

CA – Assignment 2: Argument Mining

- Group: **FakeNews**
- Group members:
 - Adnan Manzoor
 - Sajjad Pervaiz
 - Kevin Taylor
 - Christoph Schäfer

Structure

```
.
├── argument-mining-assignment
│   ├── Documentation.pdf
│   ├── README
│   ├── requirements.txt
│   └── code
│       ├── conf_bias_evaluation.py
│       ├── model.py
│       └── data
│           ├── essay_corpus.json
│           ├── predictions.json
│           ├── sample_prediction.json
│           └── train-test-split.csv
```

Scripts

- **essay_corpus.json**: Data corpus created in Data Acquisition assignment.
- **model.py**: The ML model that we use for generating predictions.
- **conf_bias_evaluation.py**: Script to evaluate the **F1** score of the **ML** model.

How to run the scripts

- On a venv install the requirements specified in **requirements.txt**
- Make sure you have the same directory structure as above otherwise adjust the paths in the **model.py** script accordingly.
- Run **model.py** to generate the predictions in **data/** directory with name **predictions.json**
- Run **conf_bias_evaluation** script with the filepath to the **data**.

Model Selection

We selected SVM as the model for mainly these 3 reasons:

- In the paper **Recognizing the Absence of Opposing Arguments in Persuasive Essays** by **Stab** and **Gurevych** they used SVM for doing the identical task and the scores were pretty high.

- We read the paper titled: [Text categorization with Support Vector Machines: Learning with many relevant features](#)[1]. In the paper the author explores why SVM are suited for text-classification tasks.
- Even then we tried multiple other models: Logistic Regression, Naive Bayes, Random Forrest and the performance of SVM was the best.
- We used Linear SVM because text classification is mostly a linearly seperable problem[1] and using kernels(rbf, poly) to map the data to a higher dimensional space did not really improve the performance in this case.
- We used GridSearch for hyperparameter tuning.

[1] : <https://link.springer.com/chapter/10.1007/BFb0026683>

Feature Selection

For features we used the following:

- **n-grams + TF-IDF** : In the range of 1-3 so that unique keywords and phrases are identified and given more importance. We used `TfidfVectorizer` of scikit-learn library to create n-grams.
- **Adversative transitional** phrases (adv): to identify conflict, contradiction, concession, and dismissal in the text which indicate presence of opposing argument. We first used all 47 adversative transitional phrases that are grouped in the following categories:
 - concession (18)
 - conflict (12)
 - dismissal (9)
 - emphasis (5)
 - replacement (3)

For each of these categories, we added features for the upper and the lower case as well as for their presence in the surrounding paragraph (introduction+conclusion or in the body) but the results were even worse than just the approach with only **n-grams + TF-IDF**. So we did a deeper analysis of training data by finding the occurrences of phrases in both the **true** and **false** classes (using `adv_trans_text_analysis` function in the code). As a result, we detected that the concession and conflict phrases are the best indicator for the **opposing arguments** in our training data. Consequently, we just used 15 phrases of these categories to get an F1 score of **.754**.

- We used 10-fold cross-validation on the training data for regularization.