Sentence Pair Classification

Vasilescu Andreea[a]

[a] *University of Bucharest, Academiei Street, no. 14, Bucharest, 077131, Romania*

Abstract

This paper is about Sentence Pair Classification task that was proposed on Kaggle. The goal is to train a classifier on a data set containing Romanian sentences.

The classification model is required to discriminate between four classes labeled from 0 to 3.

The evaluation measure is the macro score computed on the test set. The macro score is given by the mean of the scores computed for each class.

Keywords: NLP, XGBoost, SVM, Random-Forests

1. Introduction

Natural Language Processing (NLP) is a field of Artificial Intelligence that specialize in understanding the natural language, analyze human sentiments with the scope of bulding machines that manipulate human language. Applications of NLP includes sentiment analysis, speech recognition, information retrieval, taxt mining, machine translation.

2. State of the ART

The following section will provide an overview of the current advancements and achievements in the NLP field, as well as state of the art models, methodologies, limitations and potential applications. The purpose of this section aims to set a foundation for the conducted study that will be presented.

2.1. Supervised Learning

Supervised Learning is a Machine Learning technique that requires data to learn before it can make predictions, as well as labels for each data. This technique is used to solve classification and regression problems, as it help to predict unforeseen data.

2.2. NLTK

NLTK (Natural Language Toolkit) is a Python library for natural language processing (NLP) tasks. It provides tools, data sets, and resources for various NLP applications, including tokenization, tagging, parsing, and machine learning. NLTK offers pre-trained models and corpora for research and experimentation. It supports tasks like part-of-speech tagging, sentiment analysis, and syntax parsing. NLTK integrates with other NLP libraries and frameworks. It is widely used in academia and industry for NLP development and analysis. NLTK's documentation and community support make it accessible to beginners and advanced practitioners.

2.3. Tokenization

Tokenization is a crucial step in NLP, breaking down text into smaller units called tokens. These tokens can be words, subwords, or characters, depending on the task. State-of-the-art techniques, like BPE or SentencePiece, handle OOV words effectively. Tokenization is especially important for large pre-trained models like BERT, where special tokens are added. Domain-specific tokenization addresses specialized vocabulary. Overall, tokenization plays a vital role in NLP, facilitating efficient text processing and enhancing language understanding for diverse applications, such as Sentiment Analysis, Machine Translation, NER or text classification.

2.4. Lemmatization

Lemmatization is a crucial linguistic technique in natural language processing (NLP) that aims to reduce words to their base or dictionary form, called the lemma. Unlike stemming, which truncates words to their root form, lemmatization considers the word's morphological properties and applies grammatical analysis to generate accurate lemmas. By reducing words to their canonical forms, lemmatization enhances text normalization, improves information retrieval, and supports various NLP tasks such as machine translation, information extraction, and sentiment analysis. State-of-the-art lemmatization approaches often leverage linguistic resources, part-of-speech tagging, and rule-based or statistical methods to handle different word forms and languages effectively. Robust and accurate lemmatization is essential for ensuring precise analysis and understanding of text data, making it a vital component in modern NLP research and applications.

2.5. TF-IDF

Term Frequency Inverse Document Frequency (TfIdfVectorizer), as it is a commonly used technique in both NLP and Information Retrieval.

TfIdfVectorizer tool is provided by scikit-learn, and it combines the tokenization, counting and TF-IDF weighting.

Based on a collection of text documents, it will transform the text into a matrix of Tf-Idf features.

TF will measure the frequency of a term in a document by calculating the ratio of total number a term appears in a document and total number of words in the document.

IDF measures the importance of a term accross a collection of documents by calculating the logarithm of total number of documents divided by the number of documents that contain the term we want to measure importance.

TF-IDF it's focused on the terms that are frequent in a specific document, but it is not common across. The terms that are commen across all the docuemts will receive lower weights.

2.6. SVM

Support Vector Machine(SVM) is a **supervised learning algorithm** that solves a Convex Optimization Problem for the classification of objects in two or more categories, separated by a hyper-plane that maximizes the distance from it to the closest point of each category. This model can be used for text recognition, face detection, or spam filtering.

Each object we want to classify is represented as a point in n-dimensional spaces. Each of the points proposed for classification has its unique coordinates called features.

The classification consists of drawing a **hyper-plane**, also called **decision boundary**, represented by a line in 2D or a plane in 3D, splitting the categories on one side or another.

The purpose of SVM is to find a hyper-plane that separates the categories in the best way, by maximizing the distance from the hyperplane to the closest point from each category (**margin**). **Support vectors** are the closest points to hyper-plan.

SVM is a **supervised learning algorithm** because it requires a training set represented by a set of points already labeled with the correct category, needed for finding the hyper-plane. Points in each category must be on the proper side of the decision boundary.

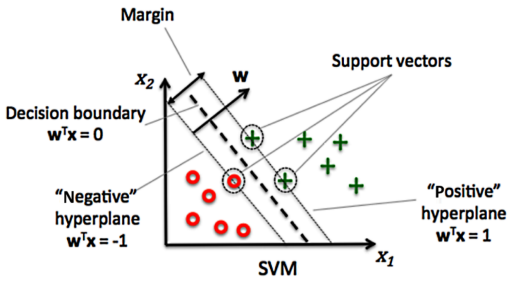

Figure 1: SVM representationAhmad (2020)

In the context of the SVM discussion, it is important to mention the Kernel Trick, a method that allows data to be mapped from 2-dimensional space (as in Support Vector Machine) to 3-dimensional space. The figure shows a graphical representation of how SVM, while the figure explains the Kernel Trick works.

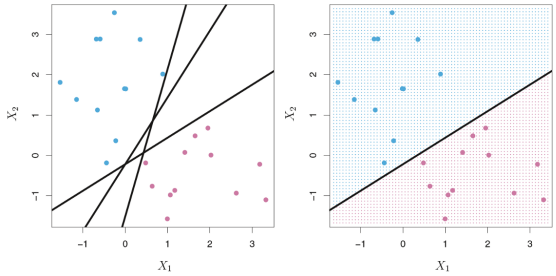

Figure 2: SVMGrace Zhang (2018)

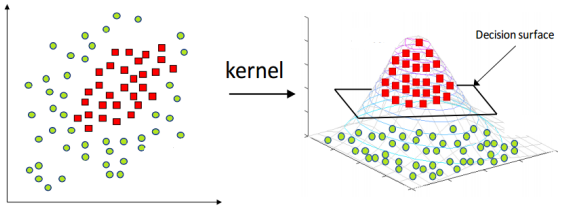

Figure 3: Kernel TrickGrace Zhang (2018)

2.6.1. Parameters

- **Cost parameter** controls the trade-off between a smooth decision boundary and classifying the points correctly. A smaller C means a simpler decision boundary (potential risk: missclassification). A larger C will be more complex, but might lead to overfitting.
- **Kernel** in SVM is used for Kernel trick described above, with the scope of transforming the input data into a higher dimensional space, aiming to find a hyperplane that will separate the classes. The choice of kernel will determine the decision boundary. Among the most used kernel functions I can ennumerate: linear, polynomial and RBF.
- **Gamma** parameter will define how far the influence of a single example can reach. A small gamma will mean a generalized decision boundary, while a large gamme will make the decision boundary sensitive to training data.

  It is important to understand the data and apply the correct parameters, as this will influence the results.

2.7. XGBoost

Extreme Gradient Boosting (XGBoost) is an ensembling method, as it combines multiple methods to provide improved results. This model performs classification by taking outputs from individual trees and then combining them. It is designed to be highly flexible, efficient, and portable, implementing an algorithm of machine learning by the Gradient Boosting framework.

The xgboost algorithm is designed especially for huge and complex datasets and this model can be used for solving both **regression** and **classification** problems.

2.8. SVC

The purpose of clustering is to classify data based on a predefined criterion to group data. **Support Vector Clustering** is

based on a Support Vector Machine. This can be achieved using methods such as:

- parametric model (only parameters are needed to predict unknown values; e.g. K-means Model, a basic model in clustering, which allows grouping data based on similar characteristics)

- grouping points based on distance, like hierarchical clustering algorithms Ben-Hur et al. (2001)

SVC transforms the points that lie in a given space into another space characteristic for them, but with a larger dimension, and then identifies a sphere in the characteristic space; In the initial space (the one before the transformation), the previously identified sphere will be transformed into a set of disjoint regions.

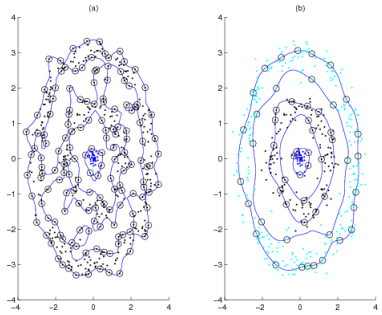

Figure 4: SVCBen-Hur et al. (2001)

2.9. Accuracy

The Accuracy gives us an overall measure of how well our model performed by calculating the percentage of correctly classified instances (TP and TN) out of total number of instances.

Precision measures how many of the predicted positive instances are actually positive. It is calculated that $\frac{TP}{TP+FP}$. A high recall indicates a low false negative rate, meaning that model has a tendancy to classify negative instances as positive.

Recall, measures how many of the actual positive instances are correctly classified as positive. It is calculated as $\frac{TP}{TP+FN}$A high recall indicates a low false negative rate, meaning that the model has a low tendency to classify positive instances as negative.

F1 Score is the harmonic mean of precision and recall and provides a single metric that balances both measures. It is calculated as $2 * \frac{(Precision*Recall)}{(Precision+Recall)}$. The F1 score considers both false positives and false negatives and is useful when you want to find a balance between precision and recall.

3. Experiment

In the following are going to explain my work done on Sentence Pair Classification task, as well as present how the experiment was conducted. This chapter contains information regarding implementation of the coding part that stands behind this paper, what I have tried and how I got to the final version. At the end of this chapter, I am going to provide the pros and cons of each model I've used on the data-set, providing a brief explanation.

3.1. Tools used

For this experiment, , I used **Python** programming language. I have used one of the main libraries in python such as:

- python3.10
- numpy for numerical operation, as it is considered to be one efficient on array operations, as well as mathematical functions.
- pandas for reading the data.
- NLTK, a library specifically designed for human language data manipulation itemspacy, popular library used for NLP processing, as it provides several models trained on large corpus texts.

Google Collaborator was my choice because I was already familiar with it from previous university projects I worked on. This tool offered the possibility to test models easier. The one biggest advantage for me was that I could easily store the data in Cloud, and then be able to access it from any device, so I was able to work even if I was

3.2. Data

For this project I have used the data from the Kaggle competition entitled Sentence Pair Classification.

The data samples are provided in the following format based on a JSON array of dictionaries (each dictionary entry contains one data sample).

4 files were provided:

- train.json - the training data samples and labels (guid, sentence1, sentence2, label)
- validation.json - the validation data samples and labels (guid, sentence1, sentence2, label)
- test.json - the test data samples (guid, sentence1, sentence2)
- sample_submission.txt - a sample submission, a sample submission in the correct format (guid, label)

3.3. Reading the data

As the data was given in JSON format and zipped, I have used zipfile to unzip it, and then I have created directories for each file provided. Using pandas library, I have created a function to load the json files to data frames.

3.4. Pre-processing

Having a different data-frame for each file given, I have started the preprocessing part.

At this point in my project, I created a function called preprocess for:

- lowercasing
- remove punctuation and digits
- tokenization using word_tokenize from nltk library
- remove stopwords specific for romanian language, using stopwords function from nltk.corpus
- lemmatization using WordNetLemmatizer() from nltk.stem

After I have created the function, I have applied the preprocessing to both sentence1 and sentence2 from all data frames (train, validation and test)

3.5. Building a model

The first approach I have tried was to build a model using TF-IDF vectorizer.

I have combined the sentences as follows:

X_train for training, X_valid for validation and X_test for testing. I vectorized the data by using vectorizer.fit_transform, in otder to convert the collection of documents into a numerical format with the scope of using it as input for the machine learning algorithms that I willfurther describe.

3.6. SVC

The first model that I tried was Support Vector Classifier. I have imported the model and then fit it on training data, predict it on validation and check it for test.

Precision: 0.49 **Recall**: 0.35 **F1-score**: 0.34

I tried different cost parameters, kernel and gamma with GridSearch.

```
param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf', 'poly'],
    'gamma': ['scale', 'auto']
}
```

Precision: 0.49 **Recall**: 0.35 **F1-score**: 0.34

To parse the data for submission I have used pandas library and then export it as CSV file.

I sent a submission with SVC as baseline and private score on Kaggle was 0.48, while the public one was 0.50. I have selected this submission to be evaluated for my final leaderboard score, but it wasn't the best one.

3.7. XGBoost

The second approach was XGBoost. I have installed XGBoost model using ! pip install XGBoost.

For this method I have used the following parameter grid for Grid Search CV:

```
param_grid = {
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'n_estimators': [50, 100, 200],
}
```

The best hyperparameters from the grid were:

- learning rate: 0.1
- max depth: 7
- number of estimators: 200

For metrics I have used sklearn.metrics, and I had the following results:

Validation Accuracy: 0.63 **Precision**: 0.58 **Recall**: 0.25 **F1-score**: 0.34

On Kaggle I got a score of 0.39 as Private, and if I would've selected this submission, I would've got a F1-score of 0.40 as public score.

3.8. Random Forest

The third and last method I have used for the sentence pair classification task was Random Forests.

The first try was on the basic model, without specifying any parameters. I am always trying the basic model just to be sure that everything works.

Then I performed a test on Random Forest Classifier with a number of 3000 estimators, and I observed that all labels were predicted as 3.

In the end, I tested out different parameters and choose the best for the latest submission:

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
```

I have selected this as one of the two final submissions and I got the highest score with it.

4. Conclusions

In conclusion, the best F1-max score obtained was with Random-Forest.

Model_Method	Valdiation Accuracy	Private score	Public Score
SVC	0.49	0.48	0.50
XGBoost	0.63	0.39	0.40
Random Forest	0.62	0.63	0.66

Issues encountered

1. It took me a while to run everything, and sometimes the runtime disconnected, so I needed to re-run everything again. As a workarount I have tried to save some models.
2. I had an issue with Tf-IDF because I had a different number of features in training data than validation data, so then it was impossible to see the accuracy for validation and predict the labels for test. This happened because I have tried an approach of working with lists, but I have fixed this issue by using pandas library.

5. Future Work

I would like to apply NN on this data-set, as on previous projects I worked on I got the best accuracy with Neural Networks. For this project I started with SVC, XGBoost and Random Forests because it was the first time I worked with pairs of sentences instead of sentences/ phrases.

References

Ahmad, I., 2020. 40 Algorithms Evry Programmer Should Know: The SVM Algorithm .

Ben-Hur, A., Horn, D., Siegelmann, H.T., Vapnik, V., 2001. Support vector clustering. Journal of machine learning research 2, 125–137.

Grace Zhang , 2018. What is the kernel trick? URL: https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d.