Position-Aware and Dependency-Aware Text Classifier

Anonymous ACL submission

Abstract

This paper presents a new text classifier that uses local biases and dependency relations. Our approach is based on finding a small set of substring patterns which classify the given documents well and also take into account the local and semantic biases in the composition of the string. We formulate the problem as a 1 norm soft margin optimization problem and solve this problem by using the combination of LPBoost and an optimal position-aware substring discovery algorithm. Since the dependency relations give rich linguistic and syntactic information, we parse the sentences by Stanford Dependency parser and replace each word by the word's dependency triplet in the Universal Dependencies representations that contains the name of the relation, governor and dependent as the pre-processing, and we evaluated our approach with the datasets that contain dependency information. Computational experiments on various datasets show that our approaches can, in many cases, significantly improve the classification performance as compared to the state-of-the-art.

1 Introduction

Text classification is an important problem in broad areas such as natural language processing, bioinformatics, information retrieval, and recommendation tasks. Machine Learning has been applied to text classification tasks in various ways: SVMs and string kernels (Ngram kernels, subsequence kernels (Lodhi et al., 2002), Local Alignment kernels (Saigo et al., 2004), an Mismatch kernels (Leslie et al., 2002)), and Boosting (e.g., Boostexter (Schapire and Singer, 2000)).

In some text classification applications, not only classification accuracy but also what makes classification accurate is important. There are several approaches, but we follow the work of Kashihara et al. (Kashihara et al., 2010) that finds a small set of patterns which induces an accurate classifier.

In this paper, we apply the Segment Structures to the pattern discovery algorithm. we consider that the pattern class is all of the possible substrings over some alphabet Σ. This time, we focus on substring patterns but our approach can be further extended for rich classes such as subsequence patterns (Hirao et al., 2003) or VLDC patterns (Inenaga et al., 2002), which is our future work (See Shinohara's survey (Shinohara, 2004) for pattern discovery algorithms).

There are several related research works.The work of Okanohara and Tsujii (Okanohara and Tsujii, 2009) and the work of Saigo et al (Saigo et al., 2009) would be most related to ours. However, both of them did not treat the position-aware patterns. Okanohara and Tsujii consider a similar problem over substring patterns and they solve logistic regression with 1 norm regularization. Saigo et al deal with 1 norm soft margin optimization over graph patterns and they also use LPBoost. Our framework is close to theirs, but we use different techniques for pattern discovery of position-aware substrings. In addition, we parse the given sentences by Stanford Dependency parser (Schuster and Manning, 2016) and Knowledge Parser (Sharma et al., 2015a,b) to add information of how two words are syntactically and semantically linked in a sentence.

2 Preliminaries

We use 1 norm soft margin optimization problem to get sparse result. This problem is to find a linear combination of classifiers associated with patterns (find a hyperplane whose each component corresponds to a pattern) which maximizes the margin w.r.t. the given labeled texts as well as minimizing misclassification. We follow the work

of Kashihara et al. (Kashihara et al., 2010), and we combine LPBoost (Demiriz et al., 2002) and our pattern discovery algorithm for solving the 1 norm soft margin optimization. This approach has two advantages, the first advantage is the increased accuracy of the resulting classifier and the other advantage is that the resulting solution is usually sparse since the 1 norm soft margin optimization is a linear program.

There has been researched parsing text by using a syntactic/semantic parser (Manning et al., 2014) and learning the vector representation (embedding) of the syntactic/semantic relationships between any two connected words in a sentence. These relation embeddings have shown to capture the rich linguistic and syntactic information (Mikolov et al., 2013). The effectiveness of using both word and relation embedding together have also been observed in tasks like knowledge extraction (Bordes et al., 2011) and aspect term extraction (Yin et al., 2016). In our work, we simply parse the sentences by a syntactic parser, Stanford Dependency parser (Schuster and Manning, 2016) and replace each word by its dependency triplet in the Universal Dependencies representations (Nivre et al., 2016).A dependency triplet consists of the governor word, the relation and the dependent word in the order mentioned. We also parse the sentences by a semantic parser, Knowledge Parser (Sharma et al., 2015a,b) that gives not only word's dependency triplet (with semantic relations instead of syntactic) but also provides each word's semantic role and conceptual class.

The work of Okanohara and Tujii (Okanohara and Tsujii, 2009) considers a similar problem over substring pattern by solving logistic regression with 1 norm regularization. And, the work of Saigo et al (Saigo et al., 2009) treats 1 norm soft margin optimization over graph patterns and also uses LPBoost, but our framework uses different techniques for pattern discovery of position-aware substrings. Both of the related researches do not consider any position-aware patterns in their works.

3 Algorithms

3.1 Segment

For a string T, we consider two kinds of segments, *Non-overlapping Segments* and *Overlapping Segments*.

Definition 1. *For a given string T and the number of layers q, the Non-overlapping Segment Structure is the set of strings $\{T^{Non-over}_{(i,j)} \mid 0 \leq i < \min\{q, \lceil log|T| \rceil\}, 1 \leq j \leq 2^i\}$ that is defined recursively as follows;*

- $T^{Non-over}_{(0,1)} = T$.
- *Let* $l = |T^{Non-over}_{(i-1,j)}|$.
  For $0 \leq i < \min\{q, \lceil \log |T| \rceil\}, 1 \leq j \leq 2^{i-1}$,
  $T^{Non-over}_{(i,2j-1)} = T^{Non-over}_{(i-1,j)}[0, \lfloor \frac{l}{2} \rfloor - 1]$,
  $T^{Non-over}_{(i,2j)} = T^{Non-over}_{(i-1,j)}[\lfloor \frac{l}{2} \rfloor, l-1]$.

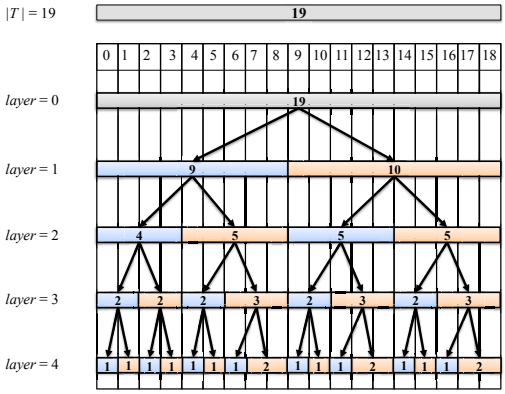

Figure 1: An example of Non-overlapping Segment for a string $T(|T| = 19)$ and the number of layers $q = 4$

Figure 1 shows an example of Non-overlapping Segments for a string $T(|T| = 19)$ and the number of layers $q = 4$. In this instance, there exists segments with length 1 in $layer = 4$, which is therefore the deepest possible layer.

Overlapping Segment Structure of the given string T and the number of layers q is the structural set of overlapping strings that is defined as below.

Definition 2. *For a given string T and the number of layers q, the Overlapping Segment Structure is the set of strings $\{T^{Over}_{(i,j)} \mid 0 \leq i < \min\{q, \lceil log|T| \rceil\}, 1 \leq j \leq 2^{i+1} - 1\}$ that is defined recursively as follows;*

- $T^{Over}_{(0,1)} = T$.
- $T^{Over}_{(i,1)} = T^{Over}_{(i-1,1)}[\lfloor \frac{|T^{Over}_{(i-1,1)}|}{2} \rfloor, |T^{Over}_{(i-1,1)}| - 1]$
- *Let* $L = |T^{Over}_{(i-1,j)}|$.
  For $0 \leq i < \min\{q, \lceil log|T| \rceil\}, 1 \leq j \leq 2^{i+1} - 1$,

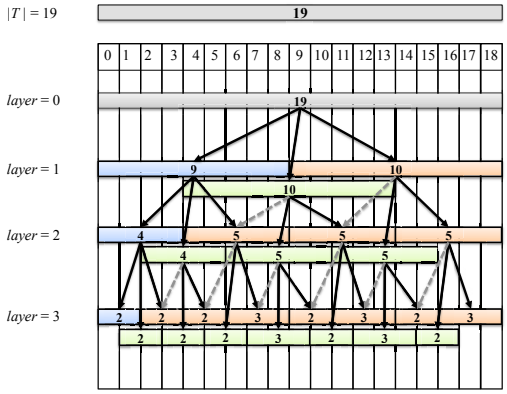

Figure 2: An example of Overlapping Segment for a string $T(|T| = 19)$ and the number of layers $q = 3$

If j is odd, then

$$T^{Over}_{(i,2j+1)} = T^{Over}_{(i-1,j)}[\lfloor \tfrac{|L|}{2} \rfloor, |L|],$$
$$T^{Over}_{(i,2j)} = T^{Over}_{(i-1,j)}[\lfloor \tfrac{\lfloor \frac{L}{2} \rfloor}{2} \rfloor, L - \lceil \tfrac{\lceil \frac{L}{2} \rceil}{2} \rceil - 1].$$

If j is even, then

$$T^{Over}_{(i,2j+1)} = T^{Over}_{(i-1,j)}[\lfloor \tfrac{L}{2} \rfloor + 1, L],$$
$$T^{Over}_{(i,2j)} = T^{Over}_{(i-1,j)}[\lfloor \tfrac{\lfloor \frac{L}{2} \rfloor}{2} \rfloor + 1, L - \lceil \tfrac{\lceil \frac{L}{2} \rceil}{2} \rceil - 1].$$

Figure 2 shows an example of Overlapping Segments for a string T and the number of layers $q = 3$.

3.2 Position-aware Sparse Substring Pattern Set Discovery

We deem the 1 norm soft margin optimization problem for position-aware string datasets, where each hypothesis corresponds to a position-aware string pattern. In other words, for a given set of labeled documents (structured by Segment), each position-aware string pattern (substring) $p_{i,j} \in \Sigma^*$ (ith layer, jth segment) corresponds to a hypothesis $h_{p_{i,j}} \in \mathcal{H}$, and $h_{p_{i,j}}(x_{i,j})$ for $x_{i,j} \in \Sigma^*$, where $x_{i,j}$ is ith layer and jth segment substring of x ($x_{i,j} \subseteq x$), is defined as follows:

$$\boldsymbol{h_{p_{i,j}}(x_{i,j})} = \begin{cases} \mathbf{1} & \boldsymbol{p_{i,j}} \text{ is a substring of } \boldsymbol{x_{i,j}} \\ \mathbf{-1} & \text{otherwise} \end{cases}.$$

So, our "weak" learner will solve the following problem. Given a set of labeled strings $S = ((x_1, y_m), ..., (x_m, y_m)) \subset \Sigma^* \times \{-1, +1\}$, the number of layer $L \leq \lceil \log |\min x| \rceil$ where $\min x = \min\{|x_1|, \ldots, |x_m|$, K_j is the jth layer's total number of segments ($k \in K_j$), and a distribution $\boldsymbol{d} \in \mathcal{P}^m$ over S, find a string $p_{j,k} \in \Sigma^*$

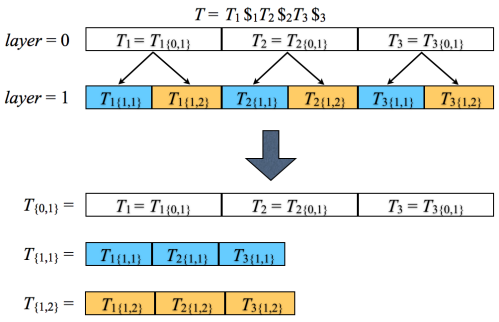

Figure 3: Example of the case that the number of layer is 2. For the given strings, $T = T_1\$_1T_2\$_2T_3\$_3$, our system creates the segment structures of $layer = 0$ and $mathitlayer = 1$. Then, it creates each Suffix Array (SA) and the height array LCP for finding optimal position-aware substring pattern (get maximum value of optimal substring pattern in each $T_{j,k}$).

such that

$$p_{j,k} = \arg \max_{q_{j,k} \in \Sigma^*} \mathbf{Edge_d}(q_{j,k}) \quad (1)$$
$$= \sum_{i=1}^{m} y_i d_i h_{q_{j,k}}(x_{i_{j,k}}) \quad (2)$$

, where $x_{i_{j,k}}$ is the segment string of ith string's kth segment of jth layer.

To solve this problem optimally and efficiently, we make use of the suffix array data structure (Manber and Myers, 1993) as well as other related data structures described in (Kashihara et al., 2010). In this paper, we apply the suffix array data structure for each segment structures. Figure 3 shows an example of how the given documents $T = T_1\$_1T_2\$_2T_3\$_3$ segmented the 2 layers non-overlap segment structure. We construct the suffix array data structure for each segment structures $T_{\{0,1\}}, T_{\{1,1\}}, T_{\{1,2\}}$.

We briefly describe how we can find the substring $p_{j,k} \in \Sigma^*$ to maximize Equation (2) in linear time. First, it is sufficient to consider strings which correspond to nodes in the generalized suffix tree of the input strings. This is because for any string corresponding to a path that ends in the middle of an edge of the suffix tree, the string which corresponds to the path extended to the next node will occur in the same set of documents and, hence, its edge score would be the same.

Also, notice that for any substring $p_{j,k} \in \Sigma^*$,

$$\mathbf{Edge}_{\boldsymbol{d}}(p_{j,k}) = \sum_{i=1}^{m} y_i d_i h_{p_{j,k}}(x_{i_{j,k}})$$

$$= \sum_{\{i:h_{p_{j,k}}(x_{i_{j,k}})=1\}} y_i d_i - \sum_{\{i:h_{p_{j,k}}(x_{i_{j,k}})=-1\}} y_i d_i$$

$$= 2 \cdot \sum_{\{i:h_{p_{j,k}}(x_{i_{j,k}})=1\}} y_i d_i - \sum_{i=1}^{m} y_i d_i.$$

Since $\sum_{i=1}^{m} y_i d_i$ can be easily computed, it is just necessary to compute $\sum_{\{i:h_{p_{j,k}}(x_{i_{j,k}})=1\}} y_i d_i$ for each $p_{j,k}$ to compute its edge score.

According to (Kashihara et al., 2010), this value can be computed for each string that corresponds to each node in the generalized suffix tree, basically using the linear time algorithm for solving a generalized version of the color set size problem (Hui, 1992; Bannai et al., 2004). When each document is assigned arbitrary numeric weights, the algorithm computes, the sum of weights (for each node of the generalized suffix tree) of the documents that contain the path of each node as a substring. For our problem, we need only to assign the weight $y_i d_i$ to each document.

4 Experiment

We conducted classification experiments for MOVIE-A and MOVIE-B datasets. MOVIE-A is a dataset by Pang and Lee (Pang and Lee, 2004)[1]. The data consists of reviews of various movies with 1000 positive reviews and 1000 negative reviews from the IMDB database. The positive reviews are judged over half of score by reviewers. For instance, if a movie's score is 7 of 10 scale score by a reviewer, then the review is positive review. The others are negative reviews. MOVIE-B is a dataset by Ifrim et al. (Ifrim et al., 2008) [2] which consisting of reviews taken from the IMDB database, for movies classified as 'Crime' or 'Drama'. There are 3720 reviews for each genre.

We evaluate our approaches with MOVIE-A and MOVIE-B as 10-fold cross validation. We preprocess the sentences in MOVIE-A and MOVIE-B for the dependency-aware approaches with Stanford Dependency parser (Dep) and K-parser (KP).

[1] http://www.cs.cornell.edu/People/pabo/movie-review-data/, polarity dataset v2.0

[2] http://www.mpi-inf.de/~ifrim/data/kdd-datasets.zip, KDD08-datasets/IMDB

Table 1: Percentage of correct classifications in classification task.

Corpus	Movie-A	Movie-B
LPSSD	91.25%	78.5%
OT	86.5%	75.1%
NOS-SSD	92.35%	92.34%
OS-SSD	**96.63%**	95.02%
LPSSD-Dep	69.83%	91.27%
NOS-SSD-Dep	58.78%	88.91%
OS-SSD-Dep	73.20%	84.4%
LPSSD-KP	51.85%	**96.62%**
NOS-SSD-KP	68.28%	61.9%
OS-SSD-KP	77.98%	77.47%

Table 1 shows the result of our method, Non-Overlap Segment position-aware substring patterns with LPBoost (NOS-SSD) and Overlap Segment position aware patterns with LPBoost (OS-SSD), as well as several other methods such as the work of Kashihara et al. (Kashihara et al., 2010) (LPSSD) and the method of Okanohara and Tujii (Okanohara and Tsujii, 2009). The range of the layer is from 1 to 5, and the most of the cases, our methods achieved higher accuracy than the other methods. However, the original sentences have lots of typo and grammatical mistakes, and they might effect the result of our approach with Dep and KP. The score for "OT (Okanohara and Tsujii, 2009)" is the score of a 10-fold cross validation taken directly from their paper.

5 Conclusions

We considered 1 norm soft margin optimization over position-aware and dependency-aware substring patterns. We solve this problem by using a combination of LPBoost and an optimal position-aware substring pattern discovery algorithm. The experiments show that our method achieves higher accuracy than other methods in most of cases. In addition, the number of patterns are very sparse, containing interesting ones and they are visible for analyze. There are mainly two points for the future work. First, extending the pattern class to more richer ones, for example, VLDC patterns (Inenaga et al., 2002) would be interesting. Second, employing other solvers for 1 norm soft margin optimization. For instance, Sparse LPBoost (Hatano and Takimoto, 2009) may faster solver than LPBoost and Entropy Regularized LPBoost (Warmuth et al., 2008) may help to get more higher accuracy.

References

Hideo Bannai, Heikki Hyyrö, Ayumi Shinohara, Masayuki Takeda, Kenta Nakai, and Satoru Miyano. 2004. An $O(N^2)$ algorithm for discovering optimal Boolean pattern pairs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 1(4):159–170.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'11, pages 301–306. http://dl.acm.org/citation.cfm?id=2900423.2900470.

Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. 2002. Linear programming boosting via column generation. *Mach. Learn.* 46(1-3):225–254. https://doi.org/http://dx.doi.org/10.1023/A:1012470815092.

Kohei Hatano and Eiji Takimoto. 2009. Linear programming boosting by column and row generation. In *Proceedings of the 12th International Conference on Dicovery Science (DS 2009)*. pages 401–408.

Masahiro Hirao, Hiromasa Hoshino, Ayumi Shinohara, Masayuki Takeda, and Setsuo Arikawa. 2003. A practical algorithm to find the best subsequence patterns. *Theoretical Computer Science* 292(2):465–479.

L. Hui. 1992. Color set size problem with applications to string matching. In *Proceedings of the Third Annual Symposium on Combinatorial Pattern Matching (CPM 92)*. Springer-Verlag, volume 644 of *LNCS*, pages 230–243.

Georgiana Ifrim, Gökhan H. Bakir, and Gerhard Weikum. 2008. Fast logistic regression for text categorization with variable-length n-grams. In *KDD*. pages 354–362.

Shunsuke Inenaga, Hideo Bannai, Ayumi Shinohara, Masayuki Takeda, and Setsuo Arikawa. 2002. S.: Discovering best variable-length-don't-care patterns. In *In: Proceedings of the 5th International Conference on Discovery Science. Volume 2534 of LNAI., Springer-Verlag*. Springer-Verlag, pages 86–97.

Kazuaki Kashihara, Kohei Hatano, Hideo Bannai, and Masayuki Takeda. 2010. Sparse substring pattern set discovery using linear programming boosting. In *Discovery Science*. pages 132–143.

Christina S. Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. 2002. Mismatch string kernels for svm protein classification. In *Advances in Neural Information Processing Systems 15 (NIPS '02)*. pages 1417–1424.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research* 2:419–444.

Udi Manber and Gene Myers. 1993. Suffix arrays: a new method for on-line string searches. *SIAM J. Computing* 22(5):935–948.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. Curran Associates Inc., USA, NIPS'13, pages 3111–3119.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA).

Daisuke Okanohara and Jun'ichi Tsujii. 2009. Text categorization with all substring features. In *Proc. 9th SIAM International Conference on Data Mining (SDM)*. pages 838–846.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*.

Hiroto Saigo, Sebastian Nowozin, Tadashi Kadowaki, Taku Kudo, and Koji Tsuda. 2009. gboost: a mathematical programming approach to graph classification and regression. *Machine Learning* 75(1):69–89.

Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. 2004. Protein homology detection using string alignment kernels. *Bioinformatics* 20(11):1682–1689.

Robert E. Schapire and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning* 39:135–168.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016.*.

Arpit Sharma, Nguyen Ha Vo, Somak Aditya, and Chitta Baral. 2015a. Identifying various kinds of event mentions in k-parser output. In *Proceedings of the The 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation, EVENTS@HLP-NAACL 2015, Denver, Colorado, USA, June 4, 2015*. pages 82–88.

Arpit Sharma, Nguyen Ha Vo, Somak Aditya, and Chitta Baral. 2015b. Towards addressing the winograd schema challenge - building and using a semantic parser and a knowledge hunting module. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. pages 1319–1325.

Ayumi Shinohara. 2004. String pattern discovery. In *Proceedings of the 15th International Conference on Algorithmic Learning Theory (ALT'04)*. pages 1–13.

Manfred K. Warmuth, Karen A. Glocer, and S. V. N. Vishwanathan. 2008. Entropy regularized lpboost. In *ALT '08: Proceedings of the 19th international conference on Algorithmic Learning Theory*. Springer-Verlag, Berlin, Heidelberg, pages 256–271. https://doi.org/http://dx.doi.org/10.1007/978-3-540-87987-9_23.

Yichun Yin, Furu Wei, Li Dong, Kaimeng Xu, Ming Zhang, and Ming Zhou. 2016. Unsupervised word and dependency path embeddings for aspect term extraction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*. AAAI Press, IJCAI'16, pages 2979–2985. http://dl.acm.org/citation.cfm?id=3060832.3061038.