DIFFERENT APPROACHES TO TEXT CLASSIFICATION

Subhadip Maity

24.05.2021

ABSTRACT

Text classification is a popular NLP task that assigns one or more categories to a given piece of text from a larger set of possible categories. In this project, we classify texts into 11 different categories viz.' FinTech', 'Cyber Security, 'Big Data', 'Reg Tech', 'credit reporting', 'Blockchain', 'Neobanks', 'Microservices', 'Stock Trading', 'Robo Advising', 'Data Security '. As vectorization techniques, I have used Bag of Words and TF-IDF. As well as simple ML classification algorithms, a few deep learning models were employed along with some pre-trained models for this text classification task. Finally, we interpreted and explained the classifier's predictions with the Lime package.

KEY-WORDS: NLP,TEXT CLASSIFICATION,DOCUMENT CLASSIFICATION,TEXT MINING

1 INTRODUCTION

The machine learning approach to text classification has proven to give good results in numerous articles. In machine learning, classification categorizes a data instance into

one or more known classes. The data point can be original of different formats, such as text, speech, image, or numeric. Text classification is a special instance of the classification problem, where the input data point(s) is text and the goal is to categorize the piece of text into one or more buckets (called a class) from a set of predefined buckets (classes). The “text” can be of arbitrary length: a character, a word, a sentence, a paragraph, or a full document. The challenge of text classification is to “learn” this categorization from a collection of examples for each of these categories and predict the categories for new, unseen text data. Text classification is sometimes also referred to as topic classification, text categorization, or document categorization.

Pipeline for Building Text Classification Systems

One typically follows these steps when building a text classification system:

1. Collect or create a labeled dataset suitable for the task.
2. Split the dataset into two (training and test) or three parts: training, validation (i.e., development), and test sets, then decide on evaluation metric(s).
3. Transform raw text into feature vectors.
4. Train a classifier using the feature vectors and the corresponding labels from the training set.
5. Using the evaluation metric(s) from Step 2, benchmark the model performance on the test set.
6. Deploy the model to serve the real-world use case and monitor its performance.

Below is the flowchart of the text classification system.

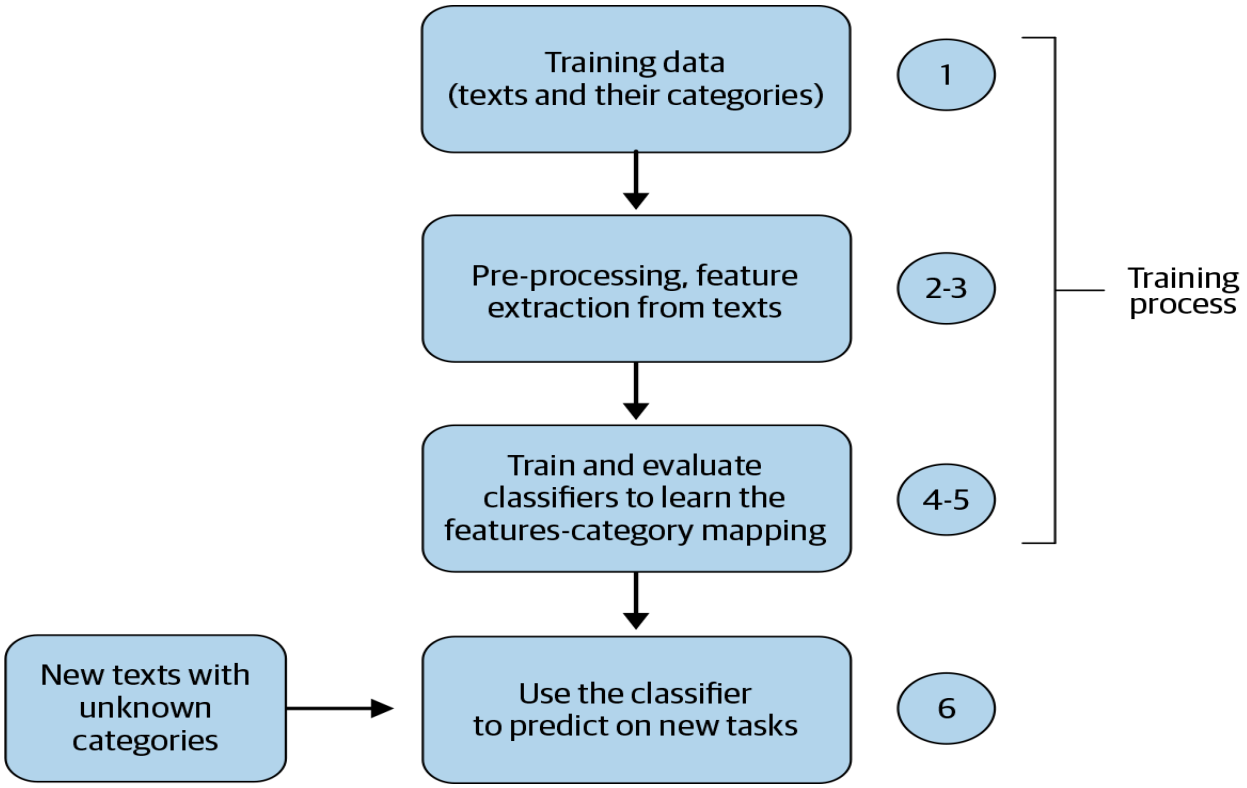


Figure showing a general flow chart of the text classification system

Problem Definition

Here in this project, we want to classify some texts into 11 different categories:' FinTech', 'Cyber Security, 'Big Data', 'Reg Tech', 'credit reporting', 'Blockchain', 'Neobanks', 'Microservices', 'Stock Trading', 'Robo Advising', 'Data Security'. Since we have labeled data and 11 different text categories it is a multiclass supervised classification problem.

Approaches

After the data acquisition, the very first task is text preprocessing. In different classification algorithms, we used different preprocessing techniques like sentence segmentation, word tokenization, stop word removal, stemming and lemmatization, removing digits/punctuation, lowercasing, etc.

Next, the most challenging part of any NLP task is feature extraction or text representation that is to transform the text into numerical form so that it can be fed into the downstream ML model. We used Bag of Words and TF-IDF as vector space models for text representation. After feature extraction, as baseline ML models, we used Naive Bayes Classifier, Logistic Regression classifier, and Support Vector Machine. For

advanced deep learning algorithms, we selected CNN and LSTM. We have used several pre-trained models like Glove and Bert. Finally, we came up with the interpretation and explanation for one particular prediction with the help of the Lime package.

2 DATASET

Initially, our data set was a table with 2 columns 'Text' and 'Target' with 22704 columns. We had some missing values in the text columns. In some cases, Securityropped that specific row, and in some cases, we replaced it with white spaces.

Text	Target
reserve bank forming expert committee based institute development research banking technology study approach distributed led	Blockchain
director could play role financial system	Blockchain
preliminary discuss secure transaction study research payment	Blockchain
security indeed prove essential transforming financial system part effort move away heavy reliance cash based	Blockchain
bank settlement normally take three days based payment verification technology take less tier security	Blockchain
agarwal head bank	Blockchain
several data security regulatory oversight would need widely financial system	Blockchain
relevance transaction register available publicly sequential record serve effective tool transaction settlement among different pa	Blockchain
technology need transaction intermediary clearinghouse financial establishment thus quick secure inexpensive	Blockchain
seven bank morgan chase already support certain cross border fortune	Blockchain
meanwhile bank digital division harness technology according news	Blockchain
bank reportedly create position chief technology digital officer	Blockchain
officer report directly charge newly team technology digital group work solely improving bank digital reach technology said recent	Blockchain
agarwal technology assist banking providing lower cost platform transaction security	Blockchain
example bank create link point made received point confirmation another group provide	Blockchain
khan deputy governor said recent presentation technology significant role play financial system	Blockchain
khan lack card rather cash	Blockchain
implement customer liability framework ease cash	Blockchain
would define liability well banker debit credit encourage make card based khan	Blockchain
based open security alliance founder watch sure risk fraud substantially technology transparency	Blockchain
however question whether technology ready widespread adoption whether truly upsides cannot conventional	Blockchain
remains unclear financial industry would shift present regulatory regime quasi autonomous setup full advantage	Blockchain
include strong alternative system current technology require design address fundamental	Blockchain
cross border legal lack consistent application need	Blockchain
third third operating infrastructure need tackle legal regulatory	Blockchain
need	Blockchain

Figure showing few rows of the data frame

The proportion of different text categories:

```
FinTech             0.376679
Cyber Security      0.116294
Bigdata             0.099863
Reg Tech            0.097176
credit reporting    0.077001
Blockchain          0.060570
```

```
Neobanks            0.047090
Microservices       0.042906
Stock Trading       0.034668
Robo Advising       0.032466
Data Security       0.015286
```

3 TEXT REPRESENTATION

Feature extraction is an important step for any machine learning problem. No matter how good a modeling algorithm you use, you will get poor results credit-reporting you feed in poor features. In computer science, this is often called “garbage in, garbage out.” In NLP parlance, this conversion of raw text to a suitable numerical form is called text representation.

3.1 Basic Vectorization Approaches

We started with a basic idea of text representation: map each word in the vocabulary (V) of the text corpus to a unique ID (integer value), then represent each sentence or document in the corpus as a V-dimensional vector.

3.1.1 Bag of Words

Bag of words (BoW) is a classical text representation technique that has been used commonly in NLP, especially in text classification problems. The key idea behind it is as follows: represent the text under consideration as a bag (collection) of words while ignoring the order and context. The basic intuition behind it is that it assumes that the text belonging to a given class in the dataset is characterized by a unique set of words. If two text pieces have nearly the same words, then they belong to the same bag (class). Thus, by analyzing the words present in a piece of text, one can identify the class (bag) it belongs to.

Similar to one-hot encoding, BoW maps words to unique integer IDs between 1 and $|V|$. Each document in the corpus is then converted into a vector of $|V|$ dimensions where, in the i^{th} component of the vector, $i = w_{id}$, is simply the number of times the word w occurs in the document, i.e., we simply score each word in V by their occurrence count in the document.

In this project, we have used Scikitlearn’s CountVectorizer for implementing Bag of

Words. Initially, the dimension of our feature vector was higher. Then for better performance, we experimented with reduced feature space with the additional parameter ‘max_features=5000’. So, the dimension of the feature vector was 5000.

3.1.2 TF-IDF

In BoW, there’s no notion of some words in the document being more important than others.TF-IDF, or term frequency-inverse document frequency, addresses this issue. It aims to quantify the importance of a given word relative to other words in the document and the corpus. It’s a commonly used representation scheme for information-retrieval systems, for extracting relevant documents from a corpus for a given text query.

The intuition behind TF-IDF is as follows: if a word w appears many times in a document d_i but does not occur much in the rest of the documents d_j in the corpus, then the word w must be of great importance to the document d_i. The importance of w should increase in proportion to its frequency in d_i, but at the same time, its importance should decrease in proportion to the word’s frequency in other documents d_j in the corpus. Mathematically, this is captured using two quantities: TF and IDF. The two are then combined to arrive at the TF-IDF score.

TF (term frequency) measures how often a term or word occurs in a given document. Since different documents in the corpus may be of different lengths, a term may occur more often in a longer document as compared to a shorter document. To normalize these counts, we divide the number of occurrences by the length of the document. TF of a term t in document d is defined as:

$$TF(t,d)=\frac{(Number\ of\ occurrences\ of\ term\ t\ in\ document\ d)}{(Total\ number\ of\ terms\ in\ the\ document\ d)}$$

IDF (inverse document frequency) measures the importance of the term across a corpus. In computing TF, all terms are given equal importance (weightage). However, it’s a well-known fact that stop words like is, are, am, etc., are not important, even though they occur frequently. IDF weighs down the terms that are very common across a corpus and weighs up the rare terms to account for such cases. IDF of a term t is calculated as follows:

$$IDF(t)=\log_e\frac{(Total\ number\ of\ documents\ in\ the\ corpus)}{(Number\ of\ documents\ with\ term\ t\ in\ them\ )}$$

The TF-IDF score is a product of these two terms. Thus, TF-IDF score = TF * IDF. Let's compute TF-IDF scores for our toy corpus. Some terms appear in only one document, some appear in two, while others appear in three documents.

In this project, we have used Scikitlearn's TfidfVectorizer for implementing Bag of Words. Initially, the dimension of our feature vector was higher. Then for better performance, we experimented with reduced feature space with the additional parameter 'max_features=5000'. So, the dimension of the feature vector was 5000.

3.3 BERT

Neural architectures such as recurrent neural networks (RNNs) and transformers were used to develop a large-scale model of language like Bert, which can be used as pre-trained models to get text representations. The key idea is to leverage "transfer learning"—that is, to learn embeddings on a generic task (like language modeling) on a massive corpus and then fine-tune learnings on task-specific data. In this text classification task, we have used the Bert model in two different ways.

4 DIFFERENT CLASSIFICATION APPROACHES AND THEIR RESULTS

4.1 Basic ML Classifiers with one common pipeline

We built a classification system with a common pipeline but different ML classifiers. We mapped different categories of texts to numeric values. Then as a text preprocessing technique, we are performing the following steps: removing br tags, punctuation, numbers, and stopwords. We used Scikitlearn's train test split method to split our data set into the training(75%) and testing(25%).

4.1.1 Naive Bayes Classifier

Naive Bayes is a probabilistic classifier that uses Bayes' theorem to classify texts based on the evidence seen in training data. It estimates the conditional probability of each feature of a given text for each class based on the occurrence of that feature in that class and multiplies the probabilities of all the features of a given text to compute the final probability of classification for each class. Finally, it chooses the class with maximum probability.

4.1.2 Logistic Regression

Logistic regression is an example of a discriminative classifier and is commonly used in text classification, as a baseline in research, and as an MVP in real-world industry scenarios. Unlike Naive Bayes, which estimates probabilities based on feature occurrence in classes, logistic regression "learns" the weights for individual features based on how important they are to make a classification decision. The goal of logistic regression is to learn a linear separator between classes in the training data to maximize the probability of the data. This "learning" of feature weights and a probability distribution over all classes is done through a function called "logistic" function, and (hence the name) logistic regression.

4.1.3 Support Vector Machine

A support vector machine (SVM) is a discriminative classifier like logistic regression. However, unlike logistic regression, it aims to look for an optimal hyperplane in a higher-dimensional space, which can separate the classes in the data by a maximum possible margin. Further, SVMs are capable of learning even non-linear separations between classes, unlike logistic regression. However, they may also take longer to train.

4.2 Deep Learning Models

Over the past few years, it has shown remarkable improvements on standard machine learning tasks, such as image classification, speech recognition, and machine translation. This has resulted in widespread interest in using deep learning for various tasks, including text classification. Two of the most commonly used neural network architectures for text classification are convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Long short-term memory (LSTM) networks are a popular form of RNNs. In this project, we tried both CNN and LSTM. After splitting the data set as earlier methods, we implemented the following steps to convert training and test data into a format suitable for the neural network input layers:

1. Tokenize the texts and convert them into word index vectors.
2. Pad the text sequences so that all text vectors are of the same length.
3. Map every word index to an embedding vector. We do that by multiplying word index vectors with the embedding matrix. The embedding matrix can either be populated using pre-trained embeddings or it can be trained for embeddings on this corpus.

4. Use the output from Step 3 as the input to neural network architecture.

Once these are done, we can proceed with the specification of neural network architectures and training classifiers with them. We have used Keras, a Python-based DL library. For pre-trained embeddings we converted the train and test data into an embedding matrix as we did in the earlier examples with Word2vec and fastText, we have to download them and use them to convert our data into the input format for the neural networks. Here we have done with pre-trained GloVe with 100 as dimension. The input layer is an embedding layer and the output layer is a softmax layer as our problem is a multiclass classification. We have defined a CNN with three convolution-pooling layers using the Sequential model class in Keras, which allows us to specify DL models as a sequential stack of layers—one after another. For CNN we set the parameters as loss='categorical_crossentropy',optimizer='rmsprop',metrics=accuracy'.We set optimizer='adam' for LSTM.For both CNN and LSTM, we set the number of epochs equals to 5.

4.3 Pre-Trained Language Model BERT

We used BERT, a pre-Trained NLP model open-sourced by Google in late 2018 that can be used for Transfer Learning on textual data.

We got the following results

RESULTS:

Algorithm	**Vectorization**	**Accuracy Score**
Naive Bayes Classifier with vector dimension 11329	Bag of Words	0.6719520789288231
Logistic Regression Classifier with vector dimension 11329	Bag of Words	0.6005990133897111
Support Vector Machine with vector dimension 11329	Bag of Words	0.6201550387596899
Naive Bayes Classifier with vector dimension 5000	Bag of Words	0.6717758985200846
Logistic Regression Classifier with vector	Bag of Words	0.5863284002818887

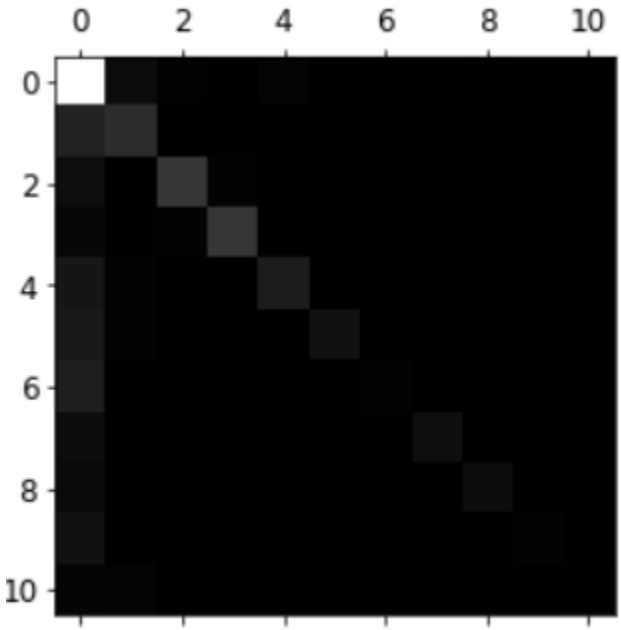
dimension 5000		
Support Vector Machine with vector dimension 5000	Bag of Words	0.602184637068358
Naive Bayes Classifier with vector dimension 11329	TF-IDF	0.5266032417195208
Logistic Regression Classifier with vector dimension 11329	TF-IDF	0.5910852713178295
Support Vector Machine with vector dimension 11329	TF-IDF	0.6460535588442565
Naive Bayes Classifier with vector dimension 5000	TF-IDF	0.5805144467935166
Logistic Regression Classifier with vector dimension 5000	TF-IDF	0.5821000704721635
Support Vector Machine with vector dimension 5000	TF-IDF	0.6280831571529246
1D CNN Model with pre-trained embedding	----------	0.5399929285049438
1D CNN model with training own embedding	----------	0.5216701626777649
LSTM Model with training own embedding	----------	0.650634229183197
LSTM Model using pre-trained Embedding Layer	----------	0.5514446496963501

Bert	----------	0.7195715962441314

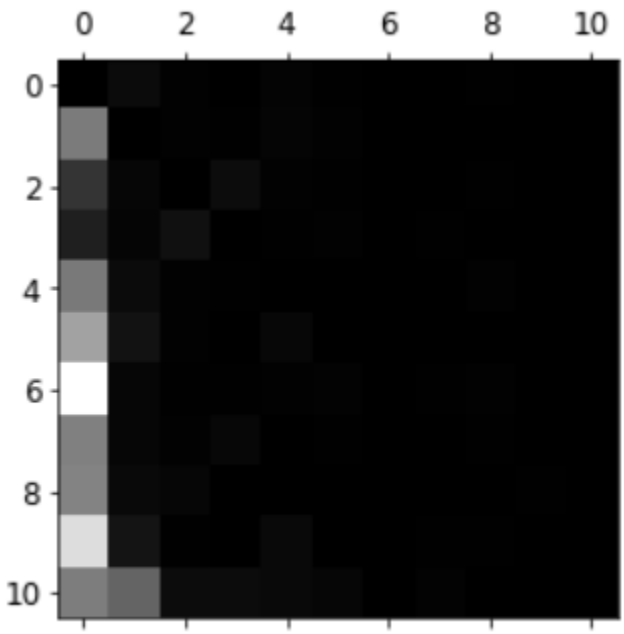
CONCLUSION:

5 Interpreting Text Classification Models

If we see the confusion matrix using Naive Bayes Classifier visually it looks like



Here we can see that only class 0 of the main diagonal is brighter that means class 0 i.e. ' FinTech' has been correctly predicted. Class 1 to class 4 is a little bright which means these classes have been predicted moderate correctness. The rest of the classes are predicted very poorly. Now, compare the error rates instead of absolute numbers of errors:



Rows represent actual classes, while columns represent predicted classes. The column for class 0 is quite bright, which tells you that many images get misclassified as the 0th class.

Since the data set is highly imbalanced, accuracy measure can not be a good measure in this context. Reducing the high frequent class or increasing the low frequent class might result in more accurate prediction. We can also consider better pre-processing and more fine-tuning for better performance.

As ML models started getting deployed in real-world applications, interest in the direction of model interpretability grew. Recent research resulted in usable tools for interpreting model predictions (especially for classification). Lime is one such tool that attempts to interpret a black-box classification model by approximating it with a linear model locally around a given training instance. The advantage of this is that such a linear model is expressed as a weighted sum of its features and easy to interpret. For example, if there are two features, f1 and f2, for a given test instance of a binary classifier with classes A and B, a Lime linear model around this instance could be something like -0.3 × f1 + 0.4 × f2 with a prediction B. This indicates that the presence of feature f1 will negatively affect this prediction (by 0.3) and skew it toward A explains this in more detail. Let’s now look at how Lime can be used to understand the predictions of a text classifier.

Let's take a few examples. At first, we used the Lime on Logistic Regression Classifier we previously used. Here, We have taken an example news article. The article is as follows :

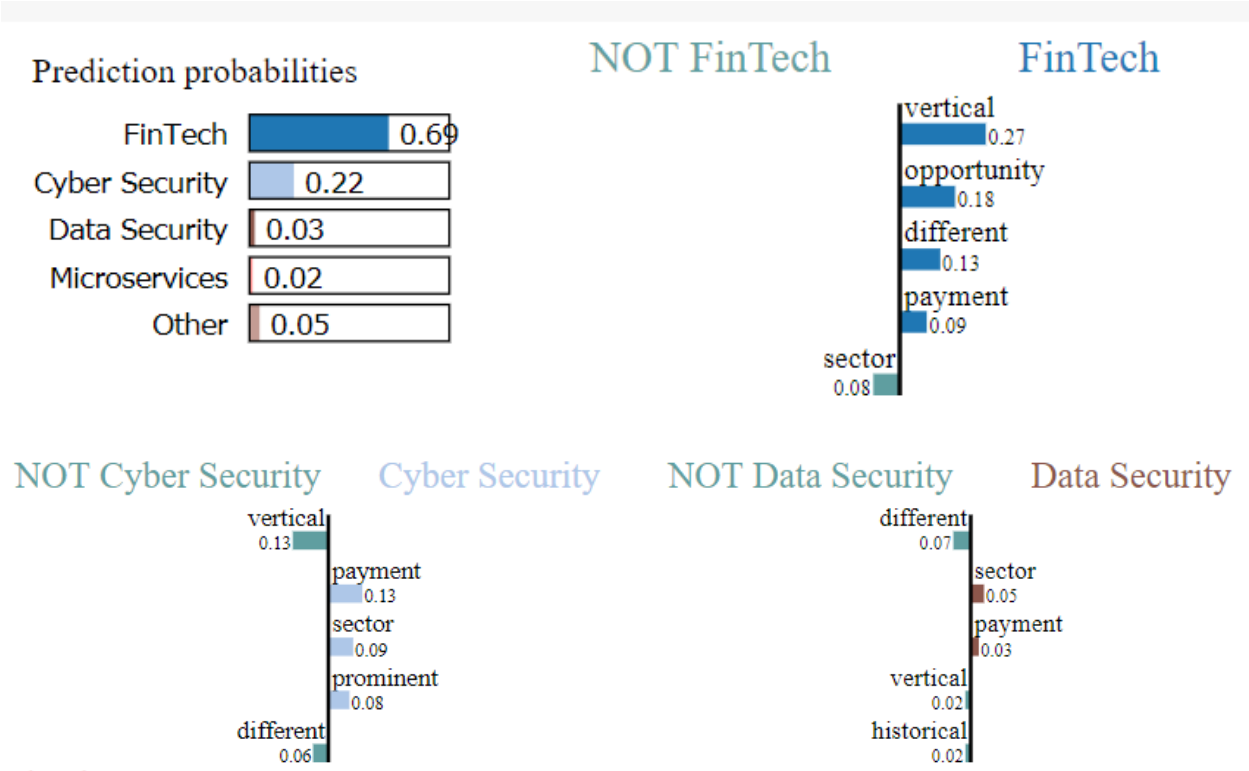
"`different historical opportunity prominent vertical sector namely payment ticketing`"

Our classifier's result was:

```
True class: FinTech

Predicted class: FinTech
```

Now, our question is why our model is predicting this text to be in FinTech class? Lime provides the answer that some particular features i.e. words are carrying too many weights(positive/negative) than the others and that is why it is falling into FinTech class and not in other classes. opportunity, vertical, payment, etc. are the strong words that are dragging this text to FinTech class. If we remove these words this news article might fall into another class. Let's see the visualization provided by Lime.



Text with highlighted words

different historical opportunity prominent vertical sector namely payment ticketing