The Use of Global Topic Information and Local Semantic Relationship in Sentiment Analysis

Linhong Li **Shiqi Xiao** **Xiangting Chen** **Jiachen Ning**
Carnegie Mellon University
Pittsburgh, PA 15213

1 Introduction

Sentiment Analysis is the task of classifying text documents by their *sentiment*, or the overall opinion towards the subject matter (17). In practice, sentiment analysis is useful for companies and policymakers to monitor large scale public opinions on specific products and/or policies, and thus extended applications of sentiment analysis can be found in security, medical, finance, and entertainment domains (12).

A survey of state-of-the-art sentiment analysis methods revealed the importance of capturing features that go beyond n-gram tokens. With that in mind, we propose a sentiment classifier that not only "reads" the text content, but also "mines" the text for grammatical structures within the sentences (local semantic relationships) and unobserved topic distributions spanning the corpus (global topic information).

To evaluate how informative these two additional sources of information are, we performed experiments on one of the most analyzed datasets for sentiment analysis – the IMDb Dataset of movie reviews. The dataset contains a collection of 50,000 reviews posted on the Internet Movie Database. Standard preprocessing ensured that the dataset is balanced, containing the same number of positive and negative reviews, where positive reviews had a score greater than or equal to 7 and negative reviews had a score less than or equal to 4 (11). Our primary task is to output a binary classification (i.e., positive or negative) of an audience's attitude towards a movie based on his or her online review. To improve classification accuracy, we encoded rich local semantic relationships during vectorization of the texts, and made use of global topic information by building an ensemble of specialized topic-specific classifiers. Our proposed method achieved 94.3% test accuracy [1] on the dataset, which is a nearly 10% improvement of our implemented baseline. Further analyses showed that most of the performance improvement came from explicitly adding local semantic relationships into the feature vectors.

2 Background

In our midway report, we identified two common components among successful sentiment analysis: (1) representative text embeddings, and (2) a robust classification scheme.

The first component usually features a n-gram tokenization combined with Bag-of-Word (BoW), term frequency (TF), or term frequency-inverse document frequency (TF-IDF) score vectorization, while a wide range of classification methods have been explored for the second component. Along these two trajectories, we established the baseline by implementing different combinations of vectorization techniques (BoW, TF, and TF-IDF) and two of the most popular classifiers for sentiment analysis – Naïve Bayes (NB) and Support Vector Machine (SVM). The best-performing combination in our baseline experiments was a uni-, bi- and trigram tokenization, the TF-IDF vectorization, and classifications done by a linear SVM, which achieved a test accuracy of 88.5%.

[1] As a comparison, the state-of-the-art performance on this dataset is 95.4% by (6) using an LSTM-based language model.

Moving on from the midway report, our investigation had evolved to finding better alternatives for these two components individually, specifically:

- **How to better encapsulate the text in a vector?** Tokenization and vectorization techniques featured in our baseline studies have limited capacity to capture longer-distance relationships between words. But, as the majority of the state-of-the-art approaches use recurrent structures such as LSTM, which are commonly used to capture long-range dependencies in spatial or time series, we inferred that incorporating long-range **lexical dependencies** might be key to a better feature representation (6; 5; 21).

- **Within a representative feature space, what type of model better untangles the two classes?** Success of neural networks on this specific task indicates that models with reduced complexity, such as SVM or NB, have insufficient capacity for sentiment analysis. While neural networks are able to learn arbitrarily complex decision boundaries in a high dimensional space, the SVM and NB classifiers used in our baseline are known to be linear classifiers. Motivated by the observation that boosting algorithms like adaboost could generate nonlinear decision boundaries using only decision stumps, we looked into ensemble methods to allow more complex decision boundaries by SVM and NB. In particular, since it is unrealistic to accurately classify the entire corpus using a single linear hyperplane, we hoped to find a way to partition the corpus into subsets with simpler decision boundaries, which motivated our experimentation with **topic modeling** to build an ensemble of topic-specific classifiers.

3 Related Work

3.1 Sentiment Analysis

Extensive work has been done on the sentiment analysis task. The standard steps include *tokenization*, *vectorization*, and *training*.

In term of tokenization, it's a standard practice to parse the text into sequences of *n* words, called *n-gram* representations. Wang and Manning (2012) showed that the inclusion of word n-gram features gives consistent gains on sentiment analysis tasks, with bigram and trigram representations being the most widely used and showing superior results than unigram tokens (22). An additional step in text preprocessing after tokenization is *lemmatization*, which refers to the process of replacing a given token with its corresponding lemma. The main idea behind it is to reduce sparsity of the document-to-feature matrix (1).

Then, to vectorize a tokenized text, common methods include counting the token occurrences (i.e., the Bag-of-Word (BoW) model) and assigning a term frequency (TF) or term frequency-inverse document frequency (TF-IDF) score to each token in the corpus dictionary, both of which reflect how important a token is to a document. Specifically, a token's term frequency measures how frequently the token appears in a document, which can be calculated as:

$$\text{TF(t)} = \frac{\text{number of times term } t \text{ appears in the document}}{\text{total number of terms in the document}}$$

The inverse document frequency is calculated as:

$$\text{IDF}(t) = \ln\left(\frac{\text{total number of documents}}{\text{number of documents with term } t \text{ in it}}\right)$$

The IDF score explicitly encodes how important a token is to a document. Intuitively, a token is more representative of a document if it only appears in a few documents in the entire corpus, which will increase the IDF score.

Lastly, in the training phase, several common choices of classifiers presented in previous works include Naïve Bayes (NB), Support Vector Machine(SVM), and (Deep) Neural Networks (NN) (6; 22) .For the IMDB dataset, the current best classifier is ULMFiT, which is a LSTM-based model proposed by Howard and Ruder in 2018 (6); SVM with NB features (NBSVM) proposed by Wang and Mannings showed superior performance among non-NN classifiers (22).

3.2 Lexical Dependency

Although the most commonly used text tokenization method is n-grams, Nastase et al. (2006) and Xia et al. (2011) argued it does not fully capture the gist of a document and thus pairs of grammatically related words (i.e., dependency pairs) should also be incorporated as features as they might capture important long-distance information(16; 23). These relations can be identified via a dependency parser based on dependency grammar (20).

Most related works are done in the product-review domain. Mukherjee and Bhattacharyya (2012) used dependency parsing to identify relations between the opinion expressions in product reviews which allowed them to merge closely associated opinion expressions (15). Qiu et al. (2011) extracted product features and sentiment words based on dependency grammar analyses (20). Both of them sought to exploit grammatical relationships that exists between words and pre-specified product-attribute-related keywords in hopes of helping sentiment analysis, ignoring dependency pairs unrelated to these attributes. Perhaps a more related work is done by Mosha and Tianfang (2010), where they demonstrated that by incorporating dependency relationship as features, they are able to improve both the recall and precision of their baseline models in the sentiment analysis task of Chinese car reviews (14). However, one criticism of this approach is in the accuracy of dependency parsing, given that real-world natural language might poorly follow standard grammar.

3.3 Topic Modeling

Topic modeling provides a powerful tool for analyzing large text collections by representing high dimensional data in a low dimensional subspace. Topic modeling is commonly associated with sentiment analysis. Lin et al. (2009) proposed a novel probabilistic modeling framework based on LDA called joint sentiment/topic model (JST), which detects sentiment and topic simultaneously from text (9). Mei et al. (2007) proposed a similar Topic-Sentiment Mixture (TSM) model that can reveal the latent topical facets in a Weblog collection, the subtopics in the results of an *ad hoc* query, as well as their associated sentiments (13). Both methods used the identified topic models to justify the predicted sentiment but did not incorporate topic information in the training phase.

Topic models have also been used to improve sentiment prediction. Lu et al. investigated the efficacy of topic-model based approaches to multi-aspect rating prediction and found that they can be used to improve unsophisticated supervised baseline performance (10). J. McAuley and J. Leskovec (2013) aligned hidden factors in product ratings with hidden topics in product reviews and got more accurate predictions on product ratings, together with highly interpretable product topics (7). These two related works showed that topic models can potentially improve the sentiment analysis performance.

4 Methods

In this section, we provide a detailed description of the training and testing procedures, and a flowchart of the entire training phase is presented in Figure 1.

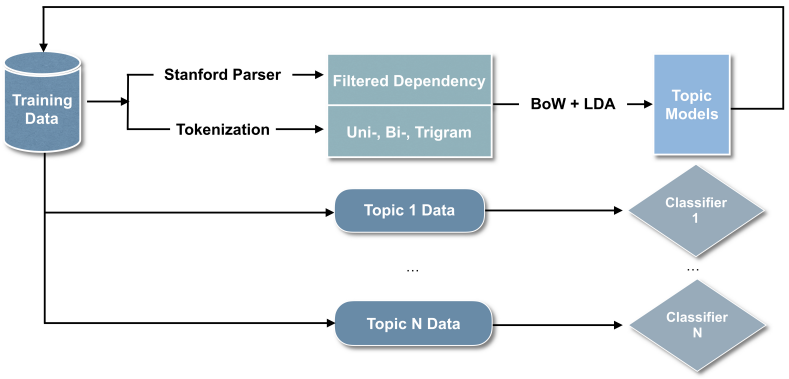

Figure 1: Training Phase Flow Chart

4.1 Training

In the training phase, given their irrelevance to the sentiment classification task, we first eliminated the remaining HTML tags in the training data (e.g., `<br>`), which was found during an coarse exploratory analysis; we also lemmatized the data using the library `SpaCy`. Next, we used the Stanford Dependency Parser to extract the lexical dependencies between words in the form of *relationship(governer, dependent)* (2). An example output from the parser is shown in Figure 2.

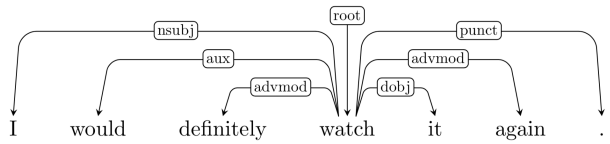

Figure 2: Example Output from Dependency Parser

Based on existing works that incorporated dependency relationship as features, we ignored the relationship tags and directly added the *(governer, dependent)* pairs as new bigram features. For instance, the sentence '*I would definitely watch it again.*' would result in the additional features `(i, recommend)` and `(film, recommend)`. Given the massive amount of dependency relationships, which is likely to produce an extremely spare feature vector, we selected a subset of them that are most relevant to the sentiment analysis task according to works by Nastase et al. (2006) (16). A full list and description of the dependency relationships retained is provided in Appendix 6.3. Then, we combined the uni-, bi-, and trigrams from regular tokenization and the new dependency bigrams to form new feature vectors using either BoW of TF vectorization. Specifically, we modified the default vectorizer method of the popular machine learning library `scikit-learn` (19). When building the dictionary, we made the arbitrary decision to keep only the top 1,000,000 n-grams ordered by term frequency across the corpus to reduce the size of the feature space.

Finally, we applied the Latent Dirichlet Allocation (LDA) to learn the topic distribution of the training data and distributed them to their most relevant classifiers for topic-specific training. In LDA under a BoW setting where we have K topics, D documents, and V words in the dictionary, the random variables are $\theta_d \sim \text{Dir}(\alpha) \in \mathbb{R}^K$ which represents the distribution of topics for document d, $\beta_k \sim \text{Dir}(\eta) \in \mathbb{R}^V$ which represents the distribution of words for topic k, $z_{d,n} \in [K]$ which indicates the topic that the n$^{\text{th}}$ word in document d belongs to, and finally $w_{d,n} \in [V]$ which is the index of the n$^{\text{th}}$ word in document d in the dictionary. The joint probability distribution is given by:

$$p(w, z, \theta, \beta|\alpha, \eta) = \prod_k p(\beta_k|\eta) \prod_d [p(\theta_d|\alpha) \prod_n p(z_{d,n}|\theta_d) p(w_{d,n}|z_{d,n}, \beta)]$$

However, the joint distribution cannot be directly used for our purpose: given a document d, we are interested in the conditional probability $P(topic = i|d)$, to determine the most relevant topics to a document. Naively applying the Bayes Theorem to compute $P(topic = i|d)$ as a posterior is computationally infeasible. Instead, sampling-based algorithms such as Gibbs sampling approximates the posterior through empirical sampling. In this project, we used `scikit-learn`'s LDA implementation, which relies on a Expectation Maximization (EM) procedure to find the topic distribution of the documents. The maximum number of iterations was set to be 5, the learning method was set to be `online` (i.e., using a mini-batch of training data to update parameters in each iteration), and the initial learning offset was set to be 50.

In choosing the underlying classifier for our model, we restricted our attention to NB and SVM, given that they are relatively fast to train and give satisfactory results, as suggested by our preliminary work presented in the Section 2. All classifiers are also constructed, trained, and tested using the library `scikit-learn` (19).

Hyperparameters of our proposed method were fine-tuned by five-fold cross-validation: (1) the number of topics learned by LDA, and (2) the number of classifiers a single document is assigned to. In addition, we would like to note that our proposed model does not require the same vectorization to be used during topic modeling and classifier training. This allows us to explore the best vectorization for each task separately.

4.2 Testing

In the testing phase, we first computed the topic distributions of the test data under the topic models generated during training. Then, according to how relevant each document is to each topic, we took a weighted vote of all the topic specific classifiers to get a final classification.

5 Results[2]

5.1 Customized Bag-of-Words

We performed the first group of experiments to evaluate the efficacy of customizing the feature vectors as proposed earlier, and reported the results in Table 1. The baseline performances were established by using BoW vectorization and text-frequency (TF) vectorization on regular 1-3 gram tokenization. Throughout the experiment we controlled the classification step to be a single NB classifier, so that any improvement from the baseline should be due to the customized feature vectors. Significantly, we showed that adding dependency pairs as new bigram features in addition to the lemmatized 1-3 gram tokens in BoW vectors increased test accuracy by 9.82%. If we were to vectorize the same tokens using tern frequency (TF), similar but smaller improvements would be observed (5.33% in our case). The advantage of BoW vectorization over TF is unexpected as TF outperformed BoW in all of our baseline experiments as seen in Table 1.

Tokenization	Vectorization	Accuracy
1-3grams	BoW	0.84476
1-3grams	TF	0.85336
1-3grams + Lemmatize	BoW	0.83812
1-3grams + Lemmatize + Dependency	**BoW**	**0.94296**
1-3grams + Lemmatize + Dependency	TF	0.90668
1-3grams + Lemmatize + Dependency + Cutoff	BoW	0.93732

Table 1: Experiment Results with Customized Bag-of-Words

5.2 Topic-based Ensemble

The second group of experiments were performed to compare the performance of a single classifier and the proposed topic-based ensemble. We fixed vectorization to be regular BoW vectors (without dependency pairs) so any change in test accuracy should be due to using the topic-based ensemble. Validation performances of the ensemble under different hyperparameter settings are reported in Figure 3. The black dashed line denotes the baseline. The yellow dashed line is the one with the best validation accuracy and the green dashed line is the one with the lowest perplexity.

Using the hyperparameter combination that had the best validation accuracy (10 topics for LDA topic modeling and distributing each sample to 2 most relevant topic-specific classifiers), the topic-based ensemble outperformed a single classifier, but only to a marginal extent. When TF vectorization was used instead of BoW vectors, the ensemble performed much worse than a single classifier, and thus we did not report the results and only used BoW vectorization in future experiments that involved topic-modeling.

Note that the test accuracy for *1-3grams & BoW* in Table 2 was slightly different from the accuracy of the 'same setting' presented in Table 1 because in this case we cut off most frequent (occur > 95% of the time) and least frequent (< 2 appearances) features.

5.3 End-to-end

In Table 2, we tuned the number of topics using cross validation and the performance improved when we used the best hyperparamater. Thus, the motivation for the end-to-end experiments is that we

[2]In all of our experiments, we fixed the n-gram setting to include all uni-, bi-, and trigram tokens, to retain more information than just using some subset of the tokens.

Tokenization & Vectorization	#Topics	Accuracy
1-3grams & BoW	-	0.84792
1-3grams & BoW	**10**	**0.85972**
1-3grams & BoW	12	0.84792

Table 2: Experiment Results with Topic-Based Ensemble

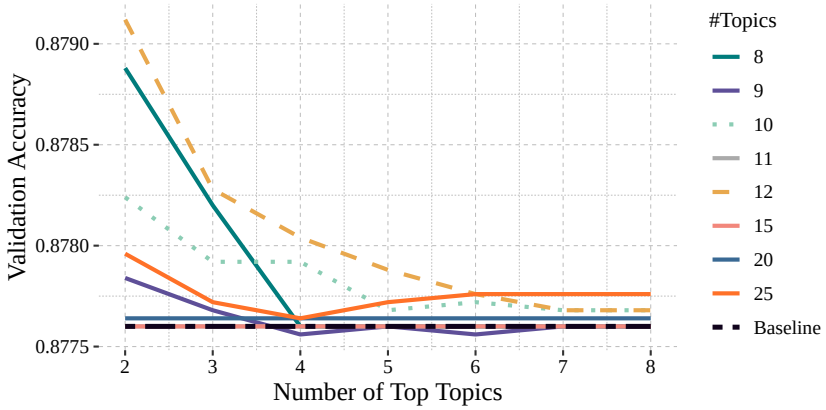

Figure 3: Relationship between Number of Top Topics, Topics, and Accuracy

expected to see improvement in the currently highest accuracy using the best parameter we got as we had better feature space. Since in topic modeling step and in classifier training step, the inputs are both vectors and hence we can use different vectorization methods in those two steps. For topic modeling we tried two methods and for training classifier we tried four different methods. The results of the 8 end-to-end experiment are reported in Table 3. Unfortunately, when we used BoW as the vectorization method in the topic modeling step where we used the best parameter we got from cross validation, only in the cases where we used only BoW and the combination of lemmatized BoW and dependency pairs with minimum and maximum cutoff in the training classifier step, the performance of the new models beat the baselines. On the other hand, when we used lemmatized BoW in the topic modeling step, our model didn't outperform the baselines, as expected as we didn't tune the parameter.

In the end-to-end experiments, none of which significantly outperformed those using a single classifier on our customized Bow vectors. The best test accuracy remained to be 94.296%, which was achieved using a single NB classifier on the customized vectors.

Topic Modeling	Training Vectorization	Accuracy	Baseline Acc
BoW	**BoW**	**0.85972**	**0.84792**
	Lemmatized BoW	0.8368	0.83812
	Lemmatized BoW + Dependency	0.9396	0.94296
	Lemmatized BoW + Dependency + Cutoff	**0.94108**	**0.93732**
Lemmatized BoW	BoW	0.84792	0.84792
	Lemmatized BoW	0.83812	0.83812
	Lemmatized BoW + Dependency	0.94296	0.94296
	Lemmatized BoW + Dependency + Cutoff	0.93732	0.93732

Table 3: End-to-end Experiment Results

As a comparison, the state-of-the-art performance on this dataset is 0.9540 using an LSTM-based language model (6) .

6 Discussion and Analysis

6.1 Regarding the Effectiveness of Customized Vectors

Our experiment results demonstrated the significant contribution of using our customized vectors to classification accuracy. Compared to regular BoW vectors, our customized vectors made two modifications: (1) the tokens were lemmatized and the remaining HTML tags were removed from the texts; (2) dependency pairs extracted by the Stanford Dependency Parser were added as new bigram features. Results from Table 1 showed that the first modification alone could not increase test accuracy; in fact, 1-3 grams with lemmatization led to worse results compared to unprocessed 1-3 grams. In another study done by Camacho-Collados and Pilehvar (2017), we also observed that lemmatizing does not seem to help (1). In fact, the relatively weaker performance of lemmatization in this experiment since the coverage of word embeddings in vanilla-tokenized datasets is limited e.g. inflected forms are missing in lemmatization. Thus, in general, more complex preprocessing techniques like lemmatizing sometimes don't outperform sole tokenization.

When we lemmatized the text and incorporated lexical dependency into our new feature vectors, the classifiers performed better than the ones that only used uni-, bi-, and trigrams. However, when only lemmatization was applied, the classifier performed worse than the baseline. Using both lemmatization and lexical dependency, the classifier using BoW feature vectorization outperforms the the one that used TF vectorization.

As a naïve attempt to investigate the reason behind the superior performance of our feature representations using dependency word relations, we used sparse singular value decomposition to project the feature vector to its first two principle components, and plotted the positive and negative data points. We want to visually see if the higher accuracy obtained using the customized feature was simply attributed to the fact that the data can be nearly linearly separable in the a lower dimension. Unfortunately, this is not the case (the figures are not shown here due to space constraints).

6.2 Comparisons with Related Work on Using Dependency Relations

We were motivated by the method proposed by Mukherjee and Bhattacharyya (2012), which used dependency parsing to exploit relations between the opinion expressions about different features of products that together form a coherent review (15). We both used Stanford parser to extract dependency relations. They evaluated their proposed system on sentiment classification of product reviews. Relative to the baseline, they increased the accuracy by around 20% from 60% to 80%.

In prior study done by Natase et al. (2006), syntactically related pairs of words were generated to provide a better description of what a document is about (16). Their main task was topic multi-class classification while we evaluate our approach on a binary classification task about sentiment analysis. Except for that, they only incorporated unigrams, lemmatized unigrams and dependency pairs while we took uni-, bi-, trigrams and customized dependency pairs into consideration. Meanwhile, we generated syntactically connective pairs using different parsers.

Perhaps a more relevant study was done by Xia et. al (2011), where they not only adopted word dependency relationships as new features but also evaluated their approach on the sentiment analysis task on the Cornell movie-review corpora (23; 18). They tested the effects of four different tokenizations: (1) using only unigrams, (2) using uni- and bigrams, (3) using only dependency word pairs, and finally (4) a *joint word relation feature* that included both uni- and bigrams and dependency word pairs. Then, using a BoW vectorization and NB as classifer, their results suggested a near 3% boost in performance using both dependency pairs and n-grams in comparison to baseline. Apart from the Cornell movie dataset, the average increase in performance on several other datasets using the joint word relation feature is about 5%. The reason why their marginal increase in accuracy was not as significant as ours might be due to several reasons, which includes the obvious difference in datasetas, a different dependency parser, and the fact aht they didn't include trigram tokens.

6.3 Regarding the Use of Topic Modeling

We found that topic-based ensemble performs better than a single classifier under specific hyperparameter settings. This indicates that the number of latent topics assumed can affect how the topics are correlated with sentiments, and how biased the ensemble classifiers are can affect their collective

decisions. In addition, the number of latent topics that scored the lowest LDA perplexity achieved the best test accuracy and improved classification accuracy by 1.18%. This suggests that a topic model that clusters the data well is suitable for building a topic-based ensemble that can enhance the baseline. The ensemble is more likely to have an edge over the baseline when its classifiers are trained under *highly biased* setting (i.e., trained on a small set of documents that are highly related to specific topics). Such an edge (though very small) vanishes and ensemble performance converges to baseline when the training sets are more diluted and less topic-specific.

At the same time, we found several limitations of topic modeling. First, *extensive fine-tuning* is required for the topic-based ensemble to outperform a single classifier. We found that when using hyperparameters that were fine-tuned, the accuracy *increased* when using "BoW" and "Lemmatized BoW + Filtered Dependency" vectorization method for classification, and *decreased* using "Lemmatized BoW" and "Lemmatized BoW + Dependency" for classification. The mixed results suggested that using hyperparameters that were fine-tuned could improve test performance in two specific cases, but not others. However, when we used the hyperparameters fine-tuned for BoW topic modeling, and applied them for the Lemmatized BoW topic modeling, we found that the accuracy will either *decrease* or *stay the same* compared to the baseline accuracy. This further indicated that hyperparameters of topic modeling need to be fine-tuned in order to be potentially better than the baseline. One of our current result's limitations is that we did not fine tune the hyperparameters for the Lemmatized BoW modeling, so in our future work, we can further improve performance by doing cross validation on Lemmatized BoW modeling.

Second, even using the hyperparameters obtained through cross validation, the accuracy did not increase significantly from the baseline (increased by 1.18%). Also when we used "Lemmatized BoW" and "Lemmatized BoW + Dependency" for training vectorization methods, the accuracy was also lower than the baseline accuracy. So this suggested that even after fine tuning hyperparameters, the performance did not guarantee to be better than the baseline.

Appendix A List of Dependency Relationships Retained [3]

nsubj (**nominal subject**) - An adjectival modifier of an noun phrase is any adjectival phrase that serves to modify the meaning of the noun phrase. (e.g., 'The baby is cute': *nsubj*(cute, baby))

dobj (**direct object**) - The direct object of a verb phrase is the noun phrase which is the (accusative) object of the verb; the direct object of a clause is the direct object of the verb phrase which is the predicate of that clause. (e.g., 'They win the lottery': *dobj*(win, lottery))

agent **(agent)** - An agent is the complement of a passive verb which is introduced by the preposition "by" and does the action. (e.g., 'The man has been killed by the police': *agent*(killed, police))

advmod **(adverbial modifier**) - An adverbial modifier of a word is an (non-clausal) adverb that serves to modify the meaning of the word. (e.g.,'Genetically modified food': *advmod*(modified, genetically))

amod**: (adjectival modifier)** - An adjectival modifier of an noun phrase is any adjectival phrase that serves to modify the meaning of the noun phrase. (e.g., 'Sam eats red meat': *amod*(meat, red))

neg**: (negation modifier)** - The negation modifier is the relation between a negation word and the word it modifies. (e.g., 'Bill is not a scientist': *neg*(scientist, not))

acomp**: (adjectival complement)** - An adjectival complement of a verb phrase is an adjectival phrase which functions as the complement (like an object of the verb). (e.g., 'She looks very beautiful': *acomp*(looks, beautiful))

xcomp**:(open clausal complement)** - An open clausal complement (xcomp) of a verb phrase or an adjective phrase is a clausal complement without its own subject, whose reference is determined by an external subject. (e.g., 'He says that you like to swim': *xcomp*(like, swim))

[3] The description and examples of the dependency relationships are adapted from the Stanford Typed Dependencies Manual (3)

References

[1] Camacho-Collados, J., Pilehvar, M. T. (2017). On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis. arXiv preprint arXiv:1707.01780.

[2] Chen, D., Manning, C. (2014). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 740-750).

[3] De Marneffe, M. C., Manning, C. D. (2008). Stanford typed dependencies manual (pp. 338-345). In *Technical report, Stanford University*.

[4] Ficamos, P., Liu, Y. (2016). A topic based approach for sentiment analysis on Twitter data. In *International Journal of Advanced Computer Science and Applications, 7(12)*, 201-205.

[5] Gray, S., Radford, A., Kingma, D. P. (2017). Gpu kernels for block-sparse weights. *arXiv preprint arXiv*:1711.09224.

[6] Howard, J., Ruder, S. (2018). Universal language model fine-tuning for text classification. arXiv preprint arXiv:1801.06146.

[7] McAuley, J., Leskovec, J. (2013, October). Hidden factors and hidden topics: understanding rating dimensions with review text. In In *Proceedings of the 7th ACM conference on Recommender systems* (pp. 165-172). ACM.

[8] Limin Yao, David Mimno, and Andrew McCallum. Efficient methods for topic model inference on streaming document collections. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 937–946. ACM, 2009.

[9] Lin, C., He, Y. (2009, November). Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management* (pp. 375-384). ACM.

[10] Lu, B., Ott, M., Cardie, C., Tsou, B. K. (2011, December). Multi-aspect sentiment analysis with topic models. In *2011 IEEE 11th international conference on data mining workshops* (pp. 81-88). IEEE.

[11] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., Potts, C. (2011, June). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1* (pp. 142-150). Association for Computational Linguistics.

[12] Mäntylä, M. V., Graziotin, D., Kuutila, M. (2018). The evolution of sentiment analysis: A review of research topics, venues, and top cited papers. In *Computer Science Review, 27*, 16-32.

[13] Mei, Q., Ling, X., Wondra, M., Su, H., Zhai, C. (2007, May). Topic sentiment mixture: modeling facets and opinions in weblogs. In *Proceedings of the 16th international conference on World Wide Web* (pp. 171-180). ACM.

[14] Mosha, C., Tianfang, Y. (2010). Combining dependency parsing with shallow semantic analysis for Chinese opinion-element relation identification. In *2010 4th International Universal Communication Symposium*.

[15] Mukherjee, S., Bhattacharyya, P. (2012, March). Feature specific sentiment analysis for product reviews. In *International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 475-487). Springer, Berlin, Heidelberg.

[16] Nastase, V., Shirabad, J. S., Caropreso, M. F. (2006). Using dependency relations for text classification. In *Proceedings of the 19th Canadian conference on artificial intelligence* (pp. 12-25).

[17] Pang, B., Lee, L., Vaithyanathan, S. (2002, July). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10* (pp. 79-86). Association for Computational Linguistics.

[18] Pang, B., Lee, L. (2004, July). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics (p. 271)*. Association for Computational Linguistics.

[19] Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. In *Journal of Machine Learning Research 12* (pp. 2825-2830).

[20] Qiu, G., Liu, B., Bu, J., Chen, C. (2011). Opinion word expansion and target extraction through double propagation. In *Computational linguistics*, 37(1), 9-27.

[21] Johnson, R., Zhang, T. (2016). Supervised and semi-supervised text categorization using LSTM for region embeddings. *arXiv preprint arXiv*:1602.02373.

[22] Wang, S., Manning, C. D. (2012, July). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2* (pp. 90-94). Association for Computational Linguistics.

[23] Xia, R., Zong, C., Li, S. (2011). Ensemble of feature sets and classification algorithms for sentiment classification. In *Information Sciences*, 181(6), 1138-1152.