SDS 2020 - Module 2: Group Assignment
Group members: Lukas Bielmeier, Marta Fedeli

STAKEHOLDER REPORT

Network Analysis, NLP exploration and ML methods for predicting tweet authors based on text

1. Introduction

The following paper aims at diving deeper into the world of online customer support through Natural Language Processing (NLP) and Network Analysis (NA). Through NA, we looked at which companies interact the most with their customers. Through NLP and later, Unsupervised Machine Learning (UML), we seeked underlying, shared patterns in the tone and the adopted word-choice while also looking into the most common topics of the dataset at hand. Finally, through Supervised Machine Learning (SML), we tried to predict whether a tweet was written by a consumer or a company representative. This is especially valuable because on the one hand, it allows to clearly differentiate between two sides of a conversation - the query (which typically comes from an individual) and the response (which typically comes from a company). On the other hand, being able to recognize customer support tweets from individuals which do not specifically mention or tag the company but still have an issue they would like a company to address, could prove to be quite useful.

2. Choice of data

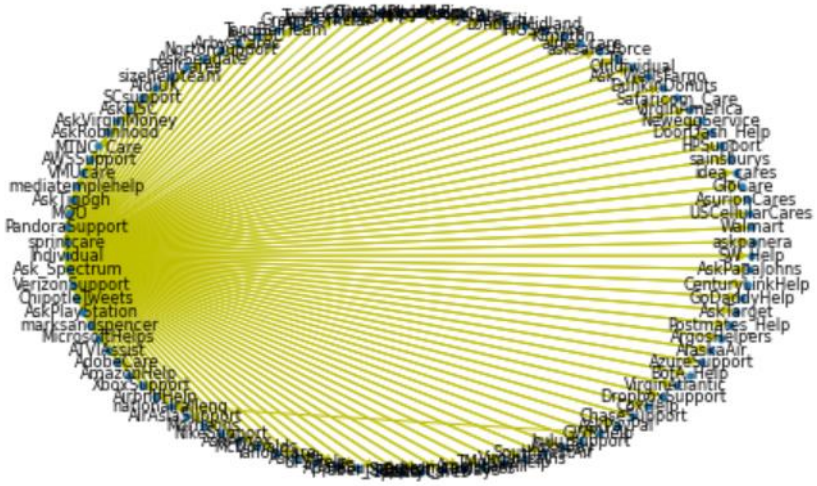
The dataset at hand was accessed and downloaded from Kaggle. In its original unprocessed form, it contains about 3 million tweets between companies and private customers, originally collected "to aid innovation in natural language understanding and conversational models, and for study of modern customer support practices and impact" (Thought Vector, Axelbrooke S., n.d.). Because of the dataset's high volume and the subsequent computational problems, we decided to create a reduced version of it, with slightly more than 50.000 observations, a number still deemed sufficient for the type of analysis conducted within our paper.

3. Network Analysis

In order proceed with NA, after preprocessing the data, we proceeded by creating an edgelist containing information on who responded to whom. At this point, we grouped all consumers into one single group called **Individual** and started to compare the interaction of this group with the one of company's customer support agents.

To create our network, the dataset feature of the author ID was defined as the source of the edge, and the ID of the responder as the target. The feature "received_response" indicates whether an edge is present. We proceeded by calculating three different centrality measures for our network observations and added them as node attributes: our graph object was now ready for further analysis.

We firstly set out to identify the most frequent authors and responders based on our edgelist. Clearly, Individuals leads in both categories because of this category being the only grouped one. We proceeded to calculate some network level characteristics of our network to gain some further insights. The network is defined by very low levels of density and transitivity, as well as by a very high level of reciprocity. These numbers are perfectly coherent with the nature of our data, as we are analysing customer support data, where in general there are specific edges between two nodes that have a conversation. It is also unlikely that when two companies are connected to the same individual, these two companies also are connected (which is indicated by the low transitivity score). The high reciprocity also makes sense, as it is typical for a customer support network to respond to queries, which in our data are represented by tweets from individuals. This means that companies usually have a conversation with consumers, thus the network is highly reciprocal. As shown below, the Individual group is leading in terms of centrality.



"Individuals to Company Representatives" network object

We can clearly see how most nodes are connected to one specific node, **Individuals**, while having almost no edges across themselves. However, to evaluate this statement, we wanted to find out if there are any communities within the network that we could detect. After applying a community detection algorithm imported from *networkx,* we found two communities in our network. Again, this appears to be logical, as we have the consumers on the one hand and the company representatives on the other.

4. Natural Language Process (NLP) Analysis

Through NLP we further explored the dataset, this time not in terms of relations between the different tweets' authors but in terms of shared linguistic characteristics in the text of the tweets. After having processed and cleaned the tweets, we started our analysis on the tokens at hand through simple word counts.

Since often a graphic object can convey a message with the highest immediacy, the first tool chosen for exploring the tweets text was the word cloud displayed below.

Customer support tweets – word cloud

The word cloud promptly gives a feel of a polite, friendly tone that is perfectly fitting with what is expected from the customer service field. As displayed in the functional notebook, a more precise word count highlighted how "please" is in fact the most used word and it is followed by other terms such as "sorry", "thanks", or the auxiliary "would", which serves the purpose of posing questions or suggestions more politely. Therefore, both the word cloud and the word count are perfectly aligned with what is to be expected of customers' service appropriate vocabulary, and this tool brought to surface nothing unsuspected in terms of the tweets word choice.

Onto a deeper level of analysis, the dataset was scouted to find potential macro-topics that the different users are most commonly dealing with. To do so, we applied first Latent Semantic Indexing (LSI) and then Latent Dirichlet Allocation (LDA). Out of the two, the best performance was achieved by the latter which was in fact able to separate topics in a more coherent way. For example, out of the ten most common topics chosen, some topics dealing with iPhone-related issues emerged. However, since the LDA division of topics was still somewhat imprecise, Word2Vec was adopted to study word embeddings within our corpus of tweets. After constructing the model, we explored it by choosing three terms that stood out to us from the previous topic clustering. Word2Vec managed quite successfully to cluster semantically similar words; "iOS" and "Account" similar words fit very well with their context. "Amazon" on the other hand, proved not to be so successful, but however it still showed a pretty good match in words such as "prime" (one of the Amazon's services) or "paid".

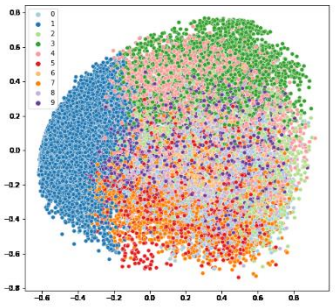
Word2Vec outputs proved to be quite interesting too, seeing how there are in fact semantic clusters emerging. Therefore, we came to conclude that perhaps, a finer tuning of the LDA model would have more clearly defined the same semantic clusters.

5. Unsupervised Machine Learning (UML)

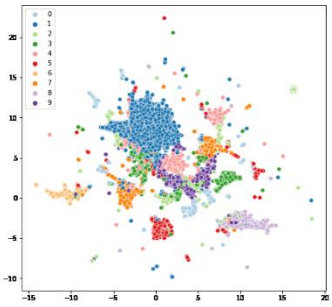
Seeking to further explore topics within the dataset, we run the LSI and LDA corpora through the classic UML steps of dimensionality reduction (we chose as algorithms PCA and UMAP) and through clustering algorithms (KMeans and Hierarchical Clustering).

Out of the plots created combining these variables (LSI vs LDA corpora, PCA vs UMAP, KMeans vs Hierarchical Clustering), the combination which seem to have performed better in terms of "traditional" UML plots is that of LSI and UMAP, while not much difference was observed with this combination in terms of KMeans clustering as opposed to Hierarchical Clustering.
Unfortunately, we must conclude that, as it often happens in NLP topic modelling, there is no clear take away from the latest analysis carried out, as also shown by the different UML plots below.

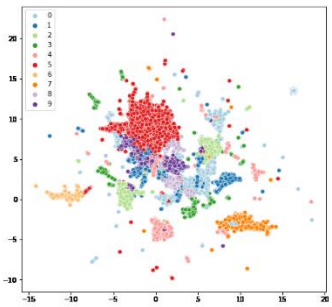
LSI



PCA and KMeans

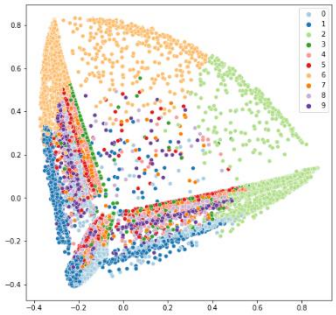


UMAP and KMeans

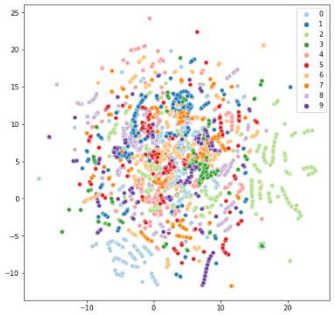


UMAP and Hierarchical Clustering

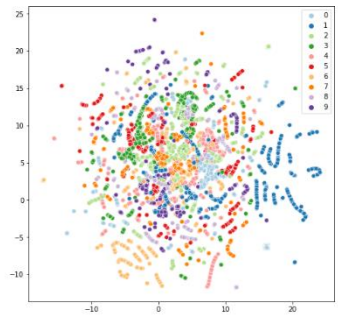
LDA



PCA and KMeans



UMAP and KMeans



UMAP and Hierarchical Clustering

While LSI provided a clear distinction among clusters in the above UMAP plots, the same distinction, as previously mentioned, was not extremely coherent from a logical standpoint. On the other hand, LDA generated a more coherent topics-set and a more reasonable topic distribution, but it failed in attaining great separation between these clusters.
However, because of its nature, LDA can be included among UML methods and through it, as also explored with Word2Vec, some underlying patterns (topics) have started to emerge.
Although for now, we turn the focus on our analysis on the sole ability of a SML model to predict whether a tweet was written by a customer or a company, further developments of this code could perhaps use SML to predict the macro-topic of a customer service tweet, alerting the most competent member in dealing with that specific topic out of the company's customer support team.

6. Supervised Machine Learning (SML)

In this section, we will utilise and build upon our previous analysis of NLP. Our goal is to be able to predict which tweet comes from a company and which tweet comes from a customer, based on their tweet text. Furthermore, we want to see how our model results differ based on the type of text

representation that it is chosen. We apply three different text representations to our SML models: The Bag of Words (BoW) representation, the tf-idf scaled version of it, and the LSI reduced text representation. Each of these is applied to the *LogisticRegression* (henceforth, LR) classifier and the *RandomForestClassifier* (RFC).

Upon inspection of the results of the LR classifier, we can draw some insightful conclusions. The LR performs quite well with the BoW text representation and seems to improve once we use the scaled tf-idf representation. Interestingly, the LR classifier performs significantly worse when we use the LSI reduced text representation, with model accuracy losing more than 10%. Below, the classification reports of the tf-idf text representation and the LSI reduced text representation are shown for comparison:

	precision	recall	f1-score	support
0	0.88	0.93	0.90	6853
1	0.91	0.85	0.88	5647
accuracy			0.89	12500
macro avg	0.89	0.89	0.89	12500
weighted avg	0.89	0.89	0.89	12500

Classification Report for TF-IDF model

	precision	recall	f1-score	support
0	0.74	0.90	0.81	6853
1	0.83	0.61	0.70	5647
accuracy			0.77	12500
macro avg	0.78	0.75	0.76	12500
weighted avg	0.78	0.77	0.76	12500

Classification Report for LSI model

Next, after having applied the different text representations to the RFC, we can observe similar patterns to the previous LR classifier. Again, the accuracy and other metrics of the model improve significantly after tf-idf scaling of the BoW. Also, the same metric also deteriorates using the LSI reduced text representation. However, we can see that the metrics do not get worse as much as they did with the first classifier. In fact, the LSI model, although less accurate than the tf-idf model, performs better than the BoW model using the RFC.

	precision	recall	f1-score	support
0	0.80	0.83	0.82	6853
1	0.79	0.75	0.77	5647
accuracy			0.79	12500
macro avg	0.79	0.79	0.79	12500
weighted avg	0.79	0.79	0.79	12500

Classification Report for BoW model (RFC)

	precision	recall	f1-score	support
0	0.82	0.88	0.85	6853
1	0.84	0.77	0.80	5647
accuracy			0.83	12500
macro avg	0.83	0.83	0.83	12500
weighted avg	0.83	0.83	0.83	12500

Classification Report for LSI model (RFC)

Overall, we conclude that the model with the strongest predictive accuracy and the best complementing metrics is the tf-idf scaled text representation applied to the LR classifier. Using this model specifications, we achieve an accuracy of almost 90% and very good precision and recall scores as well.

Considering its high performance, applying this model would prove useful in reaching the above-mentioned goals of automatically differentiating between a query and a response as well as finding “headless” tweets seeking support. In conclusion, this could potentially lead to an improved workflow for people working in customer support.