Generic feature extraction for text categorization

(PhD-Sem3)

Gregor Weiss

University of Ljubljana
Faculty of Computer and Information Science
Tržaška cesta 25, Ljubljana, Slovenia
`gregor.weiss@student.uni-lj.si`

1 Introduction

Extracting more informative text features improves the performance of text mining tasks, such as text categorization and sentiment analysis. Nevertheless researchers usually focus on applying machine learning to the main part of their problem and just preprocess texts with hand-engineered linguistic and custom methods, that are based on background knowledge about a given language, domain, and specific task. To improve and automate the initial preprocessing phase we propose a generic feature extraction method inspired by genetic programming (GP). Beginning almost without any natural language processing (NLP) knowledge it will iteratively try to define and connect primitive features and multiple layers of evolved blocks such that more complex and informative features get extracted.

2 Related work review

2.1 Text mining

Increasing amounts of unstructured textual data created a tremendous need for going beyond traditional information retrieval and towards designing algorithms for analyzing text and discovering interesting patterns. This broad and interdisciplinary area related to text analysis is called text mining[1]. Although a lot has already been accomplished, future research directions will need to work on scalable, robust, and domain-independent methods for language processing and context-aware analysis. Improving data representation with our generic feature extraction method might just be the way towards accomplishing these needs.

A traditional framework for text mining[2][3] begins with retrieving and preprocessing an unstructured text corpus (e.g. stemming, lemmatization, stop words removal) that is than transformed into a document representation more suitable for computer's abilities (e.g. bag-of-words vectors, tf-idf weighting, part-of-speech tagging, hand-engineered feature extractors). These preprocessing methods are based on background knowledge about a given language, domain, and specific task. Afterwards the knowledge discovery and text analysis methods are applied to automatically extract previously unknown information and solve the given task. Our research work is positioned in the preprocessing phase targeting feature-vector representations that are traditionally based on a lot of NLP knowledge. We believe that with enough data a GP approach is capable of extracting features as good or even better than traditional methods, but almost without any special NLP knowledge.

Automated text categorization[4][5] usually works by using a generic inductive process (learning algorithm) that builds a classifier by learning the characteristics of individual categories from a set of preclassified documents. The process is based on high dimensional feature-vector representations of text documents that has been shown to work well with Support Vector Machines (SVM)[6]. It is also one of the most researched and basic problems in text mining and can be used in many applications, such as automatic indexing, news filtering, spam detection, document routing, authorship attribution, sentiment analysis, objectivity estimation... As such we decided to use it as our target task for which we will try to improve the performance.

2.2 Feature extraction without special background knowledge

The success of machine learning algorithms generally depends on data representation, and we hypothesize that this is because different features can filter and combine different explanatory factors behind the data. Although features with specific domain knowledge work well enough, having only generic prior knowledge can be used to help design more powerful representations[7]. Because of its complexity representation learning is mainly being explored in the area of deep neural network architectures. Our feature extraction method is similar to their approach in respect of almost not using any NLP knowledge, but it differs in the way how vector representations are learned.

Although text mining without using any special background knowledge seems like a natural goal, only the deep learning community has recently approached it. It has been applied to various NLP tasks with surprising performance[8], but not yet to text categorization task that we plan to focus on. The approaches nearest our text feature extraction deal with representing words in a lower-dimensional distributed vector space that exhibits meaningful word similarity relations[9].

2.3 Genetic programming

Genetic programming (GP)[10][11] is a general technique for inducing a computer program that can produce the desired behavior for a given set of situations, consequently solving problems in a wide range of disciplines. If we can reformulate a problem in terms of program induction (define genetic operators, a heuristic fitness measure, and primitive building blocks) it would eventually evolve good programs, same as biological evolution evolves adapted organisms. The text feature extraction process in its core is just executing a carefully prepared function that transforms a collection of unstructured text documents into a feature-vector representation. This means that informative feature extraction methods for any text mining task could also be evolved without the need of manually incorporating NLP knowledge into them. Our generic feature extraction method is built around this idea that has to the best of our knowledge not yet been tried out.

In GP programs are traditionally represented as tree structures of various sizes that are inappropriate and less efficient. Cartesian genetic programming (CGP)[12] on the other hand represents programs (their genotypes) as a list of integers that are mapped to directed graphs on a fixed number of input, computation, and output nodes. Although some genes may not have an effect and doesn't make sense to compute them, such graph structures are more general than trees and can evolve considerably more efficient than standard GP methods. Our idea of having a set of primitive features that are combined into more complex and informative ones through multiple processing layers, can be naturally expressed in the form of directed graphs. Nevertheless basic CGP may not be sufficient, because the ability to express loops will probably be needed.

One of the key problems of genetic programming or program induction in general is the enormous size of the problem space. A way to accelerate the evaluation of induced programs is to parallelize the work by using multiple CPU or better GPU cores[13]. It turns out that the CGP approach we are planning on using can be represented as a classic genetic algorithm[14] and consequently accelerated on such hardware.

2.4 Fitness measures

The major problem of traditional text categorization is the high dimensionality of the feature space. But if we look at the feature extraction phase from the genetic programming approach, it becomes much worse. The feature space consists of all features that all possible feature extraction programs can produce. Even if we somehow limit the structure of induced programs, it is still enormous. The only way to make it feasible is by choosing a good heuristic fitness function for the GP method[11]. Because we are working on text categorization it would make sense to base our fitness function on a feature selection or scoring measure. Even though general feature selection methods exist, measures constructed with text categorization task in mind can in our opinion perform better.

It has been shown that a feature selection measure called bi-normal separation (BNS) outperforms general ones by a substantial margin in most text categorization problems[15]. It is defined as the difference between normal distribution's inverse cumulative probability functions of true and false positive rates. Its performance and generality make it a good candidate for our fitness function.

A text feature selection approach based on Gini index theory[16] does not contain any specific NLP knowledge and has been shown to be effective with different kinds of classifiers (SVM, kNN, fkNN). Therefore it is also a suitable heuristic for guiding our GP algorithm.

Two more feature scoring metrics have been shown to perform better for text categorization when used with the Naive Bayes learning algorithm[17]. One is a generalization of the odds ratio to multi-class problems, and the other a class discriminating measure. Although the experimental evidence is poor, they might be considered as a candidate for our fitness function.

A general and well-known feature scoring method called ReliefF[18] is based on estimating the quality of features according to how well their values distinguish between instances that are near to each other. Returned scores represent how relevant individual features are, while neglecting the effects of interactions between them (could be a useful property). High dimensionality of the feature space will probably cause problems, therefore modifications like sampling feature subsets will need to be implemented in order to use it.

2.5 Primitive building blocks and features

A typical document representation is the bag-of-words representation[4][1] where we treat individual words as terms and simply count their occurrences. Although it disregards the word order and is sensitive to spelling and grammatical errors, it surprisingly still contains enough information to be widely used in text categorization and similar tasks. Because of its simplicity our feature extraction method should be capable of building upon primitive features in this form.

A simple and commonly used generalization of the above representation is in the form of character- of word-based n-grams[19]. Here occurrence frequencies of all n consecutive characters or words that appear in the corpus are counted. More general representations, such as this, are desirable for our method, so this is one of the major candidates for supported primitive features.

It may seem that just extending the bag-of-words representation to include syntactic and semantic relationships between words (phrases, synonyms, hypernyms) would be a huge improvement[20][21]. Unfortunately without a complex voting technique the performance of learners does not improve in comparison to word-based representation. Nevertheless operators in our algorithm should be capable of expressing such concepts.

Two basic probabilistic vector representations of text documents arise naturally[22]. One is the the multivariate Bernoulli model where a binary vector simply indicates the presence or absence of feature terms. The other is a multinomial model where occurrence frequencies are retained. For expressing such concepts in our method, it must support boolean-, integer-, and real-valued variables.

The major problem of text categorization is the high dimensionality of the feature space. Some linguistic techniques for feature reduction are therefore commonly applied during the preprocessing phase. Stop-word removal is the process of filtering out the words that are generally regarded as not carrying any special meaning, but can cause learning issues due to their high frequency[23]. Stemming is a technique for reducing a word to its base or root form, consequently reducing the dimensionality by mapping related words to the same stem[24][25]. A more complex approach called lemmatization is determining the lemma of a given word where different normalization rules are applied depending on the part-of-speech role of a word[26].

Previously mentioned text representations based on counting occurrences do not reflect how important a word is to a document, because some words appear more frequently in general. Therefore the tf-idf weighting scheme[27], short for term frequency-inverse document frequency, is often used in text mining and by search engines to normalize the effects of words. Because of its intuitiveness our GP approach must have operators and data available to construct such weighting schemes.

Latent semantic indexing[28] tries to identify semantics and associations between terms contained in text documents by using a mathematical technique called singular value decomposition (SVD). It is capable of correlating semantically related terms, thus overcoming the problems of polysemy (one word having many distinct meanings) and synonymy (different words having same meaning). Our method should also be capable of dealing with these situations of natural language, so special functions used as building blocks should also support matrix manipulations, such as SVD.

Another approach for overcoming polysemy and synonymy is the use of multiwords that should capture the contextual information of individual words[29]. It can be accomplished using statistical methods based on mutual information or linguistic methods with grammatical and syntactical rules of phrases. To add this functionality to our method it seems that having a special function to compute entropies and some basic logic operators would suffice.

From the perspective of the sentiment analysis problems[30][31] we distinguish three levels of analysis (document, sentence, and entity or aspect level). We will primarily focus on the document level that is most similar to the traditional text categorization task. Not surprisingly, the most important indicators of sentiments are sentiment words whose influences are aggregated. This concept can be represented by looking up words in dictionaries, so our method should be capable of building and using such data structures. Complications in sentiment analysis arise because words can have opposite meaning in different domains, they can relate to different entities or aspects, opinions can be expressed even without those words, additionally sarcasm and figurative speech are common. These can be represented by our method as logical rules or patterns of opinions and sentiment shifting operators.

One of the text mining fields is called information extraction[32] that deals with extracting structures (e.g. entities, relationships, tags) by using various methods (e.g. manually coded or trained from examples). Although the task differs from the text categorization, we believe that features for rule-based or statistical methods presented in the survey[32] could be used as primitive features that our algorithm will iteratively try to define and combine until more complex and informative ones are induced. For example features can be in the form of looking up in dictionaries, matching patterns of words, measuring similarity and length of segments, applying grammars, ...

3 Conclusion

In our opinion the research goal of preprocessing text almost without any NLP or specific knowledge is an important step towards more robust and domain-independent processing of text. We quickly proposed a generic feature extraction method inspired by genetic programming, presented its major parts, and reviewed what has been done and could possibly be used.

References

1. C. C. Aggarwal and C. X. Zhai, *Mining Text Data*, vol. 4. Boston, MA: Springer US, 2012.
2. V. Gupta and G. S. Lehal, "A survey of text mining techniques and applications," *J. Emerg. Technol. Web Intell.*, vol. 1, no. 1, pp. 60–76, 2009.
3. M. K. Dalal and M. A. Zaveri, "Automatic Text Classification: A Technical Review," *Int. J. Comput. Appl.*, vol. 28, pp. 37–40, Aug. 2011.
4. F. Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Comput. Surv.*, vol. 34, pp. 1–47, Oct. 2002.
5. F. Sebastiani, "Text Categorization," in *Text Min. its Appl.* (A. Zanasi, ed.), pp. 109–129, WIT Press, 2005.
6. T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proc. Eur. Conf. Mach. Learn.* (C. Nédellec and C. Rouveirol, eds.), vol. 1398, pp. 137–142, Springer Berlin Heidelberg, 1998.
7. Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives.," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, pp. 1798–828, Aug. 2013.
8. R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (almost) from Scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.
9. T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *Proc. Int. Conf. Learn. Represent. (ICLR 2013)*, pp. 1–12, 2013.
10. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA, USA: MIT Press, 1992.
11. M. L. Wong and K. S. Leung, "Evolutionary Program Induction Directed by Logic Grammars," *Evol. Comput.*, vol. 5, pp. 143–180, June 1997.
12. J. F. Miller and P. Thomson, "Cartesian Genetic Programming," *Nat. Comput. Ser.*, vol. 43, pp. 17–34, 2011.
13. S. Harding and W. Banzhaf, "Fast Genetic Programming on GPUs," in *Proc. 10th Eur. Conf. Genet. Program.* (M. Ebner, M. O'Neill, A. Ekárt, L. Vanneschi, and A. I. Esparcia-Alcázar, eds.), vol. 4445 of *Lecture Notes in Computer Science*, (Berlin, Heidelberg), pp. 90–101, Springer Berlin Heidelberg, 2007.
14. M. Wineberg and F. Oppacher, "A Representation Scheme to Perform Program Induction in a Canonical Genetic Algorithm," in *Parallel Probl. Solving from Nature—PPSN III*, pp. 291–301, Springer Berlin Heidelberg, 1994.
15. G. Forman, "An Extensive Empirical Study of Feature Selection Metrics for Text Classification," *J. Mach. Learn. Res.*, vol. 3, pp. 1289–1305, 2003.
16. W. Shang, H. Huang, H. Zhu, Y. Lin, Y. Qu, and Z. Wang, "A novel feature selection algorithm for text categorization," *Expert Syst. Appl.*, vol. 33, pp. 1–5, July 2007.
17. J. Chen, H. Huang, S. Tian, and Y. Qu, "Feature selection for text classification with Naive Bayes," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5432–5435, 2009.
18. M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of ReliefF and RReliefF," *Mach. Learn. J.*, vol. 53, no. 1-2, pp. 23–69, 2003.
19. W. B. Cavnar and J. M. Trenkle, "N-Gram-Based Text Categorization," in *Proc. SDAIR-94, 3rd Annu. Symp. Doc. Anal. Inf. Retr.*, pp. 161–175, 1994.
20. D. D. Lewis, "Feature selection and feature extraction for text categorization," in *Proc. Work. Speech Nat. Lang. - HLT '91*, (Morristown, NJ, USA), p. 212, Association for Computational Linguistics, 1992.

21. S. Scott and S. Matwin, “Feature engineering for text classification,” in *Proc. ICML-99, 16th Int. Conf. Mach. Learn.*, vol. 99, pp. 379–388, 1999.
22. S.-B. Kim, K.-S. Han, H.-C. Rim, and S. H. Myaeng, “Some Effective Techniques for Naive Bayes Text Classification,” *IEEE Trans. Knowl. Data Eng.*, vol. 18, pp. 1457–1466, Nov. 2006.
23. W. Zhang, T. Yoshida, and X. Tang, “Text classification based on multi-word with support vector machine,” *Knowledge-Based Syst.*, vol. 21, pp. 879–886, Dec. 2008.
24. M. Porter, “An algorithm for suffix stripping,” *Progr. Electron. Libr. Inf. Syst.*, vol. 14, no. 3, pp. 130–137, 1980.
25. W. B. Frakes, “Stemming Algorithms,” in *Inf. Retr. Boston.* (W. B. Frakes and R. Baeza-Yates, eds.), pp. 131–160, Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.
26. J. Plisson, N. Lavrac, and D. Mladenic, “A Rule based Approach to Word Lemmatization,” in *Proc. 7th Int. multi-conference Inf. Soc.*, (Ljubljana), pp. 83–86, Jožef Stefan Institute, 2004.
27. A. Aizawa, “An information-theoretic perspective of tf-idf measures,” *Inf. Process. Manag.*, vol. 39, pp. 45–65, 2003.
28. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” *J. Am. Soc. Inf. Sci.*, vol. 41, pp. 391–407, Sept. 1990.
29. W. Zhang, T. Yoshida, and X. Tang, “A comparative study of TF-IDF, LSI and multi-words for text classification,” *Expert Syst. Appl.*, vol. 38, pp. 2758–2765, 2011.
30. B. Liu, *Sentiment Analysis and Opinion Mining*, vol. 5. Morgan & Claypool Publishers, May 2012.
31. G. Vinodhini and R. M. Chandrasekaran, “Sentiment Analysis and Opinion Mining: A Survey,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 2, no. 6, pp. 282–292, 2012.
32. S. Sarawagi, “Information Extraction,” *Found. Trends Databases*, vol. 1, no. 3, pp. 261–377, 2008.