Data Augmentation via Diversity Enhancement
Group Manual Intelligence Lab

Yuda Fan
ETH Zürich
yudfan@ethz.ch

Jinfan Chen
ETH Zürich
jinfchen@ethz.ch

***Abstract*—Text classification with deep learning has been paid increasing attention during past few years. To help models better understand the text, many data augmentation techniques were proposed. However, the majority of them oversee the structure of the sentence, making them less effective. In our paper, we proposed a simple but novel data augmentation method based on word embedding, which just duplicates one word in each data entry. Empirically, evaluating on the Twitter dataset, extensive experiments prove that our method can be generally adopted with different models and can improve their performance by a notable margin.**

I. Introduction

Text classification is the task to assign a document to a specific category. In our paper, we focus on distinguishing the sentiment emotion behind the text (positive or negative). Last decade has witnessed the popularity of microblog such as Twitter, which also raising up the importance to classify a short text. Most of the early methods rely on the word embedding technique, which tried to map the words into the vector space and directly aggregate words embedding into sentence embedding. As the attention mechanism is raised in [1], Seq2Seq models like BERT [2], GPT-3 [3] and MT-DNN [4] based on Transformer architecture have played a critical role in such tasks since then.

However, challenge is that microblogs are usually very short and sparse, relatively increasing the magnitude of the noise in the text, making our models hard to grasp the meaning and pay attention correctly. Hence, such small dataset would make trained models less generalized and more vulnerable to adversarial attacks [5]. Therefore, data augmentation on text is proposed to improve the model performance and robustness.

In our paper, we firstly analyze the drawbacks of the traditional sentence embedding method which is called **Collapse on Similar Majority**. Additionally, we define the diversity of the sentence and find the most critical word in the whole sentence. According to that, we propose a simple data augmentation technique which just replicates the most important word in each sentence located by us. This method is quite general, efficient, and works well with various models like xgboost, FastText and BERT-based Transformer.

In a nutshell, our contributions are:

1) Analyze the drawbacks of traditional sentence embedding methods based on word embedding. We discover that simply aggregate the word vectors may overlook the structure and the sentiment contribution of each single word. We deeply analyze the reason why weighted average word vectors into sentence vectors may attribute inappropriate importance to single words.

2) Propose a data augmentation strategy via diversity enhancement. Based on the latent variable generative model proposed by [6], we define the diversity of the sentiment of the sentence based on the embedding of its words. According to the diversity contribution of each single words, we can pick out the most important word in the sentence, and replicate it once to guide the attention of the learning models.

3) Extensive experiments to demonstrate the effectiveness. We adopt our data augmentation with three baselines, including xgboost, Fasttext and BERT-based Transformer, and our method achieve notable improvement on all three baselines almost without increasing training cost.

II. Related Works

Word Embedding. Word embedding is to map the words into low dimensional vector space, in which the similarity of words is represented by cosine similarity of word vectors. There are two typical ways to get word embedding among a large corpus. One is to learn vector representations according to latent representation of neural networks like [7] and [8]. The other is to compute the low rank approximation of the co-occurrence matrix like Glove [9].

Sentence Embedding. Based on word embedding, unweighted average of the word vectors is regarded as the sentence vector by [8]. Besides, weighted average associated with TF-IDF is widely used to generate the sentence vector. Addition to that, recurrent neural tensor networks on a parse tree is employed to compute the compositional effects of word sentiment in [10].

Attention Mechanism. In sequence to sequence learning, encoder-decoder models are widely used. In order to prevent the important context from being lost during processing a long sequence, attention mechanism was introduced by [11], which allows the model to focus on specific part of the input sequences. Based on this mechanism, Transformer backbone is proposed by [1] which includes scaled dot-product attention and multi-head attention.

Data Augmentation. Data Augmentation includes techniques that enhance and enrich the original data for better performance. [12] introduces several successful data augmentation methods in NLP, among which Back-translation [13] and adding random perturbation [14] are the most common approaches. This paper follows a different path of replicating important words in training sentences. Our method seeks to diversify the sentences so that the classifier can make better predictions.

III. Methodology

We first introduce the preliminary of latent variable generative model. Based on that, we analyze the drawback of traditional sentence embedding. Finally, we propose our own data augmentation that exploits the diversity of word embedding.

A. Review on Latent Variable Generative Model

Given that $v : w \to \mathcal{R}^d$ is a word embedding from vocabulary w to vector space $\mathcal{R}^d$. We regard the generation of the sentence as a temporal dynamic process where at step t is the t-th word vector v_t correspond to the word w_t is emitted. For each time step t, we model the prefix of the sentence by a latent discourse variable c_t which capture the central sentiment of the previous words,

$$Pr[\text{vector } v_t \text{ emitted at step t}] \propto \exp(c_t^\top v_t). \tag{1}$$

Proposed by [8], we can approximate the discourse vector c_t by unweighted average of the word vectors,

$$c_t = \frac{1}{t-1}\sum_{i=1}^{t-1} v_i.. \tag{2}$$

Therefore,

$$Pr[\text{sentence } s \text{ is emitted}] \propto \prod_{i \neq j} \exp(v_i^\top v_j), \tag{3}$$

indicating that we can approximate the likelihood of the sentence s by the pairwise exponential cosine similarity between its words.

B. Drawbacks of Traditional Sentence Embedding

Given a sentence s consisting of words w_i whose word vector is v_i, the sentence vector v_s is commonly regarded as the weighted average of word vectors v_i,

$$v_s = \sum_{i=1}^{t} f_i v_i, f_i \in (0,1), \tag{4}$$

where f_i is usually picked as the normalized term frequency–inverse document frequency (also known as TF-IDF score). However, TF-IDF score neglects the spatial structure of the word vectors, leading it to paying attention to those words which may not contribute much to the sentiment.

Consider the following simple restaurant review:

Environment is excellent, service is good, chef is excellent, price is also cheap, dishes are poor.

Among these four adjectives, "excellent" achieves the highest TF-IDF scores, "good", "cheap" and "poor" get the same TF-IDF score so they are associated with the same weight. However, both "excellent" and "good" are not critical to this sentence since if we eliminate any one of them in this sentence, the sentiment of the review is not changed significantly because they still have similar words remained. Nonetheless, if we remove the word "poor", the sentiment will be considerably overhauled.

We can these phenomenon by "Collapse on Similar Majority": If a single includes several similar words which is close in the word embedding, they will dominate in the sentence vector and overwhelms over the other words, whereas they are not that important.

C. Data Augmentation via Diversity Enhancement

To alleviate such phenomenon, we should guide our model to pay more attention to those inalienable words, which contribute more to the diversity of the sentence sentiment, instead of overemphasizing those words belonging to the similar majority.

According to equation (4), the sentence vector lies in the interior of the low dimensional parallelepiped spanned by those word vectors (see Figure 1 in appendix for detail illustration for parallelepiped). Therefore, we measure the diversity of sentence s by the volume of the low dimensional parallelepiped spanned by its word vectors.

Denote A as the matrix whose columns are word vectors v_i in sentence s (duplicates eliminated). Denote $div(s)$ as the diversity of sentence s.

$$div(s) = \sqrt{|det(A^\top A)|}, \tag{5}$$

where $det(\cdot)$ is the determinant of the matrix and $A^\top A$ is the Gram matrix of the word vectors v_i in the inner product space $\mathcal{R}^d$. Denote the diverse contribution from word vector v_i to sentence s as $c(s, v_i)$, we measure this contribution by

$$c(s, v_i) = \frac{div(s)}{div(s \setminus v_i)} = \sqrt{\frac{|det(A^\top A)|}{|det(M_{i,i}(A_i^\top A_i))|}}, \tag{6}$$

where A_i is the matrix by filling i-th column of A with 0 and $M_{i,i}$ is the minor of the matrix after excluding the i-th column and i-th row. As long as v_i are linearly independent, it is guaranteed that $div(s) \neq 0$, making this well-defined (see Appendix for detailed proof).

By computing the diversity contribution of each individual word, we pick out the most important word with the largest contribution among the sentence. By replicating it at its first occurrence, we got a simple data augmentation strategy via enhancing the diversity of the sentence.

IV. Experiments

A. Baselines

We include in total three baselines in our paper, training them on the original corpus and augmented corpus separately.

Xgboost: Xgboost is a highly effective, scalable tree boosting system proposed by [15]. In Xgboost, gradient tree boosting is employed to optimize the learning objective, and shrinkage together with column subsampling is adopted to prevent overfitting. However, due the limited representative ability of decision tree model, Xgboost achieves the lowest accuracy (less than 80%) among the three baselines.

Xgboost allows us to train a binary feature vectors classifier with "binary:hinge" objective. To get feature vectors, we employ one-hot encoding for the words in whole vocabulary, and the sentence vector is the weighted average of the word vectors. Hence, TF-IDF score is utilized to assign coefficient to word vectors, according to equation (4).

FastText: Compared to Xgboost which is a universal feature based machine learning system, FastText is proposed by [16] especially for learning continuous word representations. In [16], each word is regarded as a bag of character n-grams (consecutive n characters) and the word vector is the summation of character n-gram vectors, which takes morphology of the words into consideration.

FastText can be directly trained on the labeled corpus and allow us to make predictions via binary executable files. Training on the same text, FastText achieve better accuracy than Xgboost (averaging at 82.0%).

Bert Pre-trained Model: Bert [2] is a class of deep learning models that leverages the attention mechanisms [1] for various NLP tasks. Most of its effectiveness comes from unsupervised pre-training on a large corpus, and the users can fine-tune the pre-trained weights on specific down-streaming tasks with smaller-scale labeled data.

We use the publicly available Bert-Base-Uncased pre-trained model from HuggingFace. The data is preprocessed, augmented, and tokenized with Bert-Base-Uncased tokenizer. After that, the tokenized text data is fed to the sequence classification pre-trained model for the sentiment classification task.

The model achieves 87.06% test accuracy without our diversity enhancement. The performance reaches 87.54% (our best) with diversity enhancement.

B. Implementations

Preprocessing: All of the experiments runs in our paper share the common preprocessing. Since Twitter is a microblog platform in which most of the posts are quite oral and inform, causing much noise in the dataset.

To clean and normalize the noisy tweets, we first filter out all the illegal characters and meaningless symbols, including the $\langle user \rangle$ and $\langle url \rangle$ symbols generated by Twitter. Besides, we restore those most common contraction to its formal style and correct the most often seen misspellings to make sure that the same word always has the same representation. Hence, there are many words whose characters are totally separated like "w h a t" or "r e a l", we detect this and concatenate these single characters together.

Additionally, what is special in Tweets is that posts may include many tags, which is strings that start with special character "#" following by several words which are concatenate consecutively without space like "#believe15". In this case, we employ wordninja library [17] to split the words.

Diversity Enhancement: To get word embeddings, we employ the Glove library [9], which provides us with pre-trained model mapping words into 200 dimensional vector space.

In diversity enhancement step, to avoid the interference of the stopwords, we only take the 8 words with top 8 highest TF-IDF scores into consideration. Accordingly, we only conduct diversity enhancement for the sentence with at least 8 distinct words.

Hence, it should be noticed that different from the often-seen data augmentation methods, we simply replace the original text with the modified text, which means the number of the data entries remains the same. Benefit from this, our data augmentation almost bring no additional training cost: it only increases the text file size from 167Mb to 178Mb, by a ratio of 6.5%.

Back Translation: We have implemented and evaluated back-translation [13] as an alternative data augmentation technique. The model translates the preprocessed sentences from English to Germany, then back to English. The translated sentences and original sentences are concatenated for a larger dataset. However, back-translation has not experimentally shown performance improvements towards the vanilla Bert.

C. Results

Results of our experiments are showed in Table I.

For Xgboost models, we run all the experiments for "binary:hinge" learning objective and employ "error" as the evaluation metric. We use the default hyperparameters and train both models for 600 epochs.

For FastText models, we train each model 10 epochs for 5 runs. Validation Accuracy is reported as the average accuracy of 5 runs and Test Accuracy is reported as the public score of the selected model with the highest Validation Accuracy among these 5 runs.

For the Bert model, pre-trained weights from HuggingFace are loaded. We randomly split 95% data as a training set and 5% validation set. Empirically, the validation loss converges within three epochs for all Bert experiments. And the best model from 3 epochs is chosen to report accuracy values in the table. The training batch size is set to be 16. Although more than half of the sequences are longer than 64, we found that the performance does not decrease with sequence length, and hence we use a maximum sequence length of 64 for faster experiments.

Methods	Validation Accuracy	Test Accuracy
One-Hot Encoding + Xgboost	76.39	75.86
+Diversity Enhancement	77.37	76.90
FastText	82.05	81.60
+Diversity Enhancement	81.95	81.80
Bert $_{\texttt{base}}$w/o aug	88.68	87.06
+Back-translation	88.45	86.80
+Diversity Enhancement	**88.70**	**87.54**

Table I: Accuracy results for different models we evaluated. Bert with data diversity enhancement performs best on both validation and test sets.

Analysis: For Xgboost, it is noticed that diversity enhancement improve it by a notable advancement since Xgboost models are feature based and we employ the TF-IDF scores as our weights. Enhancing the diversity of sentence will significantly change the sentence embedding. Actually, after 400 epochs of training, Xgboost models with data augmentation is almost as good as the baseline model, which is trained for 600 epochs, measured in validation accuracy.

For FastText, we could observe that diversity enhancement merely improve it by a marginal difference. The reason is that FastText set minimum and maximum occurrence threshold for words, and only the word with appearances in this interval will be taken into consideration. Therefore, simply replicate those important words may make FastText excluded some words out of the dictionary, leading to the shrinkage of vocabulary size. Hence, FastText focus on the character-level n-grams, while our data augmentation pay more attention on the word level.

For Bert, we observed performance improvement in both validation and test accuracy using diversity enhancement. The validation set is large, so the validation accuracy has slight variance but is biased. However, the test set is unbiased but with a more significant variance. Diversity enhancement achieves better scores in both cases is a strong argument for its effectiveness. The Bert model has an inherent attention mechanism, and we conjecture diversity enhancement increases the probability that the model pays attention to more important words. Consequently, the model performs better with diversity-enhanced data.

V. Conclusion

In this paper, we highlight a performance collapse of traditional sentence embedding methods which is a linear combination of word embeddings. Derived from the latent variable generative model, we propose the diversity of a sentence, which is measured by the volume of parallelepiped spanned by word vectors. Based on the concept, we could simply locate the most critical words among the sentence. By replicating it just for once, we could simply improve the performance of various models almost without increasing training cost.

References

[1] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *ArXiv*, vol. abs/1706.03762, 2017.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019.

[3] L. Floridi and M. Chiriatti, "Gpt-3: Its nature, scope, limits, and consequences," *Minds and Machines*, vol. 30, pp. 681–694, 2020.

[4] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," *CoRR*, vol. abs/1901.11504, 2019. [Online]. Available: http://arxiv.org/abs/1901.11504

[5] J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi, "Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp," in *EMNLP*, 2020.

[6] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski, "A latent variable model approach to pmi-based word embeddings," *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 385–399, 2016.

[7] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," in *J. Mach. Learn. Res.*, 2000.

[8] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.

[10] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, 2013.

[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2015.

[12] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. H. Hovy, "A survey of data augmentation approaches for NLP," *CoRR*, vol. abs/2105.03075, 2021. [Online]. Available: https://arxiv.org/abs/2105.03075

[13] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[14] J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," in *EMNLP-IJCNLP*, 2019.

[15] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[16] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[17] D. Anderson, "Wordninja," https://github.com/keredson/wordninja, 2017.

APPENDIX

Parallelepiped Illustration

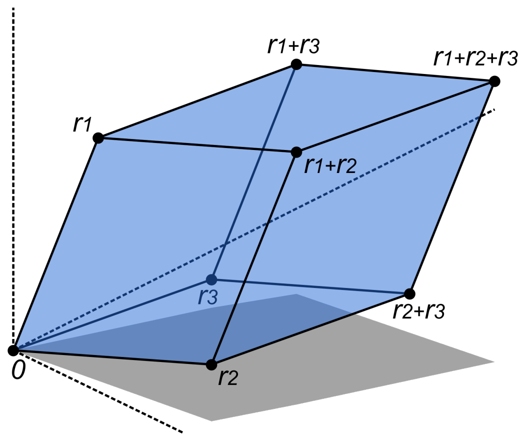

Figure 1: A illustration for parallelepiped from Wikipedia
.

Figure 1 shows a directed parallelepiped spanned by three vectors r_1, r_2 and r_3. Notice that this is a projection in a 3D dimensional subspace, r_i may have much more higher dimensions.

Proof: Invertible Gram Matrix is Equivalent to Linear Independency

Take $G = A^\top A$, we will justify that G is invertible if and only if A is linearly independent in coloumns.

$\Rightarrow$: Proof by inverse negative proposition. Suppose A is linearly dependent in columns, there exists a solution $u \neq 0$ of the linear system $Au = 0$. Therefore,

$$Gu = A^\top Au = 0,$$

implying that G is not invertible.

$\Leftarrow$: Still prove by inverse negative proposition. Suppose G is not invertible, there exist $u \neq 0$ such that $Gu = 0$. Therefore,

$$u^\top Gu = (Au)^\top (Au) = \|Au\|_2^2 = 0.$$

Thus, $Au = 0$, indicating that A is linearly dependent in columns.