Classifying Web Pages Using a Support Vector Machine

Erik Schaftenaar, s1691066
Supervised by: Marco Wiering*, Markus Jelsma†

Abstract

The World Wide Web keeps expanding at an enormous rate, tens of thousands of new pages are added daily as it becomes easier for everybody to share information. However, information is only useful when it can be retrieved. Search engines are used for this retrieval, but they are losing precision because of the ongoing expansion of the web. In order for search engines to regain precision they can be enhanced with category filters. However the web is too big to be classified by hand. In this article the possibility is explored to classify web pages in basic categories using the machine learning tool Support Vector Machines. Machine learning tools require a pre-classified training set in order to function, and manually classifying web pages for the training set can take up a lot of time. To tackle this problem, already classified home pages from a web directory are used as training set. Even though the training set is not verified the classifier scores very well. This is evidence that enhancing search engines with a category filter is not a difficult task. Because users of search engines unintentionally give feedback about the accuracy of the category filter, they help to improve it by simply using the search engine. This way more and more of the web can be categorized, and in greater detail so that information becomes more easily retrievable.

1 Introduction

1.1 World Wide Web

The World Wide Web is a beautiful source of information and it keeps on expanding at a high rate. Most new information gets stored digitally and a lot of information gets digitalized. However information is only useful when it can be retrieved. Information can be retrieved in different ways chief amongst them being: using a web directory with categorized web-sites like the [Open Directory Project, 2012] or [Yahoo! Directory, 2012], or using a search engine like Google or Altavista.

1.2 Information retrieval

Both search engines and web directories have their advantages and drawbacks. Using a web directory one can find web sites (multiple web pages) about a specific topic while using a search engine one finds web pages containing the words in the search query. If one wants to know the population of the island of Java one should use a search engine. But if one wants general information about the island of Java one might be better off using a web directory, because the search term Java would be ambiguous because of the programming language. Of course search terms can be added to the query in order to better specify what is meant, or sometimes boolean operators can be used to filter out certain words, but this decreases the user-friendliness of the search engine.
A drawback of web directories on the other hand is that the web pages are manually categorized by experts, and they can't even remotely keep up with the expansion of the information available on the web. The Open Directory Project (claiming to be the biggest web directory) has only a little more than 5 million web sites categorized (July 2012) while the World Wide Web is estimated to be containing about a trillion web pages [Kelly, 2010].

1.3 Need for classification

The problem with information retrieval using a search engine or a web directory lies in the fact that they "suffer either from poor precision (i.e.,

*Rijksuniversiteit Groningen, Department of Artificial Intelligence
†OpenIndex, http://www.openindex.io

too many irrelevant documents) or from poor recall (i.e., too little of the Web is covered by well-categorized directories)" [Chekuri et al., 1997]. A solution to this problem is to supply search engines with the possibility to only display web pages of certain categories, and in order to do this all the web pages a search engine contains need to be classified. As mentioned earlier the World Wide Web is too big to be classified by hand, and classification needs to happen automatically. Research on web page classification using machine learning has been conducted many times before [Kwon and Lee, 2003] [Kriegel, 2004], a few different approaches on using multiple web pages from a website are summarized by Qi and Davidson [Qi and Davison,]. Research regarding the use of web directories as training data for web page classification using Naive Bayes has proven to be quite fruitful [Mladenic, 1998], however this research will focus on classification using a Support Vector Machine as it "significantly outperforms Naive Bayes" [Yang and Liu, 1999].

1.4 Goals

The aim of the research is to determine the possibility of enhancing a search engine with a feature to filter on categories. Such a feature requires a good classification algorithm, and the performance of this algorithm is based on how well it is trained. Because training this algorithm requires a manually categorized training set, which is very time consuming to create, this research will determine how useful pre-categorized web pages from a web directory are in order to classify other web pages.

2 Methods

2.1 Dataset

In order to test the performance of a Support Vector Machine regarding the classification of web pages, a dataset needs to be constructed. This dataset contains already classified documents on which the algorithm can train and test. The dataset needs to contain a representative part of the World Wide Web in order to robustly classify different types of web pages into specific categories. It is very difficult however to get a true representative portion of web pages as there is no index of all the web pages out there. Other important aspects of the dataset are that all the web pages need to be classified correctly, and the dataset is large enough to train on. This poses another problem, because to be certain the web pages in the dataset are classified correctly, they need to be classified manually and to do so for a sufficiently large dataset this takes up a lot of time.

2.2 Open Directory Project

To deal with these problems the web sites listed on the Open Directory Project were used. Only the home pages listed were used, as to not influence the algorithm by blocks of text which are repeatedly used on different web pages belonging to the web site. This way a sufficiently large, quite representative dataset was constructed, however the quality of classification is unsure, because the web directory used might contain broken links and it is not certain the homepage contains information about the specific category.

2.3 Obtaining the data

The data was obtained using the web crawler Nutch [Apache Nutch, 2012], it downloaded the home pages linked by ODP in a certain category. The links supplied were the first 500 links found in the category. The data was downloaded in the html format, meaning it still contained all the html tags including header and layout data. Because this data is mostly not useful for classification, a script was written to remove such data and only keep the relevant body data for classification. Because after cleaning up the data some pages barely contained any text, or only a single line informing the page was moved, only pages greater than 1kB were added to the database.

2.4 SVM

In order to actually classify the web pages a Support Vector Machines library created was used [Chang and Lin, 2001]. SVMs should work very well for text classification for various reasons. SVMs support a "high dimensional input space" which is useful considering the large amount of words on a web page. SVMs can handle a high dimensional input because they use "overfitting protection, which does not necessarily depend on the number of features". Also "most text categorization problems are linearly separable" which is good because the "idea of SVMs is to find linear separators" [Joachims, 1998]. The main idea of SVM is to map the input data into a high dimensional input space, and find a hyperplane that will separate the different classes of input data. For optimization it will maximize the margins [Bennett and Campbell, 2000] between the hyperplane and the support vectors, which are the vectors closest to the hyperplane.

2.5 Input values

To create the vectors in the multi dimensional space certain input values are presented to the algorithm. To create the input values basically all the words in the dataset are counted. Most non-alphanumeric characters in the dataset are replaced by spaces, so that for example the quotes are removed from a word to correctly count them as the same word without quotes. For the same reason all data will be converted to lowercase. Also will be taken into account that very common or small words do not tell much about the category, therefore very common words and words smaller than 4 characters will not be counted. All the words, and the number of times they occur in the dataset form the input values for the algorithm, because not only that a word occurs in a specific category, but also how many times it occurs is valuable information to train the algorithm on.

2.6 Testing the algorithm

In order to determine how well the algorithm can classify web pages a number of tests will have to be performed. The expectation is that the closer two categories are related to each other, the more likely it will become for the algorithm to make a mistake. Therefore a test will be performed on totally unrelated categories in order to establish a control group. In order to test the robustness of the algorithm, meronym categories will be tested in order to ascertain if two subcategories belonging to the same super-category, can be successfully distinguished from each other. Because a search engine implemented with the classification algorithm mostly has to deal with ambiguity, a test will be conducted on homonyms, a word with more different meanings.

2.7 Cross validation

Because the datasets constructed for each category may not be sufficiently large, there will be made use of cross validation. The pre-labeled dataset will randomly be divided into two sets, the training set an the test set. The algorithm uses the training set to find the support vectors and create a model of hypothesis space and the hyperplane separating the different categories. After the model has been trained there will be analyzed how well it works by verifying that the test samples are placed on the correct side of the hyperplane.
The cross validation fraction refers to the percentage of the dataset that is placed in the training set, the remainder will be placed in the test set. By using different cross validation fractions there is to be expected to gain some insight on how the sample size affects the amount of correctly classified webpages.
An aspect of cross validation is the random division of the dataset. To attempt to counteract anomalous results caused by this randomness, multiple iterations of a specific cross validation fraction will be run. The average of the results of the multiple iterations will even out the anomalous results. Per test, per cross validation fraction, 15 iterations will be run.

3 Results

The results were obtained using a libSVM implementation for classifying emails [Elzinga and Znaor, 2011], modified for the use of classifying web pages. Table 1 shows the amount of samples per category, which are the web pages that made it in to the dataset, and the total number of input vectors per category, which is the number of different words the algorithm trained on. Because many words are in multiple data sets, those input values are merged when testing on categories containing those same words. For example when testing on all categories, only 62108 input vectors remain as can be seen in Table 1, which is 52.2 percent of the total input vectors per category.

Table 1: Number of samples and input vectors per category.

Category	Samples	Input vectors
Games	234	25815
Cooking	294	31892
Hockey	247	19986
Baseball	221	28315
Java Programming	69	5814
Java Island	62	7270
Combined	1127	62108

4 Discussion

4.1 Overall performance

The overall performance of the algorithm is quite good, as can be seen in Figures 1, 2, and 3. In general the tests show that with as little as about 80 samples in the training set a performance of 90 percent correctly classified web pages can be achieved. These samples are not classified by hand and therefore the performance is especially good. As expected the games versus cooking test shows the best results (about 98 percent correctly classified) because the categories differ greatly from

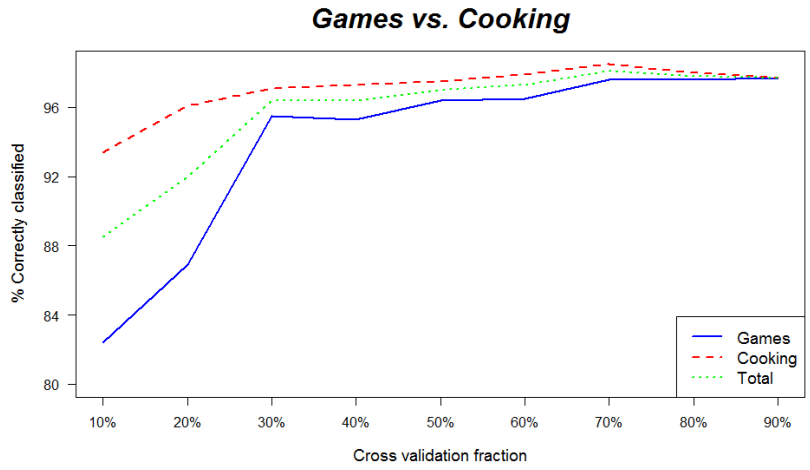

Figure 1: Graph of the percentage correctly classified web pages versus the cross validation fraction of the categories games and cooking with a total of 528 samples.

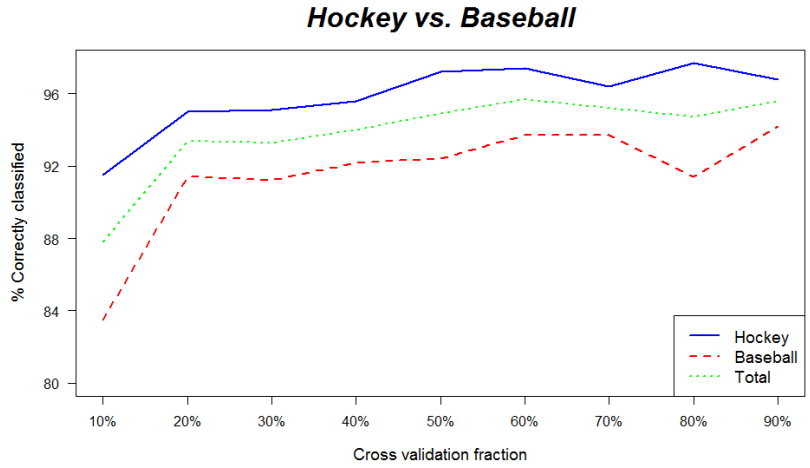

Figure 2: Graph of the percentage correctly classified web pages versus the cross validation fraction of the categories hockey and baseball with a total of 468 samples.

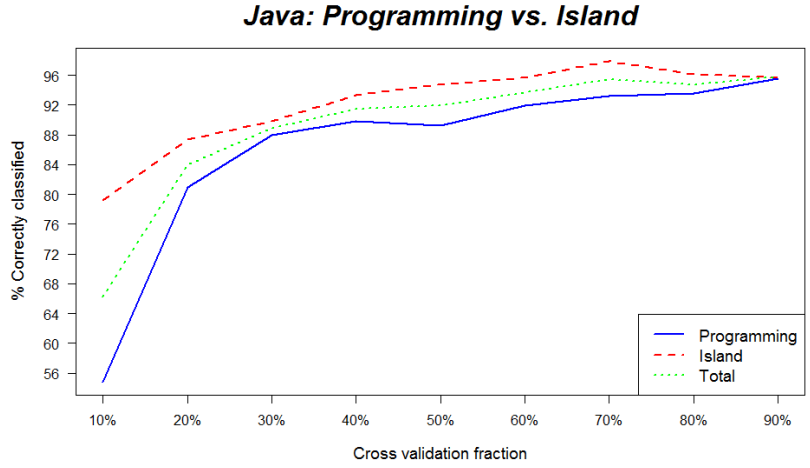

Figure 3: Graph of the percentage correctly classified web pages versus the cross validation fraction of the categories Java the island and Java the programming language with a total of 131 samples.

each other. The meronym test hockey versus baseball also shows good results (about 95 percent correct) considering they belong to the same supercategory.
This result can be explained by the fact that although overlapping sports terms are not so useful for classification, there are sports terms specifically belonging to the category. For example 'bat' and 'pitcher' can be used to classify a web page as baseball, while 'stick' and 'goal' are specific to hockey. But because of the overlap in sports terms the effectiveness of the dataset decreases, and this could explain why the results for this test are a bit lower than the results in the cooking versus games test.

4.2 Noteworthy results

4.2.1 Better scoring categories

It appears that some categories are more likely to be classified correctly than others. They keep scoring significantly better than their counterparts.
One explanation for this can be that the training set contains more data from one category than the other. While roughly containing the same amount of documents, cooking scores higher than games, and the cooking dataset has a size of 2.1 MB while the size of the dataset of games is 1.3 MB which results into a different number of input vectors. The results from the other tests show the same occurrence of one category scoring better than the other, while in those cases the lesser scoring category is larger in size than the better scoring one. However the difference is not as big as in the cooking versus games test.
Another explanation is that some categories are broader than others. It is not difficult to imagine that the cooking category is narrower than the games category. Cooking websites will mostly contain instructions to prepare food and techniques in order to do so. While the category games is broader, because there are many different types of games, for example boardgames, games with dice, card games and there are numerous different types of computer games.
This results in the fact that narrower categories have more 'tells' than others. The cooking category probably contains a lot of words like for example 'boil', 'fry' and 'microwave' which immediately give away the category. The games category also has these tells, but they occur less frequently because there are more of them.
This explanation is supported by the fact that the performance of the lesser performing category rises more rapidly than the performance of the better scoring one as more web pages are put in the training set. It continues to learn new tells, while it already trained on the tells of the narrower category.

4.2.2 Decrease in rate of improvement

There appears to be some sort of inflection point in the curves of correctly classified web pages where the rate of improvement suddenly decreases quite rapidly as more data is added to the training set. Of course it stands to reason that the curves should have an asymptote (preferably at 100 percent correct), but with as little as about 80 samples the algorithm classifies 90 percent correctly while adding another 80 samples only increases the accuracy with a few percent.
The number of samples needed to hit the 90 percent is not for all samples the same, as narrower categories need less samples. The amount of samples needed to achieve 90 percent correct classification might be useful as an indicator on how broad (or narrow) a category is. The used svmlib library places the web page from the test set in the most likely category based on probability with a threshold of 50 percent, but it would perform better if it takes into account this indicator of broadness in its threshold. The main idea behind this, is that if the algorithm is very unsure about a sample, the probability is higher that is belongs to the broader category because there tend to be more pages about a broader category than a narrower one.

4.2.3 Maximum performance

In the test results there appears to be an asymptote of less than 100 percent correct classification. However this asymptote is in theory always 100 percent, because when the algorithm is trained on an infinite amount of data every test sample should be classified correctly because it must appear in the infinity long training set. The simple explanation is that the dataset contains wrong data. Wrong data in the meaning that it is not natural language, or that it is wrongly classified. If there are two similar gibberish files, or two files belonging to a totally different category, but both are manually classified in the opposite categories a problem arises. Because when trained on one file, the other will be classified into that same category and be considered to be wrongly classified. Therefore, in order to keep the theoretical maximum at 100 percent it is important that the dataset only contains correctly pre-classified data, the data set used in this experiment clearly does not.

4.3 Misclassifications

Although most classification is done correctly, the dataset is not optimal. In order to get an insight as to where it goes wrong, the misclassified documents need to be inspected. The misclassified documents can be divided into a few categories.

Some home pages display a kind of disclaimer. An example of such a disclaimer is a disclaimer mandatory by the new EU cookie law. The disclaimer informs users that cookies may be installed on their computer, and the user needs to agree to this in order to visit the web page.
Although specifically the English section of the web directory was crawled, some web pages in the dataset were still in another language. This is harmful for the training set, because the classifier will act like all pages in that specific language belong to the category that the foreign page belongs to.
Another example of pages that should not be in the training set are semi broken links. Some web pages downloaded contained an offer from a webhoster in order to attract people for buying that specific domain.
Another problem that was encountered was that on certain web pages spam sneaked in. Advertisement from gambling websites, online pharmacies and even from porn sites. Most of the time the text from the spam message was many times greater than the text from the actual web page.
These web pages are hard to classify and drag down the performance of the algorithm. But more importantly they are not useful for the training set, because they do not train the model on the category they were supposed to. With such a distorted model it makes it a bit more difficult to classify normal web pages.

4.4 Importance of a good dataset

There can be concluded from the noteworthy results and the misclassifications that a good dataset is important in order to get a high percentage of correct classifications. A good dataset is sufficiently large so that almost all words belonging to a category are represented in it. It is also important that noise is reduced to a minimum. Noise is data considered to be irrelevant and does not help the algorithm to create a representative model, but in fact distorts the model of a category. To illustrate the negative effects of noise the baseball versus hockey test was ran again. However, all web pages were manually removed of which there was no doubt they would not be helpful for the model. This was done by looking very briefly at a web page in order to make sure there were a few coherent sentences for the classifier to train on. Figure 4 shows that removing noise from the dataset improves the results, and makes it possible to correctly classify web pages with few samples trained on.

4.5 Usability in a search engine

As the test results show it is quite possible to implement a good performing learning category filter in a search engine without much manual classification. It is however important to have large training sets with as little noise as possible. There are ways to automatically remove the noise from the crawled dataset from a web directory. It is for example possible to check whether the crawled web page contains a sufficient amount of different English words before adding such an example to the training set. Although manually checked, the hockey versus baseball dataset was modified to remove noise in a similar way.
A way to implement categorization in a search engine, is to classify the first hundred search results of a query posted by a user (if not already classified). The different categories found of these first results will be listed, and give the user the possibility to filter the results on the category they are looking for. A problem that could arise is that some pages are misclassified as belonging to a completely different category, and are never shown again. This problem will eventually solve itself, but if it is too risky for the reputation of the search provider the filtering can also happen the other way around. The user then has the choice to stop the engine from displaying a category. For example, when a website about Java programming is completely misclassified as cooking, filtering out the Java island web pages will not stop the search engine from displaying this web page.
A very useful aspect of a search engine is that it is used by a lot of people who unintentionally give feedback on the performance if such an algorithm was implemented. This feedback can be used in many ways. If for example a search result was visited a lot before implementation, but not anymore after, it is probably wrongly classified. And when a categorized search result is visited a lot, it is certain that it is classified correctly and that web page can enhance the training set.
This way users of search engines can classify the web for them by unintentionally giving feedback, this will also deal with the misclassified web pages. When a category is classified for a large enough part, subcategories can be added to the dataset to enhance precision of the classifier, and making the search engine even more useful. When enough depth in categories is achieved, users will increasingly use the classification option. Because, the more specific the search for certain information, the more irrelevant pages will be shown by search engines without a category filter.

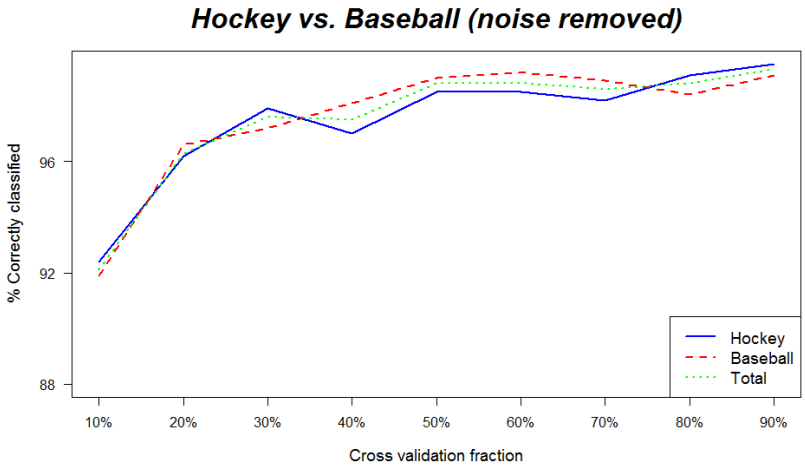

Figure 4: Graph of the percentage correctly classified web pages versus the cross validation fraction of the categories hockey and baseball with a total of 214 samples which contain little to no noise.

4.6 Conclusion

In order to keep track of all the data on the ever expanding World Wide Web, good methods of information retrieval must be designed. To retrieve information it is necessary to specify exactly what it is that needs to be found, and in order for information to be retrieved it is necessary that the information is labeled as exactly what it is. It is impossible to label all the information by hand, and this article shows that this is not even necessary, when using a web directory. By enhancing search engines with the option to search on categories, which are classified using machine learning, users will help to classify the web. The machine learning algorithm can learn from the users and subcategories can emerge until, in time, all documents on the web are labeled in so much detail that they are labeled as exactly what they are.

References

[Apache Nutch, 2012] Apache Nutch (2012). `http://nutch.apache.org/`.

[Bennett and Campbell, 2000] Bennett, K. P. and Campbell, C. (2000). Support vector machines: Hype or hallelujah? *SIGKDD Explorations*, 2(2):1–13.

[Chang and Lin, 2001] Chang, C.-C. and Lin, C.-J. (2001). Libsvm: a library for support vector machines. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`.

[Chekuri et al., 1997] Chekuri, C., Goldwasser, M., Raghavan, P., and Upfal, E. (1997). Web search using automatic classification. In *Proceedings of the 6th WWW Conference*, Santa Clara, California.

[Elzinga and Znaor, 2011] Elzinga, L. and Znaor, A. (2011). Email classifier. `code.google.com/p/email-classifier-mai-ml/`.

[Joachims, 1998] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, pages 137–142.

[Kelly, 2010] Kelly, K. (2010). *What Technology Wants*. Viking Press.

[Kriegel, 2004] Kriegel, H.-P. (2004). Classification of websites as sets of feature vectors. In *Proc. IASTED DBA*, pages 127–132.

[Kwon and Lee, 2003] Kwon, O.-W. and Lee, J.-H. (2003). Text categorization based on k-nearest neighbor approach for web site classification. *Information Processing and Management*, 29(1):25–44.

[Mladenic, 1998] Mladenic, D. (1998). Turning yahoo to automatic web-page classifier. In *European Conference on Artificial Intelligence*, pages 473–474.

[Open Directory Project, 2012] Open Directory Project (2012). `http://www.dmoz.org`.

[Qi and Davison,] Qi, X. and Davison, B. D. Web page classification: Features and algorithms. *ACM CSUR II*, 41(2). Article 12.

[Yahoo! Directory, 2012] Yahoo! Directory (2012). `http://dir.yahoo.com/`.

[Yang and Liu, 1999] Yang, Y. and Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42 – 49.