String Kernels in Sentiment Analysis

Paula Gombar

Abstract—**String kernels are a simple feature-light alternative to bag-of-word-models and worm embeddings when it comes to sentiment analysis. We experiment with different kernels and features on the task of supervised sentiment classification of tweets. We investigate whether word embeddings offer any advantage over corpus- and preprocessing-free string kernels, and how these compare to bag-of-words baselines. Results show that both word embeddings and bag-of-words baselines outperform string kernels on our informal-written, noisy dataset. The source code is publicly available on Github.**[1]

I. Introduction

Sentiment analysis (Pang and Lee, 2008) deals with the task of predicting whether the text expresses a positive, negative, or neutral opinion in general or with respect to an entity. In recent times, sentiment analysis has found popular applications in identifying political popularity, predicting stock prices, analyzing social media, etc. However, analyzing sentiment analysis in social media can be challenging since such texts are often short, informal, and noisy (Baldwin et al., 2013).

As it is the case with supervised classification tasks, common approaches amount to rich, domain-specific features, including lexicon-based and syntactic features. On the other hand, there has been a growing trend in using feature-light methods. In particular, two simple methods as an alternative to syntax-based methods are word embeddings (Mikolov et al., 2013) and string kernels (Lodhi et al., 2002). The advantage of both methods is that they are easy to set up. However, word embeddings require a large, possibly lemmatized corpus, whereas string kernels require no preprocessing at all.

In this paper we focus on sentiment classification of short text in English. We compare string kernels to more conventional methods, word embeddings and bag-of-word-models. Our goal is to research whether they offer any advantage over corpus- and preprocessing-free string kernels. We experiment on a labeled dataset consisting of positive, negative, and neutral tweets.

II. Dataset

The experiments were performed on a publicly available tweet dataset constructed for Task 4 in Semeval 2017: Sentiment Analysis in Twitter.[2] The goal of the task is to classify message polarity. More precisely, given a message, we need to classify whether it is of positive, negative, or neutral sentiment. All tweets are limited to 140 characters, which makes these short texts suitable for string kernels.

The dataset is provided in two parts, the training set and the test set. Some statistics are given in Table I. We can see that the class distribution is approximately the same in both sets. Moreover, somewhat unusually, there are more than twice as much positive instances than negative ones.

	Train		Test	
Positive tweets	3640	(37.6%)	1475	(41.6%)
Negative tweets	1458	(15%)	559	(15.7%)
Neutral tweets	4586	(47.4%)	1513	(42.7%)
Total	9684		3547	

TABLE I
Dataset statistics.

III. Models

We based all our experiments on the Support Vector Machine (SVM) classification algorithm (Cortes and Vapnik, 1995). This is because SVM offers the advantage of using various kernel functions, including string kernels. We used the LIBSVM implementation (Chang and Lin, 2011) when using string kernels, and the implementation from scikit-learn (Pedregosa et al., 2011) when using other kernels and features.

A. Bag-of-words Baselines

We evaluated two bag-of-words baselines. We used CountVectorizer from scikit-learn to construct features. In this approach, we convert all characters to lowercase before tokenizing and creating a matrix of token counts. We perform stop-words removal.

The first baseline uses word unigrams, bigrams and trigrams as features. The total number of n-grams is cut-off at 10000, and n-grams appearing less than three times in the dataset are discarded.

The second baseline uses character bigrams, trigrams, four-grams and five-grams. It has been shown that character n-grams are useful for text classification of noisy texts (Cavnar et al., 1994). The total number of n-grams is cut-off at 10000, and n-grams appearing less than three times in the dataset are discarded.

We used a linear kernel for both of these models.

B. Word Embeddings

Word embeddings (Mikolov et al., 2013) are a predictive distributional semantic model which derive dense vector representations of word meanings from corpus co-occurrences. We used pre-trained vectors trained on 100 billion words from a Google News dataset. These include 300-dimensional word vectors of 3 million words and phrases.

We used two kernels with these features: the linear kernel, and the radial basis function (RBF) kernel. Using a linear kernel results in a linear model, whereas the RBF

[1] `https://github.com/pgombar/cog-modeling-project`
[2] `http://alt.qcri.org/semeval2017/task4/`

kernel yields a high-dimensional non-linear model. The RBF kernel is expressed as the following:

$$\kappa(\mathbf{x}, \mathbf{x}') = \exp -\gamma \|\mathbf{x} - \mathbf{x}'\|^2$$

where $\|\mathbf{x} - \mathbf{x}'\|^2$ is recognized as the squared Euclidean distance between two feature vectors. For simplicity, we introduce $\gamma = \frac{1}{2\sigma^2}$.

C. String Kernels

String kernels are functions that measure the similarity of pairs of strings: the more similar two strings a and b are, the higher the value of a string kernel $\kappa(a, b)$ will be. The kernel function effectively maps the instances to a high-dimensional feature space, which eliminates the need for translating strings to fixed-length, real-valued feature vectors. We experimented with three different kernels: the subsequence kernel (SSK) (Lodhi et al., 2002), spectrum kernel (SK) (Leslie et al., 2002), and the distance substitution kernel (DSK) (Haasdonk and Bahlmann, 2004).

The subsequence kernel (SSK) maps strings to a feature vector indexed by all k tuples of characters. A k-tuple will have a non-zero entry if it occurs as a subsequence anywhere (not necessarily contiguously) in the string. The feature weight is the sum over the occurrences of the k-tuple of a decaying factor λ of the length of the occurrence. Formally, the mapping function of string s is given as

$$\phi_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}$$

where u is a subsequence searched for in s, $\mathbf{i}$ is a vector of indices at which u appears in s, l is a function measuring the length of a matched subsequence and $\lambda \leq 1$ is the weighting parameter. The corresponding kernel is defined as:

$$K_n(s, t) = \sum_{u \in \Sigma^n} \langle \varphi_u(s), \varphi_u(t) \rangle$$

where n is maximum subsequence length for which we are calculating the kernel and Σ^n is a set of all finite strings of length n.

The spectrum kernel (SK) can be viewed as a special case of SSK where vector of indices $\mathbf{i}$ must yield contiguous subsequences and λ is set to 1.

The idea behind the distance substitution kernel (DSK) is to substitute the commonly-used Euclidean distance by a problem-specific distance measure. We used the Levenshtein distance (Navarro, 2001), expressed as the number of deletions, insertions, or substitutions required to transform string s into string t.

IV. Experiments

We evaluated all models using k-folded evaluation with hyper-parameter grid search (C and γ for RBF, C for linear kernel. We used 5 folds. The models were evaluated using the average of the F1-scores for each class.

Table II shows the F1-scores for the baseline, word embeddings, and string kernel models, using different feature sets and kernel configurations. The best-performing model is word embeddings in combination with RBF kernel. This is not surprising, as word embeddings are known to work well in sentiment analysis, but also because RBF kernel can capture non-linearity in data.

Model/Features	Kernel	Score
BoW baseline		
Word n-grams	Linear	61.1
Character n-grams	Linear	60.2
Word embeddings		
Words	Linear	62.8
Words	RBF	**64.0**
String kernels		
–	SK	38.3
–	SSK	38.2
–	DSK	26.0

TABLE II
F1-scores of different models.

Word n-grams are slightly better than character n-grams, but neither perform as well as word embeddings. Unfortunately, string kernels perform much worse than any other method. Even the best-performing kernel, SK, does not come close to n-grams or word embeddings.

V. Conclusion

We investigated the usefulness of string kernels on the task of Twitter sentiment analysis and compared it to two other methods: bag-of-words and word embeddings. We trained a number of SVM models using different kernels. We find that both word embeddings and bag-of-words-models significantly outperform string kernels. We believe string kernels would perform better with domain-specific short texts, because words would appear in the same context, bearing the same meaning and sentiment. Moreover, the noisiness of the text could have impacted this method more than other methods.

Thus, word embeddings are a method of choice for Twitter sentiment analysis. In cases when it is too time-consuming and computationally not feasible to use word embeddings, word n-grams is the preferred method over character n-grams and string kernels. However, string kernels still remain an option for languages that do not have satisfactory preprocessing tools in place.

References

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. How noisy social media text, how different social media sources? In *Proceedings of the 6th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 356–364, Nagoya, Japan, 2013.

William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. *Ann arbor mi*, 48113(2):161–175, 1994.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Bernard Haasdonk and Claus Bahlmann. Learning with distance substitution kernels. In *Joint Pattern Recognition Symposium*, pages 220–227. Springer, 2004.

Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, volume 7, pages 566–575, 2002.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2 (Feb):419–444, 2002.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Neural Information Processing Systems Conference (NIPS 2013)*, pages 3111–3119, Lake Tahoe, USA, 2013.

Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.