Identifying Keywords in Random Texts

Ibrahim Alabdulmohsin
Gokul Gunasekaran

December 9, 2010

Abstract
The subject of how to identify keywords in random texts lies at the heart of many important applications such as document retrieval, bibliographic databases, and search engines. It has received a fair interest in the research community ever since the late 1960s, where most approaches to-date have mainly relied on frequency analysis and word association. In this project, we present a different approach to achieve such objective by using Machine Learning algorithms such as support vector machines (SVM) and GDA, both with uncertain label classification. While such approach uses minimal linguistic tools and relies instead on pure numeric features, which might seem at first to be at odds with the original objective, the motivation behind the adoption of such Machine Learning algorithms is driven by similar successful applications in various other disciplines such as optical character recognition. Both SVM and GDA achieve a very good performance in keyword identification, where GDA usually performs much better than SVM.

1. Introduction

Automatic keyword identification can be informally described as a process by which a short list of keywords is extracted out of a much larger text with little loss of information. It is used to provide an efficient method for both humans and machines to quickly identify the content and type of texts and documents. Such process of automatically identifying keywords in random texts using computer-aided tools and algorithms is a crucial task in the modern Information Era. It lies at the heart of many important applications such as indexing in bibliographic databases and Internet search engines as well as classification in directory services. However, because a measure for *information loss* clearly differs from one application to another, a generic customizable approach is also needed that can have a wider impact and applicability.

The subject of keyword identification has received a fair interest in the research community ever since the early dawn of computers in the 1960s. Probably, one of the earliest attempts to tackle a related subject was an experimental inquiry conducted by M. E. Maron in 1961, which demonstrated an efficient use of Bayesian analysis to categorize documents based on their "clue words" [1]. In a different study, Bayesian networks were used to assist in indexing articles through the use of a thesaurus. Because the identification method was hard-coded, no training was required in such study [2]. Furthermore, many sophisticated techniques have been developed based on frequency analysis that range from the vey simple methods, such as the use of middle frequency words, to advanced methods such as tf-idf, which takes into account the frequency of a word across multiple documents [3]. In addition, frequency analysis has also been supplemented by the use of linguistic tools such as Stop Words and stemmers [3]. The use of frequency analysis and linguistic tools is probably the most pervasive approach nowadays.

Support Vector Machines (SVM) is a learning algorithm developed by Vapnik and his colleagues in the early 1990s as a method for obtaining optimal margin classifiers in high-dimensional feature space [4][5]. Its immense success and popularity has made it one of the most important supervised learning algorithms in use today. With regard to our application, SVM enjoys key advantages that make it very suitable for automatic keyword identification.

For instance, because SVM works purely on numeric features with no regard to their underlying principles, such as language grammars, it is highly extensible to multiple languages, which is ideal for our application. In addition, SVM is easily customizable to the needs of a particular application. For instance, while keyword identification might be needed in some cases to point to the general nature of an article as in directory services, it may also be needed in other applications to indicate both nature and contents of that article such as in search engines. Using SVM, the same algorithm is used by all applications, where only the method of labeling/classifying words during training would differ. Despite such potential, we are not aware of any previous attempts to use SVM in automatic keyword identification, and we hope that this project will shed some light into this topic.

Gaussian Discriminant Analysis (GDA) is a generative learning algorithm that attempts to use the training set to create models for classes independently, as opposed to discriminative learning algorithms that merely attempt to separate classes [6]. In our application, GDA enjoys three key advantages. First, in nearly all of the features being used, such as word frequency, the probability of observing a keyword increases as the feature value increases. For example, when Stop Words are removed, remaining words with high frequencies are more likely to be keywords within the text. Thus, the probability mass of the two classes should be nearly separable in R^n using multivariate Gaussian densities, where the probability mass of the negative class lies closer to the origin. Also, because GDA is a generative learning algorithm, it can make full use of the training set by attempting to build a model of each class independently of the others, whereas discriminative learning algorithms such as SVM tend to perform poorly on highly unbalanced training sets. In our application, keywords constitute less than 5% of the entire training sets, so balancing the set in SVM means removing more than 90% of the data; a clear disadvantage. Third, GDA is a simple learning algorithm that is fast during both training and testing, and is less susceptible to overfitting.

2. Methodology

As is common in Machine Learning implementations, the general methodology is divided into four main phases: preprocessing, normalization, training & feature selection, and testing. We describe each of these steps in details next.

2.1 Preprocessing

The primary objective of the preprocessing phase is to extract feature vectors out of texts in their native-language format. In order to accomplish this task, a text goes in sequence through (1) linguistic processing, such as stemming and Stop Word elimination, (2) feature extraction, and (3) labeling/classification. For the choice of features, we have decided to extract a total of 16 features initially at our disposal. In addition to the frequency of the token across the entire text, the initial set of features include both attributes of the entire text itself such as its length in words and sentences as well as attributes of the token itself such as its type and length in characters. Depending on performance during the cross-validation phase, some of these features have been removed while others have been slightly modified as will be discussed in the Results section.

In addition, because the identification of keywords is a subjective measure that partially depends on the human reader, a strict labeling scheme may not be advantageous. For example, in the context of Machine Learning, the word *'learning'* has definitely a far greater weight than, say, *'optimization'*, while the latter, in turn, may have a greater weight than, for example, *'labeling'*. In order to incorporate uncertainty in classification, different possible approaches can be used. For instance, one approach would be to reformulate the optimization problem by incorporating uncertainty and solve its dual, in a manner similar to standard SVM implementations. Another approach to incorporate uncertainty is to use SVM-regression, which is the approach we adopted in this project. Using the latter approach, the labels $y^{(i)}$ take values in [-1, 1], where prediction is determined by a threshold on the margin predicted by SVM. In GDA, one the other hand, we decided to use two different covariance matrices for the positive and negative classes, which had led experimentally to a much better performance than conventional GDA. To incorporate uncertainty, maximum likelihood estimation yields the following equations, which are identical to the M-step update rule in the EM algorithm:

$$\phi_j = 1/m \sum_{i=1}^{m} w_i \, 1\{y^{(i)} = j\} \tag{1}$$

$$\mu_j = \left(\sum_{i=1}^{m} w_i \, x^{(i)} \, 1\{y^{(i)} = j\} \right) \Bigg/ \sum_{i=1}^{m} w_i \, 1\{y^{(i)} = j\} \tag{2}$$

$$\Sigma_j = \left(\sum_{i=1}^{m} w_i \, 1\{y^{(i)} = j\} \, (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T \right) \Bigg/ \sum_{i=1}^{m} w_i \, 1\{y^{(i)} = j\} \tag{3}$$

Here, $j \in \{0,1\}$ denotes the class type, while $w_i \in [0,1]$ denotes the certainty of classification.

2.2 Normalization

Normalization is a process of rescaling parameters so that they are all given equal weight during optimization. In the normalization phase, we have experimented with different rescaling methods such as linear rescaling and standard Gaussian normalization, given by $\left[(\text{value} - \min)/(\max - \min)\right]$ and $\left[(\text{value} - \text{mean})/(\text{std})\right]$ respectively. We also tried to use raw data directly without normalization. Finally, we also used the linear normalization method given by the equation $\text{value} := \text{value}/\max$. The latter normalization approach, combined with the feature selection method discussed in the subsequent section, yielded the best results.

2.3 Training and Feature Selection

In order to train both SVM-regression and GDA, we collected a list of 15 articles that covered various topics and varied in length from 100 to over 3000 words per article. Each document was preprocessed in Python, and a label for each word was manually assigned by the user. For stemming, we used the Porter Stemmer in Python, while Stop Words were collected in a single file from various sources. However, we have decided to retain Stop Words that are emphasized[1]. For instance, while 'new' is typically a Stop Word, 'New' in 'New York' is not. When Stop Words are removed, the final training set contained approximately 2,700 data samples, 5% of which are for the positive class.

For the SVM-regression training phase, we used *SVM-light*, which is a very popular implementation of SVM developed in C [7]. Inherently, SVM offers a large degree of freedom that needs careful assessment. For instance, we have various choices for the kernel function and its parameters, the tradeoff parameter, and also the choice of features. In GDA, by contrast, training is relatively straightforward using the closed-form equations given above, where the only freedom available is in the choice of features. In order to determine the performance of a particular choice during training, we used the *hold-out cross validation* method, where 70% of the data are used for training and 30% are used for cross-validation. To measure performance during cross-validation and testing, we computed the weighted

[1] In our application, emphasis is detected when the first letter is capitalized because only documents in text format were used. However, the method presented herein should be easily extensible to additional kinds of emphasis such as the use of **bold** or *italic* fonts in HTML documents.

average of false positives and false negatives, where a misclassified example is weighted by the uncertainty of its label. Obviously, because the negative class constitutes more than 95% of the entire data set, the probably of false positives and false negatives was a more descriptive measure.

Feature	Description	Rationale
1. Token Type	= 0: if the token is alphabetic 0.5: if the token is numeric 1: otherwise	The nature of the token should be factored in during prediction.
2. Token Length	# of characters in the token	
3. Fraction of emphasis	# fraction of time the word is emphasized (e.g. capitalized).	Emphasized tokens tend to be keywords
4. freq1 x log(Document_Length)	# frequency of occurrence of the token in the 1st block of the text, multiplied by the natural logarithm of document length (in words)	Keywords tend to appear persistently throughout the text and not only in a single block. Also, document length is important. For instance, most words tend to have high frequencies in short documents, which does not imply that they are keywords.
5. freq2 x log(Document_Length)	….	
……	….	
13. freq10 x log(Document_Length)	# frequency of occurrence of the token in the 10th block of the text, multiplied by the natural logarithm of document length (in words)	

Table 1: A list of the final features used in automatic keyword identification

2.4 Testing and Results

The final set of features is depicted in table 1 above. As shown in table 1, some of the features extracted initially were removed, while others have been slightly altered. For example, while the use of *document length* as a feature is important, e.g. all words in short documents will have unusually high frequencies, both SVM-regression and GDA, unfortunately, tended to overfit such feature. To solve the problem, we removed document length as an *explicit* feature but included it *implicitly* as shown in the table. Using this feature vector, both SVM-regression and GDA yielded good results, where GDA typically outperformed SVM as shown in figure 1. For example, GDA correctly detects, on average, 90% of all keywords while SVM detects 70% of keywords only, as weighted by the confidence of classification. In addition, the percentage of false positives is 30% in GDA and 40% in SVM. Given the nature of the task where small errors are tolerable in favor of an automated approach to keyword identification, such results are indeed promising. Also, the SVM kernel that worked best for this application was the polynomial kernel, where c=1 & d=3. In order to illustrate the effectiveness of such approach, figure 2 presents a snapshot of a short article and the top keywords identified by both SVM and GDA. Clearly, the keywords suggested by GDA are almost indistinguishable from what a human reader would suggest for that article.

3. Conclusions

The subject of automatic identification of keywords in random texts has many important applications. In the past, most approaches focused on the use of frequency & word association and Bayesian analysis to predict keywords. Such approach has limitations that can be avoided through the use of purely numeric Machine Learning algorithms such as SVM and GDA. With a carefully designed feature set, both learning algorithms can yield excellent results. In general, GDA with different covariance matrices is recommended because it is faster to train and predict, simpler, and yields much better results than SVM.

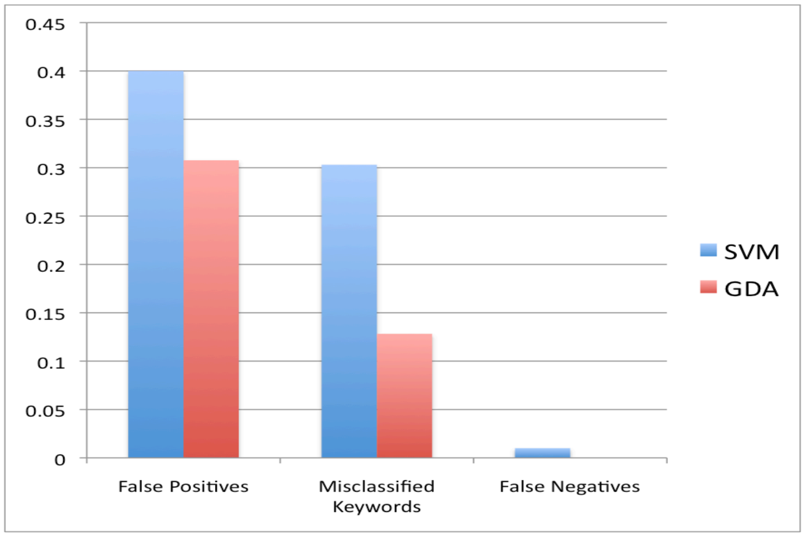

Figure 1: A summary of testing results for both SVM-Regression and GDA

PC cooling systems are one of the hottest - and coolest - computer hardware accessories available. Most computers, however, still do not come with a cooling system more advanced than the typical heat sink and cpu fan. Some computers, however, are being sold with liquid cooling systems, such as the Apple G5 Power Mac. The dual processors undoubtedly put out a great deal of excess heat that a simple copper heat sink and fan alone can not properly dissipate. Intel is also getting into the new cooling trend, claiming that it's newest processors (of the Pentium 4 line) are approximately 20 degrees hotter than any chip currently produced by AMD (Advanced Micro Devices) …. …. **(additional text)**
Here are some popular options if you are looking to add a new cooling system to your computer:

1- Water cooling system. … **(additional text)**
2- Memory cooling system …**(additional text)**
3- Heavy Duty fan system …**(additional text)**

…. …. …. **(conclusions)**

SVM Keywords (in decreasing order):
```
cooling, fan, processor, heat, system, computer, fan-type, Pentium, AMD,
high-end, CPU, Intel, micro, devices, powerful, memory, sink, heavy-duty,
RAM, advanced
```

GDA Keywords (in decreasing order):
```
cooling, system, computer, processor, heat, fan, hardware
```

Figure 2: A sample of keywords predicted by SVM and GDA for an article on "PC Cooling Systems"

References

[1] M.E. Maron, "Automatic Indexing: An Experimental Inquiry", *JACM*, Volume 8 Issue 3, (1961).

[2] L. de Campos, J. Fernández-Luna, J. Huete, and A. Romero, "Automatic Indexing from a Thesaurus Using Bayesian Networks: Application to the Classification of Parliamentary Initiatives", *Proc ECSQARU 2007*, 865–877, 2007.

[3] G. Salton, "Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computers", *Addison-Wesley, Reading*, Pennsylvania (1989)

[4] B. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers." *Fifth Annual Workshop on Computational Learning Theory*, Pittsburgh, 1992. ACM

[5] A. Ng, "Support Vector Machines", CS229 Class Notes (2010).

[6] A. Ng, "Generative Algorithms", CS229 Class Notes (2010).

[7] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.