POS tagging enhancement to NLP classification tasks

Omri Maoz, Tal Levi

Tel-Aviv University

{omrimaoz, tallevi}@mail.tau.ac.il

Abstract

In our research *we tested the hypothesis that a task of classifying contextualize phrases can be better execute with common NLP tools when feeding them with their corresponding word-to-word tagging phrases. The key observation when examine a contextual phrase is that there's a common semantic between some words in phrases that belong to the same class. For example, a positive review of a movie contains a word that holds a positive semantic meaning and this semantic is common in words in other positive reviews. We believe that this semantic might be expressed with some POS tagging pattern in the corresponding POS tag phrases. Moreover, the POS tagging phrases significantly reduce the vocabulary size, which may allow to form patterns with fewer data (collecting large, labeled data is often a problem itself). Therefore, we assume two assumptions to test in this project:*

1. *It possible to learn POS tagging pattern to achieve better results on NLP classification task.*
2. *An equivalent results can be achieved with fewer data*

1. Introduction

Classification of commercialized text phrases is a task with many applications that we use and affect our daily lives. An example is a technology that scan comments and reviews of users in social network, searching for violation of the terms and rules set by the network owner to remove and block these "bad" contents.

A lot of work and study has been done in this subject and found that deep learning tools achieve good results. Models such as CNN-GRU and BiLSTM for example achieve around 80% accuracy on Movie Reviews dataset with more than 10,000 reviews divided into 2 classes.

	Model	Movie Reviews
0	BiLSTM-rand	77.6
1	BiLSTM-static	79.5
2	BiLSTM-dynamic	79.8
3	1D CNN-static	79.0
4	1D CNN-dynamic	79.4
5	Ensemble CNN-GRU-rand	77.0
6	Ensemble CNN-GRU-static	79.8
7	Ensemble CNN-GRU-dynamic	79.4

Table 1. Some common deep learning architectures accuracy results. (Source: [1])

Improving results for those NLP models is not an easy task and there're many variables and parameters to consider alongside many approaches to test and experiment. One approach, which is supported by recent research shows that preprocessing techniques of the data manage to improve NLP models results.

Part-of-Speech (POS) tagging is a popular NLP process which refers to categorizing words in a text in correspondence with a particular part of speech, depending on the definition of the word and its context.

Recent study on question classification in Thai language has shown that text classification can

benefit from Part-of-Speech (POS) tagging as part of feature selection method to achieve better results [2]. In Thai language some words have different meaning when considered alone than when joining other words due to a meaning word based on ordering the sequence of words and context. Therefore, considering a syntactic feature for the obvious classification of Thai sentences is necessary. One feature the research editors managed to find by analyzing frequency of POS tag for each question type in the packages "PyThaiNLP" and "Stanford CoreNLP", is that each question type was sensitive to some POS tagging ratios.

To test our research hypothesis, we focused on 3 NLP models – DAN, BiLSTM and Transformer.

We fed those models with 2 datasets:

1. "IMDB reviews" dataset, divided into 2 classes (positive, negative), with more than 15k reviews.
2. "News headline" dataset, divided to many topics as classes, but we focused on the 5 most common, with more than 3,000 headlines per category.

One more motivation to use POS tagging as English words categories is that this preprocessing methodology is general and does not depend on pre known knowledge so it can be applied to any given dataset.

Our code is available at
https://github.com/omrimaoz/NLP_Project

2. Related Work

Text classification is the process of classifying or categorizing the raw texts into predefined groups. There are many studies and works developing and testing numerous approaches and methods to tackle this problem and with great results - from fundamental machine learning algorithm such as linear regression and SVM to state-of-the-art deep learning models.

Today, commercial hi-tech companies integrate text classification technologies in their products from filtering spam emails, to analyzing politician's speeches, abuse content marking and removal and much more.

3. Methodology

3.1 Datasets

For performance evaluation of the suggested hypothesis, we used two datasets to test binary classification and multi-class classification. The IMDB reviews dataset contain various length reviews dividing into positive and negative reviews. This dataset is interesting to extract feature from because positive and negative writing have many language forms (cynicism, exaggeration, straight-forward, etc.). The second dataset is the News headlines dataset, which contain 40 different subjects (as classes). In contrast to the first dataset, this dataset has approximately similar headlines length, but it may contain some interesting tagging patterns. We chose 5 subjects to focus on (Politics, Entertainment, Wellness, Travel, Sport), each with more than 3,000 headlines. We used different sizes of subset from the datasets, in decreasing order to test the 2 research assumption. Split validation factor set to be 0.2 for evaluation, and we preserved uniform distribution between classes.

3.2 Data preprocessing

3.3.1. Tagging and Tokenization

We used Stanza (Python NLP package) to create a corresponding POS tagging sentences for each sentence from the datasets by calculating each word's tag.

To tokenize words, we used spaCy (Python NLP Library) with 'en_core_web_sm' trained pipeline.

3.3.2. Feature selection

The goal is to improve text classification with the 3 tested NLP models. To do that we proposed and calculated 3 preprocess methods to produce

features which in turn were fed to the models and tried to improve results accomplished by them.

The methods:

1. **Tag Filtering** (referred as filter) – when overviewing the text in the dataset we noticed that some tags are more capable to form a combination with other tags to express a tag pattern and semantic relation. POS tags such as PUNCT (punctuation), PRON (pronoun), etc. are less likely to belong to a key subsentence which points to the semantic class of the sentence. On the other hand, NOUN, ADV (adverb), ADJ (adjective) and VERB are more likely to do so.

   With "Tag Filtering" we remove all tags but NOUN, ADV, ADJ and VERB tags from the dataset.

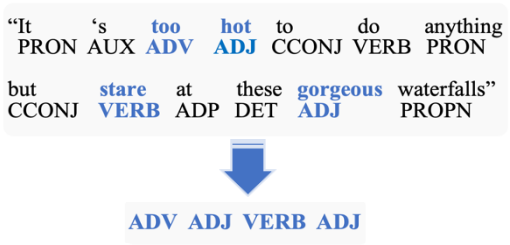

Figure 1. 'Travel' headline from the News headline dataset and its corresponding POS tag sentence.

2. **Tag Extend** (referred as extend) – most of the NLP deep-learning models work better with big vocabulary. By looking on POS tag corresponding phrases we greatly reduce the vocabulary size. To compensate for this negative effect, we extended the 4 "interesting tags" (NOUN, ADV, ADJ and VERB) to subcategories.

   - NOUN and VERB – python NLP package called NLTK with WordNet lexical database by Princeton provides subcategories for nouns and verbs. We use Synset, a tool from NLTK to look for semantic words relation in WordNet, to distribute words to these subcategories.
   - ADJ and ADV – For ADJ and ADV we used the most common adjective and adverb subcategories in the English language. In addition, we added suffixes as subcategories and used Spacy similarity feature tool to distribute words with these subcategories.

   We then concatenated the most 2 similar subcategories to the POS tag of any relevant word in the dataset.

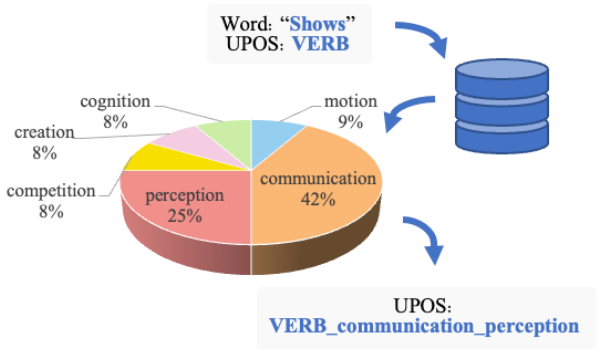

Figure 2. Calculation pipeline to extend word tag according to the most similar subcategories. Example for VERB categorized word.

3. **Bigram Interpolation** (referred as bigram) – In order to highlight tag patterns in text we used pairing 2 consecutive words in the sentence in a most common appearance in the dataset, in a decreasing order. Doing that, we emphasize common tag pairs over lesser common tag pairs.

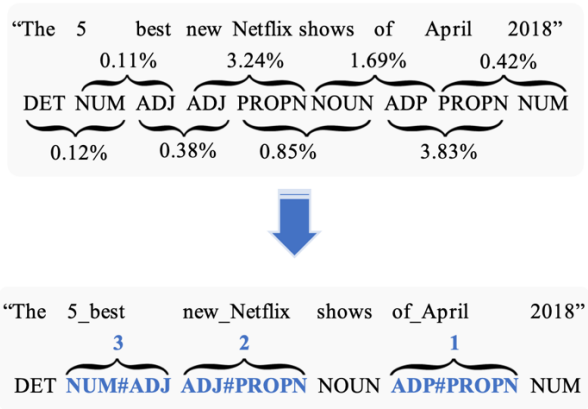

Figure 3. Example of "Bigram Interpolation" on 'Entertainment' headline from the News dataset.

Lastly, we used different combination of the methods above as feature selection to test our models with.

3.3 Classification (models)

To test our hypothesis, a comparison needed to be done and for that we needed relevant NLP models. DAN, BiLSTM and Transformer models are a good choice because while differing in their architecture, they are taking into calculation information between neighbor words in the text.

3.3.1. DAN – Deep Averaging Network [3]

We constructed an embedding matrix of size 300×300, then padded each input to size 300, followed by an average layer to average all the input words, 3 linear layers from 300 to 300 neurons with Relu activation function, and a linear layer with Softmax activation function from 300 to the number of dataset classes neurons.

3.3.2. BiLSTM – Bidirectional Long Short-Term Memory [4][5]

We constructed an embedding layer of size 300 followed by 3 recurrent layers with 128 BiLSTM cells, a 50% dropout layer, a linear layer from 128 to 32 neurons with Relu activation function, and a linear layer with Softmax activation function from 32 to the number of dataset classes neurons.

3.3.3. Transformer [6][7]

Built with Encoder and Decoder with d_model (expected feature) set to 32, heads to 4 and 2 hidden layers, followed by Linear layer with Softmax activation function from d_model (32) to the number of dataset classes neurons.

Positional Encoding - contains embedding layer of vocabulary size followed by the traditional "Vaswani et al" function for constant position-specific values

$$\begin{cases} PE_{pos,2i} = \sin\left(\dfrac{pos}{10000^{\left(\frac{2i}{\mathrm{d_{model}}}\right)}}\right) \\ PE_{pos,2i+1} = \cos\left(\dfrac{pos}{10000^{\left(\frac{2i}{\mathrm{d_{model}}}\right)}}\right) \end{cases}$$

where i refers to the position along the embedding vector and pos refers to the order of the word in the sentence.

Encoder – Contains 2 consecutive Norm layers, followed by a Multi-Head-Attention layer and a Feedforward layer.

Decoder – Contains 3 consecutive Norm layers, followed by 2 consecutive Multi-Head-Attention layers and a Feedforward layer.

4. Evaluation – Metrics

To evaluate results, we chose Accuracy and F1-score metrics. Accuracy is the measure of all the correctly identified cases. Eventually, the task is classification, so we are interested in classifying correctly. F1-score is the harmonic mean of Precision and Recall, therefore gives a better measure of the incorrectly classified cases than the Accuracy metric. We used F1-score to determine how balance our models preform.

5. Experiments

In the proposed approach, we studied the effect of using POS tags on datasets for comparing the various data preprocessing tasks described in the previous part including (1) upos, (2) upos and filter, (3) upos, filter and extend, (4) bigram[1], (5) bigram[1] and filter, (6) bigram[1] and extend, (7) bigram[1], filter and extend.

We tested our experiments with the 3 models. As evident from Table 1, the comparison results on the Accuracy score considering some feature selection with 2 out of 3 models could increase performance.

1. 'bigram' (Tag preprocessing) use UPOS tags. The different between "(2) upos and filter" and "(5) bigram and filter" for example is only with the use of "Bigram Interpolation" or not.

Model			DAN			BiLSTM			Transformer		
			Number of phrases used								
Prepro-cessing method	Dataset	Metrics	15000	5000	1000	15000	5000	1000	15000	5000	1000
original	IMDB	Acc	0.839	0.83	0.855	0.751	0.65	0.6	0.756	0.672	0.75
		F1	0.839	0.835	0.853	0.749	0.645	0.65	0.861	0.865	0.857
	News	Acc	0.858	0.782	0.805	0.6	0.51	0.45	0.841	0.792	0.688
		F1	0.64	0.78	0.812	0.76	0.66	0.55	0.643	0.746	0.56
upos	IMDB	Acc	0.628	0.612	0.655	0.584	0.59	0.58	0.752	0.5	0.5
		F1	0.626	0.608	0.655	0.605	0.63	0.61	0.859	0.5	0.5
	News	Acc	0.493	0.418	0.47	0.363	0.45	0.41	0.714	0.65	0.643
		F1	0.349	0.407	0.462	0.389	0.505	0.51	0.48	0.45	0.4
upos-filter	IMDB	Acc	0.554	0.549	0.595	0.54	0.56	0.56	0.751	**0.753**	0.764
		F1	0.554	0.533	0.566	0.55	0.56	0.57	0.858	0.859	**0.866**
	News	Acc	0/425	0.315	0.32	0.361	0.31	0.31	0.679	0.65	0.63
		F1	0.323	0.309	0.32	0.23	0.35	0.41	0.43	0.44	0.36
upos-filter-extend	IMDB	Acc	0.555	0.566	0.52	0.73	**0.69**	0.59	0.751	**0.758**	0.76
		F1	0.555	0.566	0.516	0.74	**0.705**	0.61	0.858	0.863	**0.863**
	News	Acc	0.377	0.268	0.215	0.455	0.38	0.32	0.772	0.66	0.64
		F1	0.278	0.216	0.176	0.5	0.45	0.39	0.555	0.51	0.41
bigram	IMDB	Acc	0.622	0.607	0.64	**0.857**	**0.75**	0.635	0.756	**0.754**	0.75
		F1	0.621	0.602	0.645	**0.852**	**0.745**	0.645	0.861	0.86	0.857
	News	Acc	0.492	0.417	0.48	0.493	0.51	**0.49**	0.71	0.68	0.65
		F1	0.342	0.412	0.478	0.6	0.61	**0.6**	0.48	0.55	0.45
bigram-filter	IMDB	Acc	0.555	0.552	0.595	**0.96**	**0.96**	**0.925**	0.753	**0.752**	0.756
		F1	0.552	0.534	0.579	**0.96**	**0.96**	**0.925**	0.858	0.862	**0.861**
	News	Acc	0.417	0.323	0.3	0.373	0.39	0.39	0.683	0.637	0.61
		F1	0.323	0.316	0.297	0.245	0.44	0.45	0.44	0.467	0.4
bigram-extend	IMDB	Acc	0.621	0.62	0.605	0.7	**0.67**	0.55	0.751	**0.751**	0.5
		F1	0.621	0.621	0.602	0.72	**0.66**	0.6	0.858	0.858	0.5
	News	Acc	0.442	0.389	0.44	0.43	0.45	0.41	0.79	0.688	0.61
		F1	0.442	0.384	0.43	0.44	0.56	0.49	0.58	0.55	0.638
bigram-filter-extend	IMDB	Acc	0.553	0.561	0.56	0.705	0.62	0.55	0.751	**0.755**	0.756
		F1	0.552	0.56	0.552	0.703	0.62	0.54	0.858	0.86	**0.861**
	News	Acc	0.378	0.271	0.25	0.41	0.29	0.22	0.77	0.645	0.63
		F1	0.378	0.215	0.222	0.41	0.34	0.29	0.568	0.515	0.41

Table 1. Experiments results - comparing Accuracy and F1-score of the 3 models (IMDB and News datasets) with chosen POS tags features.

6. Conclusions

Looking on POS tags and extract POS tags patterns from dataset phrases can improve classification results for some models. From the experiments results, we see a big improvement with the binary-class dataset over the multi-class dataset. One possible explanation for that might be that in the News headline dataset, in addition to it's being a multi-class (more difficult task), each phrase in the data was considerably shorter than the ones in the IMDB reviews dataset. This may have affected the overall performance of extracting POS tags patterns.

By examining the F1-score (relative to the Accuracy) from the results, we can tell that the model's performance was balanced which is the desirable behavior.

Regarding the second assumption in the beginning of this paper, we see that with the BiLSTM model, with bigram-filter preprocessing we achieve better performance than the original data even with significantly small dataset.

Lastly, at least with 2 out of the 3 models, we see that the bigram and bigram-filter preprocessing methods achieved better results than other approaches. It makes sense that these methods worked. The bigram highlights common word pairs over less common pairs, and the filter may reduce noise over the dataset (also explains why it wasn't effective with the News dataset – it didn't have much noise to reduce).

References

[1] D. Raihan. Deep Learning Techniques for Text Classification. Towards Data Science. April 2021.

[2] S. Chotirat and P. Meesad. Part-of-Speech tagging enhancement to natural language processing for Thai wh-question classification with deep learning. Heliyon, volume.7, issue 10. October 2021.

[3] M. Iyyer. V. Manjunatha. J. Boyd-Graber. H. Daume´ III. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. Association for Computational Linguistics. July 2015.

[4] R. Cheng. LSTM Text Classification Using Pytorch. Towards Data Science. July 2020.

[5] A. NS. Multiclass Text Classification using LSTM in Pytorch. Toward Data Science. April 2020.

[6] S. Lynn-Evans. How to code The Transformer in Pytorch. Toward Data Science. September 2018.

[7] A. NS. Multiclass Text Classification using LSTM in Pytorch. Toward Data Science. April 2020.