World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:1, No:9, 2007

The Influence of preprocessing parameters on text categorization

Jan Pomikálek
Masaryk University
Brno, Czech Republic
xpomikal@fi.muni.cz

Radim Řehůřek
Masaryk University
Brno, Czech Republic
xrehurek@fi.muni.cz

International Science Index, Industrial and Manufacturing Engineering Vol:1, No:9, 2007 waset.org/Publication/6976

Abstract**—** Text categorization (the assignment of texts in natural language into predefined categories) is an important and extensively studied problem in Machine Learning. Currently, popular techniques developed to deal with this task include many preprocessing and learning algorithms, many of which in turn require tuning non-trivial internal parameters. Although partial studies are available, many authors fail to report values of the parameters they use in their experiments, or reasons why these values were used instead of others. The goal of this work then is to create a more thorough comparison of preprocessing parameters and their mutual influence, and report interesting observations and results.

I. Introduction

As the number of electronic documents available (primarily through the Internet) increases, acquiring relevant ones by hand becomes impossible. One solution to automating this task is to use Machine Learning (ML) techniques. Text categorization (TC) deals with the particular subtask of assigning predefined labels to documents in natural language. Applications of text categorization include document query systems (e.g., Google), spam filtering, word sense disambiguation and others.

II. Parameters

The pipeline for obtaining learned classifier from a collection of electronic text documents comprises several non-obvious choices. The steps considered in our study, as well as the choices explored with each step, are summarized in Tab. I. For example, in the feature selection step, many different feature selection approaches are available [1], [2]. How to select one best method for our particular problem is, even after decades of research, still an open issue. In order to explore each step more thoroughly, we examine multiple possibilities.

To avoid problems connected with tuning our (or any other) IR system to a particular preprocessing pipeline, we run our experiments over whole preprocessing parameter space. Each preprocessing parameter here is viewed as a dimension in parameter space (see Tab. I). This is fundamentally different from the usual approach, which is to implicitly fix most of the dimensions (e.g., fix feature selection to MI, term weighter to tf·idf [3], stemming to Porter stemmer [4] etc.) and only vary one or two dimensions at a time (usually the number of features and machine learning algorithm). We argue that the independence assumption made by optimizing each dimension separately, may miss some crucial underlying dependencies.

What we look at is thus a *metaspace* of preprocessing parameter combinations, rather than a single concrete value of a particular test. While this approach allows us to derive powerful and robust generalizations about the joint influence of preprocessing parameters, it comes at a cost. The price to pay is computational complexity, as there is a combinatorial explosion in the number of distinct configurations (cf. $2 \cdot 8 \cdot 2 \cdot 2 \cdot 5 \cdot 9 \cdot 8 \cdot 7 \cdot 6 = 967680$, see Tab. I). Since we use a 10-fold crossvalidation, this becomes almost 10 million full test runs *on each corpus*, which is computationally infeasible. However, many of the parameter combinations are poor choices not worth investigating. Here we build on results of previous studies and research in the IR area, which suggest that for example stoplist selection, stemming and tokenization dimensions are basically independant of the rest of the space and can be investigated separately.

Such observations allow us to narrow down the final search space to 23040 tests (230400 with 10-fold crossvalidation) per corpus. In section V-B, we examine these results and report interesting patterns. It is also worth noting that we only mine the resulting database for patterns connected to the APE (amount of positive training examples) dimension. Obviously holding the APE dimension privileged is an arbitrary choice and we believe much more information, not directly connected to the IR objective here, is present in our database.

Moving away from the implicit, often unclear computational choices towards explicit parameter space allows us to represent our results more formally as a database. One obvious implication is the facilitation of automated data mining and the deployment of mathematical tools to compare and visualize subspaces that are of interest.

Our research is driven by Information Retrieval needs. Here performance on low amount of positive training data is crucial, because positive data is supplied by the user and is thus precious and scarce. This necessitates the exploration of how the amount of positive training examples influences other text categorization subtasks, such as the choice of the learning algorithm, feature selector and so on. At the same time, APE is also the least studied parameter and most studies hold it fixed and focus on the abovementioned learning algorithms or feature reduction. Studying joint performance of other subtasks with regard to varying APE will allow us to dynamically select the optimal categorization pipeline based on the amount of

TABLE I
EXPLORED PARAMETERS

step	*values*
tokenizers	word unigrams, bigrams
stoplist	yes[1], no
stemmer	Krovetz [5], Porter [4], Paice-Husk [6], Lovins [7], no stemming
feature selector	Information Gain, Document Frequency, Mutual Information, χ^2, Bi-Normal Separation [8], OddsRatio [9], Fager-McGowan, Term Strength [10], GSS [11]
number of features	100, 200, 500, 1000, 2000, 5000, 10000, all
term weighter	ann, atc, bnn, lnn, ltc, nnn, ntc [3]
classifiers	SVM[2] [12], Naive Bayes [13], kNN[3], Neural Networks[4] [14], C4.5 [15], Simple Linear Regression, Voted Perceptron=[5] [16], RepTree[6] [17]
threshold	s-cut [18], fixed[7]
number of positive examples	5, 10, 20, 50, 100, 150

International Science Index, Industrial and Manufacturing Engineering Vol:1, No:9, 2007 waset.org/Publication/6976

data factually available in the system at a certain moment.

III. DATASETS

We explore three text categorization data sets. Two of these, the Reuters-21578 ModApte and 20Newsgroups [20] data sets, are standard testbeds in text categorization research. The third one, Springer, was chosen because its documents fit more closely the type of data we are ultimately interested in for our project. It is a collection of conference proceedings published by Springer Verlag. We used the originating conferences as document classes. Thematically-related classes were manually merged to achieve a less fine-grained and more robust taxonomy.

Each corpus was trimmed to contain exactly 300 documents per category. Insufficiently populated categories were simply dropped. This was motivated by our intention to conduct experiments with different amounts of learning data while at the same time keeping the number of documents per category constant.

IV. METHODOLOGY

As mentioned above, we built our benchmark corpora so that they contain exactly 300 documents per category. Out of these three hundred, 200 are labeled as training and 100 as testing documents. Given a desired APE, we randomly select N training documents from each category. Then a binary classifier is learned for each category, using the N documents from the selected category as positive and training examples from all other categories as negative examples. All 100 documents per category are used for testing each classiffier.

To mitigate the effects of using randomness in document selection and in some of the ML algorithms, we ran each experiment with 10-fold crossvalidation. The results reported are the mean average from these 10 runs. For each run, a different subset of documents was randomly drawn from the pool of all available documents (both positive and negative examples). We trace efficiency in terms of performance measures; in the figures we report the microaveraged F1 measure. We also recorded seven other measures (Accuracy, Precision, Recall, Correlation Coefficient, 11pt Average, Top True Positives and Breakeven Point) and their respective standard deviations.

V. EXPERIMENTS

A. Reducing parameter space

Experiments with tokenizers, stemmers and stoplists confirmed conclusions of previous TC studies (e.g. [21]). Krovetz stemmer, which is the only representative of light stemmers, slightly outperformed all the other (heavy) stemmers. We also conclude that, much like stoplist selection, stemmer selection has very little impact on the overall categorization results. Even not using stemming at all (only converting tokens to lowercase) caused negligible performance loss. One novel observation in these three dimensions is the behaviour of bigram tokenization. While bigrams (i.e., using two consecutive tokens as a feature) perform miserably on 20Newsgroups and Reuters, it improves classification performance on Springer (see Fig. 1). We conjecture that bigram tokenization is only

TABLE II
CORPORA DETAILS (TRIMMED VERSIONS)

Corpus	*Number of categories*	*Number of documents*	*Average number of words per article*
Reuters	6	1800	167
20Newsgroups	20	6000	868
Springer	9	2700	4285

[1]We used the SMART system's [19] list of 524 common English words.
[2]SVMlight with linear kernel and default parameters.
[3]Based on preliminary experiments, we opted for k=11 and cosine similarity measure.
[4]We used network topology with one hidden layer of 6 *tanh* neurons.
[5]Based on preliminary tests, we chose cubic kernel with 3 iterations.
[6]WEKA implementation with default parameters.
[7]In our experiments we used a fixed threshold of 0.5 to divide documents into positive (relevant) and negative (irrelevant) based on their predicted score.

Fig. 1. Tokenizers comparison

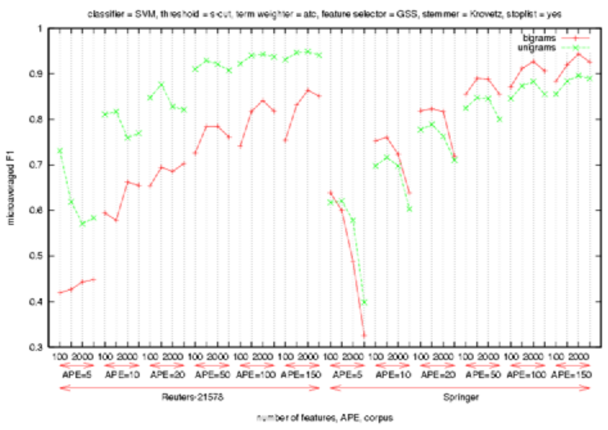


Fig. 3. Selected feature selectors comparison on 20Newgroups

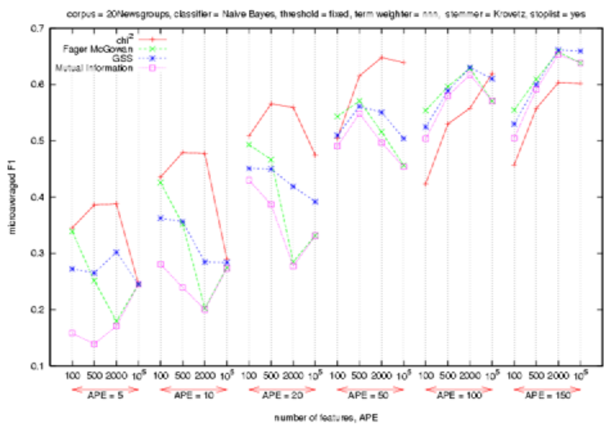


Fig. 2. Selected classifiers comparison on Reuters-21578

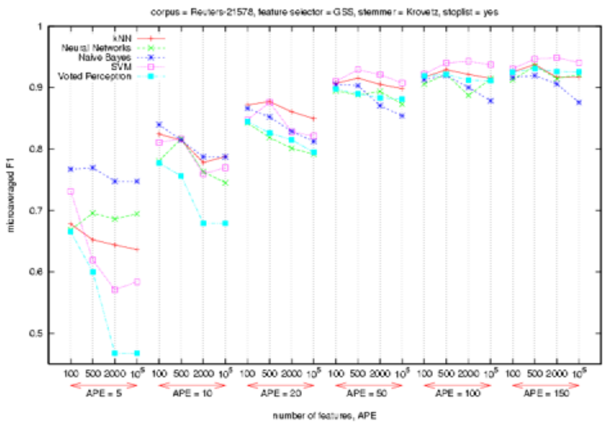
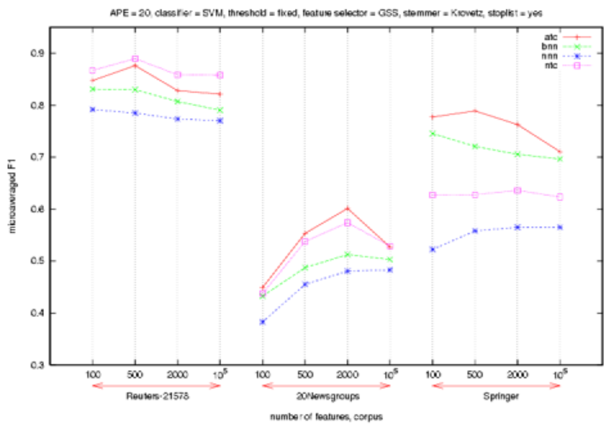


Fig. 4. Selected term weighters comparison



applicable to collections containing long documents, where it leads to an improvement in categorization at the cost of major performance loss in feature selection.

B. *Empirical observations*

An unexpected but persistent pattern in our data is the behaviour of Naive Bayes with respect to the threshold dimension. This is particularly important because Naive Bayes, along with kNN, proves to be the best performer on very low APE (see Fig. 2). While all other classifiers (including SVM) work equally well or better with s-cut, Naive Bayes prefers having a fixed threshold. This is especially true with low APE, but is also apparent for higher APE. Naive Bayes was also the only covered classifier that performed better on unnormalized (bnn and nnn) term weighters.

Another surprising observation is the dependancy of ^{2} on APE. Up to a value of approximately 50 positive training documents, ^{2} is uniformly the top performing feature selector. Then a drastic drop in relative performance occurs, and eventually at APE equal to 100, ^{2} ranks among the worst performers (see Fig. 3). This was observed across all corpora and all machine learning methods. Since we had never encountered this observation in literature before and had no theoretical explanation for it, we suspected this to be a coding bug. This was, however, ruled out by further investigation and we confirm this result.

As expected, machine learning methods are most closely tied to term weighting and APE. With increasing document length, the positive effect of using the augmented term frequency (atc, bnn etc.) instead of plain frequency (nnn) becomes apparent. In general, the results are overwhelmingly in favour of weighters that take into account inverse document frequency and normalization (i.e., *tc). With shorter documents (and therefore most notable on the Routers dataset), ntc performed best. This was observed across all dimensions (including APE, corpora and classifiers with the above mentioned exception of Naive Bayes). On 20Newsgroups, atc tied with ntc and finally on Springer (i.e., long documents), atc outperformed all other weighting schemes by a large margin (see Fig. 4 for term weighter comparison). With longer documents, we therefore suggest using atc instead of the popular tf·idf (=ntc) weighting.

VI. Conclusion and future work

In this article we presented results of our extensive empirical study of the influence of preprocessing parameters on text categorization. In the process, we developed an efficient text categorization suite operated dynamically by XML configuration files. To abstract from conclusions that may be connected only to a particular parameter settings, we ran our experiments over a large number of different parameter combinations. We presented our findings of salient behavorial patterns in this parameter metaspace. In particular, extending the tests from the popular datasets (Reuters and 20Newsgroups) to also cover Springer proved vital – some patterns proved local to the two datasets and did not extend to Springer (for example the ineffectiveness of bigram tokenization).

A potentially very beneficial innovation would be incorporating automated pattern mining from the result database. Searching for interesting patterns by hand is both tedious, error prone and in the high dimensional space difficult to do. Quantifying what constitutes an interesting pattern and letting computers do the dirty work would be immensely helpful.

Acknowledgment

This work is supported in part by MŠMT grant no. LC536 and 2C06009.

References

[1] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *Proceedings of ICML-97, 14th International Conference on Machine Learning*, D. H. Fisher, Ed. Nashville, US: Morgan Kaufmann Publishers, San Francisco, US, 1997, pp. 412–420.

[2] E. Gabrilovich and S. Markovitch, "Text categorization with many redundant features: Using aggressive feature selection to make SVMs competitive with C4.5," in *Proc. 21st Int. Conf. on Machine Learning*, 2004.

[3] J. H. Lee, "Analyses of multiple evidence combination," in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. Combination Techniques, 1997, pp. 267–276.

[4] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[5] R. Krovetz, "Viewing morphology as an inference process," in *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. Linguistic Analysis, 1993, pp. 191–202.

[6] C. D. Paice, "Another stemmer," *SIGIR Forum*, vol. 24, no. 3, pp. 56–61, 1990.

[7] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translation*, vol. 11, pp. 22–31, 1968.

[8] G. Forman, "An extensive empirical study of feature selection metrics for text classification," *Journal of Machine Learning Research*, vol. 3, pp. 1289–1305, 2003.

[9] C. J. V. Rijsbergen, *Information Retrieval*. Butterworths, 1979.

[10] J. W. Wilbur and K. Sirotkin, "The automatic identification of stop words," *Journal of the American Society for Information Science*, vol. 18, pp. 45–55, 1992.

[11] L. Galavotti, F. Sebastiani, and M. Simi, "Experiments on the use of feature selection and negative evidence in automated text categorization," in *ECDL*, ser. Lecture Notes in Computer Science, J. L. Borbinha and T. Baker, Eds., vol. 1923. Springer, 2000, pp. 59–68.

[12] T. Joachims, "Making large–scale SVM learning practical," in *Advances in Kernel Methods — Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, MA: MIT Press, 1999, pp. 169–184.

[13] A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification," in *Proceedings of AAAI-98, Workshop on Learning for Text Categorization*, 1998.

[14] J. L. Wiener, Pedersen, and Weigend., "A neural network approach to topic spotting," *Proc of the Fourth Annual Symp on Document Analysis and Info*, pp. 317–332, 1995.

[15] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1992.

[16] Y. Freund and R. E. Schapire, "Large margin classification using the perceptron algorithm," *MACHLEARN: Machine Learning*, vol. 37, 1999.

[17] E. F. Ian H. Witten, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[18] Y. Yang, "A study on thresholding strategies for text categorization," in *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-01)*, W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, Eds. New York: ACM Press, Sept. 9–13 2001, pp. 137–145.

[19] G. Salton, *The SMART Retrieval System - Experiments in Automatic Document Processing*. Prentice Hall, 1971.

[20] K. Lang, "Newsweeder: Learning to filter netnews," in *ICML*, 1995, pp. 331–339.

[21] J. Žižka and T. Hudík, "Effects of selected basic algorithm parameters and data features on text categorization by support vector machines," in *Proceedings of Znalosti 2005*. VŠB–Technická univerzita Ostrava, 2005, pp. 210–217.