Quora Insincere Questions Classification

Maksim Chernikov, Elizaveta Chefanova, Julia Sosnina

May 22, 2023

Abstract

An existential problem for any major website today is how to handle toxic and divisive content. Quora (a platform that empowers people to learn from each other) wants to tackle this problem head-on to keep their platform a place where users can feel safe sharing their knowledge with the world.

A key challenge for Quora is to weed out insincere questions – those founded upon false premises, or that intend to make a statement rather than look for helpful answers.

Based on this Kaggle competition [1], we developed models that identify and flag insincere questions.

Introduction

The topic of assessing the sincerity of online text is not new and may seem trivial. However, with the widespread availability of fast and affordable internet, the need to address such issues remains relevant. Consequently, IT companies serving millions of users are continually striving to solve this problem. As technology advances, new and more sophisticated approaches and architectures are emerging, enabling more precise identification of insincere or "toxic" content within questions and comments.

The purpose of this work is to utilize different machine learning approaches to identify insincere questions from the vast pool of inquiries posted on the popular Quora website. In this context, an insincere question is defined as one that aims to make a statement rather than seeking genuinely helpful answers. We found the dataset for this task particularly intriguing, as it comprises authentic user questions from the platform.

As a result, the work on the project was organized based on the following steps:

- searching for existing articles on the topic
- selecting best approaches for baseline and SOTA for implementation
- exploratory data analysis
- data preprocessing
- developing baseline models based on such approaches as TF-IDF, Word2Vec, LSTM, Bert
- developing SOTA approach that is currently the best for sentiment analysis
- comparing metrics and report writing

Team

3 people took part in this project. The roles were distributed in the following way:

- *Maksim Chernikov*: lead the project and was responsible for developing the transformer models, design of the project repository, report writing
- *Elizaveta Chefanova*: was responsible for developing all of the baseline models and LSTM one, as well as conducting data preparation, report writing
- *Julia Sosnina*: conducted an analysis of the raw data, including exploratory data analysis, literature review on sentiment analysis, report writing

Literature review

TF-IDF gives the relevance of a term to a document by multiplying the term frequency by its inverse document frequency. TF-IDF lowers the weight of stop words like articles or prepositions but gives high importance to familiar words in a small set of documents [2]. An example of this in sentiment analysis is the study by Das and Chakraborty, where they used TF-IDF to convert words into scaled vectors to train an SVM model. Compared with a binary BOW model, the performance of the TF-IDF is much higher. They also found that the model works better by combining subsequent word negation (NWN, negating the word after 'not') with TF-IDF [3].

With the deepening of the study on deep learning, most current studies focus more on various artificial neural networks. Rhanoui et al., proposed a framework for document-level sentiment analysis. They used Doc2Vec to embed the documents as input and put them through a CNN plus BiLSTM framework. According to their research, the model outperformed others in newspaper article prediction with 90.66% accuracy [4].

Sousa et al., tested the performance of BERT on sentiment analysis and found that BERT outperforms all classical machine learning models like Naive Bayes and Support Vector Machines. After sentiment analysis on stock articles, they also provided relevant information for decision-making [5].

Dong et al., combined BERT with CNN to extract both local (CNN) and global (BERT) semantic features when doing sentiment analysis on commodity reviews and got an elevation on F1 score in comparison with pure BERT and pure CNN models [6].

It should be noted that most of the recent articles pointed us to the original works on transformers such as BERT (Bidirectional Encoder Representations from Transformers) [7] and ROBERTA (A Robustly Optimized BERT Pretraining Approach) [8].

Some authors in their papers note that despite the fact that transformer models, such as BERT, have led to impressive improvements in NLP tasks, along with other deep learning models, they are very difficult to interpret [9].

1 Exploratory Data Analisis

1.1 Type of questions analysis

Quora is a platform that enables individuals to express their thoughts and seek answers to their inquiries. As part of our study, we obtained a collection of user questions from the Quora service. Each question has been pre-labeled as either 0 or 1, indicating whether it is deemed insincere or not. In this context, "insincere" refers to questions that violate the platform's policy [10]. We thoroughly analyzed the provided dataset and identified the prohibited types of content that fall under this category.

It is important to note that by default, profanity and sexual content are not automatically blocked. However, in the case of the former, it must be accompanied by hate speech, harassment, or bullying in order for it to be considered prohibited content. This means that such content should be excluded even if there is no presence of explicit language. Furthermore, this specific type of content encompasses a broader range and includes even subtle indications of discrimination, reinforcement of stereotypes, or assumptions about a particular group of people based on immeasurable characteristics. For example, questions like "What percentage of women cheat on their hubbies?" or "Why do good-looking guys pair up with less attractive girls (I see this a lot)?" fall into this category.

Adult content is allowed on the platform with the exception of sexually explicit graphic content involving adults which appears to depict violence or sexual exploration/abuse, particularly involving minors (child abuse).

The platform strictly prohibits content related to suicide or self-harm, despite the presence of controversial training data on this subject. It is important to note that some of the training data includes requests for sharing information, strategies, or methods related to self-harm or suicide, which are often labeled as permissible.

Activities related to spam and scams are strictly prohibited on the platform. For instance, questions such as "Answer this and you win $1,000 cash: How sick would you get from eating 50 jars of honey?" fall under this category.

On the other hand, questions regarding psychoactive drugs are not automatically excluded from the platform. They are allowed and not subject to prohibition.

The platform's policy does not specifically outline additional rules regarding political content. However, there is a noticeable enforcement of strict measures against any potential coordinated non-authentic activities, including vote manipulation. Furthermore, the platform takes steps to filter out opinions that could be perceived as either promoting or denigrating a specific political force or country, even if they are presented as personal opinions. Moreover, there are some controversial examples that are clearly labeled as prohibited. For instance, questions such as "Why do Ireland and other countries want to exit the United Kingdom?" or "Who is paying for the royal family besides the Brits?" are categorized as forbidden content.

In addition to the aforementioned observations, we came across several examples of potential errors in the dataset. However, we will disregard these instances in our work, treating them as data noise or prohibited content. Examples of such erroneous data include questions like "How do I start my own IT company in Hungary?" or "Who will win in a video game battle: Adam d'angelo vs Kim kardashian?".

The apparent rules observed in the provided data appear to deviate slightly from

the current rules on the platform. Unfortunately, we do not possess any additional information regarding the labeling process or the extent of errors within the dataset. Therefore, we are proceeding with the assumption that the dataset is accurate and working with it accordingly.

Lastly, it's worth noting that there are numerous grammatical and spelling errors present in the dataset. Examples include questions like "What is mysical law?" and "What is one thing that can make you enotional?". Additionally, there are abbreviations and concatenated words that require preprocessing, such as "Is NHL a popular sport in the US?". We also come across formulas, contractions, and instances where it is crucial to properly preprocess the text for a complete understanding, especially when numbers are excluded. For instance, questions like "Can I run gta5 smoothly in i3 3rd generation 6gb ram?" require careful handling.

1.2 Data analysis

The competition organizers supplied two datasets: a train dataset and a test dataset. The train dataset includes the true markup, while the test dataset does not. The task at hand involves binary classification since we only have two classes. A positive class signifies that the question is insincere. The train dataset comprises 1,306,122 questions, while the test dataset consists of 375,806 questions.

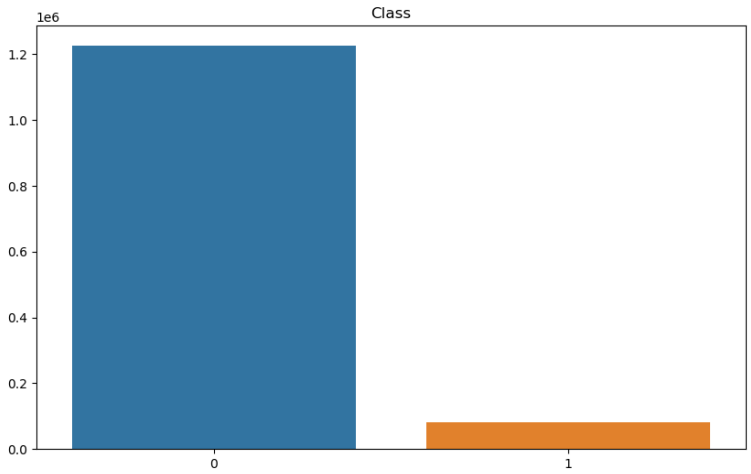

Figure 1: Class counts

The training data exhibits a significant class imbalance, with 94% of the objects belonging to class 0 and only 6% belonging to class 1 (Figure 1). Consequently, it is crucial for us to accurately identify class 1 objects and evaluate our results while considering this class imbalance. Our model would be rendered ineffective if it accurately classifies class 0 objects but performs poorly on class 1 objects.

The train dataset comprises approximately 8,508,433 words (depending on preprocessing), while the test dataset contains 2,449,841 words. Together, they constitute a dictionary of size 204,261.

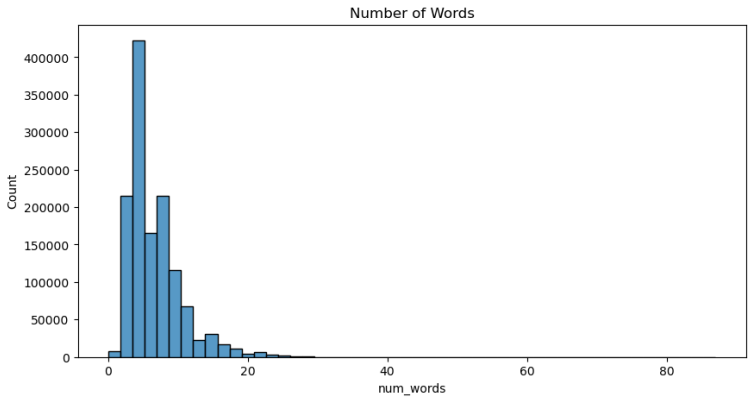

Figure 2: Words distribution (excl. stop-words and symbols)

The distribution of the number of words in questions (excluding stop-words and symbols) can be visualized as shown in Figure 2. Typically, people tend to use no more than 20 words in their questions. However, there are also longer questions that extend to 80 words or even more.

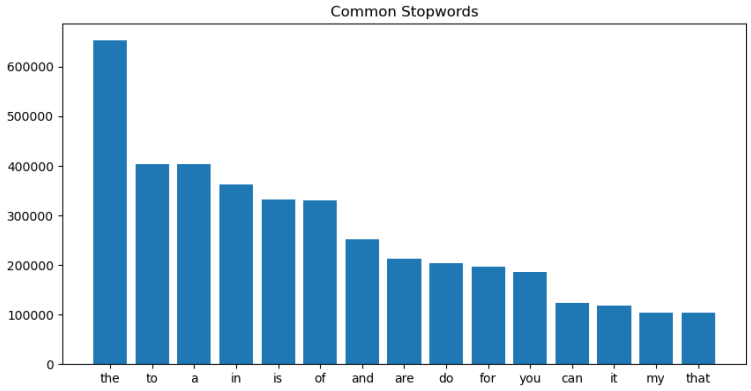

Figure 3: Stop-word counts

Figure 3 displays a count of the 15 most frequently occurring stop-words in the train dataset, with the word "the" being the most commonly used among them.

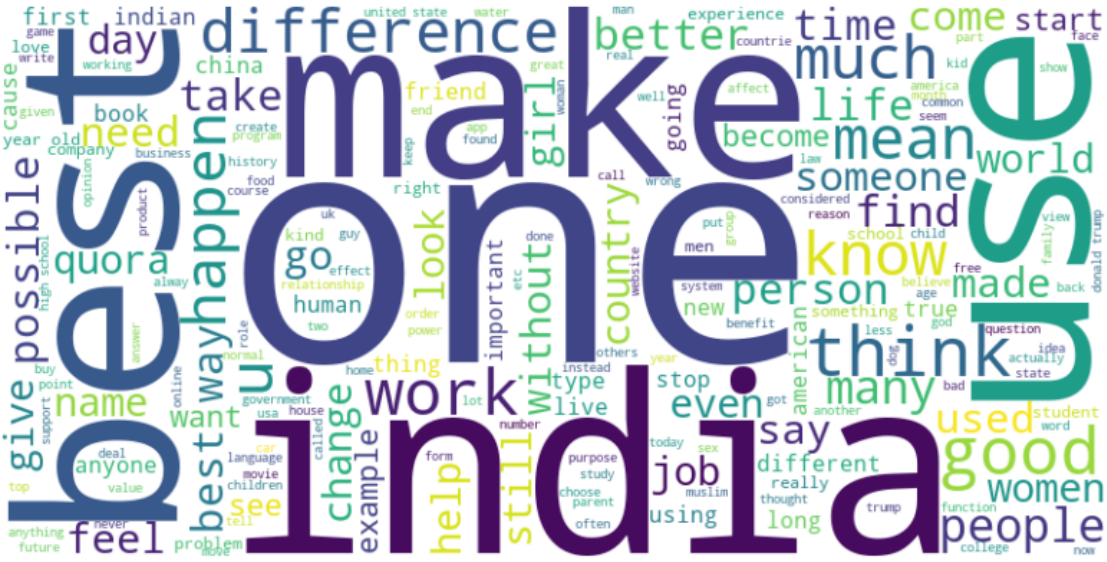

Figure 4: Wordcloud

Figure 4 highlights the most significant words found in the train dataset. Notably, words such as India, one, make, best, and use stand out among them. Based on this observation, we can infer that the training dataset likely contains a substantial number of questions related to India and inquiries about how to effectively create or utilize something in the best possible manner.

Obstacles we faced on

Since we chose one of the Kaggle competition, the only way to evaluate the quality of our models was to submit the results to Kaggle. Thus, we faced some problems. We were unable to use the "online" mode in our code due to a requirement of the competition. As a result:

- we were unable to utilize certain libraries (such as the contractions library)
- we could not use pre-trained models from the transformers library and fine-tune them

This led to several challenges throughout the project. Thus, certain portions of the code had to be commented out to ensure successful submission of the results to Kaggle. An example with preprocessing function is shown below:

```
#  The Function for text preprocessing
def tokenize_string(text):

    # Prior to lemmatization, replace all contractions with their corresponding full words
#     text_upd = contractions.fix(text)

    # Tokenize the data and use only lower letters
    words = word_tokenize(text.lower())
```

Figure 5: An example of commented code

We decided to divide our training dataset into 2 parts in order to be able to quickly conduct experiments without waiting for submissions on Kaggle and to be able to test large pre-trained models for our tasks. So, to compare quite basic models with transformers, we split train data with ratio 8:2 and use it for train and test for all our models. As it is shown below, the model metrics have changed slightly due to the additional data splitting. Thus, we can assume that the results that we have achieved, we could have achieved with a certain degree of error even when submitting at the competition.

It is worth noting that due to limitations in our computer resources and the availability of free resources on Kaggle, we were unable to conduct experiments using large-scale models. As a result, in certain cases, we resorted to utilizing distilled versions of models as a means of accommodating these constraints.

2 Training the models

2.1 Data Preprocessing

We opted for two preprocessing methods: TF-IDF and Word2Vec. Consequently, both baseline models were developed using both TF-IDF and Word2Vec approaches to consolidate the outcomes. Also, special preprocessing algorithms were used for LSTM and transformer models.

2.1.1 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents [11, 12].

In our project, we applied the TF-IDF technique to transform our textual data into a format suitable for machine learning algorithms. Each document was represented as a vector, where each dimension corresponds to a specific word in the corpus and the value in each dimension corresponds to the TF-IDF score of the word in that specific document. This method allowed us to quantitatively represent the significance of each word, providing a foundation for further analysis like clustering, classification, or sentiment analysis. The formula for TF-IDF algorithm is the following:

$TF - IDF = TF * IDF = \frac{n_t}{\sum_{i=1}^{k} n_i} * ln(\frac{n_c}{\sum_{j=1}^{m} n_j})$,

where:

1. $n_t = number\ of\ times\ the\ term\ appears\ in\ the\ corpus$,
2. $\sum_{i=1}^{k} n_i = total\ number\ of\ terms\ in\ the\ document$
3. $ln(\frac{n_c}{\sum_{j=1}^{m} n_j}) = ln(\frac{number\ of\ documents\ in\ the\ corpus}{number\ of\ documents\ in\ the\ corpus\ contain\ the\ term})$

The advantage of using TF-IDF lies in its ability to pull out the most important words that are unique to each document. It gives less importance to words that appear frequently across all documents, such as 'the', 'is', and 'and', which are often irrelevant to the underlying themes. However, TF-IDF has its limitations. It disregards word order, thus losing any semantic meaning that comes from the arrangement of words. That is the reason why we did not use TF-IDF for LSTM model. Additionally, it treats every document independently and does not capture any inter-document relationships.

For our baseline models, the *TfidfVectorizer* from *sklearn.feature_extraction.text* module was used, with all parameters set to their default values. Once the TF-IDF model was trained using all of the available training data, we utilized it to create all of our baseline models.

While this approach may be considered somewhat old-fashioned, it allowed us to create some basic baseline models that achieved F1 score placing us at **1346th out of 4037** submissions.

2.1.2 Word2Vec

In our project, we utilized the Word2Vec model[13] to improve our understanding of textual data. By converting words into numerical vectors, we could effectively feed this information into various machine learning algorithms to accomplish the task.

Word2Vec's ability to maintain semantic context was invaluable in the task, providing far superior results than traditional Bag-of-Words or TF-IDF approaches.

Word2Vec has numerous advantages. Its compact vector representations are efficient to compute and use. Moreover, they capture a significant amount of the semantic information, which is crucial for many NLP tasks. However, Word2Vec is not without its limitations. For instance, it does not handle out-of-vocabulary words well, and it does not account for the polysemy of words (words with multiple meanings). Probably, that is the reason why Catboost with Word2Vec failed in contrast with other baseline models result. Despite these challenges, the model's overall performance and flexibility make it a valuable tool in the field of NLP.

For the baseline models and LSTM model, the *Word2Vec* from *gensim.models* module [14] was used with the following parameters:

- Vector size was 300 for baseline models and 700 for LSTM (*vector_size = 300, vector_size = 700*)
- The maximum distance between the current word and its neighboring words was 5 (*window = 5*)
- The minimum frequency value of words was 3 for the baseline models and 5 for LSTM (*min_count = 3, min_count = 5*)

Additionally, we generated a UMAP projection of the embeddings to assess their quality.

UMAP Projection of Word2Vec Embeddings of top 100 the most common words that was used with the baseline models was the following:

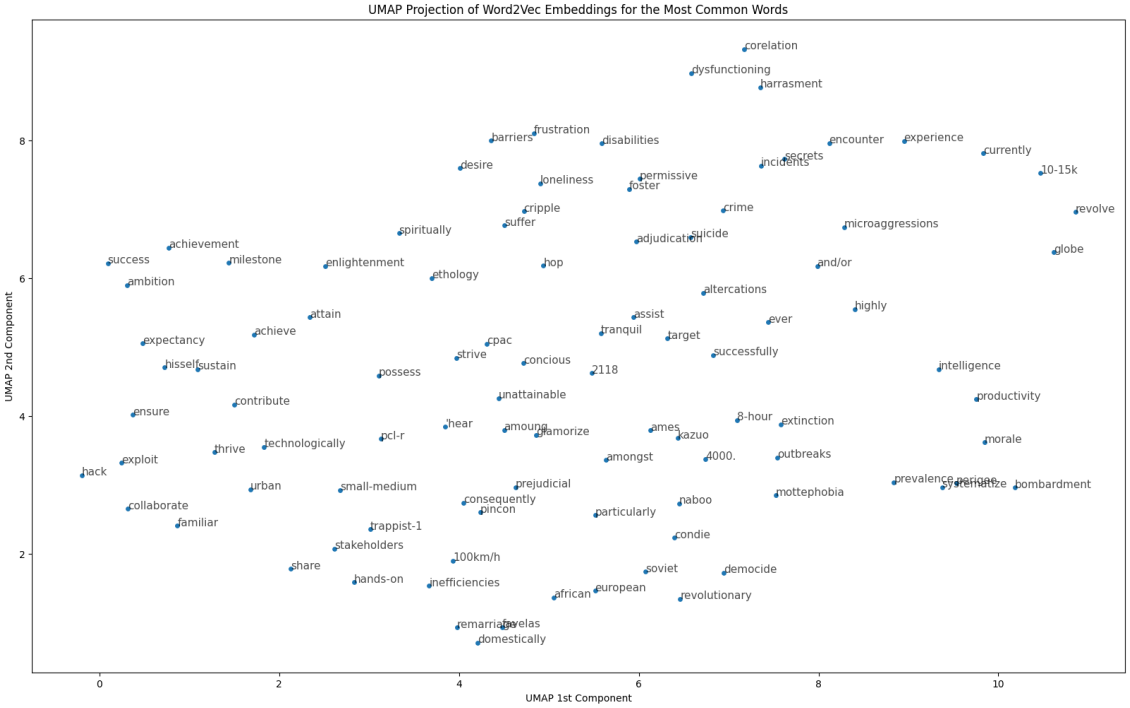

Figure 6: UMAP Projection of Word2Vec Embeddings for the most common words of the train Data set (vector_size = 300, window = 5, min_count = 3)

UMAP Projection of Word2Vec Embeddings of top 100 the most common words that was used with LSTM was the following:

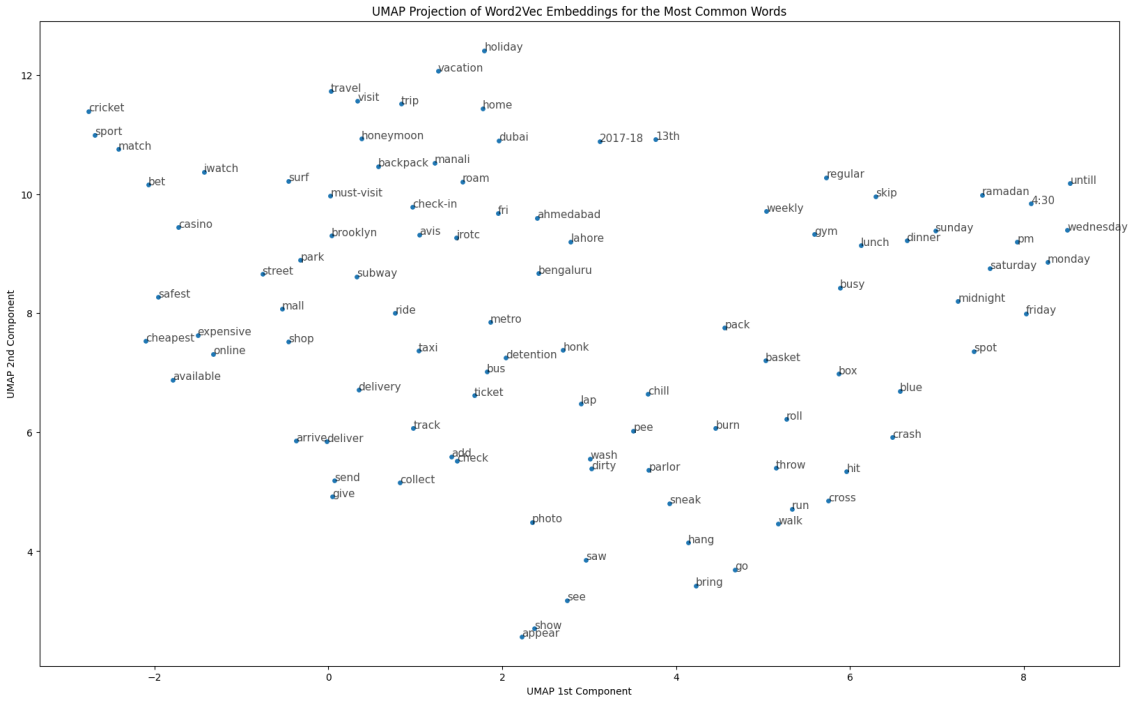

Figure 7: UMAP Projection of Word2Vec Embeddings for the most common words of the train Data set (vector_size = 700, window = 5, min_count = 5)

2.2 Baseline Models

2.2.1 Logistic Regression

We developed two baseline models using the Logistic Regression approach:

- One model used the TF-IDF algorithm for preprocessing.
- The other model used the Word2Vec model for preprocessing.

To ensure comparability of the results we developed both models with the same parameters. Specifically, we used the *LogisticRegression* model from the *sklearn.linear_model* Python library [15], with all parameters set to default except for the maximum number of iterations, which was set to 10 000 (*max_iter = 10000*).

It is worth noting that the Logistic Regression model using the TF-IDF preprocessing algorithm outperformed the one using the Word2Vec model, with F1 scores of **0.50** and **0.32**, respectively. The total time required to study both models on a CPU was approximately **30 minutes**.

2.2.2 CatBoost

Catboost model was chosen as another baseline one to understand the difference and find the best baseline model for this and future projects.

To ensure comparability of the results we developed both models with the same parameters. Specifically, we used the *CatBoostClassifier* model from the *catboost* Python library [16], with the following parameters:

- Number of iterations was 10 000 (*iterations = 10000*)
- Learning rate was 0.1 (*learning_rate = 0.1*)
- The Tree depth was 6 (*depth = 6*)
- Loss Function was the LogLoss (*loss_function = 'Logloss'*)
- Frozen random seed was 42 (*random_seed = 42*)

Catboost model using TF-IDF and Word2Vec were quite the same: F1 scores were **0.54** and **0.52**, respectively. The total time required to study both models on a CPU (unlike neural networks, CatBoost and LogisticRegression could not be run on a GPU on Kaggle) was approximately **6 hours**.

2.3 RNN models (LSTM)

Long Short-Term Memory (LSTM) model[17] was chosen as part of Recurrent neural networks (RNN).

In our project, we utilized LSTM networks to analyze sequential data. The LSTM's ability to understand long-term dependencies in sequences allowed us to model complex patterns that would be challenging for traditional machine learning models to capture. Also, as far as LSTM was designed to handle sequential data, TF-IDF approach was not used in preprocessing for RNN models.

Since we were restricted from using *keras.layers.Embeddings* due to the competition's limitations, we opted to employ LSTM with Word2Vec preprocessing as an alternative approach.

We observed the success of the Bidirectional LSTM model on Kaggle, we decided to explore a similar architecture. A Bidirectional LSTM, or BiLSTM [18], extends the traditional LSTM by adding a second layer that processes the sequence in the reverse order. The rationale behind this is that the context from both preceding and following time steps can provide enhanced context for the current time step, compared to traditional LSTMs which only account for past context. In essence, BiLSTMs can learn information from both past and future states simultaneously. This makes them particularly effective for tasks where understanding context is crucial.

In our model, we have employed several layers of these BiLSTMs, each decreasing in size. This structure allows the model to extract and condense crucial information, functioning as a form of information funneling. The final output is then passed through a fully connected layer with a sigmoid activation function, transforming the output to a probability score between 0 and 1, perfect for binary classification tasks.

Our LSTM model that achieved an F1 score of 0.6 comprised of four Bidirectional layers and one dense layer with a linear pipeline. Each layer had a varying number of units, ranging from 256 to 32, with the first and fourth layers corresponding to these values respectively. Furthermore, the callback mode was enabled to utilize the

best model based on the evaluation of the validation data, specifically focusing on the maximum validation AUC (Area Under the Curve).

Additionally, the LSTM model was trained using the following parameters:

- Loss function was set to "Binary Focal Crossentropy" (*loss = 'BinaryFocalCrossentropy'*)
- The optimizer was Adam (*optimizer = 'adam'*)
- The evaluation metric used was AUC (Area Under the Curve), as we aimed to improve the model's performance not only for the 0 class (*metrics = ['AUC']*)
- The activation function employed was sigmoid (*activation = 'sigmoid'*)
- The model was trained for a total of 10 epochs (*epochs = 10*)

All of the other parameters were set to default.

Finally, the F1 score for LSTM model was **0.60** that put us t **1314th out of 4037** submissions.

It is worth noting that although the Bidirectional LSTM model emerged as the winner in the competition, we did not observe any significant difference in the F1 score between the "classic" LSTM and the Bidirectional LSTM models.

2.4 Transformer models

Our choice of transformer models focused on two specific ones: BERT [7] and a modified version of RoBERTa [8]. Despite being introduced in 2018-2019, these models remain highly relevant and continue to deliver exceptional accuracy in their respective tasks. To address computational limitations, we utilized distilled versions of these models, namely DistilBERT [19] and DistilRoBERTa.

DistilBERT [19] is a compact and efficient Transformer model based on the BERT architecture. With 40% fewer parameters than bert-base-uncased, it runs 60% faster while preserving 97% of BERT's performance on the GLUE language understanding benchmark. DistilBERT achieves this by employing knowledge distillation, a technique that compresses a larger model (the teacher) into a smaller model (the student). By distilling BERT, we obtain a streamlined Transformer model that shares many similarities with the original BERT model, but is lighter, more compact, and faster to execute. DistilBERT is thus a compelling choice for deploying large-scale trained Transformer models in production. This distillation approach can also be applied to other Transformer architectures, such as RoBERTa, GPT2, and more.

Common params for models:

- learning rate (*lr = 0.003*)
- number of epochs (*n_epochs = 2*)
- weight decay (*weight_decay = 1e-6*)
- batch size (*batch_size = 128*)
- max sequence length (*max_len = 138*)

3 Results and discussion

The following F1 score were obtained for the developed models:

Model	F1 (Kaggle)	F1 (Train/Test split)
Logistic Regression (TF-IDF)	0.50	0.49
Logistic Regression (Word2Vec)	0.32	0.31
CatBoost (TF-IDF)	0.54	0.54
CatBoost (Word2Vec)	0.52	0.54
LSTM (Word2Vec)	0.60	0.60
DistillBERT	–	0.59
DistillRoBERTa	–	0.43

Table 1: Metrics for each model

In summary, the least effective model is the Logistic Regression model based on Word2Vec. While the best baseline models in terms of F1 are the CatBoost models (with the highest score being 0.54), the time required to train these models is still too long. Therefore, the best baseline model overall is the LSTM model, with an F1 score of 0.6.

Despite our expectations, the transformer models did not perform much better than the LSTM model. The best result for transformers was achieved on the DistilBERT[19] model with F1 score 0.59.

In order to get better results on this task in the future, in our opinion, for transformers it is necessary to retrain the models on more texts related to sentiment analysis. It is worth trying to use more epochs and experiment with learning rate.

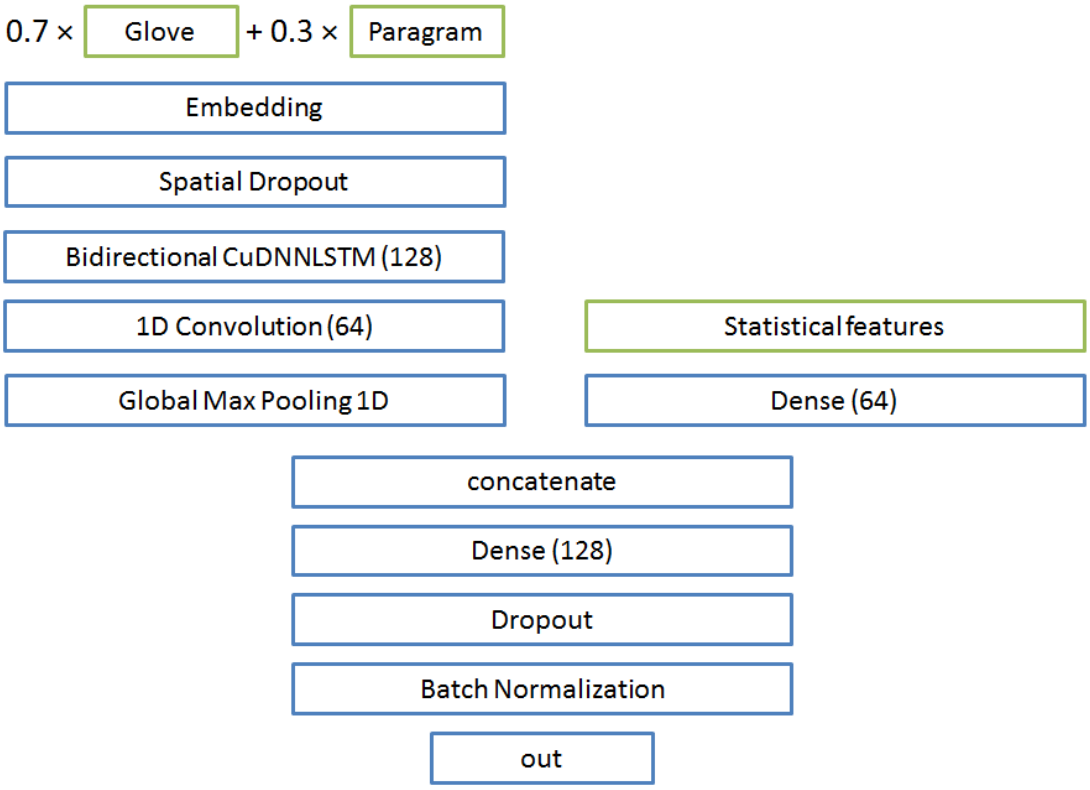

Figure 8: 1st place solution

It is intriguing to examine the winning approach employed in the competition. The approach incorporated the use of Glove and Paragram embeddings, which were provided to participants. Additionally, the model utilized the BI-LSTM architecture and integrated statistical features obtained from a corpus of texts. Proper threshold selection was also crucial for achieving optimal results in this task. For a detailed analysis of the winning solution, refer to the link provided [20].

This winning solution highlights the importance of not solely relying on pre-trained embeddings but also leveraging statistical information derived from a corpus of texts. Moreover, constructing complex models that integrate information obtained through diverse methods into a unified layer has shown to be advantageous. This approach enables the model to effectively capture nuanced patterns and relationships, ultimately enhancing its performance.

References

[1] "Quora insincere questions classification." https://www.kaggle.com/competitions/quora-insincere-questions-classification. Accessed: 2023-04-12.

[2] J. e. a. Ramos, "Using tf-idf to determine word relevance in document queries. in proceedings of the first instructional conference on machine learning," 2003. Accessed: 2023-03-01.

[3] B. Das and S. Chakraborty, "An improved text sentiment classification model using tf-idf and next word negation.," 2018. Accessed: 2023-03-01.

[4] Y. S. Rhanoui M., Mikram M. and B. S, "A cnn-bilstm model for document-level sentiment analysis. machine learning and knowledge extraction," 2019. Accessed: 2023-03-01.

[5] d. S. R. L. Sousa M. G., Sakiyama K., "Bert for stock market sentiment analysis. in 2019 ieee 31st international conference on tools with artificial intelligence (ictai)," 2019. Accessed: 2023-03-01.

[6] G. Y. Dong J., He F. and Z. H., "A commodity review sentiment analysis based on bertcnn model. in 2020 5th international conference on computer and communication systems (icccs)," 2020. Accessed: 2023-03-01.

[7] "Bert: Pre-training of deep bidirectional transformers for language understanding." https://arxiv.org/pdf/1810.04805.pdf. Accessed: 2023-05-02.

[8] "Roberta: A robustly optimized bert pretraining approach." https://arxiv.org/abs/1907.11692. Accessed: 2023-03-01.

[9] "Deriving contextualised semantic features from bert (and other transformer model) embeddings." https://aclanthology.org/2021.repl4nlp-1.26.pdf. Accessed: 2023-03-01.

[10] "Quora platform policy." https://help.quora.com/hc/en-us/articles/360000470706-Platform-Policies. Accessed: 2023-04-08.

[11] "Term frequency-inverse document frequency approach." https://monkeylearn.com/blog/what-is-tf-idf/. Accessed: 2023-04-15.

[12] "Term frequency-inverse document frequency formula." https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/. Accessed: 2023-05-08.

[13] "Efficient estimation of word representations in vector space." https://arxiv.org/pdf/1301.3781.pdf. Accessed: 2023-05-02.

[14] "Gensim word2vec – a complete guide." https://www.askpython.com/python-modules/gensim-word2vec. Accessed: 2023-05-14.

[15] "Logisticregression, sklearn.linear_model python library." https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html. Accessed: 2023-04-29.

[16] "Catboostclassifier, catboost python library." https://catboost.ai/en/docs/concepts/python-reference_catboostclassifier. Accessed: 2023-05-01.

[17] "Long short-term memory." http://www.bioinf.jku.at/publications/older/2604.pdf. Accessed: 2023-05-02.

[18] "Bidirectional lstm." https://medium.com/analytics-vidhya/what-does-it-mean-by-bidirectional-lstm-63d6838e34d9. Accessed: 2023-04-23.

[19] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter." https://arxiv.org/abs/1910.01108, 2019. Accessed: 2023-03-01.

[20] "1-st place solution." https://www.kaggle.com/competitions/quora-insincere-questions-classification/discussion/80568. Accessed: 2023-03-02.