First Homework Report
BLM6144 Computational Linguistics

Rayene Bech
18011115
03/04/2023

Content

- **Models**:
  - Latent Semantic Analysis (LSA)
  - Latent Dirichlet Allocation (LDA)
  - Term frequency Inverse Document Frequency (TF-IDF)
  - TF-IDF character grams
  - FastText
  - BERT

- **Combination of embeddings:**
  - First method: concatenate words (BERT, FastText), sum for sentence.
  - Second method: sum words (BERT, FastText), sum for sentence.
  - Third method: compute each sentence representation separately (sum words), then combine the two sentence representations by sum.

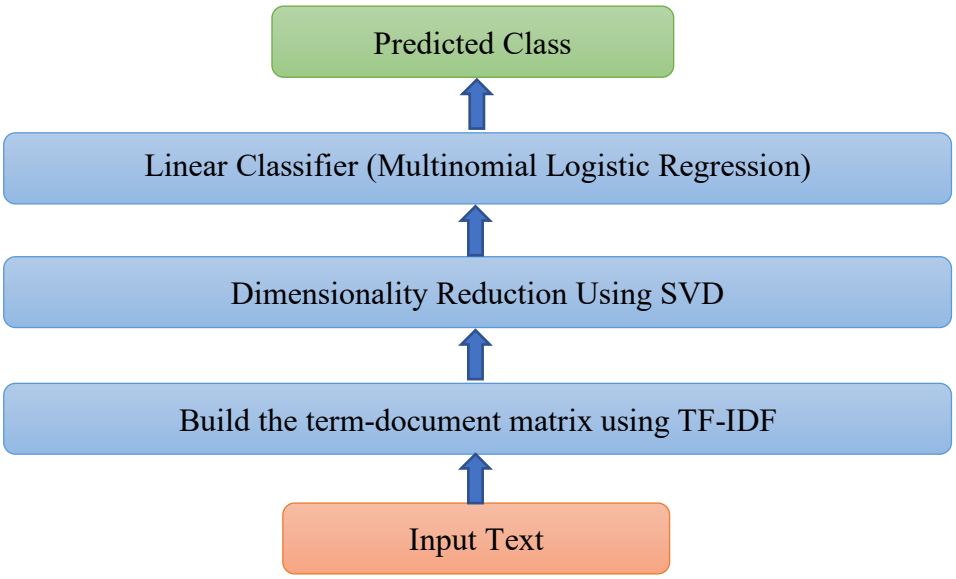
- **Experiments**
  - Topic Modeling
  - IMDB Movies Reviews Analysis
  - Twitter Sentiment Analysis

- **Conclusion**

Models

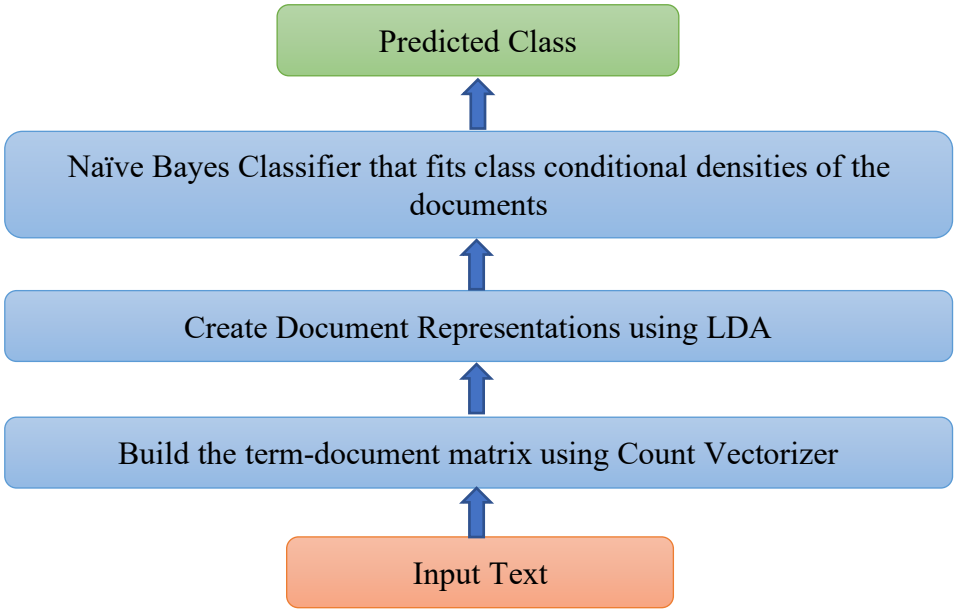
1- Latent Semantic Analysis (LSA):

The objective of LSA is reducing dimension for classification. The idea is that words will occurs in similar pieces of text if they have similar meaning.
The model's architecture is:

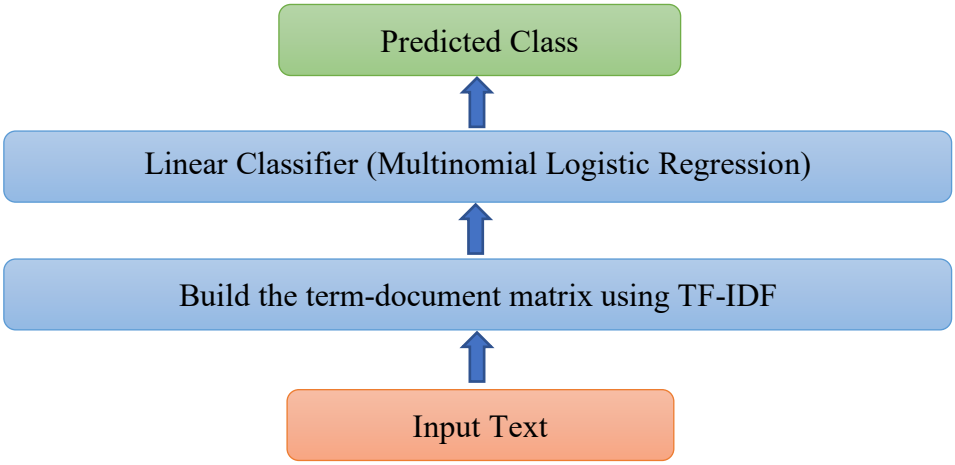


2- Latent Dirichlet Allocation (LDA)

In LDA, documents are represented as a mixture of topics and a topic is a bunch of words. Those topics reside within a hidden (latent) layer.

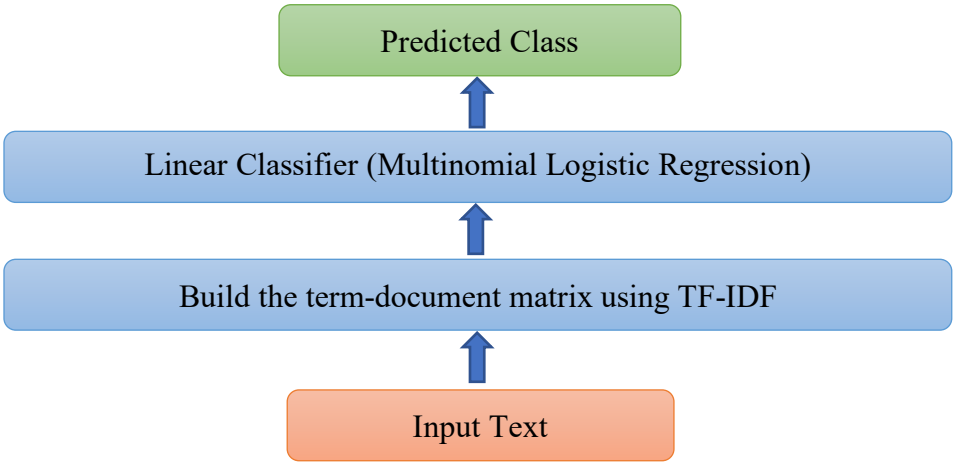


3- Term frequency Inverse Document Frequency (TF-IDF)



4- TF-IDF Character Gram

In this case we applied TF-IDF on character level. Instead of computing the occurrences of the words, the occurrences of the characters are computed. by setting the range (2,6) which means using from the bigrams, trigrams, 4-grams, 5-grams and 6-grams.



5- FastText Model:

A very fast model for computing word representations. Each word is represented as a bag of character n-grams in addition to the word itself. For example, for the word “apple”, with n = 3, the fastText representations for the character n-grams is <ap, app, ppl, ple, le>. This allows to capture meaning for suffixes/prefixes. The n-gram featuers are averaged to form the hidden layer.

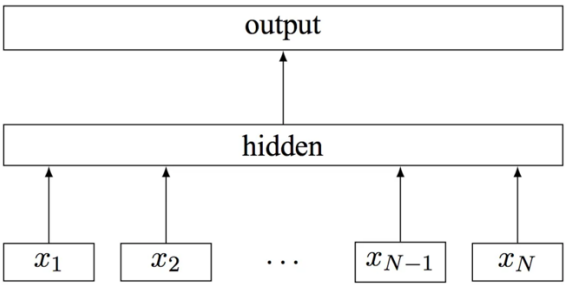
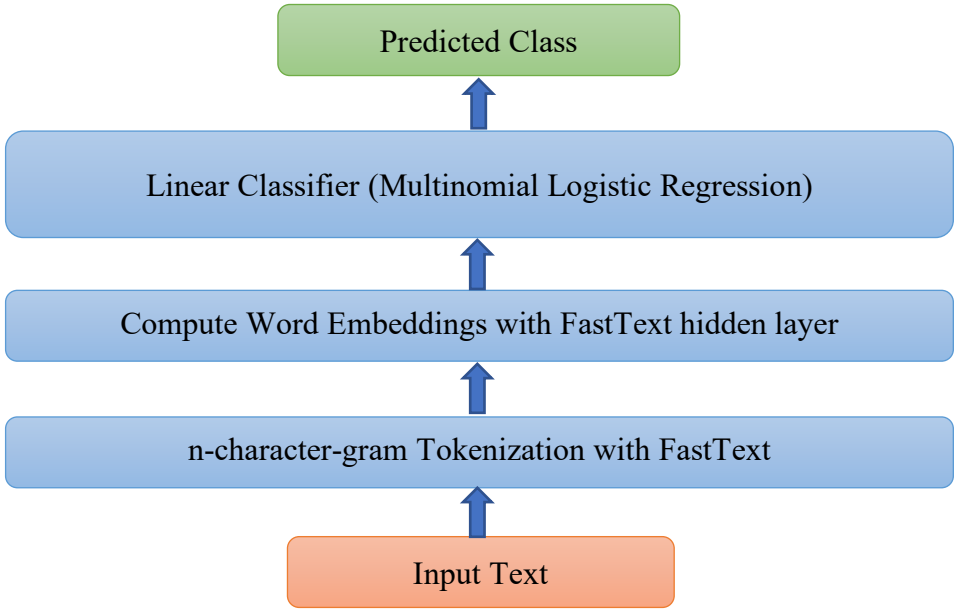


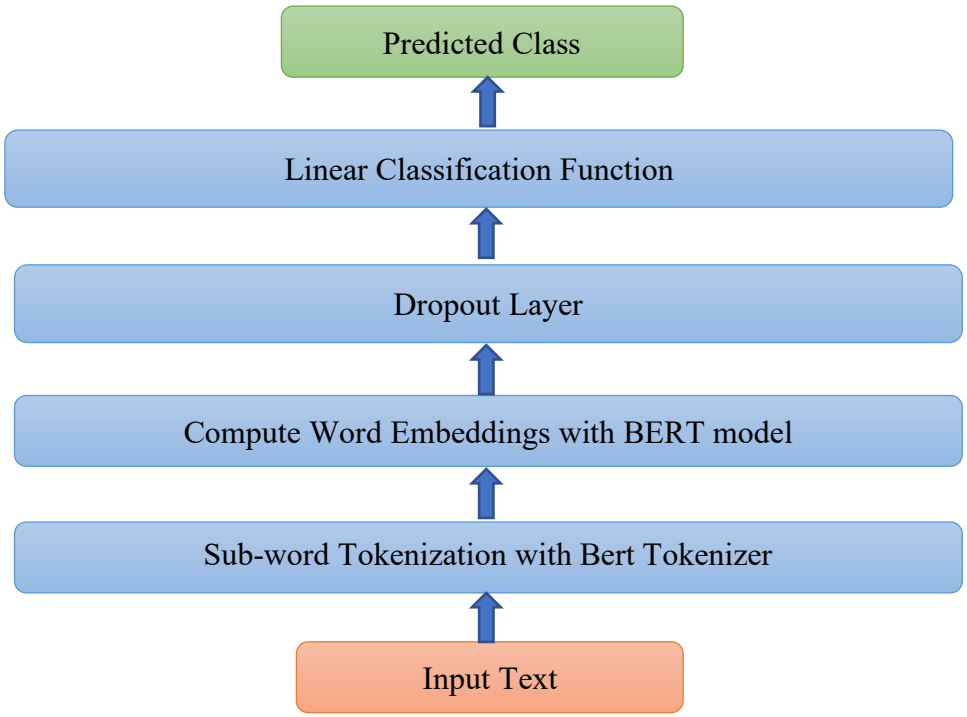
Figure 1: Model architecture of `fastText` for a sentence with N ngram features $x_1, \ldots, x_N$. The features are embedded and averaged to form the hidden variable.

Image taken from the original paper



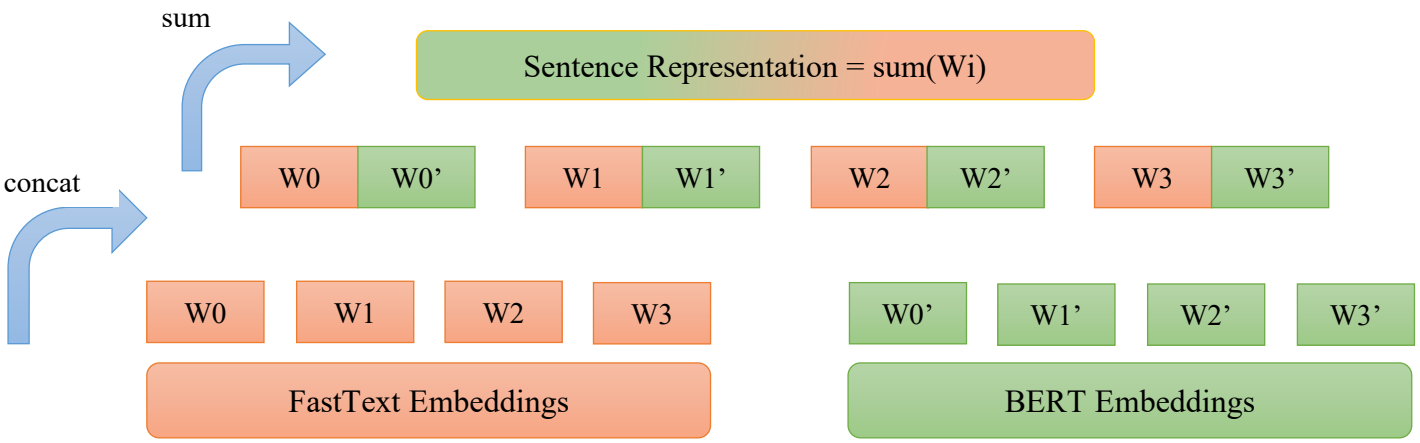
6- BERT Model:

Bidirectional Encoder Representations from Transformers (BERT) is based on the self-attention mechanism of the Transformers models. Every token in the input sequence is related to other tokens. The powerful mathematics behind this concept enables the model to capture context-based features while generating word embeddings.
In this project, the "bert-base-uncased" model was used in all experiments to extract the word embeddings. The embeddings are then fed to a linear classifier to predict the classes. Also a dropout layer was added between the embeddings layer and the classifier function. The model's architecture can be visualized as follows:

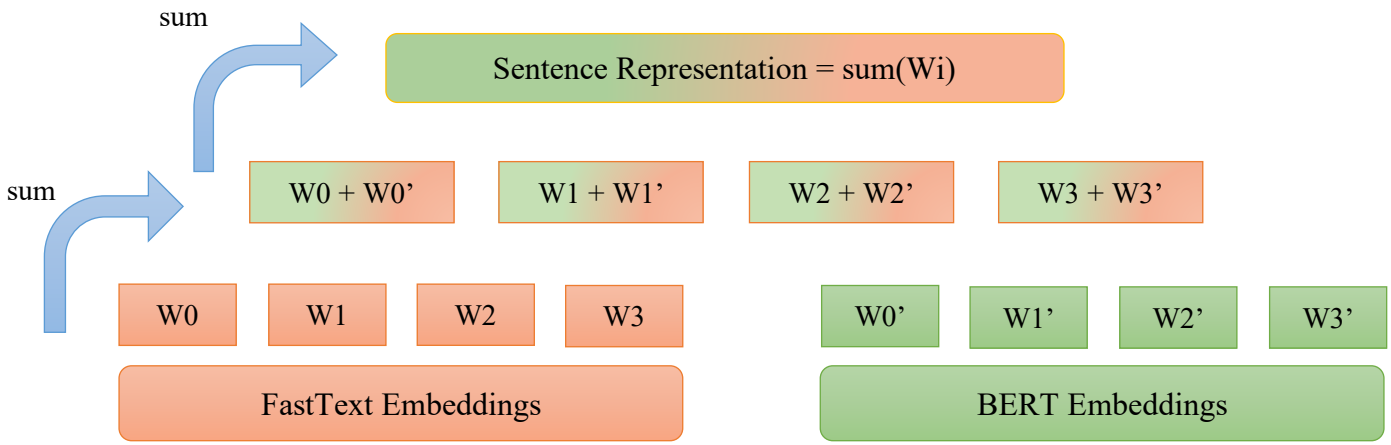


Combination of the embeddings

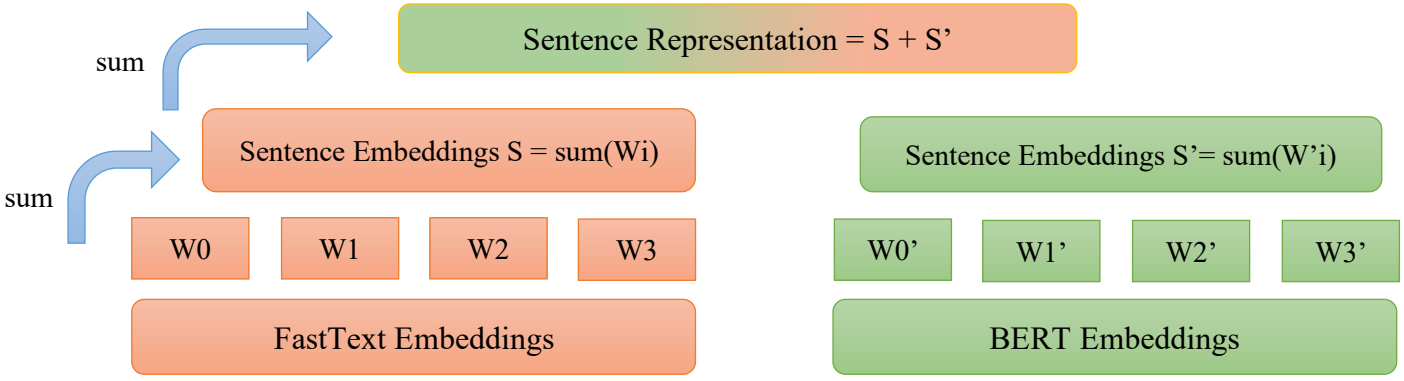
- **First method: concatenate words (BERT, FastText), sum for sentence.**
  - Compute BERT words embeddings for each word separately.
  - Compute FastText words embeddings for each word separately.
  - Combine embeddings for each word separately by concatenating BERT word embeddings with FastText word embeddings.
  - At this stage we have a combined representation for each single word. To compute the sentence embedding, we sum the embeddings of all the words



- **Second method: sum words (BERT, FastText), sum for sentence:**
  - Compute BERT words embeddings for each word separately.
  - Compute FastText words embeddings for each word separately.
  - Combine embeddings for each word separately by summing BERT word embeddings with FastText word embeddings.
  - At this stage we have a combined representation for each single word. To compute the sentence embedding, we sum the embeddings of all the words



- **Third method: compute each sentence representation separately (sum words), then combine the two sentence representations by sum.**
  - Compute BERT words embeddings for each word separately.
  - Compute BERT sentence embeddings by summing BERT word embeddings.
  - Compute FastText words embeddings for each word separately.
  - Compute FastText sentence embeddings by summing FastText word embeddings.
  - Combine the two Sentence Embeddings by summing BERT sentence embeddings with FastText sentence embeddings.



Notes:

- To be able to combine the FastText Embeddings with BERT embeddings, we trained the FastText model to generate vectors of size 768 to match the size of BERT embeddings.
- There are many ways to obtain word embeddings from the BERT model. According to the litrature, the best method is sum the last four vectors. In all the experiements, where BERT is involved, we applied this method.

Experiments

1- Topic Modeling:

This dataset consists of 2225 news texts with five categories.
The dataset was split with 80% training, 10% validation and 10% test. The results of 5-kfold cross validation of all the models are shown below:

sport	511
business	510
politics	417
tech	401
entertainment	386

Model	Precision scores	Recall scores	F1 scores
LSA	95.91% (+/- 0.0219)	96.40% (+/- 0.0185)	96.15%
LDA	96.53% (+/- 0.0303)	96.68% (+/- 0.0257)	96.60%
TF-IDF	97.06 (+/- 0.0133)	97.26% (+/- 0.0162)	97.15%
TF-IDF character grams	97.17% (+/- 0.0206)	97.35 (+/- 0.0185)	97.25%
FastText	95.78 (+/- 0.0225)	95.78 (+/- 0.0225)	95.78%
BERT	99.77% (+/- 0.0067)	99.75% (+/- 0.0080)	99.76%
Combination: concatenate words (BERT, FastText), sum for sentence.	65.67% (+/- 0.0739)	58.74% (+/- 0.0371)	62.01%
Combination: sum words (BERT, FastText), sum for sentence.	67.84% (+/- 0.0399)	56.74% (+/- 0.0415)	61.79%
Compute each sentence representation separately (sum words), then combine the two sentence representations by sum	65.20% (+/- 0.0868)	60.11 (+/- 0.0728)	62.55%

Notes:

- The BERT classifier model outperformed all other models. This is thanks to its ability to capture context-based features which is crucial in tasks such as topic modeling.
- LSA and LDA performance were close to each other. This is because both algorithms work well for topic modeling. Both extract the latent features to classify documents per topics.
- TF-IDF based models surprisingly outperformed the pretrained word embeddings od FastText model. Looking to the literature, many works has proved that TF-IDF performs better in all measures than FastText.
- Combining word embeddings of BERT and FastText performed poorly on this dataset. This could be due to the big difference in the space representations between the two vectors.

2- IMDB Movies Reviews:

This dataset contains 50k movies reviews. For test purposes, a sample was taking from the dataset (10% from the original data). The sample distribution of the labels is shown on the right. The dataset was split with 80% training, 10% validation and 10% test. The results of 5-kfold cross validation of all the models are shown below:

```
negative    2520
positive    2480
```

Model	Precision scores	Recall scores	F1 scores
LSA	78.33% (+/- 0.0248)	78.35% (+/- 0.0246)	78.34%
LDA	66.65% (+/- 0.1203)	66.66% (+/- 0.1199)	66.66%
TF-IDF	84.52% (+/- 0.0341)	84.61% (+/- 0.0332)	84.56%
TF-IDF character grams	84.65% (+/- 0.0313)	84.84% (+/- 0.0341)	84.74%
FastText	82.96% (+/- 0.0431)	82.96% (+/- 0.0431)	82.96%
BERT	91.87% (+/- 0.0846)	91.87% (+/- 0.0851)	91.87%
Combination: concatenate words (BERT, FastText), sum for sentence.	66.24% (+/- 0.1961)	74.57% (+/- 0.0384)	70.16%
Combination: sum words (BERT, FastText), sum for sentence.	60.79% (+/- 0.1282)	73.95% (+/- 0.0457)	66.73%
Compute each sentence representation separately (sum words), then combine the two sentence representations by sum	60.55% (+/- 0.1797)	73.66% (+/- 0.0846)	66.46%

Notes:

- Again, the BERT classifier model outperformed all other models thanks to the transformers architecture.
- LDA performed poorly on this dataset. This could be due to its inability to capture specific topics since the classification objective here is not based on topic categorizations but rather on semantic analysis of the review's sentiment.
- LSA outperformed by a large margin LDA. This could be due to LSA's ability to capture the semantic relationships between words and identify similar words based on their context which is suitable for this task. LDA is more about Topic Modeling.
- An interesting observation is about combining word embeddings of BERT and FastText by concatenating words of each model then summing to get sentence representations. This method outperformed other combination methods.

3- Twitter Sentiment Analysis:

The dataset used was from the "Sentiment 140 Dataset". The original dataset contains around 1.6 million tweets. However only a sample was used in all experiments. The sample distribution of the labels is shown on the right. The dataset was split with 80% training, 10% validation and 10% test. The results of 5-kfold cross validation of all the models are shown below:

```
0    2409
4    2391
```

Model	Precision scores	Recall scores	F1 scores
LSA	60.94% (+/- 0.0559)	60.98% (+/- 0.0560)	60.96%
LDA	54.76% (+/- 0.0673)	54.99% (+/- 0.0701)	54.87%
TF-IDF	72.71% (+/- 0.0286)	72.73% (+/- 0.0289)	72.72%
TF-IDF character grams	73.80% (+/- 0.0281)	73.90% (+/- 0.0285)	73.85%
FastText	69.96% (+/- 0.0412)	69.96% (+/- 0.0412)	69.69%
BERT	83.91% (+/- 0.1011)	83.90% (+/- 0.1019)	83.90%
Combination: concatenate words (BERT, FastText), sum for sentence.	66.82% (+/- 0.0698)	67.54% (+/- 0.0674)	67.18%
Combination: sum words (BERT, FastText), sum for sentence.	65.46% (+/- 0.0764)	66.81% (+/- 0.0420)	66.13%
Compute each sentence representation separately (sum words), then combine the two sentence representations by sum	69.26% (+/- 0.0636)	69.51% (+/- 0.0584)	69.38%

Notes:

- Same observations can be made with regards to BERT, LSA, LDA, TF-IDF and FastText models.
- However, unlike other experiments, fort his dataset, computing sentence representations of each model separately worked well comparing to other methods. This combination had scores very close to the FastText Classifier. A possible reason could be that the combined sentence representations were close to the FastText sentence representations in the space.

Conclusion

- The BERT classifier model outperforms all other models. Being based on transformers architecture, it can generate contextual word embeddings which are the key for a good classification model.

- Topic Modeling using LDA works very well. However, this algorithm does not perform well with classifications tasks such as sentiment analysis. Because LDA is a probabilistic modeling technique that aims to identify topics that are present in a collection of documents. That's why it may not be as effective as LSA as it does not directly capture semantic relationships between words.

- TF-IDF outperforms FastText on all tasks. The reason could be that FastText was trained on a very large corpus of data, while TF-IDF models are more task specific.

- Combining word embeddings from BERT and FastText didn't perform well. This could be due to the incompatible embeddings of the two models. BERT uses a transformer-based architecture that generates contextualized embeddings, while FastText generates word embeddings based on character n-grams. As a result, the embeddings from the two models may not be directly comparable or combinable.