June 10, 2024

Natural Language Processing

Project 1.4: Oppositional thinking analysis

Organization

- Please read the project description carefully. All the details are presented in this document.
- Submit your solution no later than June 25th, 23:55. Submit your solution using the corresponding ISIS section, where you have downloaded this document.
- You can make unlimited re-submissions until the deadline, but the latest received submission will be graded only.
- Submit your solution as a zip file with the following name format: 'NLP_project1_4_[Your Names separated by underscores].zip' containing your codes and report files.
- For those who decided to work on the project in a group, one submission of the solution would be enough.
- For each task, you see a percent that show the task's score in the whole project.
- Please use Python 3 for coding.
    - You are free to use any module/library for different tasks.
- Please put comments on your code, wherever you think it would help to improve the readability and understandability of your code.
    - You can submit your codes as a .py file or Jupyter notebook
- You should submit a short report (e.g., 4 pages), describing your approach to solve different tasks and provide the obtained results (e.g., the evaluation result of your models).
    - For the report, please use ACM proceedings template from here (https://www.acm.org/publications/proceedings-template). The office word and latex templates are provided in the page.
    - The report should be between 3 to 5 pages.
- There is a bonus task in the end of the document, as the name implies it is not a mandatory task, but you can take your time and solve it to get more scores.

Plagiarism Statement

Your project, includes the report and the code, will be checked against other submissions in the class. We trust you all to submit your own work only. Copying someone else's code and report and submit it with minor changes, will be treated as plagiarism.

If you have any further questions, please write to: salar.mohtaj@tu-berlin.de

Introduction:

Oppositional thinking analysis can be formulated as a text classification task (like spam filtering), where conspiracy theories and critical thinking narratives represented as a category. This distinction is vital because labeling a message as conspiratorial when it is only oppositional could drive those who were simply asking questions into the arms of the conspiracy communities. This project is a binary classification task differentiating between (1) critical messages that question major decisions in the public health domain, but do not promote a conspiracist mentality; and (2) messages that view the pandemic or public health decisions because of a malevolent conspiracy by secret, influential groups.

Data:

Each instance in the Json dataset corresponds to a dictionary that contains tokenized text, the binary category, and the span annotations. For this project, you can ignore the span annotations. So, the text and the category are the input and output of your model, respectively. The data includes 4000 instances of spam and normal (ham) email texts. Please apply the following tasks on the dataset.

Tasks:

Task 1: Extract insights from data (15%)

For this task you should extract some insights (i.e., some statistics and graphs) from the provided data. Here are some ideas:

- It could be a graph compared the length of text in different categories.
- Number of unique words in each category and also in the whole dataset
- …

Please don't limit yourself to these two examples and try to summarize the data with numbers and graphs. Please highlight some of the most important findings in your report.

Task 2: Pre-processing (15%)

In this task, first, apply all the necessary pre-processing steps that you think they would help to better prepare your data for the next steps. You don't have to apply all the pre-processing tasks which are covered in the course. Regarding the report, you should briefly mention it in your report that why you decided to apply the chosen pre-processing steps (and why not the others).

Task 3: Text classification (50%)

In this task you should do the following sub-tasks:

- Split data into train and test sets. Use 20% of the data as the test set. Make sure to under or over-sample in case of imbalance in classes.
- Train a naïve Bayes model on the training part and test it, using the test set.

    - Compare the impact of different vectorization models (e.g., count vectorizer, TF-IDF, and …) on the final performance of your naïve Bayes model.
    - Compare the impact of different pre-processing pipelines (e.g., with and without stop words, stemming, and …) on the final performance of your naïve Bayes model.
    - Perform error analysis on the model's prediction. In other words, analyze errors that have been made by the model and describe why your model couldn't work well in case of these errors.
- Train a feed-forward neural network model and report its performance (F1 score) on test data.
    - Again, compare the impact of different vectorization approaches on the final performance of your model.
    - Again, Compare the impact of different pre-processing pipelines (e.g., with and without stop words, stemming, and …) on the final performance of your model.
    - Perform error analysis on the model's prediction.
- Compare the performance of your naïve Bayes model with the achieved results from the feed-forward model. What can you conclude from the differences between the performance of the two models?

Please report all the achieved results with either model in your report document. Moreover, describe the hyper-parameters of your neural network model in the report.

Task 4: PMI based word similarity (20%)

Convert words to vectors using PMI (window size = 2). For 10 random words in the data, provide the most similar words based on the computed word-word pointwise mutual information matrix. Please highlight the 10 randomly chosen terms and their most similar words in your report. (Please don't use available packages to solve the task)

Bonus Task: Textual similarity (+10%)

As the bonus task, you should fine-tune two pre-trained models on the provided spam detection dataset. You can choose between the wide range of pre-trained models (e.g., BERT, RoBERTa, and so on). It is very important to be careful about the hyper-parameters and fine-tune the models as accurate as possible without too much overwriting of the current weights. Please describe your selected models and the training process in details in the report.