, and therefore cannot discriminate between words that have different meanings based on the part of speech.

The lemmatiser itself can be used in place of the stemmer because this technique can incorporate the same stemmer processes at the same time.

We use an algorithm of lemmatization based on WordNet[4]. There are two types of processes used in our algorithm to convert

[4] WordNet is a lexical database for English.

inflected words to their base form. The process starts with checking the exception list for each syntactic category, followed by executing the detachment rules based on the list of the inflectional endings on each syntactic category. The exception lists contain a list of strings (in alphabetical order) that cannot be processed by the morphological transformation algorithm.

Each line of the exception list contains an inflected word with its base form(s). If the inflected word is not found in exception list, it will be passed to the morphological transformation process, which is based on detachment rules, in order to find the base form of the word by suffix matching.

We use three different analysers. This allows us to see how different strategies affect classification. The three analysers available are: the Lucene StandardAnalyzer, which chains together a StandardTokenizer that breaks the text into a stream of words, discarding whitespace and punctuation, followed by a StandardFilter, a LowerCaseFilter, and then a StopFilter, which uses a list of English stop words; an analyser that chains together just a StandardTokenizer and a LowerCaseFilter; and an Analyser that uses only a Lucene NGramTokenizer to index the data using 4-grams.

6.2 *Using the Index for Classification*

To use Lucene to classify a document from the test set, we tokenize the new segment and create a Lucene query over its tokens. We do a search on the index that was built on the training data. The results of the research are a set of segments from the training corpus that are the most similar to the segment we want to classify (the query). These results are sorted according to the similarity score. As we have explained before, we indexed each segment and its topics in one Lucene document (with two fields) in order to retrace the topics of each segment. We use the topics of the top-scoring segment as output of the classifier.

We consider the top 10 segments (with the highest scores). The discriminative algorithm computes from all the topics of these 10 segments the topics to be assigned to the new segment.

Table 1 shows the result of this classifier on the test data, in terms of precision, recall and F1-measure (see the second column, named Lucene). We consider that these results are a baseline for our next experiments; we want to design a method that can obtain better results than this baseline.

In the next section, we propose to improve the performance of this version by using of an item-based recommender algorithm to predict, based on a statistical model, other topics that may be considered, knowing the topics resulting from the first version (our baseline algorithm).

7 Recommender

An item-based recommender is a flexible and easy to implement algorithm with a diverse range of applications.

The item-based recommendation algorithm takes as input customer preferences in terms on preferred items and generates an output recommending similar items, with a score indicating how much a customer will ”like” the recommended item [15].

The input file has the structure (customer: item: score), where the customer gives a recommendation score for an item. We use the same structure, but we adapt it to our problem. Hence the input file has the structure (topic1: topic2: number of co-occurrence). Recommenders use numeric score to rank topics. For example, a score of 0.64 means that there is 64% shared relevance between the two topics. The score is constructed based on the number of occurrences of the two topics in the same segment.

The idea for this version of our method is that instead of considering the topics of the top 10 segments directly as in the first version, we take only half of these topics (the best ones) and apply the recommendation algorithm. For each of these topics, it recommends a topic that has the highest probability of appearing with the given topic. Table 1 shows that this version improves the results of our first version (see the third column, named Lucene/Recommender).

8 Noun Phrases

Until now, we compared the segments. We used the classes of the segments that are similar to the new segment to classify it. In the case of multi-label classification, a text is assigned to several classes because it covers several topics. When a text is similar to another text, this does not mean that both texts discuss exactly the same topics. We want to exclude topics that are not covered in both texts and to keep only the topics that they have in common.

Any given segment is composed of multiple phrases such as noun phrases, verb phrases, etc. These phrases can be used to represent the corresponding segments. Within a specific domain, a phrase usually has very specific meaning. Single words can be have several meanings (they can be ambiguous), but a phrase is usually less ambiguous. A new segment might contain some of the phrases learned from the training data. Hence, based on the meaning of these phrases, the system can provide more accurate tags.

By extracting important noun phrases from a text, the meaning can be deducted. Within a specific domain, a noun phrase has limited meaning (usually one meaning). Phrases tend to be more specific than words. To understand a text, we extract the best noun phrases; each noun phrase has metadata which contains a validated meaning and context from previous learning. Hence, we can understand the meaning of the paragraphs.

Most of the meaningful noun phrases are between 2 to 3 words long, without stopwords. We use the part of speech tagger (POS) of the OpenNLP java library[5] to extract nouns phrases (NPs). For all these NPs, we eliminate the stopwords and apply the WordNet lemmatizer. Instead of indexing the entire text in Lucene (as for version 1 and 2), we index the NP separately. The learning process in this version goes through two steps.

First, for each NP extracted from a segment, we assign all its classes. We index all the NPs with their respective classes in Lucene. Second, to find the meaning of the NP (its class), we look

[5] https://opennlp.apache.org/documentation/manual/opennlp.html

Table 1. Classifier results

Method	P	R	F
Lucene	0.59	0.81	0.69
Lucene/Recommender	0.63	0.79	0.70
NP	0.75	0.81	0.78

for NPs that are repeated more than once in the corpus. We only consider the classes that come back each time to the same NP (intersection).

Finally, we re-index all NPs (without repetition) with the new classes assigned to each NP. To classify a new segment, we use the same process. We extract all the NPs, we eliminate stopwords and we look for the base form of each word using the morphological analyser. We keep only the NPs composed of 2 to 3 words. For each NP kept, we looking into the corpus for the most similar NP. Classes indexed with the most similar NP are attributed to the NP of the new segment. The classes that are final result for the classification of the new segment are the set of classes allocated to its NPs. This version of our method works well for complex text covering several topics. This version obtained the best results (see the last column in the Table 1) .

9 Conclusion and Future Work

In this paper, we proposed three variants of an innovative approach for multi-label text classification. We used Lucene to index textual data and metadata. We computed the similarity of a new text to all indexed documents. We used the classes of the best results to classify new texts. Depending on the nature of the corpus, we can use one of three versions of our method. If the text is quite simple and the number of classes is not high, one of the first two versions would be more appropriate. If the text is more complicated and we have a fairly high number of classes for each text, the third version should lead to better results.

In this work, we relied on an annotated corpus. Such a corpus is not always available. Annotating a corpus in the same manner as the one we used in this work is a time-consuming task. In future work, we propose to develop methods to help human experts to annotate a corpus based on the discussed topics. We believe that a method of automatically-extracting keywords could be used to help with this.

REFERENCES

1. Tsoumakas, G., Katakis, I.: Multi-label classification: An overview. Dept. of Informatics, Aristotle University of Thessaloniki, Greece (2006)
2. Aggarwal, C.C., Zhai, C.: A survey of text classification algorithms. In: Mining text data. Springer (2012) 163–222
3. Carvalho, V.R., Cohen, W.W.: On the collective classification of email speech acts. In: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, ACM (2005) 345–352
4. Cohen, W.W.: Learning rules that classify e-mail. In: AAAI spring symposium on machine learning in information access. Volume 18., California (1996) 25
5. Chakrabarti, S., Dom, B., Agrawal, R., Raghavan, P.: Using taxonomy, discriminants, and signatures for navigating in text databases. In: VLDB. Volume 97. (1997) 446–455
6. Liu, B., Zhang, L.: A survey of opinion mining and sentiment analysis. In: Mining text data. Springer (2012) 415–463
7. Lang, K.: Newsweeder: Learning to filter netnews. In: Proceedings of the 12th international conference on machine learning. (1995) 331–339
8. Srivastava, A., Han, E.H., Kumar, V., Singh, V.: Parallel formulations of decision-tree classification algorithms. Springer (2002)
9. Jordan, A.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Advances in neural information processing systems **14** (2002) 841
10. Gopal, S., Yang, Y.: Multilabel classification with meta-level features. In: Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, ACM (2010) 315–322
11. Hirsch, L., Hirsch, R., Saeedi, M.: Evolving lucene search queries for text classification. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM (2007) 1604–1611
12. Gospodnetic, O., Hatcher, E.: Lucene in Action. Manning (2005)
13. Porter, M.F.: An algorithm for suffix stripping. Program **14**(3) (1980) 130–137

14. Porter, M.F.: Snowball: A language for stemming algorithms. http://snowball.tartarus.org/texts/introduction.html (2001)
15. Owen, S., Anil, R., Dunning, T., Friedman, E.: Mahout in action. Manning Shelter Island (2011)

MOHAMED MOUINE
UNIVERSITY OF OTTAWA
800, KING EDWARD STREET, OTTAWA, ON, K1N 6N5, CANADA
E-MAIL: <MMOUINE@UOTTAWA.CA>

DIANA INKPEN
UNIVERSITY OF OTTAWA
800, KING EDWARD STREET, OTTAWA, ON, K1N 6N5, CANADA
E-MAIL: <DIANA.INKPEN@UOTTAWA.CA>

PIERRE-OLIVIER CHARLEBOIS
REDOCK INC.
5369 CANOTEK RD #3, OTTAWA, ON, K1J 9J3, CANADA
E-MAIL: <POCHARLEBOIS@REDOCK.COM>

TRI HO
REDOCK INC.
5369 CANOTEK RD #3, OTTAWA, ON, K1J 9J3, CANADA
E-MAIL: <THO@REDOCK.COM>