Bias in ML

Archaeology of Intelligent Machines

1st Semester of 2023-2024

Sefcic Adrian-Ionuț
adrian-ionut.sefcic@s.unibuc.ro

Hârnagea Andrei-Alexandru
andrei-alexandru.harnagea@s.unibuc.ro

1 Introduction

Bias in AI means unfairness or mistakes in how AI makes decisions. This can cause problems in many areas, like accuracy and how AI treats society.

Since code cannot make assumptions, we reached the conclusion that the problem must be with the databases that train a certain model. The project aims to perform sentiment analysis using machine learning to analyze and classify text based on the sentiments originated from texts in the Romanian languages.

Contributions

Adrian Sefcic

- organised the code
- tested the code to ensure accuracy

Andrei Hârnagea

- gathered information
- translated into Romanian the lexicon

Approach Summary

We approached the project in more than one way, but using the similar methods. First, we collected the data necessary, that being word embeddings and lexicons. After that, we trained a model to evaluate sentiment in text data. Finally, we evaluated the detection accuracy.

Motivation

With the increasing popularity of AI, it appears to be "biased". Understanding and addressing the source of the bias can help ensure fairness and prevent discrimination.

Previous Work

The main idea of the project, as well as the guide for the code, came from Robyn Speer's idea, in a blog from 2017. Despite the code being considers "outdated", it still holds up. Even if new technologies appeared and data became more in-depth, the prejudice still can be seen in the results.

2 Approach

Link towards the code here, and for Github.

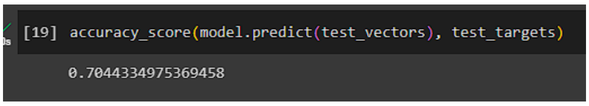
Software tools used:

- **Python:** The primary programming language used for development.
- **Various Python libraries** such as NumPy, Pandas, Matplotlib, Seaborn, Scikit-learn, and SpaCy: Used for data manipulation, analysis, visualization, and machine learning tasks.
- **Google Colab:** Utilized for running Python code in a cloud-based environment with access to GPU resources.
- **Multiple word2vec resources** such as:
  - MUSE, a library developed by Facebook Research for training multilingual word embeddings
  - Spacy the ro-legal-fl library, it includes pre-trained models and tools for tasks like entity recognition, which can be useful for analyzing legal documents
  - JurBERT, a Romanian BERT model fine-tuned for legal text understanding, Jurbert is trained on a large corpus of Romanian legal texts and can be used for various natural language processing tasks specific to the legal domain

- **Word lexicons:** Two lists of words in Romanian, one list containing negative words, the other one containing positive words. These words were collected from the internet, but some are also translated from English.

Evaluation method The 'text to sentiment' function is used to compute the sentiment of a given text by averaging the sentiment scores of individual words extracted from the text. Also, for each model, an accuracy score text is used.

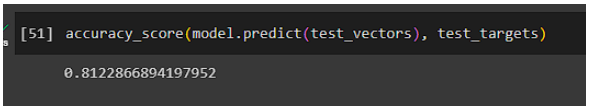
Accuracy score



```
[19] accuracy_score(model.predict(test_vectors), test_targets)

     0.7044334975369458
```

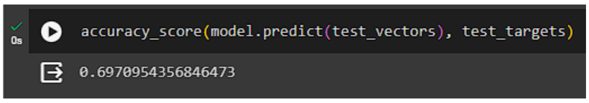
Figure 1: Accuracy score using 'ro-legal'



```
[51] accuracy_score(model.predict(test_vectors), test_targets)

     0.8122866894197952
```

Figure 2: Accuracy score using 'MUSE' word2vec

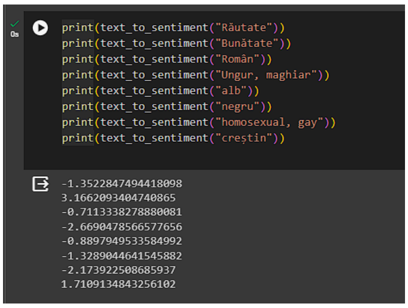


```
accuracy_score(model.predict(test_vectors), test_targets)

0.6970954356846473
```

Figure 3: Accuracy score using 'JurBERT'

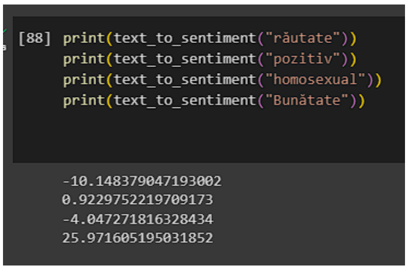
Text to sentiment results



```
print(text_to_sentiment("Răutate"))
print(text_to_sentiment("Bunătate"))
print(text_to_sentiment("Român"))
print(text_to_sentiment("Ungur, maghiar"))
print(text_to_sentiment("alb"))
print(text_to_sentiment("negru"))
print(text_to_sentiment("homosexual, gay"))
print(text_to_sentiment("creștin"))

-1.3522847494418098
3.1662093404740865
-0.7113338278880081
-2.6690478566577656
-0.8897949533584992
-1.3289044641545882
-2.173922508685937
1.7109134843256102
```

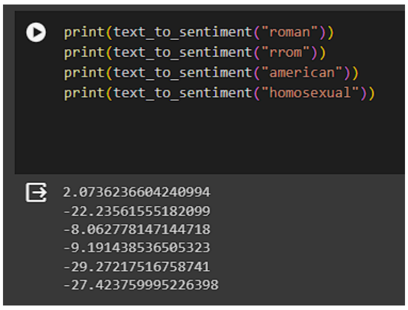
Figure 4: Text to sentiment results using 'Muse'



```
[88] print(text_to_sentiment("răutate"))
     print(text_to_sentiment("pozitiv"))
     print(text_to_sentiment("homosexual"))
     print(text_to_sentiment("Bunătate"))

     -10.148379047193002
     0.9229752219709173
     -4.047271816328434
     25.971605195031852
```

Figure 5: Text to sentiment results using 'bert'



```
print(text_to_sentiment("roman"))
print(text_to_sentiment("rrom"))
print(text_to_sentiment("american"))
print(text_to_sentiment("homosexual"))

2.0736236604240994
-22.23561555182099
-8.062778147144718
-9.191438536505323
-29.27217516758741
-27.423759995226398
```

Figure 6: Text to sentiment results using 'ro-legal'

Algorithm explained step by step

1. **Loading Word Embeddings:** Word embeddings are dense vector representations of words in a high-dimensional space, often pre-trained on large texts. A function is defined to load pre-trained word embeddings from a text file, or from a library.

2. **Loading Lexicons:** Lexicons are lists of words categorized based on certain criteria, such as sentiment (positive or negative). Positive and negative word lists are loaded from separate text files ('cuv-pozitive-v2.txt', 'cuv-neg-v3.txt') into lists.

3. **Creating Random Word Samples:** Random samples of words are selected from the loaded word embeddings for further analysis. This was used when there were too many word embeddings (there is a size limit of arrays on Colab).

4. **Processing Text with Spacy:** The words are processed using the Spacy NLP pipeline with a specific model ('ro-legal'). This pipeline tokenizes the text and assigns linguistic annotations such as part-of-speech tags and syntactic dependencies (only used for ro-legal).

5. **Training a Sentiment Classifier:** A sentiment classifier is trained using SGD with logistic loss. The classifier predicts the sentiment of words based on their embeddings.

6. **Converting Words and Text to Sentiments:** Functions are defined to convert words and text to sentiment scores using the trained classifier.

3 Limitations

The biggest problem we had was not finding resources available in Romanian. We had to rely on resources available online (including AI tools, Chat-GPT specifically). When working on the lexicon, we used Google Translate to translate a lexicon that we found in English. We found in the translated result repeating words or some words weren't translated properly. We searched words with positive meaning in Romanian and we noticed that the words were biased towards religion. When we requested Chat-GPT for a list of negative words, we were given words such as 'homosexual', 'lemon', as well as words that were directed towards minorities.

Another problem we encountered was the availability of word embeddings. They weren't available at a first Google search (like the one in English) and they required a lot of processing.

Due to the unavailability of resources in Romanian, some words aren't recognisable by the model. Also, we noticed that if the data had some oversights (when we didn't pay attention to the words it was trained on), the results were noticeably abnormal. For example, 'pozitiv' had a negative score. This further proves that any model is sensitive to the database it was trained on and that some data can be biased (by mistake or on purpose).

4 Conclusions and Future Work

- *Is there anything we could have done different?*
  The code could have been more organised. Sometimes we had to take *shortcuts* since the libraries weren't so easy to work with.

- *Is there any way of improving this project?*
  As said previously, if resources in Romanian were more widespread, better results would have been achieved.

- *Did we learn anything new by doing this?*
  We both learnt more about word2vec, embeddings and how they works. Also, we revised some knowledge about how to train an AI model

- *What is something that you feel you didn't fully understand?*
  We feel like we didn't fully get how word2vec works exactly.