NLP@UIT at SemEval-2019 Task 4: The Paparazzo Hyperpartisan News Detector

Duc-Vu Nguyen$^\diamond$**, Dang Van Thin**$^\diamond$**, and Ngan Luu-Thuy Nguyen**$^\heartsuit$
$^\diamond$Multimedia Communications Laboratory
$^\heartsuit$Faculty of Computer Science
University of Information Technology, Vietnam National University Ho Chi Minh City, Vietnam
{vund,thindv,ngannlt}@uit.edu.vn

Abstract

This paper describes the system of NLP@UIT that participated in Task 4 of SemEval-2019. We developed a system that predicts whether an English news article follows a hyperpartisan argumentation. Paparazzo is the name of our system and is also the code name of our team in Task 4 of SemEval-2019. The Paparazzo system, in which we use tri-grams of words and $hepta$-grams of characters, officially ranks thirteen with an accuracy of 0.747. Another system of ours, which utilizes tri-grams of words, tri-grams of characters, tri-grams of part-of-speech, syntactic dependency sub-trees, and named-entity recognition tags, achieved an accuracy of 0.787 and is proposed after the deadline of Task 4.

1 Introduction

Fake news is a noteworthy term in recent years. The rise of users and rapid spread information on social networking have made on automatic controlling of fake news more difficult. Fake news articles are typically extremely one-sided (hyperpartisan), inflammatory, emotional, and often riddled with untruths (Potthast et al., 2018). The influence of misinformation varies depending on the style it is written in. For example, sarcasm in a sports news article will have less of an impact than news written in the hyper-partisan argumentation style, which can sway voter decision in an election.

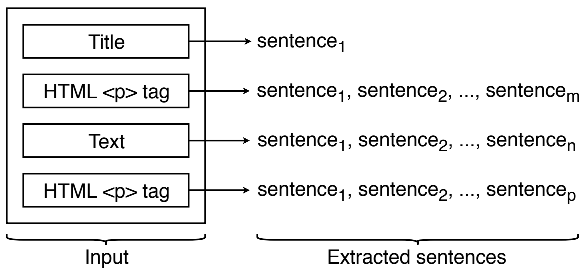
Hyperpartisan detection in news articles is one of the ways to control fake news on the media and public. Kiesel et al. (2019) provided a new task, which they name "Hyperpartisan News Detection," to decide whether a news article text follows a hyperpartisan argumentation. We approach this task following traditional text classification by extracting style features. The bag-of-words model is the way of text representation and is applied to sentiment analysis effectively (Pang et al., 2002). Matsumoto et al. (2005) applied text mining techniques on dependency sub-trees as features for sentiment analysis at the document level. Our results show that n-grams of words and dependency sub-trees features from sentences of the document have certain impacts on the performance of the classifier. The details of the features in our systems and the results are described in Section 3 and Section 4.

2 Task Description

SemEval2019 Task 4 has only one task, in which participants are required to build the systems for hyperpartisan news detection. The task is to predict which category ("hyper-partisan" vs. "not hyper-partisan") an argumentation belongs to when given the news article in English (Kiesel et al., 2019). There are 645 articles in the for-ranking training set, and 628 articles in the for-ranking testing set (all of them are labeled through crowdsourcing on an article basis). Besides, the organizers of this task provided another dataset with the training/validation/testing set having 600,000/150,000/4,000 articles (all of them are labeled in accordance with the judgment of the publisher). The organizers use the *accuracy* as the main metric in the *for-ranking testing set* to evaluate the performance of the participants' systems. All submissions and results are validated by the organizers via the evaluation service TIRA (Potthast et al., 2019).

3 System Description

In this section, we describe the major stages we followed, as well as the prediction models we utilized in our detection system.



Title
HTML <p> tag
Text
HTML <p> tag
sentence₁
sentence₁, sentence₂, ..., sentenceₘ
sentence₁, sentence₂, ..., sentenceₙ
sentence₁, sentence₂, ..., sentenceₚ
Input
Extracted sentences

Figure 1: Diagram of data preprocessing.

3.1 Data Preprocessing

Data preprocessing of the given input is the important phase for every task related to natural language processing. The input of SemEval-2019 Task 4 is an XML file, containing a title and many paragraphs in the body text. Paragraph segmentation is based on the HTML <p> tag because the <p> tag defines a paragraph. While many paragraphs are wrapped by the <p> tag, some are not. Observation of some inputs from the dataset shows that paragraphs that are not wrapped by any HTML tag may contain "noise," such as advertisements and the browser's error messages. On the other hand, texts displayed in HTML <p> tags can also contain "noise," such as notifications for redirecting a page (e.g., "Click here to..."). We did not handle the aforementioned noises in our experiment.

The next step after paragraph segmentation is sentence segmentation. During this process, we used spaCy tool (Honnibal and Montani, 2017) to extract sentences from titles, HTML <p> tags, and paragraphs not wrapped in any HTML tag of input (as we can see the diagram in Figure 1).

3.2 Features Extraction

3.2.1 N-grams of words

Before extracting n-grams of words, we break the sentences into words in three ways:

1. WS_1: The sentence is split by space/multi-space into tokens.
2. WS_2: The sentence is split by space/multi-space into tokens. After that, we discard tokens which are punctuations or English stop-words.
3. WS_3: The sentence is segmented into words. And then, we lemmatize words into lemmas. All is done by using the spaCy tool (Honnibal and Montani, 2017).

After splitting/segmenting the sentence into tokens/words, we put tokens/words are all in lowercase and implement extracting n-grams of them. The specific values of n for prediction models are mentioned in section 3.3.

3.2.2 N-grams of characters

Extracting n-grams of words is effective for text classification that is word-based representation, but this approach requires reliable tokenizers for breaking the sentences into words. Experiments on unsolicited e-mail messages (spam) and a variety of evaluation measures, Kanaris et al. (2007) show that n-grams of characters are more reliable to classify texts than n-grams of words. Potthast et al. (2018) show how a style analysis can distinguish hyperpartisan news from the mainstream, and they also use tri-grams of characters as features for the classifier in their experiments. As we described in section 3.1, unfortunately, the input of SemEval-2019 Task 4 contains a small number of strange n-grams of characters towards the tokenizers. Therefore, we decide to use n-grams of characters as the features in our system. We use the sentence with all of its tokens being rejoined after the segmentation in WS_1 (we described in section 3.2.1) with character space for extracting n-grams of characters. In our experiments, the value of n ranging from 2 to 7 and the specific values for prediction models are mentioned in section 3.3.

3.2.3 N-grams of part-of-speech

Argamon et al. (2003) found that n-grams of part-of-speech can efficiently capture syntactical information and gender-based style of the writer. Potthast et al. (2018) used tri-grams of part-of-speech to make a comparative style analysis of hyperpartisan (extremely one-sided) news and fake news. Although the efficacy of using n-grams of part-of-speech on fake news was not examined in their study, we decided to experiment by using n-grams of part-of-speech as features for hyper-partisan news detection. We used the spaCy tool (Honnibal and Montani, 2017) for part-of-speech tagging and extract tri-grams of part-of-speech as features.

3.2.4 Sub-trees of dependency tree

In our experiment, dependency parsing involves extracting from a dependency tree a dependency sub-tree, which is defined by Matsumoto et al. (2005) as "a tree obtained by removing zero or more nodes and branches from the original de-

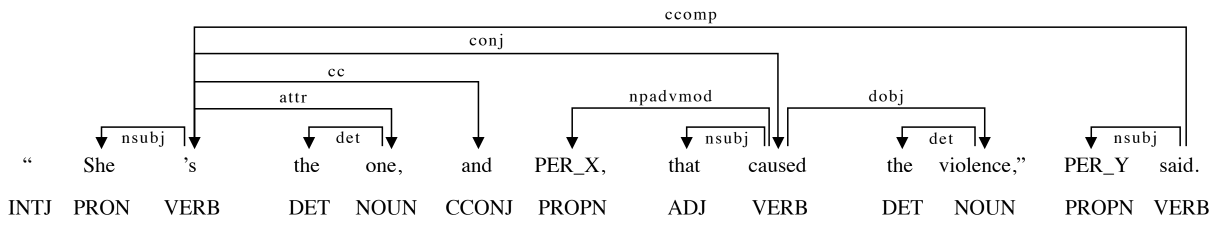


Figure 2: Visualization of the dependency tree of the sentence within the bracket (“She’s the one, and PER_X, that caused the violence,” PER_Y said.). This sentence is taken from a news article of the training for-ranking training set which is mentioned in Section 2. The person’s name is replaced by PER_{uppercase letter} in this example (we did not do that in our experiment).

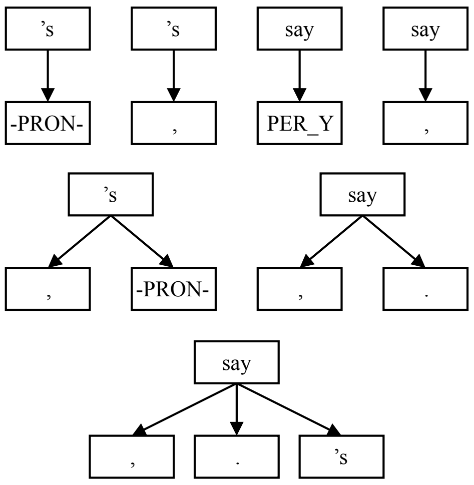


Figure 3: Visualization of seven sub-trees which are extracted from the dependency tree in Figure 2. There are four sub-trees with two nodes, two sub-trees with three nodes, one sub-tree with four nodes in the current figure. All words in this example are lemmatized.

pendency tree.” Figure 2 illustrates a dependency tree of a sentence parsed with spaCy tool (Honnibal and Montani, 2017), and its shortcoming that shows the double quotation mark on the left does not have any child node or parent node. This shortcoming, however, did not affect the extraction of all sub-trees of the dependency tree, but we resolved this issue by considering each group of subtrees as one connected component, and the dependency tree as a graph that can contain more than one connected component. Figure 3, the number of nodes in a sub-tree can range from 2 to 4, and NetworkX tool (developed by Hagberg et al. (2008)) was used to extract all the sub-trees of the original dependency tree as one connected component for each node. All words at each node of sub-trees are lemmatized in our experiment. As we can see in Figure 3, some sub-trees can capture words which are not located close to each other.

3.2.5 Named-entity recognition tags

Characteristics of the input of SemEval-2019 Task 4 contains names of people, names of organizations. Therefore, we decided to use mentions of specific terms in named-entity recognition as features. In our experiments, a feature is represented by concatenating a mention and a named-entity recognition tag. We used the spaCy tool (Honnibal and Montani, 2017) for the named-entity recognition task.

3.3 Prediction Models

In this section, we describe the four models which we have summited to the organizers. In all models, we used linear SVM (SGDClassifier from Scikit-learn (Pedregosa et al., 2011)) as the classifier, and the loss function which is hinge loss with L2 regularization. In all models, we did not run validation experiments for turning regularization term α of all models. We used just the default value of $\alpha = 0.0001$ following SGDClassifier from Scikit-learn (Pedregosa et al., 2011). Most importantly, we concatenated different count vectors by way of extracting features described in section 3.2, to obtain the input representation of the model.

3.3.1 First model

This model uses *tri*-grams of words which are split from the text of the article (we did not segment the text into sentences) the way described in WS$_1$ in section 3.2.1). Besides, the first model uses *hepta*-grams of characters from the text of the article as features. We discarded the title when extracting the features for the first model, and we do not distinguish between texts with the HTML <p> tag wrapping and those without (as mentioned in section 3.1).

3.3.2 Second model

We extracted bi-grams of characters from the body text regardless of whether the text is wrapped by the HTML <p> tag or not, and for the title, we followed the way mentioned in WS_2 in section 3.2.1) to extract its bi-grams of words. Additionally, we extracted all mentions of named-entity recognition from all sentences of the article, and we distinguished between features from the title and those from the body text.

3.3.3 Third model

This model shares similar features with the second one, except for our extraction of the dependency sub-trees.

3.3.4 Fourth model

In this model, the title, the text with the HTML <p> wrapping, and those without are distinguished. All sentences are segmented to tokens in the same way described in WS_3 in section 3.2.1). We used tri-grams of words, tri-grams of characters, tri-grams of part-of-speech, syntactic dependency sub-trees, and named-entity recognition tags to extract the text before performing the TF-IDF transformation with Scikit-learn tool (developed by Pedregosa et al. (2011)) on the combined features with min_df at 0.05 and max_df at 0.95.

4 Results

	Accuracy	Precision	Recall	F1
First model[1]	0.747	0.754	0.732	0.743
Second model	0.685	0.687	0.678	0.683
Third model	0.707	0.666	0.831	0.739
Fourth model[2]	0.787	0.796	0.771	0.783

Table 1: Metric summary of fully trained models on the official test dataset.

We did not use the training dataset of 600,000 articles for training all the models in our experiments. The result (Table 1) shows a decrease in performance of the second and the third models when the n-grams of words were not used as features. The accuracy of the third model, however, increased by 2% compared with the second model when the extra dependency sub-trees were used as features. On the other hand, the fourth model achieved the highest accuracy, up to 0.787. This accuracy level, however, is still lower than that achieved via deep learning techniques, such as the Convolutional neural network and pre-trained ELMo representations, employed by "Bertha von Suttner" team who were ranked first in SemEval-2019 Task 4.

[1]The first model officially ranks thirteen in Sem-Eval 2019 Task 4.

[2]The fourth model is proposed after the deadline of Sem-Eval 2019 Task 4.

5 Conclusion

Our major contribution to SemEval-2019 Task 4 is that using n-grams of words and dependency sub-trees as features for extracting has a positive impact on the performance of the classifier: In our experiment, we were able to achieve the accuracy of 0.787 with the proposed model that uses tri-grams of words, tri-grams of characters, tri-grams of part-of-speech, syntactic dependency sub-trees, and named-entity recognition tags. That model can also capture words which are not located close to each other through dependency sub-trees. The disadvantages of our models, however, are that extraction of dependency sub-trees is a time-consuming process, and the relations between sentences of the articles are not represented.

References

Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. 2003. Gender, genre, and writing style in formal written texts. *Text*, 23:321–346.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Ioannis Kanaris, Konstantinos Kanaris, Ioannis Houvardas, and Efstathios Stamatatos. 2007. Words versus Character n-Grams for Anti-Spam Filtering. *International Journal on Artificial Intelligence Tools*, 16:1047–1067.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. SemEval-2019 Task 4: Hyperpartisan News Detection. In *Proceedings of The 13th International Workshop on Semantic Evaluation (SemEval 2019)*. Association for Computational Linguistics.

Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura. 2005. Sentiment Classification Using Word Sub-sequences and Dependency Sub-trees. In

Advances in Knowledge Discovery and Data Mining, pages 301–311, Berlin, Heidelberg. Springer Berlin Heidelberg.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86. Association for Computational Linguistics.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830.

Martin Potthast, Tim Gollub, Matti Wiegmann, and Benno Stein. 2019. TIRA Integrated Research Architecture. In Nicola Ferro and Carol Peters, editors, *Information Retrieval Evaluation in a Changing World - Lessons Learned from 20 Years of CLEF*. Springer.

Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. 2018. A Stylometric Inquiry into Hyperpartisan and Fake News. In *56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 231–240. Association for Computational Linguistics.