

**Multi-Label Tagging of Medium Headlines**

Student: Yoni Pick

Student ID: 300554730

Head Advisor: Dr. Dovi Yellin

Advisor: Dr. Yehudit Aperstein

**Acknowledgments**

I wish to express my gratitude to Dr. Dovi Yellin for taking this journey with me.

Dr. Yellin’s ideas and guidance were clear and insightful. This project would not have been possible without his support.

In addition, I would like to thank Dr. Yehudit Aperstein for her support and productive comments along the way.

Last but certainly not least, I lovingly thank my amazing wife, Rona, and my two beautiful, smart,

energetic and amazing kids, Amit and Yuval, for supporting my ambition to complete this

thesis while working mostly from home.

Daddy’s excuses have run out! I have more time to play now!

**Abstract**

Multi-label classification is one of few highly challenging research tasks in the Natural Language Processing (NLP) space. In this project, I aimed to automatically tag unstructured data, provided as free text, at highest possible accuracy. My goal was to arrange the free text in structured format allowing to obtain relevant information from it, by using algorithms from the NLP field and a classifier for tagging the unstructured data.

In this project, I used a free-text title and the labels from a Medium-website text classification dataset that was uploaded to Kaggle [Jansma 2018]. Preprocessing of the data indicated it is highly imbalanced necessitating special care upon dividing the data to train and test sets. Different multi-label classification models trained using this data were compared to explore the alternatives for methods that facilitate the search for information on this type of headline, starting from classic machine-learning methods to newest deep learning methods. I explored the effectiveness of deep learning and transfer learning in text classification, by fine-tuning different pre-trained language representations, such as GloVe with vanilla LSTM, and Hybrid models such as LSTM with CONV. In addition, a model with attention was used, as well as Pooled RNN and Max pool Text CNN. Finally, I tested the state-of-the-art BERT model.

Results (in terms of micro-averaged F1 and AUROC scores) showed clear advantage to the BERT-based model. However, it is also interesting that, although I used an imbalanced dataset, the number of samples per label did not affect the predictions of these labels, and all the models had the same difficulty with the more abstract labels. Yet overall, we saw the clear advantage of using deep learning models.

**Table of** **Content**

[1. Abbreviations 5](#_Toc49259856)

[2. List of Figures 6](#_Toc49259857)

[3. List of Tables 6](#_Toc49259858)

[4. Introduction 7](#_Toc49259859)

[*4.1.* *Background* 7](#_Toc49259860)

[*4.2.* *Motivation* 8](#_Toc49259861)

[*4.3.* *Challenges* 8](#_Toc49259862)

[5. Data 9](#_Toc49259863)

[*5.1.* *Data Imbalance* 10](#_Toc49259864)

[*5.2.* *Data Analysis* 10](#_Toc49259865)

[6. Literature Review 12](#_Toc49259866)

[*6.1.* *Text Classification* 12](#_Toc49259867)

[*6.2.* *Feature Engineering* 13](#_Toc49259868)

[*6.3.* *Short Text Classification* 14](#_Toc49259869)

[*6.4.* *Machine Learning in Text Classification* 14](#_Toc49259870)

[*6.5.* *One vs. All* 15](#_Toc49259871)

[*6.6.* *Deep Learning* 16](#_Toc49259872)

[*6.7.* *Evaluation Metrics* 19](#_Toc49259873)

[*6.8.* *Focal Loss* 21](#_Toc49259874)

[7. Experimental Setups 21](#_Toc49259875)

[*7.1.* *Setup* 22](#_Toc49259876)

[*7.2.* *Evaluation and Cross-Validation* 22](#_Toc49259877)

[*7.3.* *Machine Learning Methods* 22](#_Toc49259879)

[*7.4.* *Deep Learning Methods* 23](#_Toc49259880)

[8. Methods 30](#_Toc49259881)

[*8.1.* *Parameters Selection* 30](#_Toc49259882)

[9. Results 33](#_Toc49259883)

[10. Discussion 36](#_Toc49259889)

[11. References 37](#_Toc49259890)

[Appendix A 39](#_Toc49259891)

## Abbreviations

|  |  |
| --- | --- |
| AUC | Area Under Curve |
| AUROC | Area Under the Receiver Operating Characteristic |
| Bert | Bidirectional Encoder Representations from Transforms |
| BiLSTM | Bidirectional Long Short Term Memory |
| CE | Cross Entropy |
| CNN | Convolution Neural Network |
| CV | Cross Validation |
| NLP | Natural Language Processing |
| ML | Machine Learning |
| RNN | Recurrent Neural Network |
| SOTA | State of the Art |
| SVM | Support Vector Machines |
| TF– IDF | Term Frequency Inverse Document Frequency |
| OVR | One vs. All |
| LSTM | Long Short Term Memory |
| GolVe | Global Vectors. |
| GRU | Gate Recurrent Unit |
| YAKE! | Yet Another Key Extraction |

## List of Figures

[Figure 1: Example of Multi Label in Movies 7](#_Toc49260076)

[Figure 2: Distribution of Tags in Training Set 10](#_Toc49260077)

[Figure 3: Distribution of Tags in Test Set 10](#_Toc49260078)

[Figure 4: Number of Tag Set in Training 11](#_Toc49260079)

[Figure 5: Correlation between the Tags 11](#_Toc49260080)

[Figure 6: Vanilla BERT 17](#_Toc49260081)

[Figure 7: Pipeline Model for Classic Methods 22](#_Toc49260082)

[Figure 8: Vanilla LSTM Architecture 23](#_Toc49260083)

[Figure 9: Hybrid Model - 2 LSTM and CONV Architecture 25](#_Toc49260084)

[Figure 10: 2 BiLSTM with Attention Architecture 26](#_Toc49260085)

[Figure 11: Pooled RNN Architecture 27](#_Toc49260086)

[Figure 12: Max Pool Text CNN Architecture 28](#_Toc49260087)

[Figure 13: BERT Based Architecture 29](#_Toc49260088)

[Figure 14: The F1 Score vs Max Feature and Max df 30](file:///C:\Users\Yoni%20Pick\Dropbox\Msc\Msc%20final%20project\Final%20Report\Final.docx#_Toc49260089)

[Figure 15: The F1 Score vs Log (alpha) 31](#_Toc49260090)

[Figure 16: Focal Loss vs Cross Entropy 32](file:///C:\Users\Yoni%20Pick\Dropbox\Msc\Msc%20final%20project\Final%20Report\Final.docx#_Toc49260091)

[Figure 17: Normalized Number of Samples per Class (blue line); the F1 Score for Each Class from BERT (red line) 35](#_Toc49260092)

## List of Tables

[Table 1: Confusion Metrix 18](#_Toc49260094)

[Table 2: Selected Parameters in Classic Model 30](#_Toc49260095)

[Table 3: The Effect of Threshold on Precision, Recall F1 Score and Hamming Loss in Max Pool Text CNN 32](#_Toc49260096)

[Table 4: Micro Averaged Precision, Recall and F1, and AUC for Every Classifier on the Test Set 33](#_Toc49260097)

[Table 5: Top 5 Labels 33](#_Toc49260098)

[Table 6: Bottom 5 Labels 34](#_Toc49260099)

[Table 7: The Effect of the Number of Supports on Label Classification 34](#_Toc49260100)

[Table 8: Examples of Prediction on Test Set 35](#_Toc49260101)

[Table 9: The Micro Averaged F1 Scores per Class in Each Classifier 40](#_Toc49260102)

## Introduction

## *Background*

Text classification, also known as text categorization, is a classical problem in natural language processing (NLP). It aims to assign labels or tags to textual units such as sentences, queries, paragraphs and documents. It offers a wide range of applications, including question answering, spam detection, sentiment analysis, news categorization, user intent classification, content moderation and so on. Text data can come from different sources, for example, web data, emails, chats, social media, tickets, insurance claims, user reviews, questions and answers from customer services and many more. Text is an extremely rich source of information but extracting insights from it can be challenging and time-consuming, due to its unstructured nature. In this project, I focused on multi-label classification. People sometimes tend to mix up multi-label and multi-class classifications. Multi-class classification is a classification task with more than two classes where labels are mutually exclusive. The classification premise is that each sample belongs to one and only one class. On the other hand, multi-label classification assigns to each sample one or more target classes. This task can be thought of as predicting properties of a data-point in a multi-dimensional space, where properties are not mutually exclusive. For example, movies that are often categorized as being of the romantic genre as well as comedies. Multi-class classification has many real-world applications, such as categorizing businesses on Yelp or movies (see Figure 1) into one or more genres.

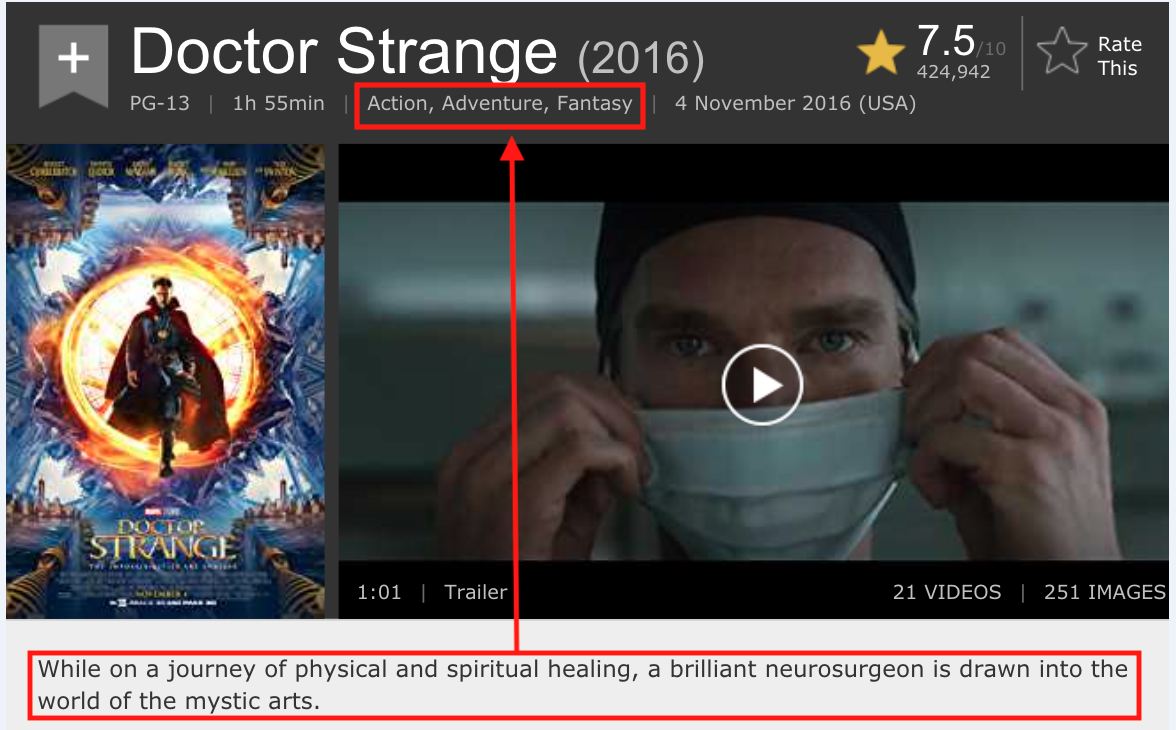


Figure 1 Example of Multi-Label in Movies

## *Motivation*

Getting information from free text is one of the biggest unresolved challenges in NLP. Our goal is first is to take unstructured data provided in free text format, especially short text and arrange the free text in a more structured way allowing the relevant information to be obtained from it by using algorithms from the NLP field. Our second goal is to build an automatic classifier that able to automatically tag the data to the relevant domains.

## *Challenges*

The focus of the project was to extract information from short text segments, from case titles of blogs from the Medium site, giving them a set of prediction labels.

This is very challenging because the original data also takes into consideration the full body of text, due to the use of less information. Some labels cannot be predicted without the full text. On the other hand, I cannot devalue the quality of the tagging.

One more challenge that I faced was that writers sometimes try to draw in readers by giving a deceiving title, and that can affect the labeling. Beside that the data is highly imbalanced that can cause mispredictions on minority classes.

## Data

The data for this project came from one of Kaggle's open source repositories. The data consists of 1.4 million stories from the top 95 of Medium's most popular story-tags. The stories were published between August 1st, 2017 and August 1st, 2018. The dataset created by Harrison Jansma[1], was initially created to help understand the Medium's clap-metric. The dataset contains Title, Sub-Title, Author, Publication, Date, Tags, Read-Time, Claps-Received, Story-URL, and Author-URL for each story.

For this project, I focused only on titles and tags. According to Medium guide, a story can receive up to five tags. Medium site[2] has a hierarchy order of subjects that start from the top five subjects such as art & entertainment, industry, innovation & tech, life and society. Each subject has many topics. The site shows the top five related picks for each topic, with each topic receiving a variety of tags. In accordance with this, certain data cleaning was done, which included the removal of blank text, non-English stories and duplicated titles. In addition, a threshold of less than three words was set for the removal of short texts, as well as texts consisting of more than 14 words. This was due to the fact that titles usually consist of short texts. Vague titles that did not have any key-term words were removed and the labels were minimized from 95 to 62 tags.

## *Data Imbalance*

Data imbalance is a well-known problem in machine learning and real-life applications. While some classes in the dataset are more frequent than others, these should not go untreated. The model tends towards learning to predict frequent classes. For example, if a dataset consists of 100 positive and 900 negatives, the model on this data would tend towards learning to predict negative more often than detecting positive in a manner that would degrade the overall accuracy. In this case, I can try to balance the data using sampling techniques such as over-sampling the duplication of the minority class or under-sampling the majority class[3]. However, this problem gets harder when you have multi-label data. Even though the dropping of samples from class can impact the other classes as well, and over-sampling the minority class can increase other classes by using weight class [27,28], the information will not get lost. However, the model will take more time to converge.

## *Data Analysis*

Before starting any analysis, the data should be divided into a training set and a test set. Figures 2 and 3 show the distribution of number of tags vs. the labels in the training and test datasets, and this confirms that the training set and test set come from the same distribution.

Also, the data is highly imbalanced, and some techniques for dealing with imbalanced sets should be considered. An additional task is also to check the distribution of the number of tags per sample in training data (Figure 4) and verify it fits with Medium policy that each title can get at most five tags and at least one tag. Lastly, there is also a need to examine the correlation between the tags (Figure 5) that show that very small correlation between the tags, but we expected to be more correlation between them.

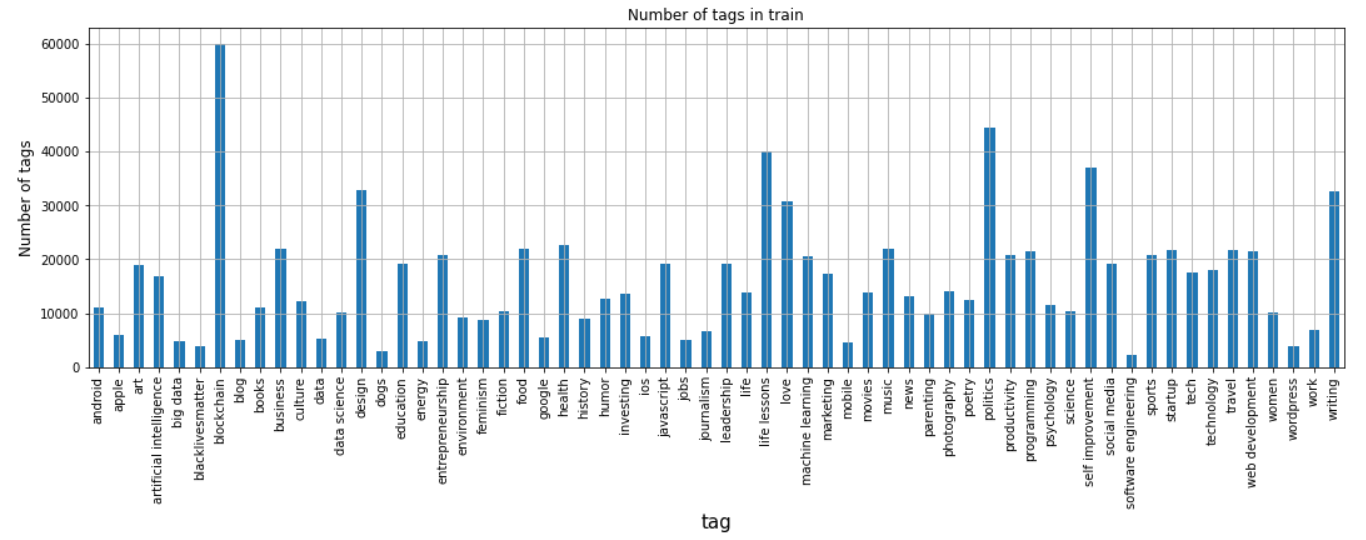


Figure 2: Distribution of Tags in Training Set

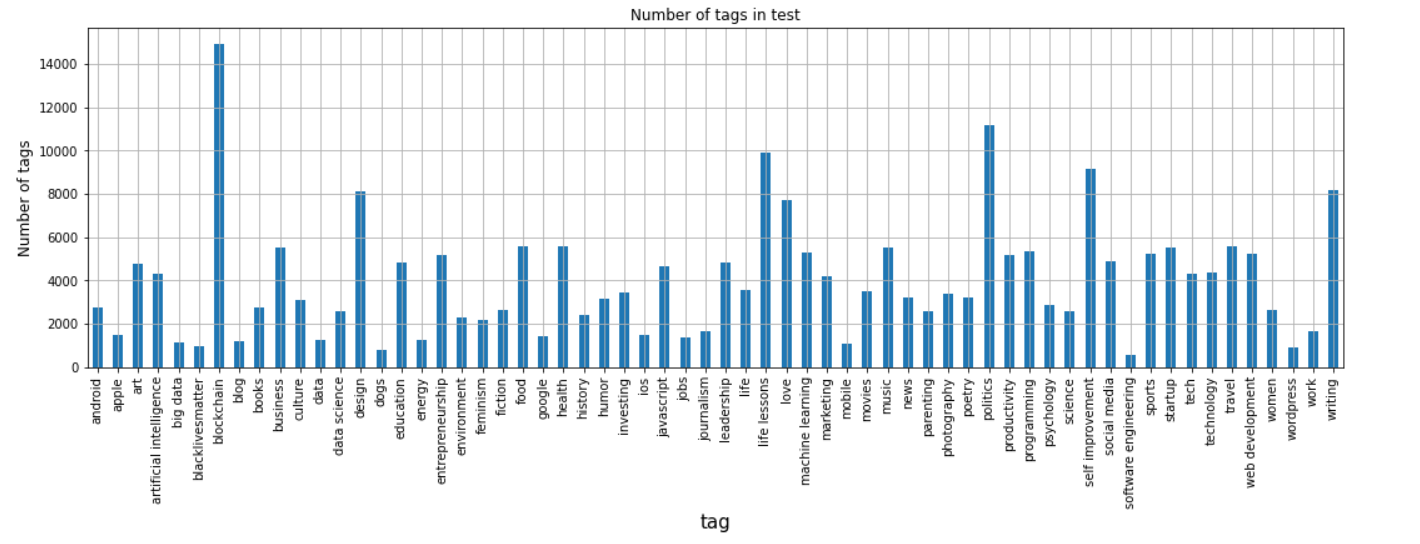


Figure 3: Distribution of Tags in Test Set

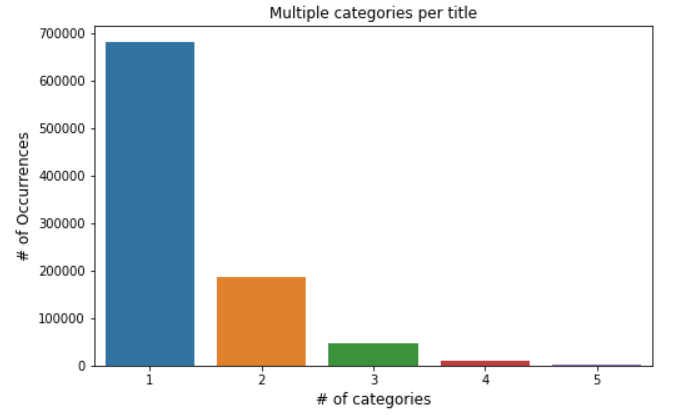


Figure 4: Number of Tag Sets in Training

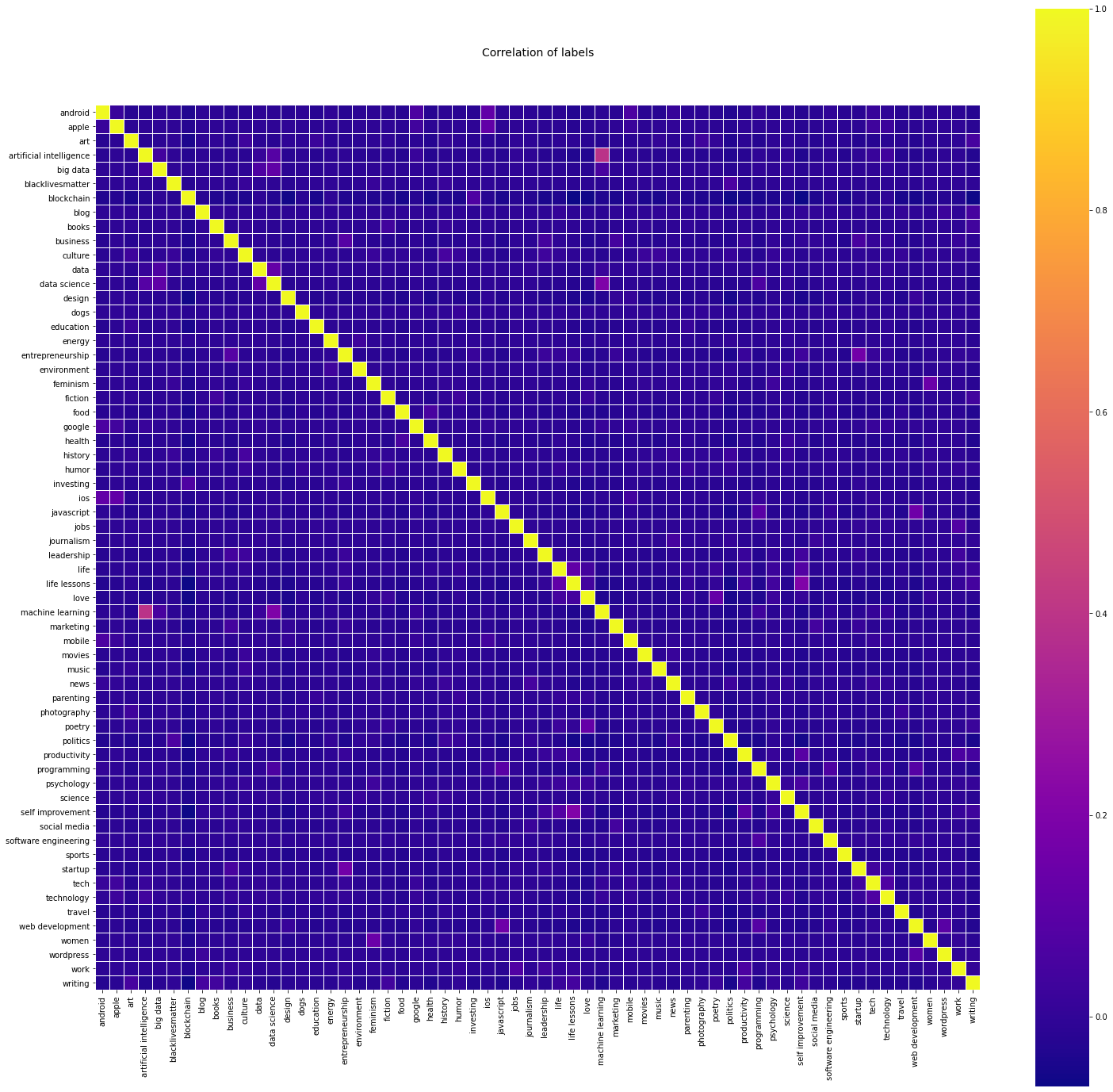


Figure 5: Correlation between the Tags

## Literature Review

In this section, I will explore previous works related to multi-label classification.

## *Text Classification*

Text classification is the area of studying the information retrieval concerned with the classification of text samples into predefined classes.

There are three main types of text classification: binary, multi-class and multi-label.

Binary text classifiers assign one of two possible labels to each sample. This is the common approach, for instance, for sentiment analysis. A good example in the field is the IMDB dataset of movie reviews. There are two pre-defined classes indicating whether the movie review is positive or negative[4].

Multi-class text classifiers assign one label from a set of more than two classes to each sample. An illustrative example is the BBC News dataset, which has the following classes: business, entertainment, politics, sport and tech.

Multi label classification assigns each sample a set of labels that can be from one or more labels in the data[5]. A good example is the IMDB dataset in which labels are set by using a description of the movie, e.g.: drama, comedy, romantic, thriller, crime, action.

## *Feature Engineering*

Working with unstructured text data is hard because the intelligent system does not interpret and understand natural language like humans beings are able to do. The algorithm needs standardized input to work efficiently, thus the processing and transforming of noisy, unstructured textual data into some structured, vectorized format is required, in order to be understood by any machine learning algorithm. To deal with this challenge, I added the methods of YAKE! and TF-IDF as part of the project.

* + 1. *Key Term – YAKE!*

YAKE! is an unsupervised method to extract key terms from text[9]. YAKE! is quick, not domain-specific, nor language-dependent. Every text goes through a pipeline that includes text pre-processing, feature extraction, individual terms score, candidate keywords list generation, data duplication and ranking.

I devised a set of five features to capture the characteristics of each individual term - Casing, Word Positional, Word Frequency, Word Relatedness to Context and Word DifSentence.

The method’s output is a list of relevant keywords formed of N-grams. In my project, N-grams were set to three, meaning only the top three terms were taken per each sample[6].

* + 1. *Term Frequency - Inverse Document Frequency (TF-IDF)*

TF-IDF is a scoring measure that is widely used in information retrieval (IR) or summarization. The TF-IDF function is to reflect how relevant a term is in a given document. The intuition behind it is that if a word occurs multiple times in a document, we should boost its relevance as it is supposedly more meaningful than other words that appear fewer times (TF). At the same time, if a word occurs many times in a document as well as in many other documents[7], then it’s significance is lower as it is likely a prevalent word in the language.

The first part of this method, Term Frequency (TF), represents how often a given term *t*

occurs in one sample or document. The second, Inverse Document Frequency (IDF), represents the total number of samples or documents # over the number of samples or documents in which that given term C occurs, calculated with the log to base 10 as seen in Equation 6.1

## *Short Text Classification*

Most of literature has focused on long text contexts that shows successful results in classification tasks. However nowadays, a large volume of text data is generated from the social communities, such as blogs, tweets, and comments providing more datasets of short text such as Twitter, Stack Overflow and Google News[8].

How to evaluate short text topic models is still an unresolved issue. A lot of metrics have been proposed for measuring the coherence of topics in texts[9,10].

Although some metrics tend to be reasonable for long texts, they can be problematic for short texts[24]. Most conventional metrics try to estimate the likelihood of held-out testing data based on parameters inferred from training data. However, this likelihood is not necessarily a good indicator of the quality of the extracted topics[25].

## *Machine Learning in Text Classification*

Below are the most common approaches to the multi-label classification problems[11]:

data transformation, method adaptation and ensembles approaches.

Data transformation consists of transforming the multi-label problem into a simpler one. The most common approach of data transformation is binary relevance. With this method, a classifier is trained on each class and needs to learn whether a label is relevant or not for a given sample.

The second approach - method adaptation, consists of adapting binary classification algorithms so that they can deal with multiple labels.

The last approach - ensemble, consists of combining several classifiers with the assumption that they will be able to capture different biases, which when combined, will achieve better results than if they were applied individually. This method, therefore, does not tackle the multi-label problem directly, but deals with related problems such as class imbalance, i.e. having considerably more samples for some classes than for others.

* + 1. *Multinomial Naive Bayes*

In its most basic form, Naive Bayes text classifiers, assign each sample-class the probability that a sample, represented by a vector of binary weights, belongs to that class by applying the Bayes theorem. The binary weights indicate whether a term appears in the sample or not, and once all the probabilities have been calculated, the classifier assigns the class with the highest probabilities to the corresponding sample. However, this form of Naive Bayes does not consider the frequency of the terms. This is something that is considered by the multinomial Naïve Bayes, which is generally more effective for text classification [12].

* + 1. *Support Vector Machines*

Support Vector Machines (SVM) is a kernel method and has proven to be considerably effective in text classification[12]. The idea behind the most basic SVM is a binary classifier in a high-dimensional vector space where samples are represented as points and where the objective is to find the decision hyperplane that divides them into one of two classes. To calculate this hyperplane in a linear separable dataset, the classifier first locates the support vectors, which are the samples delimiting the boundaries of the classes. The lines or planes that cross the support vectors are called delimiting hyperplanes and it is between them that the classifier needs to find the decision hyperplane. Finally, the decision hyperplane is selected to be the line or plane in the space that maximizes the distance to the delimiting hyperplanes. Once the decision hyperplane is learned, the classifier examines a new sample and assigns it a label depending on which position of the decision hyperplane it is located.

* + 1. *Logistic Regression*

The most basic logistic regression algorithm is a binary classifier that first multiplies the features of each sample by its weights and adds a bias, and later applies a sigmoid function to obtain a probability of the sample belonging to one class or the other. The difference between this probability and the actual value of the class we want to predict is calculated with the cross-entropy loss function, which is then minimized with gradient descent. Contrary to Naive Bayes, logistic regression does not assume the independence of each of the sample’s features, which means that when data is available in considerable size, logistic regression tends to be more effective[13], although Naive Bayes generally works well with limited data and is faster to execute.

## *One vs. All*

Multi-label learning deals with problems where each example is represented by a single instance while being associated with multiple class labels simultaneously. Binary relevance is arguably the most intuitive solution for learning from multi-label examples[16, 17]. It works by decomposing the multi-label learning task into several independent binary learning tasks (one per class label). One of the weaknesses is ignoring correlations between labels. In this project, each tag was treated as a separate classification problem. In total, there were 62 classifiers.

## *Deep Learning*

Deep learning can be defined as a subfield of machine learning that learns representation from data through successive layers of increasing representation stacked in neural networks that are learned simultaneously. The main advantage of these models, over the machine learning methods which we explained about in the previous section, is not only are they more effective when lots of data is available, but they fully automate the feature selection as well.

* + 1. *Convolution Neural Networks - CNNs*

Convolutional Neural Networks (CNNs) create a matrix of word embedding for each sentence, to which convolutional filters called kernels are applied with the option of possible windows of words that extract different features. After that, a max-pooling strategy is applied to reduce the layers’ output dimensionality and keep them in a fixed size. CNNs are competitive with RNNs and can be a much faster alternative to RNNs for text classification tasks.

Therefore, multiple studies have used CNNs for multi-label text classification[14]. The proposed CNNs have generic and domain-specific word embedding for multi-label text classification with microblogging data, and therefore obtain improved accuracy compared to machine learning approaches.

* + 1. *Recurrent Neural Networks - RNNs*

Recurrent Neural Networks (RNNs) process the elements of a sequence one by one and recurrently apply certain tasks to each element, whose computation is dependent on the previous computations. This means that the model keeps memory of the previous elements with respect to the target. This memory is crucial for RNNs since the semantic meaning of sequences often relies on previous elements.

Another important aspect of RNNs is their availability to handle inputs of variable length, which is something that CNNs cannot do and which may have an impact on text classification tasks.

* + 1. *Word Embedding*

An example of feature-based pre-trained language representations is word embedding - word representations in vectors that have been learned from their context, following the hypothesis that similar words have similar contexts. The most common word embedding architectures are Word2vec, FastText, GloVe and BERT.

* + - 1. Global Vectors for Word Representation - *GloVe*

Global Vectors for Word Representation (GloVe)[15] is a log-bilinear regression model that uses the occurring counts of the words in the entire corpus when training, instead of only using a shallow window-based method such as that of Word2vec.

* + 1. *Bidirectional Encoder Representations from Transformers - BERT*

Bidirectional Encoder Representations from Transformers[19][20] is a recent paper published by researchers at Google AI Language. It has caused a stir in the machine learning community by presenting state-of-the-art results in a wide variety of NLP tasks. They propose the Masked Language Model (MLM) with a deep bidirectional Transformer, which randomly masks parts of the unlabeled input, so that the model learns how to predict the masked elements from both

directions.

Additionally, the model also learns how to predict the next sentence with Next Sentence Prediction (NSP).

BERT has only two steps: pre-training and fine-tuning. Multiple successful examples of using BERT have been proposed for text classification. Chang et al[19], for example, have proposed X-BERT - a system which fine-tunes BERT by assembling different models trained on heterogeneous label clusters. They have achieved a precision of 67.80% in an Extreme Multi-label Text Classification.

How BERT works is that it makes use of the Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms - an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT’s goal is to generate a language model, only the encoder mechanism is necessary. As opposed to directional models that read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore, it is considered bidirectional, although it would be more accurate to say that it is non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

Figure 6 shows a high-level description of the Transformer encoder. The input is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors of size H, in which each vector corresponds to an input token with the same index. When using language models, there is a challenge of defining a prediction goal. Many models predict the next word in a sequence (e.g. “The child came home from \_\_\_”) - a directional approach, which inherently limits context learning.



Figure 6: Vanilla BERT

## *Evaluation Metrics*

Multi-label text classifiers return predictions per sample in the form of a set of one or more labels evaluated against the ground truth set of labels. They might predict all the labels, in a larger or smaller proportion, or none of the labels, therefore entailing us to use metrics that capture the different degrees of effectiveness.

* + 1. *Precision, Recall and F1 Score*

In text classification, it is common to use precision, recall and F measure metrics. To understand them we should first draw a contingency table (see Table 1).

|  |  |  |
| --- | --- | --- |
|  | Relevant | Not Relevant |
| Retrieved (1) | True Positives (TP) (1-β) | False Positives (FP) (α) |
| Not Retrieved (0) | False Negatives (FN) (β) | True Negatives (TN) (1-α) |

Table 1: Confusion Metrix

We can then calculate precision and recall (E.g. 3.1 and 3.2)

In our context and having a sample as reference, precision is defined as by how many of the predicted labels belong to that sample, while recall is defined by how many relevant labels from that sample were found. A metric that compromises between these two is F measure, which is formally defined as the weighted harmonic mean of precision and recall (E.g. 3.3)

With α = 0.5 or β = 1 both precision and recall have the same weight, which corresponds to the most commonly used parameters for F measure, called F1 measure.

In multi-label classification, this is nevertheless more complicated since these metrics can be calculated at the level of the sample or the class. For that reason, two averaging methods are usually proposed in the literature - i.e. the macro average and the micro average.

Micro-averaged scores are calculated per sample instance ***j*** (E.g. 3.4 and 3.5), whereas macro averaged scores are calculated by class ***m*** and averaged over all the classes ***k*** (E.g. 3.6 and 3.7)***.***

The micro and macro-averaged F measure scores are then calculated by plugging these micro and macro scores into the F measure formula shown in E.g. 3.3.

The main difference between these metrics is that macro-averaged F1 treats all labels equally, whereas micro-averaged F1 gives the same weight for decisions that the classifier takes in each document. In an unbalanced dataset, larger classes have a bigger impact[5], which is not considered in macro-averaged F1 measure. Because the dataset is unbalanced in terms of samples per class (see Section 5.1), I decided to use micro-averaged F1 scores.

* + 1. *Hamming Loss*

The hamming loss is an example-based metric common in multi-label classification (E.g. 3.8). It calculates the symmetric difference (Δ) between the set of predicted labels ***h*** and the set of true labels ***y. F***or every label ***l*** and sample ***d***, it counts all elements in this difference (which represent the wrong predictions) and normalizes them over the number of labels and samples. This is a loss function, so the optimal value is zero.

* + 1. *Micro AUROC*

Area Under the Receiver Operating Characteristic Curve (AUROC) is a metric composed of two parts.

First, Receiver Operating Characteristic (ROC) which is based on plotting the TPs on the y-axis of a graph and the FPs on its x-axis, and which creates a curve on the plot and tells us how effective our classifier is. The more data points on the northwest side of the plot, the more effective the classifier is.

Once the ROC is plotted, the Area Under the Curve AUC is calculated, which is a number between 0 and 1. Random guessing usually produces an AUROC of 0.5, so our classifier should always have more than that. Similarly, to the micro and macro averaging scores of Section 6.6.1, we can either obtain AUC by scoring all labels or samples. In this case, I decided to use the micro-averaged score, which means that the ROC is calculated for every text sample.

* + 1. *Cross-Validation (CV)*

Cross-validation consists of creating **k** classifiers and dividing our dataset in **k** folds, which is subsequently split differently in training, validation and test subsets. Then the classifiers are trained using the training folds and the hyper parameters are found by experimenting with different settings and validating the results against our validation folds. Once we have found the best parameters, the final models are tested on the test folds.

## *Focal Loss*

The focal loss was designed to address the scenario in which there is an extreme imbalance between the classes. I introduced the focal loss from the cross-entropy (CE) loss for binary classification[18].

## Experimental Setups

This section will show the different system implementations and experimental setups of the project. These experiments explored multi-label text classification, since each sample or text is labeled and can be assigned from one up to five labels. Furthermore, I took into consideration that it would be interesting to have a baseline with machine learning methods explained and see how they would compare against the deep learning methods.

## *Setup*

All the experiments were carried out in Jupyter notebooks, using Google collaboration tools with one NVIDIA Tesla T4 GPU and 12GB of RAM.

## *Evaluation and Cross-Validation*

The classifiers were evaluated with the implementation from Scikit Learn for the multi-label stratified cross-validation. All experiments were carried out with 5-fold cross-validation, instead of the more common 10-fold, since some of the classifiers were very computationally expensive. Therefore, I first saved the indices of the samples for each fold with a split of 80% training, 10% validation and 10% test. I then tried each classifier on each fold and fine-tuned the hyper-parameters with the validation set. Once I found the best hyper-parameters, I used the trained classifier to predict the test set labels. The results were obtained from each fold and averaged over 5.

## *Machine Learning Methods*

The baseline experiments with the multinomial Naive Bayes, SVM and logistic regression algorithms were carried out using Scikit Learn’s implementations: MultinomialNB, LinearSVC and logistic regression, respectively.

For the first step in my experiments with machine learning methods, I found the best parameters for each algorithm through GridSearchCV with a Pipeline, for the training and validation data. GridSearchCV runs each classifier meticulously with different parameters, while performing cross-validation with the five-folds, which then returns the best parameters and scores. The Pipeline (see Figure 7), on the other hand, helped me combine transforms that I did to the data with the classifier. I chose an OVR transform approach for the classifiers, which means that a classifier was created for each class.

Figure 7: Pipeline Model for Classic Methods

## *Deep Learning Methods*

Besides the use of classic ML approaches, the aim is to train deep learning models, especially with sequence to sequence linking like LSTMs and GRUs, which shows a good result in a NLP test. Naturally, the biggest challenge is to choose the right architecture, to compare between the few models such as Vanilla LSTM, LSTM with CNN, as well as the more complex models like Pooled RNN and Max pool text CNN.

I tried to find the best parameters for each network and to focus on the following parameters: the learning rate, batch size, number of epochs, network weight initialization, number of neurons in the hidden layer and activation functions.

Each model was evaluated by the same metrics, as explained in section 6.6.

* + 1. *The Architecture of the Models*

This section will explain the architecture of each model. I added embedding layers in all the models.

* + - 1. Vanilla LSTM

A Vanilla LSTM is a LSTM model with a single hidden layer of LSTM units and a dense output layer, used to make a prediction (see Figure 8).

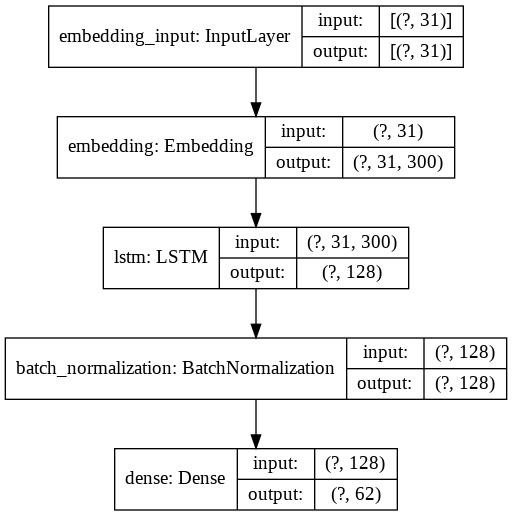


Figure 8: Vanilla LSTM Architecture

* + - 1. 2LSTM and CONV

Many Hybrid models have been developed to combine LSTM and CNN architectures to capture local and global features of sentences and documents[21]. Here multiple hidden LSTM layers are built and then stacked one on top of another, in what is referred to as a Stacked LSTM model.

A LSTM layer requires three-dimensional input, and LSTMs, by default, will produce two-dimensional output as an interpretation from the end of the sequence. You can address this by giving the LSTM output a value for each time step in the input data by setting the return\_sequences = True argument on the layer. This allows to have 3D output from the hidden LSTM layer as input to the next feed of the convolution layers to generate the document representation (see Figure 9).

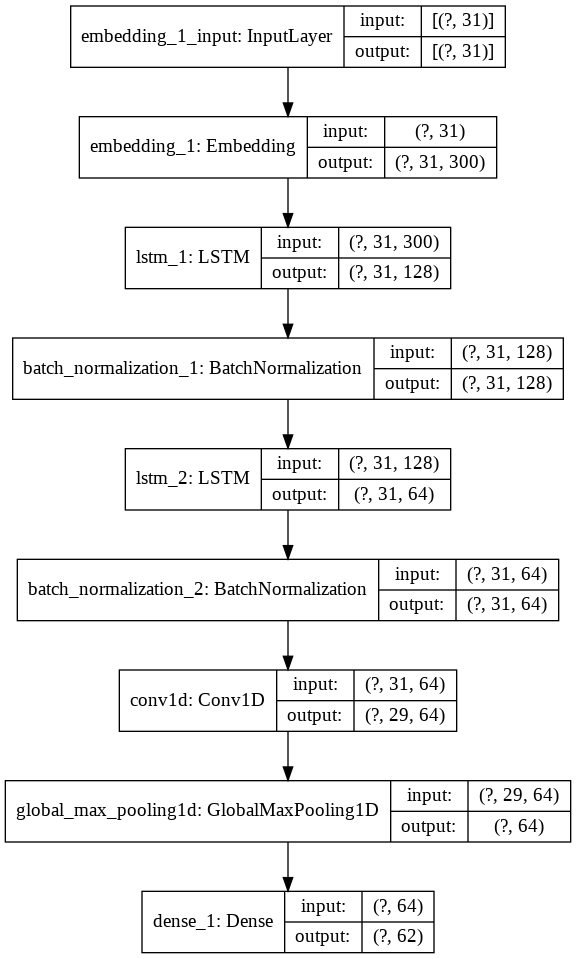


Figure 9: Hybrid Model - 2 LSTM and CONV Architecture

* + - 1. 2 BiLSTM with Attention

Some sequence prediction problems can be beneficial to allow the LSTM model to learn the input sequence both forward and backward and concatenate both interpretations. That is called a Bidirectional LSTM. We can implement a Bidirectional LSTM, for univariate time series forecasting, by wrapping the first hidden layer in a wrapper layer called Bidirectional and adding the attention layer (see Figure 10). Attention has become an increasingly popular concept and a useful tool in developing deep learning models for NLP[22].

In short - attention in language models can be interpreted as a vector of importance weights. In order to predict a word in a sentence, we estimate, using the attention vector, how strongly it correlates with or “attends to” other words, and then take the sum of their values weighted by the attention vector as the approximation of the target.

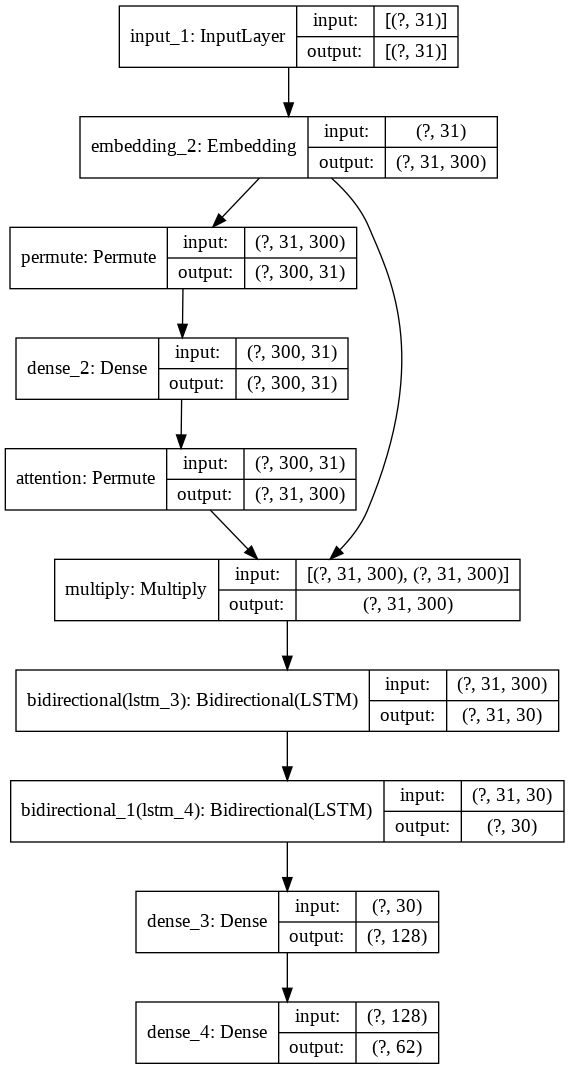


Figure 10: 2 BiLSTM with Attention Architecture

* + - 1. Pooled RNN

You can take the 2 BiLSTM and pool the max and average variables from them and then connect them. Then, you can use dropout for regularization and dance layer with 62 units and sigmoid activation (see Figure 11).

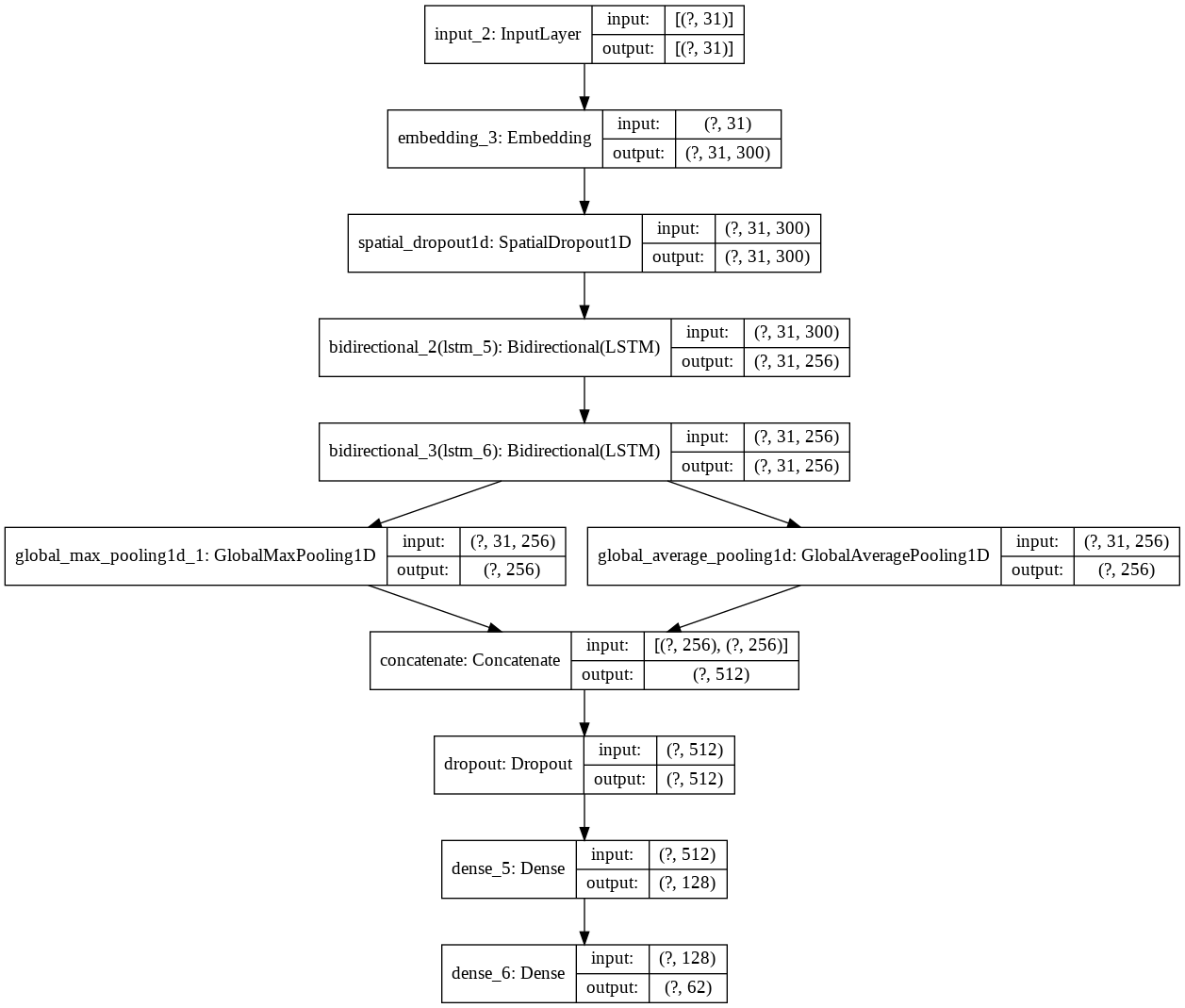


Figure 11: Pooled RNN Architecture

* + - 1. Max Pool Text CNN

Similarly, as the section above replaces the BiLSTM with CONV and Max pooling, you can connect them to dropout for regularization and try to predict the correct labels[23] (see Figure 12).

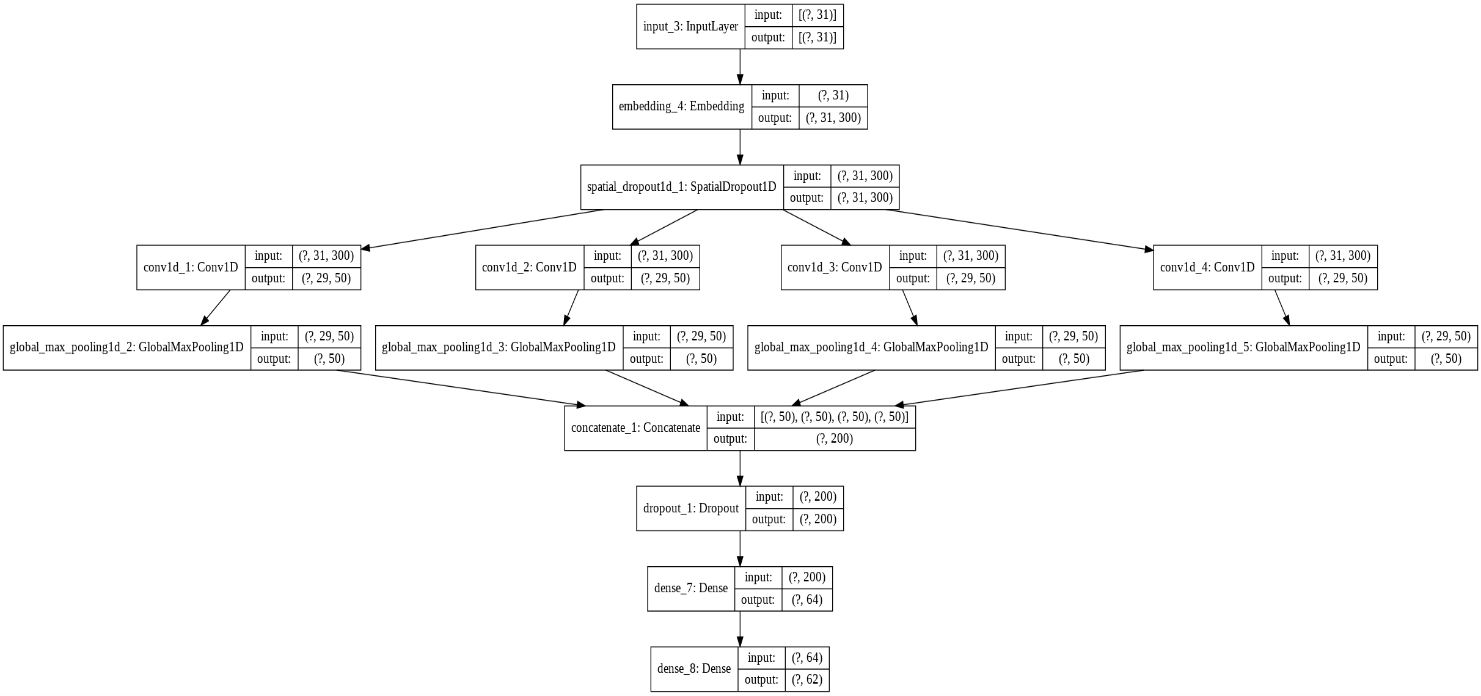


Figure 12: Max Pool Text CNN Architecture

* + - 1. BERT

In this project, we take a BERT base (Figure 13), including 12 layers, 768 hidden units, 12 heads and 110M parameters [29,30].

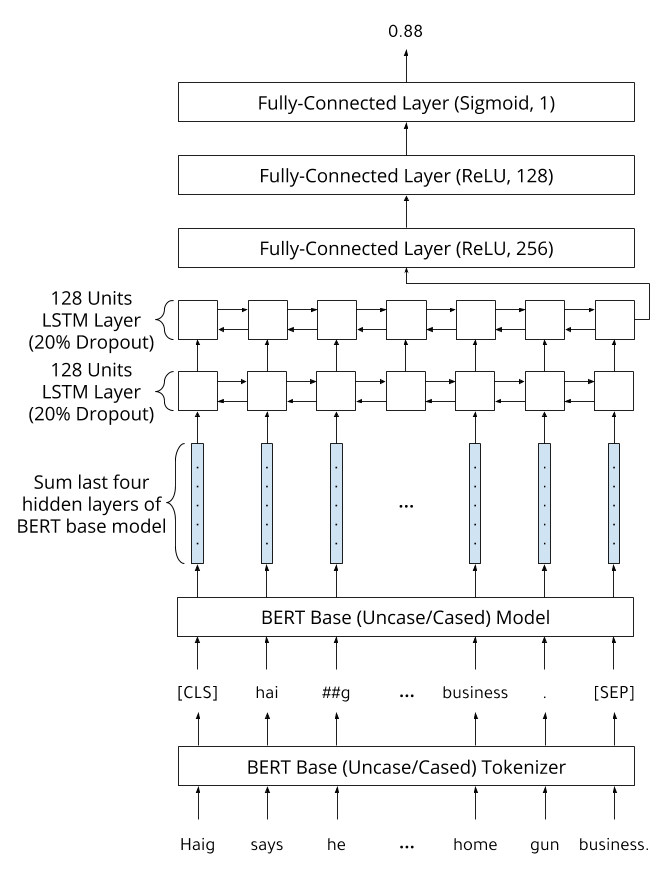


Figure 13: BERT Based Architecture

## Methods

## *Parameters Selection*

This section will show the selected parameters for each model with GridSearchCV.

* + 1. *TF-IDF*

I run TF-IDF with GridSearchCV. The parameters selected for the experiment were with the highest number of max features, N-grams and max df. The scoring methods were micro F1 score, and CV was set to five (see Table 2). In Figure 14 we can see how the F1 score rises with the number of features; therefore, the max df does not impact the F1 score.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Max df | N-gram | Max features |
| Naïve Bayes | 0.1 | 2 | 20000 |
| LinearSVC | 0.5 | 2 | 20000 |
| Logistic Regression | 0.1 | 3 | 20000 |

Table 2: Selected Parameters in Classic Model

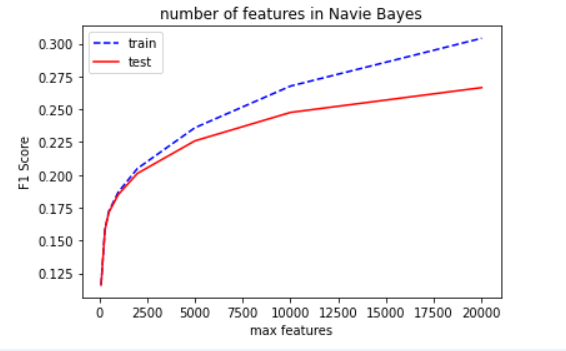
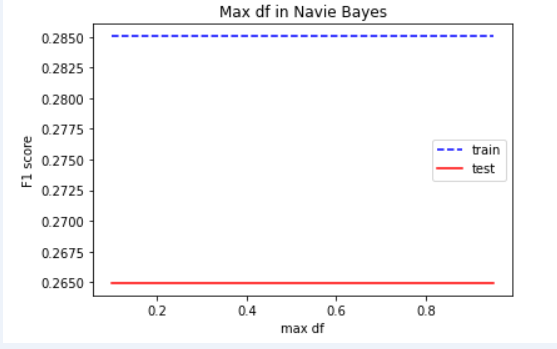


Figure : The F1 Score vs Max Feature and Max DF

Figure 12: the F1 score vs max feature and max df

* + 1. *The Models’ Parameter Selections*

In both LinearSVC and Naïve Bayes I tested the alpha parameters, but it represents something different in each method.

For NB, it represents the additive smoothing parameter (see Figure 15) the smaller the alpha (in log scale) the better the F1 score.

For SGD, it performs regulation for the learning rate and logistic regression tested C, which also represents the regularization for the model.

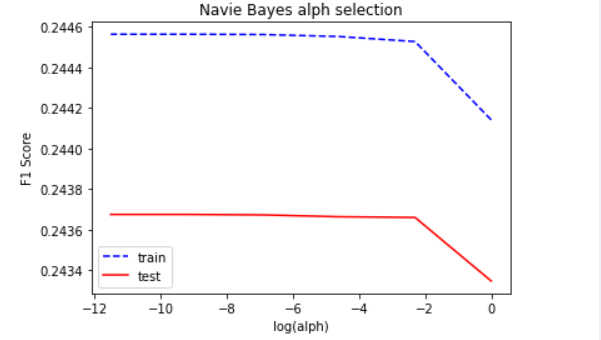


Figure 15: The F1 Score vs Log (alpha)

* + 1. *Threshold*

For classification problems that have a severe class imbalance, the default threshold can result in poor performance. As such, a simple and straightforward approach to improving the performance of a classifier that predicts probabilities on an imbalanced classification problem, is to tune the threshold used to map probabilities to class labels.

In some cases, such as when using ROC Curves and Precision-Recall Curves, the classifier's best or optimal threshold can be calculated directly. In other cases, it is possible to use a grid search to tune the threshold and locate the optimal value[26].

In this project, every classifier has its own threshold value. For the classic methods, the best threshold is 0.2. The best option for deep learning is 0.3, and for BERT, it is 0.4.

In Table 3, you can see the result of moving the threshold for pool max text CNN.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Threshold | Pre | Rec | F1 Score | Hamming Loss |
| 0.1 | 0.0504 | 0.9336 | 0.0957 | 0.3809 |
| 0.2 | 0.2374 | 0.5983 | 0.34 | 0.0501 |
| 0.3 | 0.5503 | 0.3639 | 0.4381 | 0.0201 |
| 0.4 | 0.7623 | 0.2243 | 0.3466 | 0.0182 |
| 0.5 | 0.8788 | 0.118 | 0.2093 | 0.0194 |
| 0.6 | 0.9381 | 0.0503 | 0.0955 | 0.0206 |
| 0.7 | 0.9721 | 0.0126 | 0.0249 | 0.0213 |
| 0.8 | 0.989 | 0.001 | 0.002 | 0.0216 |
| 0.9 | 1.0 | 0.0 | 0.0 | 0.0216 |

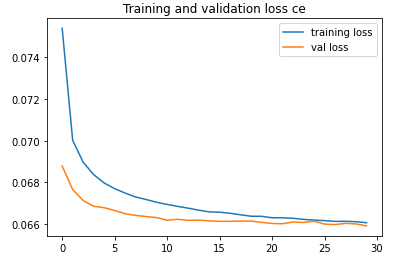
Table 3: The Effect of Threshold on Precision, Recall F1 Score and Hamming Loss in Max Pool Text CNN

* + 1. *Loss*

In this project, I compare two different loss functions - one is built-in Keras “binary cross-entropy” (CE), and the other one is FocalLoss (FL) (see section 6.8), which according to written literature, should work better in imbalanced data.

From Figure 16, we can see that FL has a lower loss than CE, 0.0065 vs. 0.066.

Figure 16: Focal Loss vs Cross Entropy



## Results

In table 4, you can see the results of my experiments on micro-averaged precision, recall and F1 score, as well as AUROC.

What is interesting is 7 out of 9 the models tend to have higher precision than Recall except LinearSVC and Logistic Regression.

Furthermore, it is worth noting that BERT is not only the most effective model with micro-averaged F1 scores of 0.51, but it also presents the best scores - better than all other classifiers in AUROC - 0.93 and precision - 0.66 . The LinearSVC got the best Recall - 0.54.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Model |  |  |  |  |
| Naïve Bayes | 0.44 | 0.41 | 0.89 | 0.42 |
| LinearSVC | 0.18 | **0.54** | 0.86 | 0.27 |
| Logistic Regression | 0.21 | 0.44 | 0.82 | 0.28 |
| Vanilla LSTM | 0.49 | 0.38 | 0.89 | 0.43 |
| 2 BiLSTM and Attention | 0.51 | 0.32 | 0.87 | 0.4 |
| 2 LSTM +CONV | 0.51 | 0.32 | 0.88 | 0.4 |
| Pooled RNN | 0.52 | 0.36 | 0.9 | 0.43 |
| MaxPool Text CNN | 0.52 | 0.43 | 0.91 | 0.47 |
| BERT | **0.66** | 0.42 | **0.93** | **0.51** |

Table 4: Micro-averaged Precision, Recall and F1, and AUC for Every Classifier on the Test Set

In Table 8 (Appendix A), we can see the F1 scores of the models per class and each class's support in the test data.

Looking at the best model results, BERT, you can see the highest scores were obtained with the majority of the classes (49 from 62).

Also, when a class was hard to classify, it was hard for all the classifiers, such as "life."

In Table 5, you can see the top five labels, and that BERT got the best results, except in "Blockchain", where Max pool text CNN is better.

Table 6 shows the bottom labels, which all classifiers have a hard time predicting, but the classic methods predicted a better result.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Label | Support | NB | SVM | LR | LSTM | 2BILSTM | Pooled RNN | Max Pool Text CNN | BERT |
| Blockchain | 14910 | 0.73 | 0.68 | 0.66 | 0.74 | 0.7 | 0.7 | 0.9 | 0.82 |
| WordPress | 905 | 0.62 | 0.56 | 0.75 | 0.78 | 0.72 | 0.75 | 0.78 | 0.81 |
| Sports | 5209 | 0.65 | 0.46 | 0.44 | 0.6 | 0.52 | 0.65 | 0.68 | 0.76 |
| JavaScript | 4666 | 0.6 | 0.54 | 0.55 | 0.67 | 0.6 | 0.63 | 0.71 | 0.75 |
| Dogs | 774 | 0.52 | 0.65 | 0.64 | 0.65 | 0.63 | 0.68 | 0.7 | 0.74 |

Table : Top 5 Labels

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Label | Support | NB | SVM | LR | LSTM | 2BILSTM | Pooled RNN | Max Pool Text CNN | BERT |
| Life | 3447 | 0.1 | 0.1 | 0.1 | 0 | 0.0 | 0.01 | 0 | 0 |
| Work | 1665 | 0.17 | 0.12 | 0.14 | 0.12 | 0.05 | 0.11 | 0.15 | 0.12 |
| News | 3216 | 0.17 | 0.11 | 0.11 | 0.13 | 0.06 | 0.01 | 0.18 | 0.2 |
| Culture | 3098 | 0.14 | 0.09 | 0.12 | 0.12 | 0.06 | 0.06 | 0.13 | 0.13 |
| Blog | 1185 | 0.15 | 0.06 | 0.09 | 0.1 | 0.13 | 0.0 | 0.12 | 0.07 |

Table : Bottom 5 Labels

We can see in Table 7 that the top three labels represent the tags with the most examples and the bottom three rows, represent those with the least examples. You can see that the amount of samples does not necessary guarantee a good result. For example, “Dogs” tag got a high score of 0.74 , even though it had few examples and this is probably due to the fact that “Dogs” was unique in the data. On the contrary, “Software Engineering”, which is close to other tags got a low score.

In Figure 14 it can be seen that the number of samples does not guarantee a high score. For example “WordPress” (905) and “Software Engineering” (576) got F1 scores of 0.81 and 0.22, respectively.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Label | Support | NB | SVM | LR | LSTM | 2BILSTM | Pooled RNN | Max Pool Text CNN | BERT |
| Blockchain | 14910 | 0.73 | 0.68 | 0.66 | 0.74 | 0.7 | 0.7 | 0.9 | 0.82 |
| Politics | 11108 | 0.54 | 0.47 | 0.47 | 0.53 | 0.47 | 0.54 | 0.57 | 0.63 |
| Life Lessons | 9719 | 0.32 | 0.26 | 0.25 | 0.32 | 0.24 | 0.24 | 0.31 | 0.3 |
| Black Lives Matter | 974 | 0.31 | 0.17 | 0.18 | 0.23 | 0.22 | 0.2 | 0.31 | 0.34 |
| Dogs | 774 | 0.52 | 0.65 | 0.64 | 0.65 | 0.63 | 0.68 | 0.7 | 0.74 |
| Software Engineering | 576 | 0.22 | 0.08 | 0.13 | 0.15 | 0.09 | 0.0 | 0.12 | 0.15 |

Table : The Effect of the Number of Supports on Label Classification

Figure 17: Normalization of the Number of Samples per Class (blue line); the F1 Score for Each Class from BERT (red line)

In Table 8, you can see some prediction labels and their actual labels.

You can see that sometimes the classifier was able to retrieve one of the sets of the label and yet other times it misclassified the predicted label.

|  |  |  |
| --- | --- | --- |
| **Text** | **True Labels** | **Predicted Labels** |
| **Top Things to do in San Francisco Today** | Travel | Travel |
| **Ways to Effectively Monitor and Improve Employee Morale** | Leadership | Leadership |
| **Books to Kick Start your New Year’s Resolution** | Business, Life Lessons, Health, Self-improvement | Books |
| **Traction Trumps Everything** | Start-up | Politics |
| **A Great Psychologist Tests which Door you are Afraid to Enter** | Self-improvement | Psychology |
| **Programming Practices may not Stand the Test of Time** | Programming, Software engineering | Programming |
| **Book Review Hold on to Your Love** | Books, Fiction, Writing | Books |
| **How to Remain Productive No Matter What** | Life lessons, Productivity, Self-improvement | Productivity,  Self-improvement |

Table 8: Examples of Prediction on Test Set

## Discussion

We have tested different multi-label text classification approaches to try to give the best suitable labels for each title. This was a challenging task due to various reasons, mainly related to limits on data content and structure. We know that some of the labels come from the text body (rather than from information available in the title), thus remaining concealed relative to the model. Therefore, the model has no actual capacity to predict them. We can also see the keywords imports in the text from table 8 implying that if a tag appears in the title, it naturally got a high score (being classified directly to this same tag). But, sometimes when the tag has more than a single underlying meaning it can also fail the model. These results may explain a high score in AUROC we generally observe in all the models, suggesting it is due to high success rate on trivial predictions, but failure (mainly of the simpler models) on the harder cases. Looking deeper into the results, it seems fair for some non-trivial labels, but the problem was perhaps too hard for others. Some cases of failed classification can be understood, as we can see in tables 8 and 9. Importantly, we can see that Bert outperformed all other classifiers with an F1 score of 0.51. However, the best Recall but the worst Precision was achieved with LinerSVC 0.54 and 0.18, respectively. In model that precision is high and recall is low that mean that our model is very picky in what it classify but if classify correctly is probability right but it miss a lot of actual true labels in contrast when recall is high and precision is low that shows any of was wrongly classified from that we will prefer in this project It can explain that if the model could retrieve the relevant title, it got the probability of classified correctly. Also, due to time limitations, I did not fine tune and optimize the Bert model because every run took so much time, so there is also a potential room for improvement here.

Furthermore, for labels that were very hard to predict, usually consisting of abstract labels such as blog, business, culture and life style that do not have unique key words; it turned out that for these words all the models had difficulty, but the classic methods got a better result than some DL models. Perhaps because they give less focus to keywords. It was interesting to find that the number of samples in labels did not have any significant effect on the result. The labels that achieved good results are labels that have a unique keyword such as “android” or “dog“ compared to “life.” I think that if we invest more efforts on clustering the labels (into families), the result can be improved and the model may become more realistic. We see that from classic cases of title mislabeling in tech cluster – with labels such as Programming and software engineer. For future work, we can try to best tune the Bert model and try some of the newest models published like GPT-3 besides that for the other deep learning methods. We could try different embedding, such as fast text, Elmo. We could also try using deep graph networks that take the hierarchy of structured data (medium website) into account.

## References

1. <https://www.kaggle.com/harrisonjansma/medium-stories>
2. <https://medium.com/topics>
3. Li, Y., Sun, G., & Zhu, Y. (2010, October). Data imbalance problem in text classification. In *2010 Third International Symposium on Information Processing* (pp. 301-305). IEEE.‏
4. Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011, June). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies* (pp. 142-150).‏
5. Manning, C., Raghavan, P., & Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, *16*(1), 100-103.‏
6. Campos, R., Mangaravite, V., Pasquali, A., Jorge, A. M., Nunes, C., & Jatowt, A. (2018, March). YAKE! collection-independent automatic keyword extractor. In *European Conference on Information Retrieval* (pp. 806-810). Springer, Cham.
7. Ramos, J. (2003, December). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning* (Vol. 242, pp. 133-142).‏
8. Qiang, J., Qian, Z., Li, Y., Yuan, Y., & Wu, X. (2020). Short text topic modeling techniques, applications, and performance: a survey. *IEEE Transactions on Knowledge and Data Engineering*.‏
9. Newman, D., Lau, J. H., Grieser, K., & Baldwin, T. (2010, June). Automatic evaluation of topic coherence. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics* (pp. 100-108).‏
10. Mimno, D., Wallach, H., Talley, E., Leenders, M., & McCallum, A. (2011, July). Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing* (pp. 262-272).‏
11. Herrera, F., Charte, F., Rivera, A. J., & Del Jesus, M. J. (2016). Multi-label classification. In *Multi-label Classification* (pp. 17-31). Springer, Cham.‏
12. Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval* (Vol. 463). New York: ACM press.‏
13. Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and Naive Bayes. In *Advances in neural information processing systems* (pp. 841-848).‏
14. Peng, H., Li, J., Wang, S., Wang, L., Gong, Q., Yang, R., & He, L. (2019). Hierarchical taxonomy-aware and attentional graph capsule RCNNs for large-scale multi-label text classification. *IEEE Transactions on Knowledge and Data Engineering*.‏
15. Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532-1543).‏
16. Zhang, M. L., Li, Y. K., Liu, X. Y., & Geng, X. (2018). Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, *12*(2), 191-202.‏
17. Wu, W., Gao, X., & Gao, S. (2006, January). One-versus-the-rest (OVR) algorithm: An extension of common spatial patterns (CSP) algorithm to multi-class case. In *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference* (pp. 2387-2390). IEEE.‏ Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense
18. object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980-2988).‏
19. Chang, W. C., Yu, H. F., Zhong, K., Yang, Y., & Dhillon, I. X-BERT: eXtreme Multi-label Text Classification using Bidirectional Encoder Representations from Transformers.‏
20. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.‏
21. Zhou, C., Sun, C., Liu, Z., & Lau, F. (2015). A C-LSTM neural network for text classification. *arXiv preprint arXiv:1511.08630*.‏
22. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).‏
23. Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.‏
24. Quan, X., Kit, C., Ge, Y., & Pan, S. J. (2015, June). Short and sparse text topic modeling via self-aggregation. In *Twenty-fourth international joint conference on artificial intelligence*.‏
25. Chang, J., Gerrish, S., Wang, C., Boyd-Graber, J. L., & Blei, D. M. (2009). Reading tea leaves: How humans interpret topic models. In *Advances in neural information processing systems* (pp. 288-296).‏
26. Fan, R. E., & Lin, C. J. (2007). A study on threshold selection for multi-label classification. *Department of Computer Science, National Taiwan University*, 1-23.
27. <https://scikit-learn.org/stable/modules/generated/sklearn.utils.class_weight.compute_class_weight.html>
28. King, G., & Zeng, L. (2001). Logistic regression in rare events data. *Political analysis*, *9*(2), 137-163.‏
29. <https://huggingface.co/transformers/pretrained_models.html>
30. <https://huggingface.co/bert-base-uncased>

# Appendix A

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Label | Support | NB | SVM | LR | LSTM | 2BILSTM | Pooled RNN | Max Pool Text CNN | BERT |
| Android | 2742 | 0.6 | 0.47 | 0.53 | 0.63 | 0.62 | 0.61 | 0.68 | 0.72 |
| Apple | 1482 | 0.51 | 0.47 | 0.54 | 0.59 | 0.57 | 0.58 | 0.62 | 0.66 |
| Art | 4710 | 0.34 | 0.24 | 0.21 | 0.38 | 0.37 | 0.39 | 0.41 | 0.46 |
| Artificial Intelligence | 4269 | 0.49 | 0.38 | 0.41 | 0.51 | 0.51 | 0.5 | 0.55 | 0.56 |
| Big Data | 1142 | 0.38 | 0.2 | 0.25 | 0.4 | 0.36 | 0.4 | 0.43 | 0.45 |
| Black Lives Matter | 974 | 0.31 | 0.17 | 0.18 | 0.23 | 0.22 | 0.2 | 0.31 | 0.34 |
| Blockchain | 14910 | 0.73 | 0.68 | 0.66 | 0.74 | 0.7 | 0.7 | 0.9 | 0.82 |
| Blog | 1185 | 0.15 | 0.06 | 0.09 | 0.1 | 0.13 | 0.0 | 0.12 | 0.07 |
| Books | 2719 | 0.45 | 0.4 | 0.44 | 0.49 | 0.48 | 0.48 | 0.51 | 0.55 |
| Business | 5483 | 0.23 | 0.2 | 0.22 | 0.2 | 0.18 | 0.19 | 0.18 | 0.14 |
| Culture | 3098 | 0.14 | 0.09 | 0.12 | 0.12 | 0.06 | 0.06 | 0.13 | 0.13 |
| Data | 1239 | 0.21 | 0.19 | 0.26 | 0.2 | 0.25 | 0.19 | 0.3 | 0.25 |
| Data Science | 2555 | 0.4 | 0.26 | 0.26 | 0.4 | 0.34 | 0.39 | 0.43 | 0.45 |
| Design | 8093 | 0.5 | 0.45 | 0.44 | 0.51 | 0.5 | 0.52 | 0.55 | 0.59 |
| Dogs | 774 | 0.52 | 0.65 | 0.64 | 0.65 | 0.63 | 0.68 | 0.7 | 0.74 |
| Education | 4787 | 0.41 | 0.34 | 0.35 | 0.42 | 0.35 | 0.42 | 0.46 | 0.49 |
| Energy | 1264 | 0.45 | 0.31 | 0.36 | 0.4 | 0.34 | 0.47 | 0.51 | 0.57 |
| Entrepreneurship | 5163 | 0.26 | 0.19 | 0.2 | 0.25 | 0.25 | 0.27 | 0.27 | 0.25 |
| Environment | 2299 | 0.4 | 0.28 | 0.29 | 0.37 | 0.31 | 0.44 | 0.47 | 0.57 |
| Feminism | 2164 | 0.36 | 0.3 | 0.33 | 0.37 | 0.33 | 0.39 | 0.41 | 0.44 |
| Fiction | 2632 | 0.21 | 0.11 | 0.12 | 0.14 | 0.04 | 0.08 | 0.2 | 0.27 |
| Food | 5578 | 0.58 | 0.53 | 0.48 | 0.58 | 0.49 | 0.63 | 0.64 | 0.68 |
| Google | 1409 | 0.4 | 0.31 | 0.51 | 0.49 | 0.48 | 0.46 | 0.51 | 0.53 |
| Health | 5575 | 0.5 | 0.38 | 0.39 | 0.45 | 0.41 | 0.51 | 0.53 | 0.58 |
| History | 2370 | 0.27 | 0.18 | 0.25 | 0.26 | 0.21 | 0.15 | 0.29 | 0.38 |
| Humor | 3145 | 0.18 | 0.1 | 0.11 | 0.08 | 0.07 | 0.07 | 0.14 | 0.19 |
| Investing | 3447 | 0.39 | 0.29 | 0.3 | 0.38 | 0.36 | 0.35 | 0.45 | 0.48 |
| iOS | 1471 | 0.46 | 0.33 | 0.36 | 0.53 | 0.49 | 0.49 | 0.57 | 0.61 |
| JavaScript | 4666 | 0.6 | 0.54 | 0.55 | 0.67 | 0.6 | 0.63 | 0.71 | 0.75 |
| Jobs | 1350 | 0.31 | 0.21 | 0.27 | 0.34 | 0.12 | 0.35 | 0.32 | 0.43 |
| Journalism | 1644 | 0.33 | 0.22 | 0.25 | 0.37 | 0.3 | 0.39 | 0.42 | 0.48 |
| Leadership | 4792 | 0.37 | 0.26 | 0.3 | 0.4 | 0.34 | 0.39 | 0.42 | 0.47 |
| Life | 3447 | 0.1 | 0.1 | 0.1 | 0 | 0.0 | 0.01 | 0 | 0 |
| Life Lessons | 9719 | 0.32 | 0.26 | 0.25 | 0.32 | 0.24 | 0.24 | 0.31 | 0.3 |
| Love | 7525 | 0.38 | 0.3 | 0.31 | 0.36 | 0.34 | 0.39 | 0.42 | 0.48 |
| Machine Learning | 5260 | 0.57 | 0.42 | 0.41 | 0.57 | 0.52 | 0.54 | 0.61 | 0.64 |
| Marketing | 4191 | 0.39 | 0.32 | 0.33 | 0.44 | 0.42 | 0.43 | 0.46 | 0.47 |
| Mobile | 1105 | 0.28 | 0.19 | 0.22 | 0.3 | 0.26 | 0.18 | 0.32 | 0.35 |
| Movies | 3496 | 0.57 | 0.39 | 0.4 | 0.54 | 0.5 | 0.56 | 0.62 | 0.68 |
| Music | 5491 | 0.56 | 0.46 | 0.45 | 0.53 | 0.48 | 0.59 | 0.61 | 0.7 |
| News | 3216 | 0.17 | 0.11 | 0.11 | 0.13 | 0.06 | 0.01 | 0.18 | 0.2 |
| Parenting | 2540 | 0.4 | 0.33 | 0.38 | 0.45 | 0.41 | 0.46 | 0.49 | 0.56 |
| Photography | 3390 | 0.5 | 0.43 | 0.5 | 0.52 | 0.49 | 0.53 | 0.56 | 0.61 |
| Poetry | 3091 | 0.23 | 0.13 | 0.14 | 0.23 | 0.1 | 0.23 | 0.18 | 0.27 |
| Politics | 11108 | 0.54 | 0.47 | 0.47 | 0.53 | 0.47 | 0.54 | 0.57 | 0.63 |
| Productivity | 5115 | 0.39 | 0.23 | 0.24 | 0.38 | 0.33 | 0.35 | 0.4 | 0.44 |
| Programming | 5314 | 0.44 | 0.34 | 0.33 | 0.47 | 0.43 | 0.46 | 0.52 | 0.57 |
| Psychology | 2836 | 0.17 | 0.1 | 0.1 | 0.06 | 0.04 | 0.1 | 0.15 | 0.18 |
| Science | 2578 | 0.33 | 0.21 | 0.25 | 0.18 | 0.16 | 0.32 | 0.4 | 0.47 |
| Self-improvement | 8997 | 0.33 | 0.25 | 0.25 | 0.33 | 0.29 | 0.32 | 0.33 | 0.36 |
| Social Media | 4846 | 0.49 | 0.44 | 0.44 | 0.51 | 0.47 | 0.51 | 0.53 | 0.56 |
| Software Engineering | 576 | 0.22 | 0.08 | 0.13 | 0.15 | 0.09 | 0.0 | 0.12 | 0.15 |
| Sports | 5209 | 0.65 | 0.46 | 0.44 | 0.6 | 0.52 | 0.65 | 0.68 | 0.76 |
| Startup | 5495 | 0.27 | 0.21 | 0.22 | 0.24 | 0.23 | 0.25 | 0.27 | 0.24 |
| Tech | 4281 | 0.22 | 0.14 | 0.15 | 0.14 | 0.12 | 0.12 | 0.21 | 0.16 |
| Technology | 4379 | 0.17 | 0.14 | 0.14 | 0.04 | 0.01 | 0.07 | 0.1 | 0.08 |
| Travel | 5541 | 0.53 | 0.37 | 0.39 | 0.49 | 0.46 | 0.53 | 0.55 | 0.62 |
| Web Development | 5244 | 0.48 | 0.41 | 0.39 | 0.52 | 0.49 | 0.5 | 0.53 | 0.57 |
| Women | 2596 | 0.3 | 0.32 | 0.34 | 0.35 | 0.33 | 0.34 | 0.37 | 0.41 |
| WordPress | 905 | 0.62 | 0.56 | 0.75 | 0.78 | 0.72 | 0.75 | 0.78 | 0.81 |
| Work | 1665 | 0.17 | 0.12 | 0.14 | 0.12 | 0.05 | 0.11 | 0.15 | 0.12 |
| Writing | 8120 | 0.4 | 0.35 | 0.39 | 0.42 | 0.41 | 0.39 | 0.44 | 0.48 |

Table 9: The Micro-averaged F1 Scores per Class in Each Classifier

**תקציר**

סיווג רב תוויות הוא אחת מתוך כמה משימות הקיימות בתחום עיבוד שפות טבעיות (NLP).

בפרויקט זה, אני שואף לתייג אוטומטית נתונים לא מובנים המסופקים בפורמט טקסט חופשי. קבלת מידע מטקסט חופשי היא אחד האתגרים הפתוחים הגדולים ביותר ב- NLP.

מטרתי היא לסדר את הטקסט החופשי בצורה מובנית ובכך לקבל ממנו מידע רלוונטי בעזרת שיטות מתחום עיבוד השפה ע"י תיוג מידע זה.

הדאטה נלקחה מאתר KAGGLE ונוצרה ע"י הריסון ג'ונסמה למטרת ניתוח שיטת כפיים אשר בצורה זאת נמדד טיבה של כתבה באתר MEDIUM בו מפרסמים אנשים בלוגים מכל התחומים.

השתמשתי בדאטה זו ובניתי מודלים אשר מטרתם היא לתייג באופן אוטומטית את כותרות הבלוגים.

בצעתי השוואות בין שיטות שונות. למשימה הזאת השוואנו בין שיטות קלאסיות כגון Naive Bayes, LinerSVC ו- logistic regression.

בכל השיטות האלו ניסיתי להגיע לתוצאה הטובה ביותר ע"י חיפוש הפרמטרים הטובים ביותר לכל מודל.

בנוסף לכך, השתמשתי בשיטות יותר מתקדמות של רשתות עמוקות כגון רשתות קונבולוציה, זיכרון עם שכבת אימבנדג של GLOVE ושכבות Attention. כל אלו הביאו לתוצאות טובות יותר אבל לא באופן משמעותי.

לבסוף, ניסיתי שיטה חדשנית אשר קידמה רבות את תחום עיבוד השפה ומכילה בתוכה טכניקות רבות. השיטה שנקראת BERT, הצליחה בשלל המשימות. שיטה זו באמת שיפרה את התוצאות בכלל המדדים כגון, AUC Precisionו- F1 Score.

כמו כן, למרות שהדאטה לא היתה מאוזנת מבחינת תגיות, התגיות עם מעט דוגמאות הצליחו כמו אלה עם הרבה, ולהיפך.

לסיכום הבנתי שלא כמות הדוגמאות היא זו שהביאה לתוצאה טובה אלה מילת המפתח בתוך הטקסט. זה אימת את התחושה ראשונית שלי שכותרות מכילות מידע מרכזי על גוף הטקסט אך לא את כולה, ולכן אפשר להוציא מתוך הכותרת חלק מהתגיות במידה ותגיות שהוגדרו הן לא משהו מופשט. תגיות מופשטות היו קשות לחיזוי וזה נראה בכל המודלים באופן שווה.