NLP classification applied to logistics tracking

Jorge da Costa Ferreira and Tiago Rabaça Vaz Craveiro

Economics Faculty, University of Oporto

up200908673@fep.up.pt
up201204029@fep.up.pt

Abstract. On this work an overview of the entire NLP classification process is presented. From the most challenging aspects related with text mining such as ambiguity, synonymity, multi-language. The un-structured data source promotes data sparsity and consequently affects the susceptibility of model to overfitting. The use case under study is to apply a text classification model in a production data base of HUUB tracking messages, a digital logistics company that ships worldwide. Techniques of data cleaning and different type of vectorizations using Bag of Words (BoW) or alternatively Word Embedding like Word2Vec, GloVe and BERT are analyzed. These embeddings promote a higher density on space of features which tends to be beneficial. A python implementation using data techniques, such as data cleaning and different type of vectorizations using Bag of Words (BoW) or alternatively Word Embedding like Word2Vec were applied. In order to find a feasible solution to be implemented by the company to classify tracking messages, distinct NLP classification models, such us Naïve Bayes, Logistic Regression and SVM were trained and then compared using distinct evaluation methods.

Keywords: NLP; Text mining; Syntactic analysis; Stemming; Lemmatization; BoW (Bag of Words); One hot encoding; Word2Vec; GloVe; BERT; RNN; Tracking; Logistics.

Table of Contents

1 Problem Description and Motivation

HUUB is a supply chain management startup that enables its customers - fashion brands - to manage their logistic operations, from suppliers to end customer. It supports what is so called the backbone of the Ecommerce industry. One important step of the Ecommerce journey is the last mile delivery, meaning the delivery of the package to the customer's home.

Nowadays most digital customer relies on the tracking visibility that carriers provide so that the order journey can be followed since the package leaves the warehouse until it's delivered to the customer. The main pain is that the information provided by the carriers on their digital tracking solutions can be very confusing, either because the information provided is not clear, meaning delivery status can get the customer confused "your order is in transit" can mean that is still in transit or that is being delivered at the customer's address. This get even more complex when for distinct location the words used to provide the tracking are completely different and carriers among them use distinct status for the same thing.

For example, UPS a well-known shipping carrier send this type of message to an ecommerce customer "Arrived at Delivery Facility in MANCHESTER - UK" and the status associated with the message is "Returned", meaning that the package is being returned to the carrier warehouse. But for FedEx other recognized carrier the same message "Arrived at Delivery Facility in MANCHESTER - UK" can be correlated with "In Transit", because the package is in transit and have made a pit-stop in warehouse.

This is an example that makes fashion brands worried because this not homogeneous experience among carriers makes customers anxious and nervous turning the customer support activity a nightmare. This was looked by HUUB has an opportunity to improve the experience given to Ecommerce customer when tracking their order.

The challenge was the following: get access to the tracking information from the carriers and then create a "mask" over it capable of simplifying and standardizing the tracking information for the customer. This means that tracking details from distinct carriers in different languages had to be used to classify the tracking status of an order that can go from "Shipped", "In transit", "Out for delivery", "Delivered", "Returned". To avoid any human interaction, a text mining tool was thought as a way to classify the tracking status and then allow the brands to send proper notifications to the customers.

2 Challenges and Issues

Tracking details are small typified messages that are made available to a customer, either via email or by a web portal, like "Arrived at Delivery Facility in MANCHESTER - UK", that are then associated with a pre-defined status. On what concerns NLP, the challenges associated with this type of text can be either data or understanding related.

Data-related problems are regularly associated with the fact that data is scarce, unbalanced, as well as too heterogeneous, leading to a decrease of the effectiveness of NLP models. For the proposed problem and the type of text used it can be identified the following challenges: low-resource languages and unstructured data.

It is a known challenge that less spoken languages, for example African spoken dialects, that receive less attention when compared with world recognized languages, such as English, Mandarin, Spanish and Portuguese. There are some interesting approaches to solve this issue such as cross-lingual transformer language models and cross-lingual sentence embeddings that try to find universal connections/commodities between languages.

Unstructured data, another data-related issue, emerges due to the fact that data doesn't fit neatly into the data base structure of relational databases, a reality that represents the vast majority of data available. For the problem in question, where most of the text represent actions, in not so digital carriers, these details are written in free text by the delivery postman. To tackle this there are many digital initiatives that were created in order to turn open fields into close selection type options, making that way much easier to structure the inputted data.

When one looks into understanding-related challenges things like text ambiguity and synonymy, emerge. These are some specifies that make so unique the NLP field and the problems around it. Ambiguity of the text is to understand and model words around the context they are inserted, meaning that it's necessary to model the elements within a variable's surroundings because words are indeed unique objects. They can acquire different meanings depending on environment that they are included in resulting in ambiguity on the lexical, syntactic and semantic levels. Other phenomenon of natural languages is Synonymy. Completely different words or sentences can express the same idea with different terms which are also dependent on the context they are used. Applied to the problem under study, this phenomenon doesn't have great impact because there is no utility on mimicking human dialog.

Lastly there is normalization vs. information paradox. One of the biggest challenges of NLP is the size that normally text structured representation can get, meaning the very large number of features they can have. To tackle this, natural language needs to be processed, in other words, it has to be normalized. There are several techniques that can be applied from converting plural terms, to tacking out punctuation or even looking at the definitive form of a verb. However, all these processes of natural language normalization can lead to losing part of the information in exchange of being able to generalize it. This normalization/information trade-off is quite common in the study of NLP and its resolution greatly depends on the type of problem and associated data.

Looking specifically for the text classification type of problems, when doing tokenization, one must be aware of the noise among words otherwise a significant number of features will be generated creating problems afterwards when modelling it. Another

important remark is the fact that feature vectors have to absorb the semantic complexities of the text in order to produce better results. On the model training phase, there is the possibility of the attained knowledge may diverge from the real data, resulting in a reduction in the quality of the results. Volume and size of the training data play also a critical role in a learning model; hence data should be large and wide enough to cover most of the text classes under study.

3 Main references, methods and techniques

NLP is an already mature field that mainly focus on the cross-section of computer science, artificial intelligence and most recently data mining. For an NLP researcher the ultimate goal is to read, understand and mostly to make sense of the human languages by machines that translates into converting real-world human tasks into automated tools such as online chatbots, text summarizers, auto-generated keyword tabs among other applications.

3.1 Standard NLP process

A standard NLP process uses either Syntactic Analysis or Semantic Analysis as techniques that helps on the process of understanding natural language. Together they access the validity of a sentence. The syntax refers to the grammatical structure, while semantic refers to the conveyed meaning.

Syntactic Analysis

Syntactic Analysis, also called “Parsing”, focusses on the process of analyzing natural language conforming to the rules of formal grammar. Grammatical rules are applied to categories and groups of words, not individual words. This method basically assigns a semantic structure to text by applying techniques such as stemming and lemmatization. The goal of such tools is to reduce inflectional forms and derivationally related forms of a given word to a common base form.

Stemming is related with simple heuristic process that chops off the ends of words eventhough not always real words are created. One of the most known Stemmers is Porter's algorithm (Porter, 1980). On the other hand, Lemmatization ensures that a correct form of word is created on the language used. Looking to this specific tracking classification context, the words "delivered", "delivery", "delivering" and "deliver" when applied to the Porter’s Algorithm would result in “deliv”, “deliveri”, “deliv” and “deliv”, respectively, while using a Lemmatization technique, all words may result in something that actually exists in English language like “deliver”. Both techniques reduce dimensionality but due to its simple form Stemming process is faster.

Semantic Analysis

Semantic Analysis or Natural Language Understanding (NLU) looks into the way context influences the language, as well as intuition and experience. Semantic Analysis is the process of understanding the meaning and interpretation of words, signs, and sentence structure. One of the biggest differences from NLP is that NLU goes beyond understanding words as it tries to interpret meaning dealing with common human errors like mispronunciations or transposed letters or words.

3.2 NLP text classification

The standard Syntax and Semantic approach are very challenging because they require to program exhaustively pre-defined rules. There are other ways to tackle NLP problems without needing to go through the hard work of parsing sentence grammar. A classification algorithm does not require a specific language or to precisely respecting grammar rules, as long as it can break the text into separate words and measure the effects of those words.

Through induction learning by a sufficient number of classified observations the learners can create classifier models that are able to correlate documents of the same type and its characteristics. Those models when presented to an unclassified document are able to use the inducted rules to correctly predict the class of the document. On the tracking messages context, these models, will try to correctly identify if the message represents an “in transit”, “exception”, “arriving”, “delivered” or “returned” status of a shipment. The big advantage of this approach comparing with Syntax and Semantic interpretation is the computation and modelling speed. The process to build up a text classification problem follows four steps: text cleaning, text encoding, model training and then text prediction.

Text cleaning.

Every data mining process starts with data cleaning and text mining is not different. Text cleaning or text pre-processing includes several techniques such as translation, word case, punctuation, stop words, number filter, accented characters, stemming and lemmatization among other mechanisms that make the next step of encoding much easier because object dimension is greatly reduced without removing important meaning.

Data sparsity is one of the big struggles in text mining. Translating every document to the same language and lowering the text are some of the important processes to ensure a denser set of features are extracted with the relevant information. Also removing punctuation, highly frequent words without information gain (i.e. stop words), numbers and accented characters improve the quality of data. Stemming and lemmatization were

already explained above on the syntactic techniques. All of this will reduce dimensionality due to irrelevant text information and possible typos. The dimensionality is the second worst problem in machine learning and the first is overfitting.

Feature extraction

Since the generality of machine learning models do not deal with attributes in the form of categorical labels but rather with numerical variables, it's necessary to do some kind of text encoding. For NLP problems which have as an input unstructured, weekly structured or semi-structured documents it means that prior training any model to learn patterns and perform predictions the data must be cleaned, previous step, and then converted to sequences of numbers, a vector. The first step is to divide the plain text into words (or alternatively called token), this method is referred as *tokenisation*.

A document however is a sequence of words (tokens) and to extract the features of it, one must apply encoding or vectorization, which is converting that sequence of tokens into a numerical vector. There are 2 different main approaches for that, (1) using Bag-of-words (BoW) just based on words frequencies at the document or (2) using word embeddings, pre-trained models that try to capture words meaning, semantic relationships and its context. On the BoW method each word present on the data set, will be represented by a position on a sparse vector of features and the observation, in this case a tracking message, will only have positive values on the words that belong to it, while on word embeddings it is produced a more dense vectors with lower dimensionality.

Bag of words

Examples of the vectorization encoding with Bag of Words are One-hot Encoding, Count Occurrence, Normalized Count Occurrence or Term Frequency-Inverse Document Frequency (TF-IDF). One-hot Encoding is a basic representation of converting a word into a vector. Each word has an id and the vector has 1 on the positions in which the words with that id are present on the document and 0 if the word does not exist. Count occurrence and normalized count occurrence save the information of frequency, so that words that are represented on a document more than once have a higher impact on its vector representation. The normalization process allows documents with different sizes to be compared without causing bias.

TF-IDF instead of only counting the occurrence of a word in a document it also does so in relation to the entire data set. Words that are common in every message, such as "the", "if" and "is" will rank low even though they may appear many times on a message, since they do not carry a lot of meaning for that particular document. When stop words are applied on pre-processing this is already done, however since problems have specific contexts there are words that are not generally used, but on the corpus of the problem are disseminated equally to all classes (e.g. the word "shipment" on tracking messages).

Although very simple, Bag-of-Words encoding has some shortcomings, such as the data sparsity (high dimensionality with each feature having low representation) and the context of the written words is ignored. This makes the models more prone to overfitting and promotes a process that is computationally expensive.

Word embeddings

Word embeddings are a dense vector representation of words in lower dimensional space. The first well known neural network un-supervised training to create a word embedding was published by Google and is called word2vect. This model retrieves a vector representation of the words from where similarity between words can be captured and defined by measuring how far or near a word is in the vector space to other words vector representation.

For example, using One-hot encoding, a BoW method, and being w1 the vectorial representation of the word “apple”, w2 of the word “banana” and w3 of the word “billion”, the Manhattan distance between every two words results in 1. On the other hand, Word2Vect will capture the semantic similarity of “apple” and “banana”, both fruits, so the vectors representing those words will be closer to each other being its difference lower than the distance between w2 and w3, for example. Words2Vect allows to perform mathematical operations with words. This embedding method was trained on an unsupervised manner, the main issue of Word2Vec is that it still provides a unique representation for a word independently of the context it is used.

Besides the Word2Vect that uses neural networks, another way of generating word embeddings is the use of the co-occurrence matrix. GloVe, that stands for Global Vectors is a method developed in Stanford, that is based on co-occurrence/count matrixes where the global corpus statistics are captured directly by the model. The co-occurrence matrixes are built by considering the words around a given word and counting how many times that word appears in the context of other word. Then the conditional probability of that word appearing near other word is computed and calculated the division between conditional probability of those two words in a presence of specific word k. This is called word-word co-occurrence probabilities. The values can be very high (>1), very low (<1) or proximally the same (~1). The co-occurrence probability ratios along with local statistic component are then used to formulate a loss function responsible to generate the vector values. In the end GloVe leverage the use of both global and local statistics when performing word embedding. In the end each text embedding approach, windows versus full document, can have distinct performance and accuracies depending on the size of the text corpus under study, therefore its usage needs to be access case by case.

Another advancement on this area was produced by Goggle on 2018 and it is named, Bidirectional Encoder Representations from Transformers or simply BERT. It was trained with BooksCorpus (800 million words) and English Wikipedia (2.5 billion

words) data sets and it has a new way of pre-training by introducing bidirectional context on the positional encoder. BERT is a transformer model that uses attention concept to help improving the performance of the word encoding. Self-attention allows to associate each word of input with other positions in the input sequence to obtain information that can help lead to a better encoding for that word. BERT has multiple use cases such as sentiment detection, classification, machine translation, named entity recognition, summarization and question answering.

Both BERT with Recurrent Neural Networks (RNN) proved that more than one representation of a word based on the context it appears is one important notion for word embeddings. However, RNN, which are used for their ability to capture and maintain long term dependencies in their hidden states, for its inherently sequential nature tend to be slow to train and very hard to parallelize. BERT overcome this issue by using self-attention. For this classification problem it can be used as a trained Transformer Encoder stack. On each layer, it applies self-attention, and passes its results through a feed-forward network, and then hands it off to the next encoder. Each position outputs a vector. For the sentence classification example address on this work, the output of that position that received the token CLS can be used as input of the classifier.

3.3 Prediction model

After having a set of features, literally all classification models can be used to create learners able to predict defined classes in a supervised learning manner. Not being the objective of this work to fully understand the functioning of all prediction models some will be named. From logistic regressions, to Naive Bayes, which was one of the initial models used in text mining, there are several options on the model selection for training. When SVMs started to emerge they had a notable success solving this type of problems, since, due to margin maximisation, SVMs resist to overfitting on high dimensions that are typical on text mining. But the last researches on NLP are more focused on deep learning techniques. Recurrent Neural Networks, and more precisely, long short-term memory (LSTM) architecture is beneficial for text mining, since they use the RNNs allow previous outputs to be used as inputs while having hidden states, being this extremely useful to define the meaning of a word having knowledge about the previous words. As a final layer an Activation softmax function function for multi-class classification must be used.

4 NLP Classification implementation

To solve the problem under study it was implemented an application in Python, using Jupyter Notebook, capable of applying data pre-processing/cleaning techniques to the tracking text messages, train distinct algorithms and then evaluate their accuracy to predict a tracking status given a new text input.

4.1 Tracking text dataset

As explained before, tracking messages are texts send by the carrier giving details about the journey of an order. For the problem under study it was used a dataset of 854 tracking messages with 2 columns:

- "Message", input variable with the tracking text details;
- "label_class", normalizad tracking status that can assume five values: "Exception", "Transit", "Pickup", "Delivered" and "Out for delivery";

The first step was to load the csv file with the tracking dataset. In the figure 1 bellow, it is shown a snipped of the used code and a preview of the dataset already loaded.

Fig. 1. Tracking Dataset being loaded using Pandas in Jupyter Notebook

Bellow, figure 2, is also displayed the amount tracking messages for each tracking status. It can be easily concluded that this dataset has a higher number of tracking messages with an associated tracking status "Exception", when compared with the other status. This can be easily explained due to the fact that carriers share an higher frequency of exception messages when an order enters in an exception situation, for example orders that are blocked at customs for tax clearance.

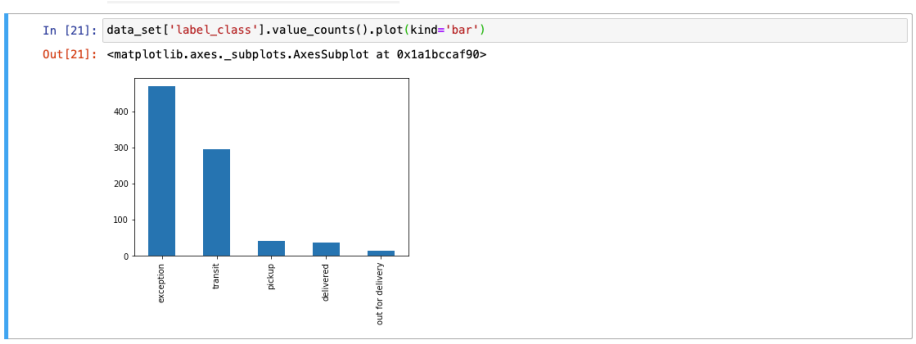

Fig. 2. Number of tracking messages for each type of tracking status using Jupiter Notebook

4.2 Text pre-processing

Data pre-processing or data cleaning is one of the most important steps on any machine learning implementation, especially in NLP problems where the features are generated from words, leading to gigantic number columns, a data sparsity problem, with significant impact on the performance and quality of the algorithm.

The first thing that was analyzed was the language of the tracking message. In order to reduce the data sparsity, it is important to translate all tracking messages to the same language, otherwise the same details in distinct idioms, would become distinct words and therefore distinct features, that in the end represent the same redundant information.

For the use case under study all tracking details were already in English. Nevertheless, if needed there are well-known Python libraries, such as *textblog*, that allows one to get an accurate translation of texts. An important remark is that tracking messages are quite objective type of text, not subjected to implicit interpretation. It can be safely said that no relevant information is lost when applying translation techniques.

The raw dataset is going to be named “original” for the sake of interpretation. The original dataset has 1221words, that would represent 1221 features, if a BoW would be applied. To reduce the number of words it was firstly used a lowercase technique, that ensures that all words are in lower case. The number of unique words reduced from 1221 to 999 words. Then was applied a punctuation tool that removes all characters apart from numbers and letters, achieving a 22% (777 words) reduction on the number of unique words.

Another interesting technique is the removal of stop words, basic words like “the” and “to” used as connectors that don’t bring much value. A snippet of the python code for this technique is displayed on the figure 3, bellow.

```
stop = stopwords.words('english')
data_set['message_pre_processed'] =
data_set['message_pre_processed'].apply(lambda x: ' '.join([word for word in  x.split
() if word not in stop ]))
```

Fig. 3. *Stopwords* function in python for stop words removal

Then, numbers were also removed since they don't bring relevance to the classification, reducing the number of unique words to 646.

It was also analyzed both non-frequent words and small words, less than 3 digits, due to the fact that they can represent a big chunk of the words that a corpus might have. More importantly, by being less frequent those words can lead to a model over-fit, since they don't have significant occurrences that prove its relevance to explain the output variable, in this case the tracking status. The impact of its removal is impressive by leading to a 54%, from 646 to 293, reduction on the number of unique words.

Lastly, it was applied the Porter Stemmer technique that, as previously explained, removes the suffix from a word and reduce it to its root word. After its application it was achieved a number of unique words of 247.

Bottom line, after applying all of these text cleaning techniques it was possible to reduce the original number of words from 1221 to 247, a meaningful 79% reduction, as it is shown either on the table and bar chart displayed on figure 4.

	Data	n_words
0	original	1221
1	lower	999
2	ponctuation	777
3	stop_words	693
4	remove_digits	646
5	TF<3	316
6	nchar<3	293
7	PorterStemmer	247

Fig. 4. Data cleaning techniques impact on the number of unique words using Jupyter Notebook

4.3 NLP classification model evaluation for distinct vectorization methods

For the case under study it was used Naïve Bayes, SVM and Logistic regression as classification algorithms. To understand the impact of the data cleaning and vectorization methods on each model, it was analyzed how the model accuracy performed for each step of the data cleaning process, using a specific vectorization method from 3 possible candidates: one hot encoding, TF-IDF and word2vec. The accuracy of the model is calculated based on the number of the correctly identified class. It also used a 30% split of the original dataset for training and prediction.

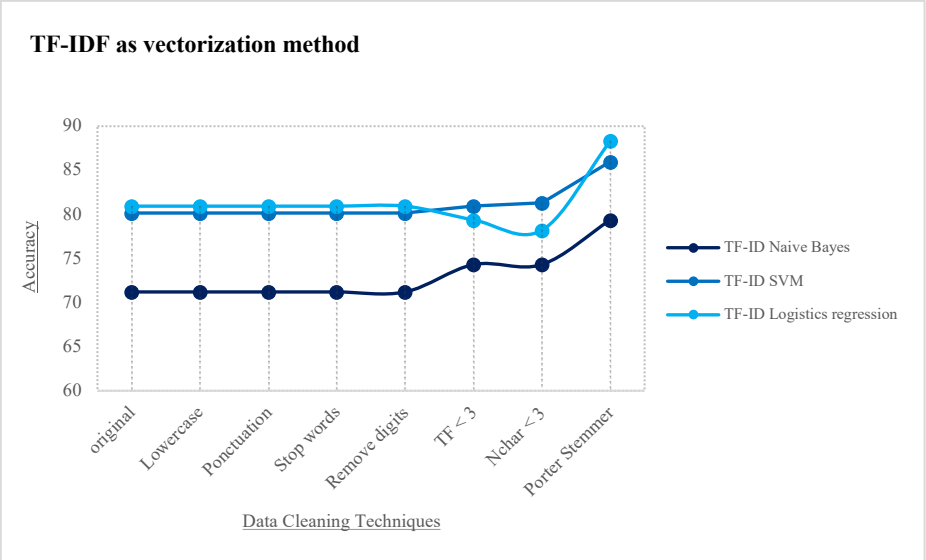

Fig. 5. Analysis comparing the accuracy of the models using TF-IDF

On figure 5 is displayed the comparison between the three distinct classification models using as vectorization method the TF-IDF. It can be concluded that both SVM and Logistic regression models get good accuracies overall and the last one improves significantly using the Porter Stemmer technique, achieving a final accuracy around 88,32%. It is also important to highlight that the effect of each data cleaning method is applied sequentially, meaning that the reduction on the number of features is accumulated along the way.

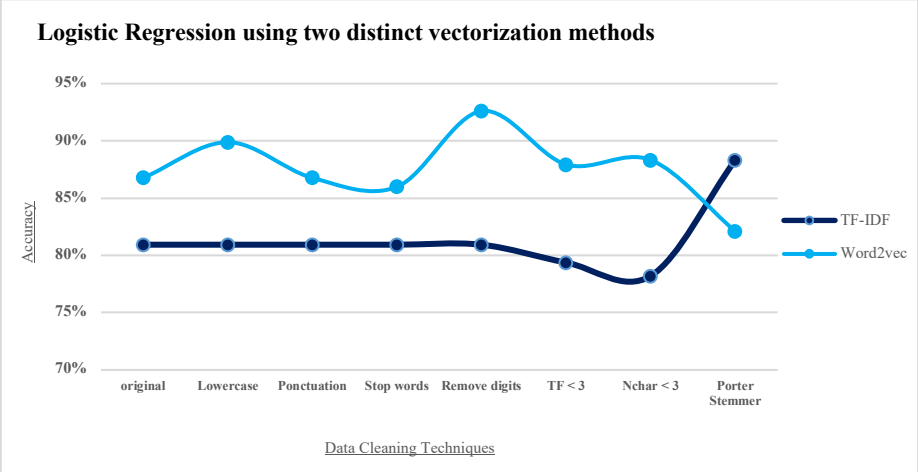

Fig. 6. Impact on using distinct vectorization methods on Logistic Regression

Now, it was selected the logistic regression model and it was tested the impact of the data cleaning techniques and the two distinct vectorization methods, TF-ID and Word2vec, on the accuracy of the model. As it can be easily concluded, word2vec proves to be a method that brings the accuracy up when compared with TF-IDF. However, when applied the Porter Stemmer method, the accuracy of the logistic regression model decreases significantly for the Word2vec and increases for the TF-IDF. This effect happens because this data cleaning technique can create simpler forms of words or terms, something that jeopardizes the ability of the Word2vec to access the context of the words. At the remove digits stage, the Logistic regression model using Word2vec achieves an interesting 93% accuracy when predicting a tracking status. The confusion matrix of the model is displayed on the figure 7, bellow:

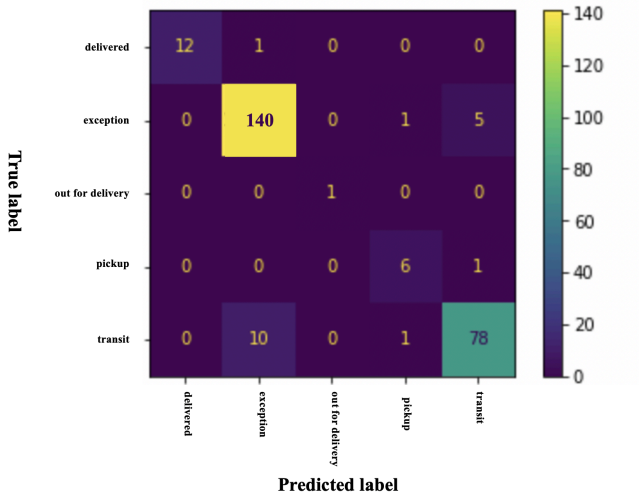

Fig. 7. Confusion Matrix

Analyzing the confusion matrix, it can be concluded that the class with most false positives is “Exception”, in 10 times was predicted when the true label was “Transit” and 1 time “Delivered”. This may have to do with the fact that the origin dataset is unbalanced with a significant predominance of “Exception” observation when compared with the other tracking status. To overcome this, data augmentation (DA) techniques could be applied on the data preparation stage it could be applied a class balancing strategy, either oversampling the less frequent classes or under-sampling the most frequent one. Alternatively, to increase the size of the training data, synthetic data can be generated to reduce the prediction bias and enhance the model robustness.

The accuracy is not always the best metric for defining the usefulness of the model. For example, when we have very few positive cases in a huge dataset (class imbalanced dataset), even if the model fails to detect any true positive case, the accuracy will remain high (due to the high value of true negative). On the specific context of the tracking experience a false negative when an order is in exception can generate negative impact because the customer/logistic operatior won’t be notified for the issue in time and no one would act to solve the shipment problem.

On top of accuracy, a model can also be evaluated using the recall and precision scores.

	Precision	**Recall**	**f1-score**	**Support**
“Delivered”	1.00	0.92	0.96	13
“Exception”	0,93	0,96	0,94	147
“Out for Delivery”	1.00	1.00	1.00	1
“Pickup”	0,75	0,86	0,80	7
“Transit”	0,93	0,88	0,90	89
Accuracy			0,93	257
Macro avg	0,92	0,92	0,92	257
Weighted avg	0,93	0,93	0,93	257

Fig. 8. Table summarizing evaluation metrics

The metrics displayed on figure 8 shows that the Logistic Regression model, using Word2vec embedding, represent satisfactory precision predicting “Delivered”, “Exception”, and “Transit” shipments, meaning most of the times that labels were predicted the true tracking status was equal. Analyzing the recall metric one can deduct that the model has a higher capability detecting the “Delivered” and “Exception” shipments, achieving a prediction of those labels on 92% and 96% of the times they have occurred, respectively. On the testing dataset “Out for the delivery” has a poor representation, not being possible to take conclusions on the goodness of fit of the learner to predict this class. Looking into the “Pickup” tracking status, there is an indication that the model is not performing well predicting this class due to the fact that on the test dataset it was predicted 2 times a “Pickup” when the message represented a distinct class and also failed to classify 1 observation, by confusing with the “Transit” class.

5 Conclusions

NLP has a wide subset of problems, such as sentiment analysis, text classification, machine translation, named entity recognition, summarization and question answering, among others. The big challenges of all these problems are the unstructured way that the data source is presented, and its high dimension caused by all the variance a text document can have (misspelling, figures of speech, ambiguity of words meaning and so on). This makes the data pre-processing process and the feature extraction task extremely relevant parts of creating a successful classifier.

On the data cleaning some methods were presented: the removal of features or words that usually do not carry any problem related information is something that is usually helpful. That is done for common words in every document cluster like determinants, connectors and frequent verbal forms, but also with numbers, case variations, verbal forms terminations and punctuation. However, the context of the problem must be considered while performing data pre-processing and, on this work case, the tracking messages of the carriers may have for instance frequent words in every state that are usually not considered as a stop word. For that TF-IDF may help to find those cases.

The feature engineering has different approaches based on one hand on a Bag of Words that look at word's frequencies or on the other hat based on word embeddings that are more complex but dense word representation of the words that are trained with a big corpus and are enable to relate words on feature space that are connected to each other. Generally pre-trained models are used but adding some problem context training may help to increase the accuracy. Bag of Words example techniques are one-hot-encoding or TF-IDF, while word embeddings explored on this paper are word2vec (Google), GloVe (Standford) and BERT (Google).

On the implementation section of this work, it was proved that word embedding outperform bag of words based on frequencies TF-IDF in what concerns vectorization preprocessing. Other important subject is the combination of data cleaning techniques with vectorization that sometimes can have contradictory results. By applying Porter Stemming while using a vectorization method with semantic context, Word2vec, the trimming caused a loss of syntax in the corpus jeopardizing that way the performance of the embedding algorithm.

After the data pre-processing and feature extraction the model training can be applied. On this work, logistic regression, SVMs and Naive Bayes models were used for text mining purposes combined with both BoW and embedding in order to predict a tracking status class. In the end, this combination, in specific the Logistic Regression and Word2vec, demonstrated interesting prediction capacity with an accuracy of around 92%.

As for future work, the authors intend further investigate the use of deep learning, Recurrent Neural Networks, that are able to recurrently use the state of the hidden layer

as in input for passing the vector the next word into the network, or a transformer technique (BERT). Another interesting approach prior to implement distinct learners would be the optimization of the dataset by balancing the prediction classes aiming bias reduction and also the empirical test of hyper parameter tuning on the already implemented models.

6 References

1. Porter, M.F. “An algorithm for suffix stripping”, Program, Vol. 14 No.3, pp. 1980.
2. Domingos, Pedro. “The Master Algorithm”, 2015.
3. Jeffrey Pennington, Richard Socher, and Christopher Manning. “Glove: Global vectors for word representation.” Proc. Conf. on empirical methods in natural language processing (EMNLP). 2014.
4. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. “Distributed representations of words and phrases and their compositionality.” Advances in neural information processing systems. 2013.
5. Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. “Efficient estimation of word representations in vector space.” ´preprint:1301.3781 (2013).
6. Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” preprint: 1810.04805 (2018)
7. Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention Is All You Need” prepint: 1706.03762 (2017)