POLITECNICO DI TORINO
&
EURECOM SOPHIA ANTIPOLIS

Double Master's Degree in
Data Science and Engineering

Politecnico
di Torino

EURECOM
Sophia Antipolis

Double Master's Degree Thesis

Maximizing the Voice of the Customer with NLP: A Tool to detect Insights with Sentiment Analysis and Named Entity Recognition

Supervisors

Prof. Paolo GARZA

Prof. Maria ZULUAGA

Eng. Nikos SYMEONIDIS

Candidate

Simone PAPICCHIO

December 2022

English abstract

The use of machine learning techniques to analyze customer feedback has the potential to provide valuable insights for businesses. In this thesis, we present a tool for extracting Voice of the Customer (VoC) data from tweets and analyzing it using Natural Language Processing (NLP) techniques. The tool consists of four macro steps that are flexible and can be run in parallel, allowing for efficient and effective data extraction and analysis.

We propose two architectures: one for Sentiment Analysis and one for Named Entity Recognition (NER), and demonstrate that our approach outperforms the baselines reported in the literature in both tasks. In the case of Sentiment Analysis, we use Human Level Performance to demonstrate Data & Conceptual shift. For NER, we propose a method for processing the data to leverage different word embeddings and a way to conduct ablation studies using TensorBoard.

The tool has been deployed and is being used by Toyota Motor Europe to conduct real-time VoC analysis. The results of research show that the integration between NLP and VoC analysis can provide valuable insights for businesses and contribute to the NLP literature.

French abstract

L'utilisation de techniques d'apprentissage automatique pour analyser les commentaires des clients a le potentiel de fournir des informations précieuses pour les entreprises. Dans cette thèse, nous présentons un outil permettant d'extraire les données de la voix du client (VoC) à partir de tweets et de les analyser à l'aide de techniques de traitement du langage naturel (NLP). L'outil se compose de quatre macro-étapes qui sont flexibles et peuvent être exécutées en parallèle, permettant une extraction et une analyse des données efficaces et effectives.

Nous proposons deux architectures : l'une pour l'analyse des sentiments et l'autre pour la reconnaissance des entités nommées (NER), et nous démontrons que notre approche surpasse les lignes de base rapportées dans la littérature pour les deux tâches. Dans le cas de l'analyse des sentiments, nous utilisons la performance au niveau humain pour démontrer le changement de données et de concepts. Pour le NER, nous proposons une méthode de traitement des données afin d'exploiter différents encastrements de mots et un moyen de mener des études d'ablation à l'aide de TensorBoard.

L'outil a été déployé et est utilisé par Toyota Motor Europe pour effectuer une analyse de la VoC en temps réel. Les résultats de la recherche montrent que l'intégration entre la PNL et l'analyse de la VoC peut fournir des informations précieuses pour les entreprises et contribuer à la littérature PNL.

Acknowledgements

I am deeply grateful to my family for their love and support throughout my academic journey. Without their constant encouragement and sacrifice, I would not have been able to complete this thesis. Special thanks go to my parents, Marco and Lucia, for their unwavering belief in me and for always being there to provide guidance and advice.

I am also thankful to my partner, Eleonora, for being my rock during this long and sometimes difficult journey. Her support, encouragement and understanding have kept me going, even on the most difficult days. I am truly grateful for your presence in my life and the impact you have on it. I could not have done it without you and I will be forever grateful.

I am also deeply grateful to my friends, who provided me with endless love, support, and motivation throughout this process. Especially, Riccardo Prestigiacomo and Francesca for consuming several cans of oil during the lock down and Martina for the litres of herbal tea drunk between promenades. I must also mention both Riccardo Pastorino and Nour for making this journey unforgettable and full of memories through the many trips to Liguria and beyond. I also have to mention Alessio for always being helpful and supportive, the path would have been very different without you; Andrea Torredimare for the numerous tech tips and coffees shared between one study break and another; Federico Pes unfailing travelling companion between numerous storms called projects and many others for the endless amount of jokes, memes and puns that entertained me and helped me stay focused and motivated. Thank you for being there for me and helping me stay sane during the long hours spent in the study room.

I must also thank the Europe+ program for providing me with the funding and support to participate in Double Degree program. The resources and opportunities provided by Europe+ have been instrumental in my academic success, and I am grateful for the impact that the program has had on my life.

To my father, who ignited the spark of imagination in me.
To my mother, who showed me the power of determination and hard work.
To my sister, who taught me to never let adversity stand in my way.
To my brother, who showed me the beauty of being unique and standing out from the crowd.

Table of Contents

List of Tables

List of Figures

Chapter 1

Background

1.1 Toyota Business Practice (TBP)

The Toyota Business Practice (TBP) is an important tool for Toyota associates because it provides a standard approach to problem-solving that is based on the principles of the Toyota Way. The Toyota Way is a set of guiding principles that are used to shape the culture and practices of Toyota, and it includes two pillars: continuous improvement and respect for people. TBP can be applied to all types of problems across the company, and it helps to create a common language and understanding among global Toyota associates.

The TBP approach to problem-solving is based on the "plan, do, check, act" framework, which is a systematic way of identifying and addressing problems. This approach emphasizes the importance of viewing problems as opportunities for improvement, rather than as negative events. By using the TBP mindset, Toyota associates can approach problems productively, systematically, and can work together to identify and solve problems in a way that helps the company grow and improve.

TBP is composed of the following steps

1. Clarify the problem: Identify the problem that needs to be addressed and define it clearly.

2. Break down the problem: Analyze the problem and its underlying causes in order to better understand it.

3. Set a target: Determine the ideal state that you want to achieve, and define specific and measurable goals that will help you to reach that state.

4. Analyze the root cause: Identify the underlying causes of the problem and determine how they contribute to the problem.

5. Develop countermeasures: Develop a plan to address the problem, taking into account the root causes and the desired target state.

6. See countermeasures through: Implement the plan and monitor its progress to ensure that it is effective.

7. Evaluate both results and process: Evaluate the results of the plan to determine its effectiveness, and assess the process used to implement it to identify any areas for improvement.

8. Standardize successful processes: Once the problem has been effectively addressed, document the process used to solve it so that it can be used as a model for addressing similar problems in the future.

By following these steps, Toyota associates can approach problem-solving in a systematic and effective way, and can work together to identify and solve problems in a way that helps the company grow and improve.

The TBP approach to problem solving was successfully implemented in this industrial thesis. The process was thoroughly understood and applied to address several problems that arose during the development of the project. As a result of using the TBP approach, the project was completed successfully and met all necessary objectives. This demonstrates the effectiveness of the TBP approach in industrial settings and its utility for addressing complex problems.

1.2 Project Lifecyle: MLOps

MLOps, or "Machine Learning Operations," is a practice that combines software engineering and data engineering techniques to develop, deploy, and manage machine learning models in production. Because MLOps is a relatively new field, there is no one "best" project lifecycle for it. However, there are some general best practices that can help ensure the success of an MLOps project.

One of the key principles of MLOps is collaboration and communication between different teams, such as data scientists, software engineers, and IT operations. This means that the project lifecycle should be designed to facilitate collaboration and cross-functional communication.

Another important aspect of MLOps is the automation of certain processes, such as model training, testing, and deployment. This can help improve the speed, reliability, and reproducibility of machine learning models in production. As such, the project lifecycle should include steps for automating these processes, as well as for monitoring and maintaining the deployed models.

Here is a general outline of a project lifecycle for MLOps:

1. Define the business problem and objectives of the project, as well as the data and algorithms that will be used to solve the problem. This is an important first step, as it helps to ensure that the project is aligned with the organization's goals and objectives. It also helps to identify the data and algorithms that will be used for the project.

2. Set up the MLOps infrastructure, including the tools and platforms that will be used for model training, testing, and deployment. For example, the project team might choose to use Git for version control, AWS SageMaker for model training and deployment, and Datadog for monitoring the performance of the deployed models.

3. Develop and train the machine learning model using the data and algorithms defined in the previous step. This step involves the actual development and training of the machine learning model. This might include steps such as data preprocessing, feature engineering, model training, and performance evaluation. It is important to perform these steps in a repeatable and reproducible manner, so that the model can be trained and evaluated consistently over time.

4. Automate the testing and deployment of the machine learning model. This step involves creating automated processes for testing and deploying the

machine learning model. This might include steps such as creating test data sets, running automated tests, and deploying the model to a production environment. Automation can help to ensure that the model is tested and deployed consistently and reliably, without the need for manual intervention.

5. Monitor and maintain the deployed model. This step involves monitoring the performance of the deployed model in production, and performing regular maintenance and updates as needed. This might include steps such as tracking the model's performance over time, detecting and addressing any issues that arise, and updating the model with new data and algorithms as needed. Regular monitoring and maintenance can help to ensure that the deployed model continues to perform well and meet the organization's needs.

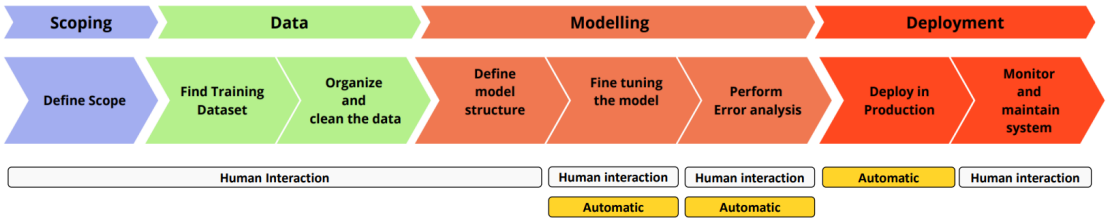


Figure 1.1: Project Lifecycle used to deploy Sentiment Analysis and NER models

The project life cycle that we use in our work for the implementation of the Deep Learning (DL) architecture comprises several phases, as illustrated in the figure 2.1. Some of these phases require human interaction, while others are automated. The initial stages of the process are based on human interaction. However, as the project progresses, fine-tuning and error analysis involve automated components specific to the library used. Once the model is implemented, the tool covers all stages, from the raw data set to prediction. Monitoring and maintenance of the system takes place manually to ensure that the system continues to function properly and provide accurate forecasts. Overall, this project life cycle provides a structured approach to the implementation of the DL architecture, incorporating both human skills and automated processes.

In order to protect the confidentiality of the organizations involved in the research presented in this thesis, we have omitted certain details used in the development of our findings. This may include the names of specific software or tools, details about Toyota's infrastructure, or other potentially compromising information.

Chapter 2

Project scope: Voice of Customer from Tweet

The automotive industry is highly competitive and constantly evolving, making it essential for companies to stay attuned to the needs and wants of their customers. One way for companies to do this is by detecting and analyzing the Voice of Customer (VoC), which refers to the feedback and opinions of customers about a company's products, services, and overall brand. By detecting the VoC, automotive companies can gain valuable insights into what customers like and dislike about their products, as well as identify areas for improvement. This information can be used to inform product development, design, and marketing efforts, helping companies to stay ahead of the competition and increase customer satisfaction. Additionally, detecting and responding to the VoC can improve a company's reputation and build customer loyalty, leading to increased sales and revenue.

One way to extract this information is through social media, as many people use social media platforms to share their thoughts and experiences with a company's products or services. Additionally, using social media to extract VoC can help companies to track and monitor customer sentiment in real-time. This can be particularly useful for identifying potential problems or issues before they escalate, as well as for responding to customer inquiries and concerns quickly and effectively.

The tool reported in this thesis can be used on any kind of social media but the project so far focus only on Twitter. Twitter is one of the most popular social media platforms, with millions of users around the world. It is known for its real-time nature, allowing users to quickly and easily share their thoughts and experiences with a wide audience.

There are several reasons why companies might choose to focus on Twitter when extracting Voice of the Customer (VoC). One reason is that Twitter is a public platform, which means that anyone can see the tweets that are posted on it. This can make it a valuable source of information for companies looking to gather feedback from a wide range of customers.

Another reason why companies might focus on Twitter is that it is easy to search for specific keywords or hashtags on the platform. This can make it easy for companies to find and track customer feedback about their products or services.

Additionally, Twitter has a robust API (Application Programming Interface) that allows developers to access and analyze the data on the platform. This can make it easy for companies to extract and analyze customer feedback from Twitter in order to gain insights into their customers' experiences.

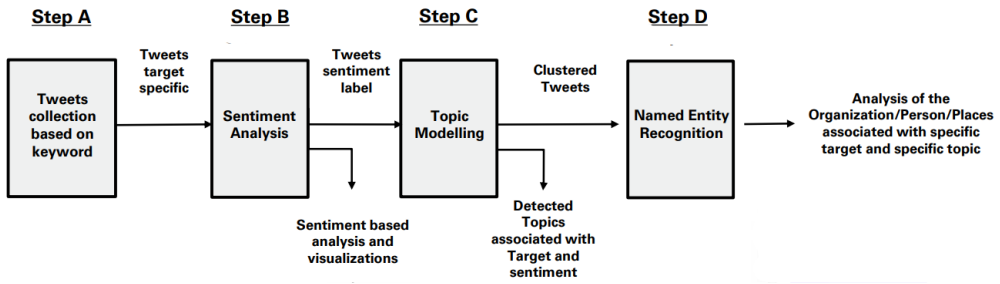


Figure 2.1: Proposed tool to extract Voice of Customer from tweet

The proposed tool for this thesis is illustrated in the figure 2.1. It is composed of four macro steps that are flexible and can be executed in parallel. The decision-making flow may differ from the one shown in the figure.

The first step, labeled as step A, is responsible for collecting tweets based on a specified keyword in any language. This step allows users to tailor the tool to their specific needs by providing a keyword that is relevant to their research.

Once the tweets have been collected, the sentiment analysis model in step B takes over. This model is trained to label each tweet as positive, neutral, or negative based on the language used in the tweet. This step is used to understand the overall sentiment of the collected tweets.

In step C, the output of the sentiment analysis model is fed into the topic

modeling algorithm. This algorithm groups the tweets into clusters in an unsupervised fashion, allowing users to quickly identify common themes and topics in the collected data.

Finally, step D is used to extract the names of the organizations, people, and places mentioned in each tweet. This step allows users to easily identify entities mentioned in the tweets and potentially gain insights into the relationships between these entities

Overall, the proposed tool is designed to provide users with valuable insights into the content of tweets based on a target keyword. By using advanced natural language processing techniques, the tool is able to extract entities, analyze sentiments, and identify common themes and topics in the collected data.

One of the key features of the tool is its flexibility. Users can use the tool to analyze the collected data in a variety of ways, depending on their specific research needs. For example, they can explore the entities mentioned in the positive tweets related to a specific topic, or they can analyze the sentiment of tweets containing a specific target keyword. The tool also allows users to analyze the collected data either together or individually. This means that users can gain insights into the relationships between different entities and how they are mentioned in relation to a specific topic or target keyword.

This work does not explain Topic Modelling, as it has already been implemented in Toyota for a different target. However, steps A, B, and D are analyzed in detail, providing a reproducible approach for other researchers to follow. We have also discussed the challenges and limitations we encountered during our analysis, with the aim of helping other researchers avoid common pitfalls and quickly address similar problems in the future.

2.1 Step A: Tweet collection based on keyword

Step A shown in figure 2.2 involves collecting tweets based on a specified keyword in any language. This step enables users to customize the tool for their research by providing a keyword that is relevant to their topic of interest.

Step A.1 involves translating the keyword specified by the users into any language they choose. To perform the translation, we use DeepL[1], a state-of-the-art neural machine translation service developed by the German company DeepL GmbH. DeepL uses deep learning algorithms to provide high-quality translations between multiple languages, and it has been shown to produce more accurate and fluent translations than other machine translation services. This is due to its use of advanced neural network architectures and large amounts of training data.

Step A.2 involves using the Twitter API to search for tweets that match the translated keywords. To do this, you first need to obtain a developer account with Twitter and create a new application. Once you have done this, you can use the API's search endpoint to specify the keywords you want to search for, and the API will return a response containing a list of matching tweets. The response will be in JSON format, and you can use it to extract the relevant information from the tweets, such as the text, the user who posted it, and the time it was posted. It is important to note that the Twitter API has limitations on the number of search queries that can be made within a certain time period, as well as on the amount of data that can be retrieved in a single request. Therefore, it is crucial to carefully plan and optimize your use of the API to avoid reaching these limits. This will ensure that you can efficiently collect the tweets you need for your analysis.

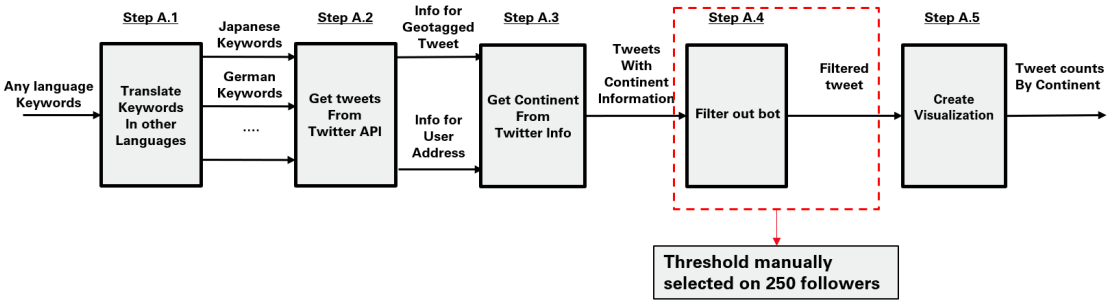


Figure 2.2: Step A: step-by-step methodology

After extracting the continent from each tweet, the tweets that are detected as

[1]https://www.deepl.com/

bot are filtered out. One possible way to detect if a tweet is from a bot is to look at the user's account and activity. Bots often have a large number of followers, but a low number of tweets and a low engagement rate (e.g. likes, retweets, replies). They may also have a generic profile picture and a lack of personal information in their bio. In addition, bots may have a high tweet frequency and a lack of variation in their tweets. Another way to detect bots is to analyze the content of the tweet itself. Bot-generated tweets often contain repetitive or generic language, and may include hashtags, links, or other content that is unrelated to the main message of the tweet. They may also include keywords or phrases that are commonly used by bots, such as "follow me" or "retweet this". The overall pattern of tweets from the user can be also a factor to determine if they are likely to be a bot. For example, if a user consistently tweets the same content at regular intervals, or if they retweet a large number of tweets from other users without adding their own comments, this may indicate that they are a bot.

We filter out tweets from users with fewer than 250 followers, based on our manual analysis of the collected tweets. This threshold is not set in stone, and it may be adjusted in the future if necessary but, it has proven to be effective at removing a significant number of bot-generated tweets from our dataset. It is important to note that detecting bots is not an exact science, and some bots may be able to evade our detection methods. However, we continue to store all the collected tweets so that we can apply new detection methods and re-analyze our dataset if necessary.

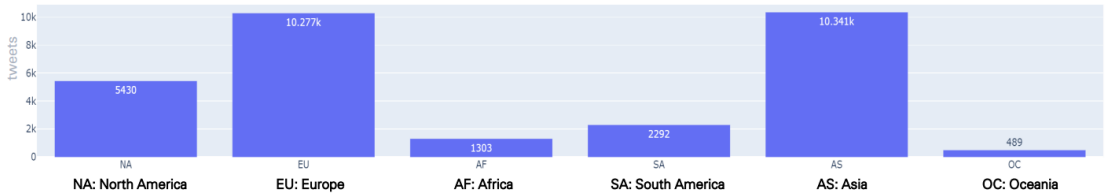


Figure 2.3: Step A: Trend analysis for CARBON NEUTRALITY from 15/Jul to 10/Nov based in continent

In the final step of our process, we are focusing on creating a visualization of the data that we have collected. This data contains a wealth of information, so we have the ability to create a range of different types of plots. One example that we could consider is a figure that illustrates the trend analysis of Carbon Neutrality by continent Fig. 2.3. This type of information is essential for marketing purposes. By looking at the trends in tweets about Carbon Neutrality by continent, a company may identify areas where their marketing efforts could be improved in order to increase awareness about the topic. For instance, if the number of tweets

about Carbon Neutrality is lower in certain countries, the company may decide to increase their marketing efforts in those countries in order to raise awareness and engagement on the topic.

2.2 Step B: Sentiment Analysis

Sentiment analysis is a technique used to determine the sentiment or emotion expressed in a piece of text. There are several approaches to sentiment analysis, including dictionary-based methods and machine learning-based methods. Dictionary-based methods involve comparing the words in a piece of text to a pre-defined list of words with associated sentiment scores (e.g. positive, negative, or neutral). Machine learning-based methods (the one that we use) involve training a model on a large dataset of labeled text data and then using the model to classify the sentiment of new text.

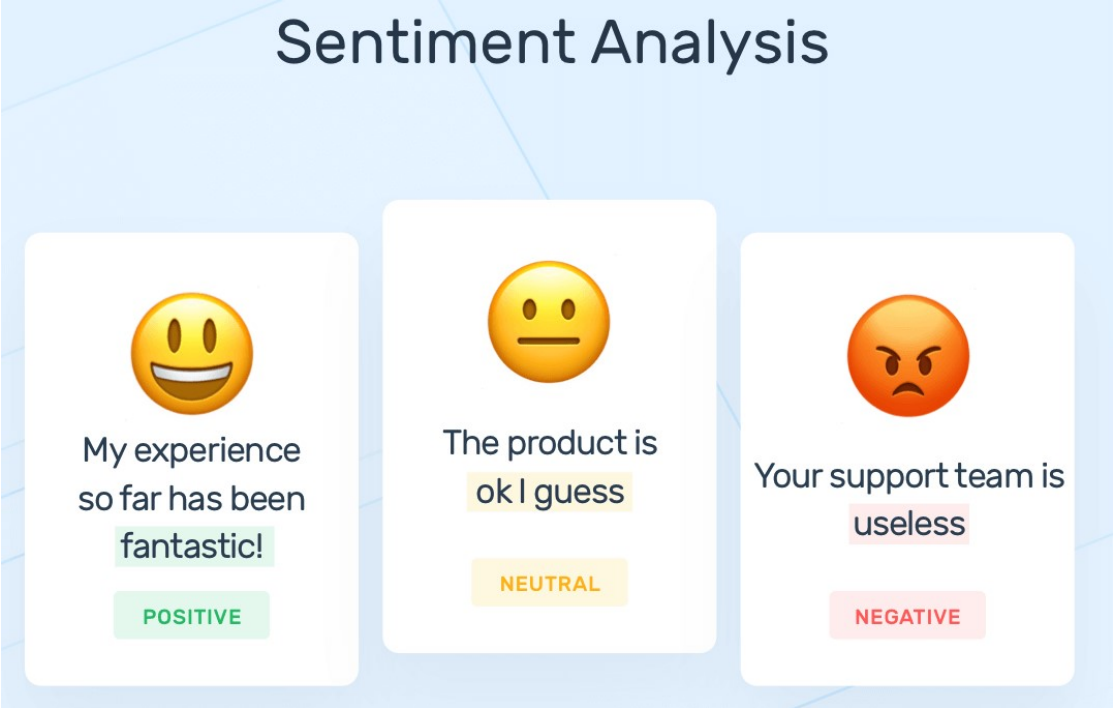
When applied to social media, it can help companies understand how people feel about their brand, products, or services. This can be useful for a variety of purposes, including:

1. Identifying trends and patterns in customer sentiment: By analyzing the sentiment of social media posts, companies can identify common themes and trends in how people feel about their brand. This can help them understand what people like and dislike about their products or services, and identify areas for improvement.

2. Improving customer satisfaction: By tracking and analyzing customer sentiment, companies can identify and address issues that may be causing dissatisfaction. This can help improve customer satisfaction and loyalty.

3. Identifying potential Public Relations (PR) crises: By monitoring social media sentiment for negative sentiment or issues that could damage the company's reputation. A PR crisis refers to a situation where an organization faces negative public scrutiny or criticism that could harm its reputation. Examples of PR crises include negative publicity, customer complaints, and product recalls. By using sentiment analysis to identify potential issues early on, companies can take proactive steps to address and mitigate the impact of a PR crisis before it becomes a bigger problem.

4. Improving marketing and branding efforts: By analyzing the sentiment of social media posts about their brand, companies can better understand how their target audience feels about them and use this information to improve their marketing and branding efforts. This can help them create messages and campaigns that are more likely to resonate with their audience and be more effective at reaching their marketing goals



Source: `Source:https://monkeylearn.com/sentiment-analysis/`

Figure 2.4: Step B: Sentiment Analysis examples

2.2.1 Training Dataset: Sentiment140

Since we are not using dictionary based methods, we need a training dataset to train a machine learning model. The model is trained on these examples by adjusting the values of its parameters based on the errors it makes when predicting their labels. This allows the model to learn the patterns and relationships that are indicative of different sentiments.

Once the model has been trained, it can be used to make predictions about the sentiment of unseen text. The quality of the predictions made by the model will depend on the quality of the training dataset and the effectiveness of the model at learning from it.

It is important to choose a training dataset that is similar to the serving dataset because the model's performance on the serving dataset will depend on how well it has learned to generalize from the training dataset. If the training and serving datasets are significantly different, the model may not perform well on the serving dataset because it has not seen enough examples of the types of inputs it will encounter during serving.

For example, if the training dataset is primarily composed of product reviews and the serving dataset is composed of tweets about politics, the model may not perform well on the serving dataset because it has not been trained on sufficient examples of political tweets. On the other hand, if the training and serving datasets are similar, the model will have a better chance of performing well because it has seen a range of examples that are similar to the inputs it will encounter during serving

The sentiment140[2] collection is comprised of 1,582,416 training tweets, 16,000 validating tweets, 1,600 testing tweets which have been fetched using the Twitter API in the 2009. The tweets are annotated as negative and positive (0 and 4 respectively). It includes six fields listed below:

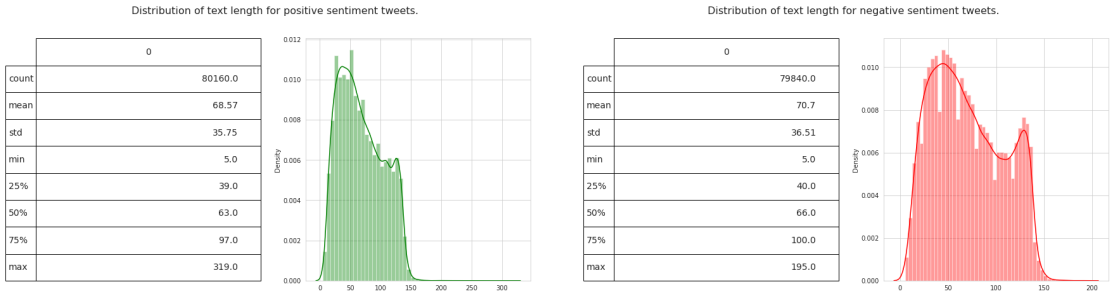
1. target: the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
2. *ids*: The id of the tweet (2087)
3. *date*: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
4. *flag*: The query (lyx). If there is no query, then this value is NO_QUERY.

[2]https://huggingface.co/datasets/sentiment140

5. *user*: the user that tweeted (robotickilldozr)

6. *text*: the text of the tweet (Lyx is cool)

The figure 2.5 shows that the lengths of positive and negative tweets are similar on average, which means that it is not necessary to "normalize" the sequences to prevent the model from making predictions based on sentence length. The figure also shows that more than 75% of tweets are less than 100 characters, which is important to keep in mind in case we want to fine-tune a transformer model which has been trained on text with specific maximum length.



	0
count	80160.0
mean	68.57
std	35.75
min	5.0
25%	39.0
50%	63.0
75%	97.0
max	319.0

	0
count	79840.0
mean	70.7
std	36.51
min	5.0
25%	40.0
50%	66.0
75%	100.0
max	195.0

Figure 2.5: Tweet lengths for positive and negative tweets in Sentiment140

Figure 2.6 shows the word clouds for positive and negative tweets. The word cloud for positive tweets includes words such as "love," "thank," and "quot," which may be indicative of positive sentiment. The word cloud for negative tweets includes words such as "now," "work," and "today," which may be indicative of negative sentiment. These word clouds can provide insights into the types of words and phrases that are commonly associated with positive and negative sentiment in tweets.

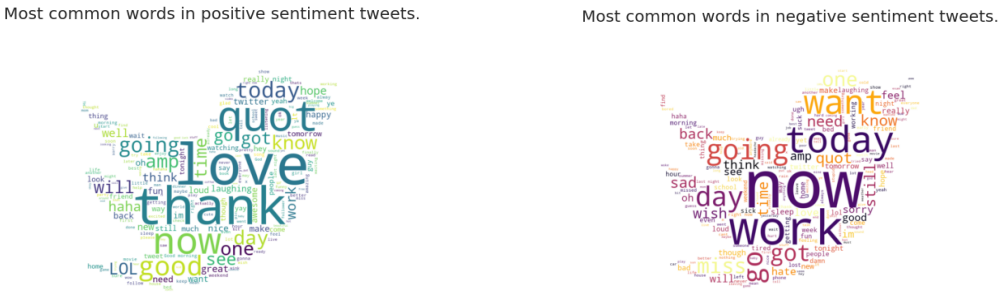


Figure 2.6: Word-Cloud for positive and negative tweets in Sentiment140

2.2.2 Model Architecture: BERTweet and Classification Head

The figure 2.7 shows the steps of the model designed to deploy Sentiment Analysis. The model consists of a transformer as the backbone and a simple head for classification. In step B1, the input text is pre-processed into tokens that can be understood by the transformer. In step B2, the transformer is used to extract features from the entire sentence. These features represent the knowledge that the transformer has learned during its training. The model uses only the feature associated with the special token "CLS", which represents the entire input sentence. The size of this feature depends on the transformer being used as backbone. The "CLS" feature is then fed into a simple two-layer neural network, which serves as the classification head. This neural network makes the final prediction about the sentiment of the input text.

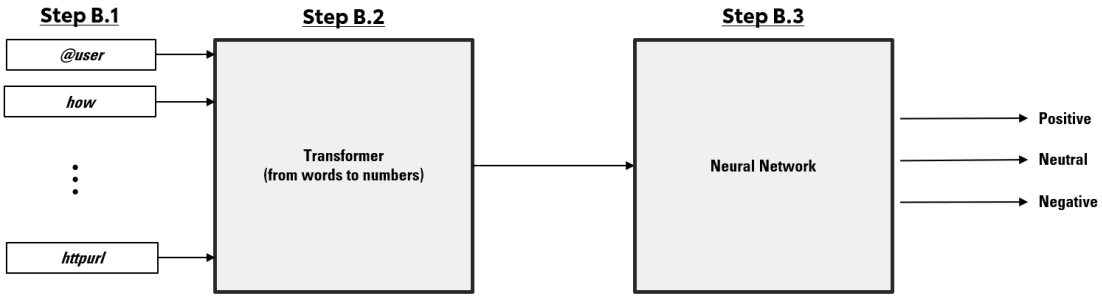


Figure 2.7: Sentiment analysis architecture

Transformers are a type of neural network architecture that have proven effective for natural language processing tasks, including sentiment analysis. There are several reasons why transformers may be a good choice for sentiment analysis. First, they use an attention mechanism that allows the model to focus on specific parts of the input when making predictions, which can be helpful for sentiment analysis by allowing the model to weigh the importance of different words or phrases in the input text. Second, many transformer models, such as BERT and GPT, have been pre-trained on large datasets and fine-tuned for specific tasks, which can make it easier to train a model for sentiment analysis because the model has already learned a lot about language structure and meaning. Third, transformers have been shown to outperform other types of neural networks on a variety of natural language processing tasks, including sentiment analysis, which may make them a good choice for tasks where high accuracy is desired. Fourth, transformers are more efficient than some other types of neural networks because they can process input in parallel rather than sequentially, which can make them faster to train and more efficient to use for inference.

Since the attention mechanism of a transformer model operates on the full input sentence, it is sensitive to the specific words and phrases present in the input text. As a result, it may not be a good idea to use a typical preprocessing pipeline that includes stop-word removal, lemmatization, or other forms of normalization, as these steps could alter the input text in a way that would affect the attention mechanism and potentially degrade the model's performance. Instead, it may be more effective to preserve the original form of the input text as much as possible, to ensure that the attention mechanism has access to all of the information it needs to make accurate predictions. For thess reason, the pre-processing phase (step B.1) it is composed of:

1. Replacing Emojis and Emoticons: Icons made from punctuation marks, letters, and numbers, typically depicting some sort of feeling or emotion. Using the emoji[3] library, we transform them into the textual equivalent of the emotion they intend to convey. However, a more comprehensive and efficient parsing required the addition of some missing entries to the original dictionary. These entries are taken from the Wikipedia page.[4]

2. Replacing slang and informal language: Tweets often contain slang, abbreviations, and other forms of informal language that may not be understood by a machine learning model. Replacing these terms with more standard language can help improve the model's performance. The conversion dictionary is compiled by hand from a variety of resources.

3. Removing hashtags, mentions, and URLs: These elements of a tweet may not be directly relevant to the sentiment being expressed, and they can often be removed or replaced with a generic placeholder without affecting the meaning of the tweet. we replace them into unique tokens, @USER and HTTPURL, respectively as part of "soft" normalization.

4. Tokenization: Finally, the tweets will need to be tokenized, which involves splitting the text into individual words or subwords and encoding them as numerical values that can be processed by the machine learning model. We tokenize the tweets using "TweetTokenizer" [1] from the NLTK toolkit.

After the pre-processing phase, step B.2 extract features from the entire tweet. Tweets are not like other types of written content, such as encyclopedia articles or news reports. This is because Tweets are often shorter and include more informal

[3]https://pypi.org/project/emoji/

[4]https://en.wikipedia.org/wiki/List_of_emoticons

syntax and terminology, such as abbreviations, typos, and hashtags. Since Tweets lack the structure and uniform vocabulary of typical text corpora, it may be challenging to apply generic transformer models to this data.

For this reason, we decide to use the first large-scale language model for English Tweets. BERTweet (BERT for Tweet) [2] has been trained on a 80GB dataset of 850M English Tweets. The model is pre-trained using the RoBERTa [3] method, and it employs the BERTbase model setup. According to experimental results, this model is superior to RoBERTabase and XLM-Rbase [4] on tasks—Part-of-speech (POS) tagging, Named-entity recognition (NER), and text classification. The advantage of BERTweet is that it has been pre-trained on a large dataset of tweets, which means that it has already learned a lot about the structure and meaning of Twitter language. This can make it easier to train a model for tasks like sentiment analysis, because the model has already learned a lot of the information it needs to make accurate predictions.

For this task, we do not use the transformer to extract features from each token, but to analyze the entire tweet. The special token "CLS" (short for "classification") is used to represent the entire input sequence. The representation for the "CLS" token is typically used as the final output of the model, after it has been processed by a classification head (step B.3). This representation is intended to capture the meaning and context of the entire input sequence, and it is used to make the final prediction about the label or category of the input text. The "CLS" token is an important part of the transformer architecture, and it plays a central role in tasks like sentiment analysis, where the goal is to predict the overall sentiment of a piece of text. In order to make accurate predictions, the model needs to be able to understand the context and meaning of the entire input text, and the "CLS" token is used to represent this information.

Final step B.3 is the classification head. It is responsible for generating the model's final output and making the prediction that is used to evaluate the model's performance. In this case, it consists of a simple neural network with no hidden layers (only input and output layer).

To train the model we use the categorical cross entropy loss which is a common loss function used in classification tasks, where the goal is to predict a categorical label or class for an input sample. It is used to measure the difference or "error" between the predicted probability distribution over the classes and the true probability distribution.

$$CategoricalCrossEntropyLoss = -\sum_{i} y_i \cdot log(\hat{y}_i) \tag{2.1}$$

where y_i is the target value of the i^{th} and $\hat{y}_i$ the respective model prediction. This loss measure how different two probability distributions are. In this context, y_i is the probability that event i occurs and the sum of all y_i is 1, meaning that exactly one event may occur. A lower loss value indicates that the model's predictions are more accurate, while a higher loss value indicates that the model is making more errors. The model is typically trained to minimize the categorical cross entropy loss by adjusting the values of its parameters to reduce the error between the predicted and true probability distributions.

2.2.3 Training Phase: Results on Sentiment140

The training phase is the process of fitting a machine learning model to a set of training data. During training, the model is presented with a set of input data and the corresponding desired output, and the model's parameters are adjusted to minimize the difference between the predicted output and the desired output.

The goal of the training phase is to find a set of model parameters that result in good performance on the training data. This is typically done by minimizing a loss function (Categorical Cross Entropy loss in our case), which measures the difference between the predicted output and the desired output. The model's performance is then evaluated on a separate test set to ensure that it generalizes well to unseen data.

The training phase is an iterative process, and the model's parameters are typically updated multiple times based on the training data. The process of adjusting the model's parameters based on the training data is known as training the model or fitting the model. The trained model can then be used to make predictions on new, unseen data.

However, since we are using BERTweet as backbone, we are fine-tunng the model. Fine tuning is a process of adapting a pre-trained machine learning model to a new task by updating the weights of the model based on additional training data. It involves retraining the model on the new data, adjusting the hyperparameters, and possibly adding or removing layers from the model.

The dataset used for generating the results is Sentiment140 and it contains more than 1,5 millions of tweets. More details about the dataset in section 2.2.1. The dataset is splitted in 1,582,416 tweets for training, 16,000 tweets for validating and 1,600 tweets for testing.

several hyperparameters are available for fine tuning the model, below are reported the hyperparameters with the best performances:

```
from transformers import TrainingArguments

training_args = TrainingArguments(
    num_train_epochs=1,
    learning_rate=2e-5,
    weight_decay=0.01, # small regularization
    per_device_train_batch_size=32,
    per_device_eval_batch_size=32,
    evaluation_strategy="steps", # rather than epoch
```

```
        logging_steps=50, # each 50 batch validation
        fp16=True, # work with float precision 16bit r.t. 32 bit
        metric_for_best_model="f1",
        gradient_accumulation_steps=3 # batch = 128
    )
```

The learning rate is set at a low number since we don't want to alter entirely the transformers' properties. The performance improvements were greatly aided by *fp16* and *gradient accumulation steps*. The first modification reduced the 32-bit floating point precision to 16-bit. This roughly cuts the training time for the model in half. Even though there is a potential loss of performance, our results for both precision are comparable. The *gradient accumulation steps* method specifies how many forward steps must be taken before calling the backpropagation, i.e. the batch size may be increased without using extra GPU RAM. To prevent overfitting, an early stopping callback is added with a patience of 3. This means that the training will be stopped after 3 validation cycles without an improvement in performance. This helps to prevent the model from fitting too closely to the training data and potentially performing poorly on new, unseen data.

The metrics used to assess the performances of the model are *accuracy* and $f_1 score$

$$f_1 score = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2} \cdot (FP + FN)} \tag{2.2}$$

$$accuracy = \frac{TP + FP}{TP + FP + TN + FN} \tag{2.3}$$

Accuracy 2.3 is a common metric used to evaluate the performance of a machine learning model. It is defined as the number of correct predictions made by the model divided by the total number of predictions.

It is also easy to understand because it is a simple ratio. It is a single number that can be easily compared to other models or to a baseline. For example, if a model has an accuracy of 0.8, it means that it made 80% correct predictions. This can be easily understood and interpreted by people with a wide range of backgrounds, making it a widely used metric in machine learning.

However, it can be misleading when the classes are imbalanced (i.e. there are significantly more examples of one class than the others). For example, if a model is trained to classify email as spam or not spam, and 99% of the emails in the training data are not spam, the model could achieve a high accuracy simply by always predicting not spam. In addition, accuracy does not take into account the relative importance of different types of errors. For example, in a medical diagnosis task, it

may be more important to avoid false negatives (i.e. incorrectly predicting that a patient is healthy when they are actually sick) than false positives (i.e. incorrectly predicting that a patient is sick when they are actually healthy). In this case, using a metric like precision or recall, which specifically focus on false negatives or false positives, may be more appropriate.

To overcome the problem of the Accuracy, we also use the $f1-score$ 2.2 because it is the harmonic mean of precision and recall, and is used to balance these two metrics. It penalizes models that have poor precision or recall and rewards models that have high precision and recall.

Transformers	train loss	validation loss	accuracy	$f_1 score$	step
BerTweet-large	**0.3031**	**0.2855**	**0.8834**	**0.8834**	1200
BerTweet-base	0.3107	0.2994	0.8762	0.8762	1000
Roberta-Large	0.3830	0.352	0.8560	0.8570	1100
Roberta-base	0.3900	0.3650	0.8460	0.8550	**900**

Table 2.1: Fine tuning results on Sentiment140, average of 3 experiments per transformers. Training 1,582,416; Validation 16,000; Test 1,600

In table 2.1 are reported the results for the fine-tuning phase. Accuracy and $f_1 score$ are calculated over the test set.

The main differences among the transformers are the number of parameters (large vs base) and whether it was trained on a twitter corpora or on a plain English corpora (BerTweet vs Roberta). The columns *step* represents the number of steps required by the model to achieve the best results and to not overfit the data.

The results clearly reveal that transformers trained on twitter corpora outperform the others in all criteria. However, we cannot say the same about the transformer size because there is only a minor variation between large and base models.

For the sake of brevity, only the best model's results are shown in Fig.2.8. The model was validated each 50 steps. The figure shows that the model was able to generalize well, as indicated by the nearly overlapping loss functions and high accuracy and $f_1 score$. It is worth noting that the training stopped after using only 4% of the dataset, which suggests that the dataset may not be diverse enough. This can lead to the model learning patterns and relationships that are specific to the training data and may not perform well on new data.

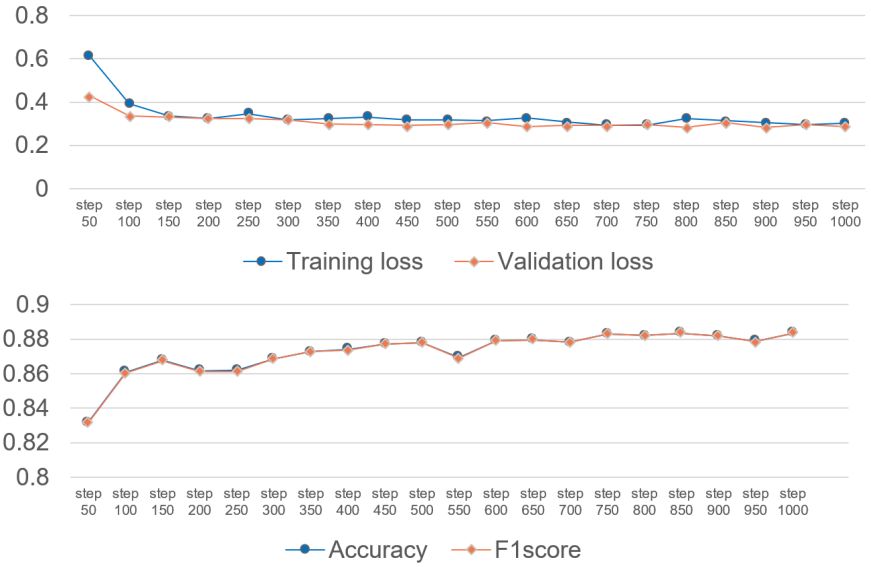


Figure 2.8: BerTweet-Large fine-Tuning results for best run.

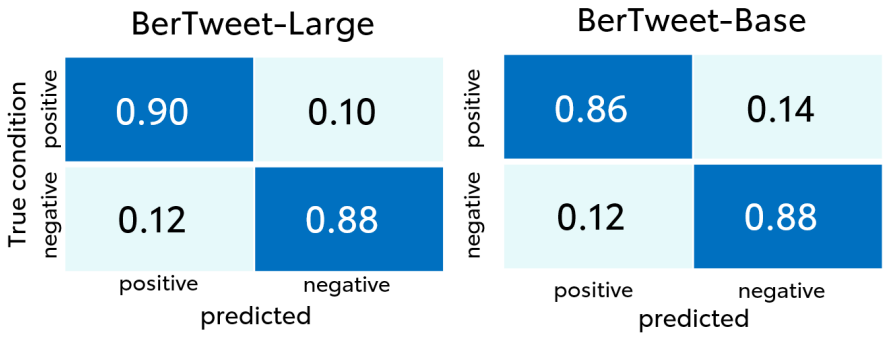


Figure 2.9: Confusion Matrix comparison between BerTweet-Large and BerTweet-base

It's also interesting to look more closely at the error of the two top models. The main difference is that BerTweet-Base has a larger value for False Negative (predicted Negative but positive). However, when compared to the huge model, it is faster during inference phase (more or less half of the time required). Depending on the model's application, one can pick between a faster model with some inaccuracies for the tweet classified as negative and a more exact but slower model.

2.2.4 Human Level Performance: Data and Conceptual Shift

Before deploying the model to production, we decide to conduct a thorough analysis to ensure that the model's performance is comparable to that of a human. One way to assess this is by comparing the model's error rate to what is known as Human Level Performance (HLP). HLP is calculated by averaging the error rates of different individuals who manually label the dataset. The ultimate goal is to create a proxy for Bayesian error, which is also known as irreducible error. This is the lowest potential prediction error that can be achieved, even if we have a complete understanding of the process that generated the data. This is because errors will still occur if the process includes random elements. Therefore, in practice, we strive to minimize the gap between the performance of our model and the HLP. The procedure for doing so is outlined in the figure 2.10.

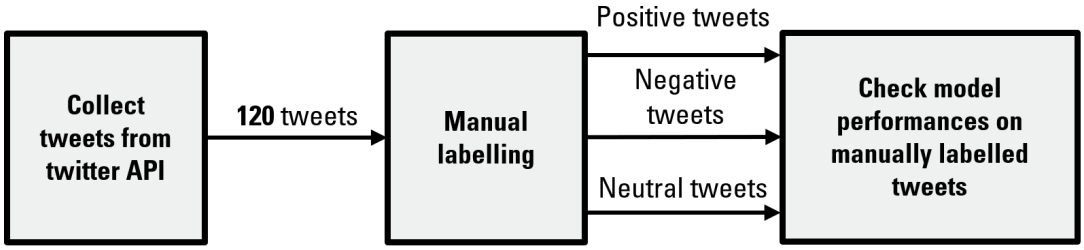


Figure 2.10: Human Level Performance analysis step-by-step procedure

We manually label 120 tweets collected from the twitter API platform according to two simple but clear rules:

1. Each tweet has to be labelled thinking about the writer's sentiment and not about the content

2. If a tweet only states fact, it is neutral

It is important to have clear rules for manually labeling data because these rules help ensure that the labeling process is consistent and accurate. Without clear rules, the labeling process may be prone to errors or biases, which can negatively impact the quality and reliability of the labeled dataset.

Since our model does not consider neutral tweet, we introduce the second rule to understand what is the impact of this relaxation. The first rule is more subtle but crucial. Having in mind that the model is only an approximation function, this rule specifies which function we want to focus on. Indeed, we want to approximate

the function which associates the tweet to the sentiment of the writer and not the tweet with a positive/negative impact. For instance considering the tweet:

*"in Turin the stop continues from 8am to 7pm for euro 1 petrol vehicles, I do not give a *** about the pollution, I just wanna go to work!"*

This tweet may be either positive, if you believe that Turin is correctly addressing the pollution in the city or negative if you focus on the writer sentiment. We decide to be strict with the second option.

Name	transformer	train dataset	fine-tuned dataset
twitter-roberta-base-sentiment	RoBERTa-base	58M tweets 05/2018 to 08/2019	SemEval 2017
twitter-roberta-base-sentiment-latest	RoBERTa-base	124M tweets 01/2018 to 12/2021	SemEval 2017
twitter-xlm-roberta-base-sentiment	RoBERTa-base with adapters train. Multi lang	198M tweets 05/2018 to 03/2020	60M tweets 8 lang.
Bertweet-Large	RoBERTa-large	850M tweets 01/2012 to 08/2019	Sentiment140
Bertweet-base	RoBERTa-base	850M tweets 01/2012 to 08/2019	Sentiment140

Table 2.2: Comparison among the different models used for HLP analysis.

We decide to compare our fully fine-tuned models (BerTweet) with the three most downloaded sentiment-analysis models for tweets on Hugging Face[5]. The differences among the models are reported in Table 2.2. Almost all the models share the same architecture RoBERTA [3]. The only difference is the *twitter-xlm-roberta-base-sentiment* model which is based on RoBERTa-base but trained on a cross-lingual classification (XLM-R)[5]. In addition, the model is not full fine-tuned but it exploits the adapter technique [6] and the adapter configuration is the same proposed in Pfeiffer et al. [7]. The datasets used for fine-tuning the models contain more recent tweets (up to 2017) than Sentiment140 (2009) as well as Neutral tweets.

In Fig. are reported the count of miss-classified tweets over the manually labelled dataset. Since, our model classify all the tweets as positive, the miss-classified

[5]https://huggingface.co/

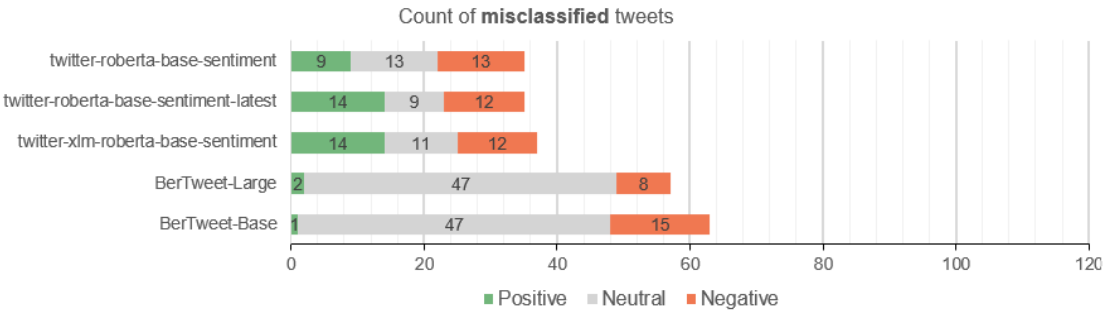


Figure 2.11: Count of miss-classified tweets over the HLP dataset.

tweets are either neutral or negative. Even if this results seems destructive, we need to remember that our models does not expect to classify neutral tweets. However, we cannot overlook the negative errors. Interesting, it is to analyze the errors reported by the other models. We can notice how changing the transformer's training dataset (refer to Table for more details), can improve the performances. The most miss-classified sentiment is the Positive one. In Fig.2.12 we can see that the Neutral tweets have an huge impact over the results since the positive and the negative tweets are mostly miss-classified as neutral.

The number of miss-classified tweets over the manually labeled dataset is shown in figure 2.11. Because our approach classifies almost all tweets as positive, the incorrectly classified tweets are mostly either neutral or negative. Even if this finding seems harmful, we have to remember that our methods are not designed to classify neutral tweets. We cannot, however, ignore the negative errors.

Analyzing the errors that the other models have reported, we can see how modifying the transformer's training dataset (see table 2.2 for additional information) might increase performance. Positive sentiment is the most commonly miss-classified but it is also the most present class (50% of the tweets are positive). In Fig.2.12, we can observe that Neutral tweets have a significant impact on the outcomes since both positive and negative tweets are frequently miss-classified as neutral.

In figure 2.13 is reported the $f_1 score$ for each model. We define a safely threshold over which we can safely use the model in production. the Threshold is set to 0.8. The gap between our model (BerTweet) and the threshold is almost 0.4%. The next design choices will be dedicated in reducing this gap. The $f_1 score$ has been selected among the other metrics since the HLP dataset is imbalanced. (60 positive, 29 neutral, 31 Negative)

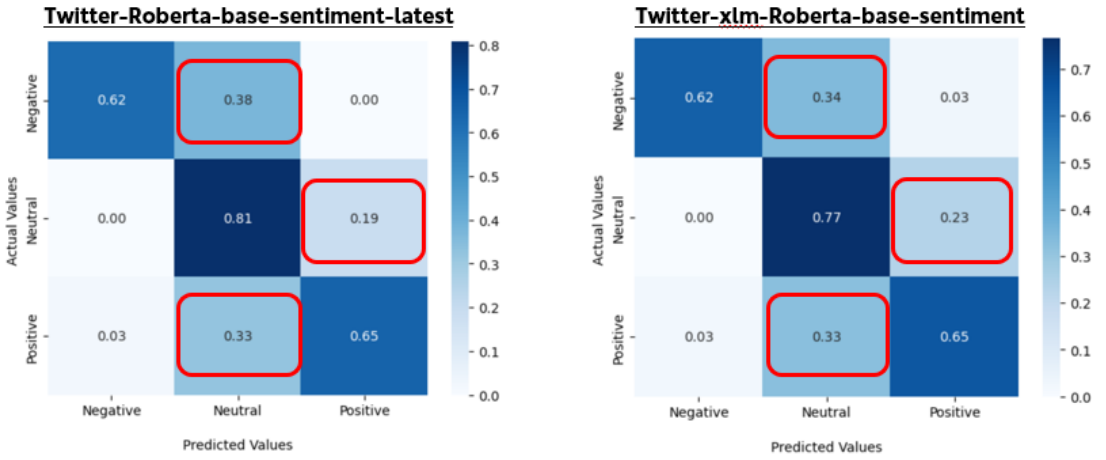


Figure 2.12: Confusion Matrices of best models for HLP. In red is reported a common error path among the models.

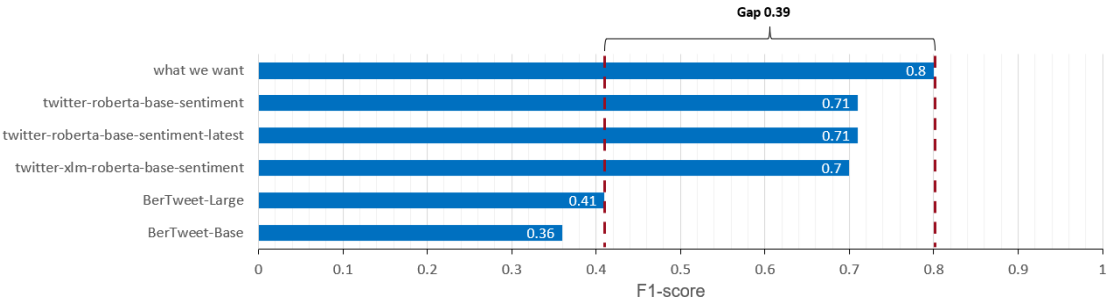


Figure 2.13: $f_1 score$ for each model on HLP dataset

The root cause analysis is performed to determine why our models are performing so poorly and what are the possible countermeasures (Fig.2.14). The function that the model has learnt to label the tweet is not representative. This can be caused by two problems: the model is not able to capture the relation; the distribution of the dataset used for fine-tuning the model is different from the inference dataset.

The first possibility is discarded, since our model achieve almost 90% of $f_1 score$ over the Sentiment140 test dataset.

The second possibility is decomposed in several reasons: the dataset is too old (Data shift); the dataset does not contain neutral tweets; the label procedure is different (Conceptual Shift). The first two options may improve performance but, other models, that have integrated them, are still unable to successfully close the gap.

The final root cause is the most promising. The labeling technique for HLP differs from that of sentiment140. The most effective countermeasure is to build our training dataset using tweets obtained via the Twitter-API.

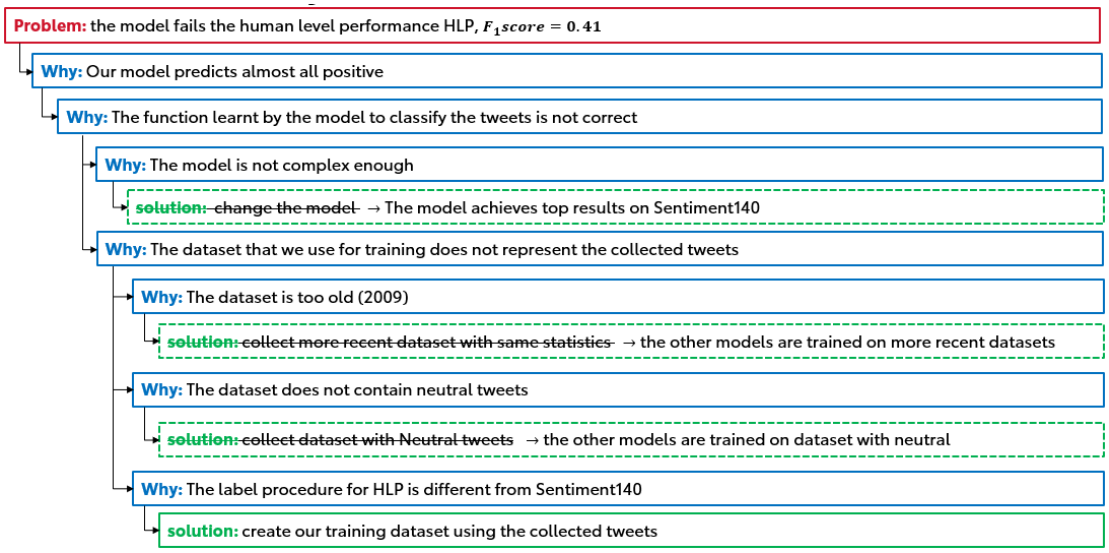


Figure 2.14: Root Cause Analysis for BerTweet HLP results

In figure 2.15 is reported the process for generating the training dataset. First step is to apply the same pre-process technique explained in 2.2.2. Then we use an ensemble composed of the state-of-the-art sentiment analysis models where the final output is the most frequent predicted class. Since this models are not perfect the last step is human based i.e. a labeller checks whether the label is correct or not (step 4). The last step is performed only in case the state-of-the-art models have different output. The rules used to check the labels are the same expressed for generating the HLP dataset.

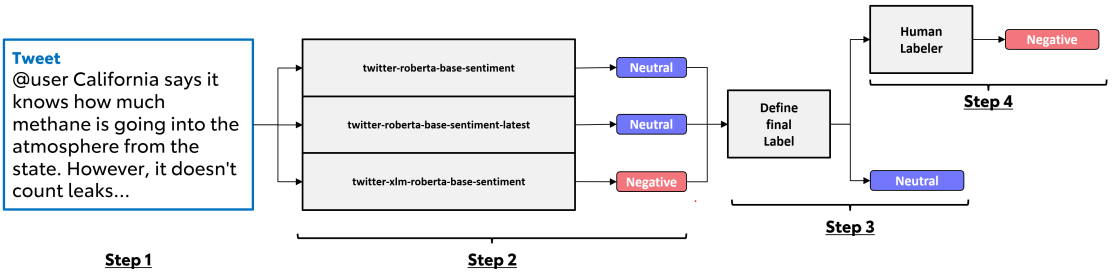


Figure 2.15: Process to create the training dataset to address data and conceptual shifting

2.2.5 Countermeasure Analysis and Results

According to the root cause analysis, the labeling approach for HLP varies from that of sentiment140. The effect of the countermeasure on the gap is examined in this section.

The production of training dataset takes time because the labeling process is not totally automated. As a result, we gradually increase the amount of tweets in the training dataset to find the point where time complexity and performance meet. In Fig. 2.16 are shown the result for the validation[6] loss and $f_1 score$ calculated over the HLP dataset and we successfully close the gap with 1000 tweets. We stop creating training datasets since a substantial increase in $f_1 score$ can be seen between 100 and 500 tweets but not between 500 and 1000 tweets. As a result, we expect a plateau for $f_1 score$ expanding the dataset. Given that the datasets are more expensive than useful, we safely stop at 1000 tweets.

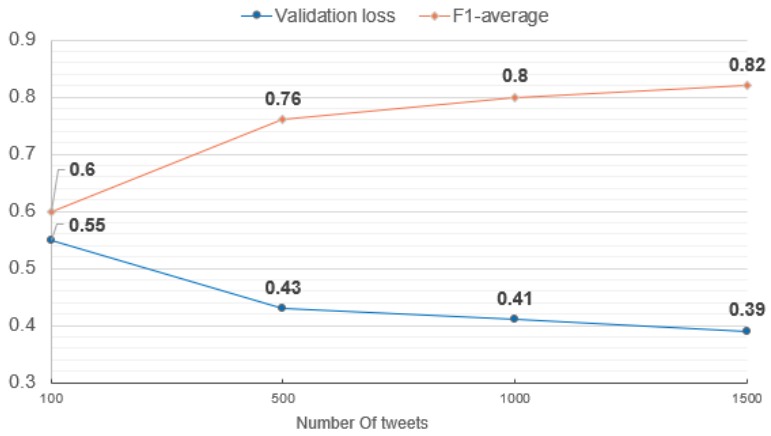


Figure 2.16: Validation loss and $f_1 score$ evaluated for different size of training dataset. validation size 10%. $F_1 score$ is calculated on the HLP dataset

In Fig. 2.17 are reported the training and validation loss for the dataset with 1000 tweets. We can clearly notice that the model overfits the data after step 120. However, the best validation loss is achieved at step 160, At this step, the model has performed 12 epochs, i.e. each tweet has been used for training 12 times.

[6]size validation dataset10% i.e. 1000 tweets, 900 train, 100 validation

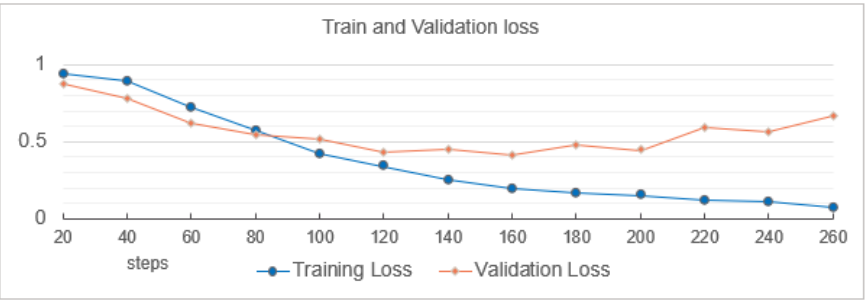


Figure 2.17: Training and Validation loss for 1000 tweets, 10% validation size. At each iteration (step), the model receive as input 64 tweets (batch size)

In Fig. 2.18 are reported the confusion matrices calculated over the HLP datasets. On the left the results of the countermeasure model. On the right the best state-of-art model. The countermeasure significantly reduces the percentage of mistakes in all three patterns. The neutral being labeled as negative has the greatest decline, a decrease of 0.22%.

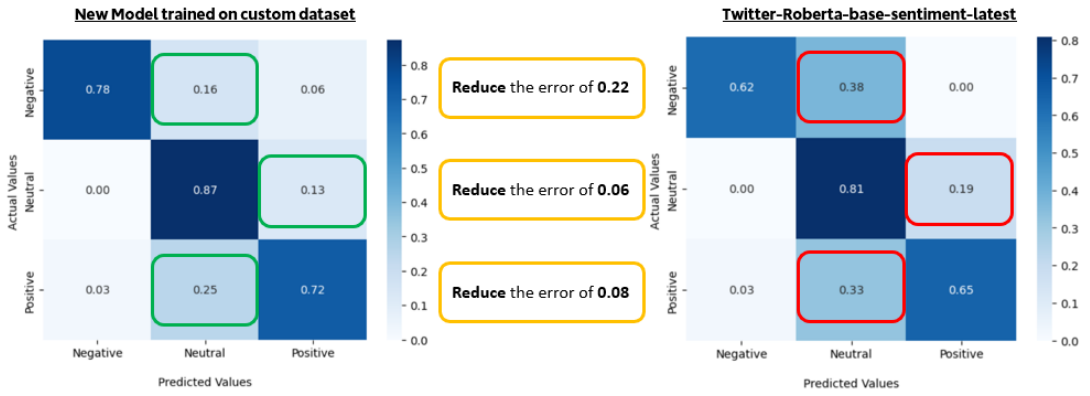


Figure 2.18: Differences in confusion matrices between countermeasure and best state-of-art sentiment analysis model

We successfully close the gap and for this reason the model is deployed in production. In Fig. 2.19 are reported three random tweet with the predicted label. Given the analysis so far and after a manual inspection of the predicted label, Can we trust the results? The Negative label with high confident instead the neutral and positive labels with medium confident. The error that our model can have are between positive miss-classified as neutral (and vice-versa) but almost never miss-classified with negative. This is also confirmed in Fig 2.19 where the negative tweet has a complete different structure from the others.

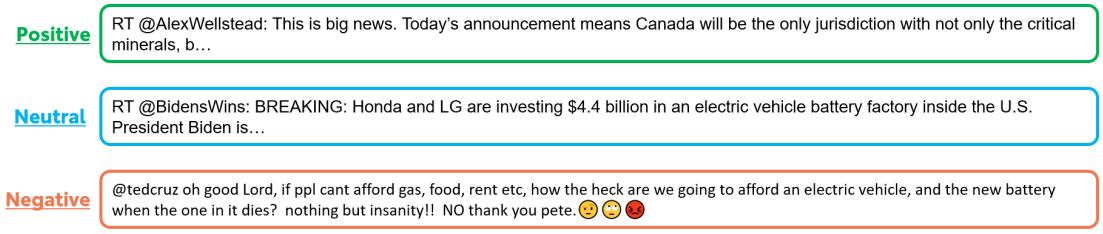
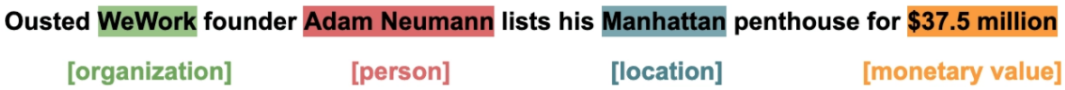


Figure 2.19: Three random sample prediction for each label

2.3 Step D: Named Entity Recognition

Named entity recognition (NER) is a subfield of natural language processing (NLP) that involves identifying and classifying named entities in text into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc. In Fig.2.20 is present an example of NER application over unstructured data.

There are many different approaches to performing NER, including rule-based systems, dictionary-based systems, and machine learning-based systems. Machine learning-based approaches, such as conditional random fields and transformers, have achieved state-of-the-art results on various NER benchmarks.



Source: `https://monkeylearn.com/blog/named-entity-recognition/`

Figure 2.20: Visual Example of possible application of NER on unstructured data

By identifying and extracting named entities from social media posts, an automotive company can gain insights into how its brand is perceived, what products are being discussed, and who is talking about them. Here are some examples of how an automotive company might use NER in social media:

1. Brand monitoring: An automotive company can use NER to identify mentions of its brand in social media posts, including hashtags and mentions of the company's name. This can help the company track the overall sentiment of these posts and identify any potential issues that need to be addressed.

2. Product tracking: NER can be used to identify mentions of specific products in social media posts, allowing the company to track the popularity of its products and identify any potential problems or issues with them.

3. Influencer identification: By identifying named entities that correspond to people in social media posts, an automotive company can identify influencers and key opinion leaders who are talking about its products. This can be useful for identifying potential partnerships or collaborations.

The NER is usually the first and most important step for Information Extraction Pipelines. However, NER can be challenging on Twitter due to the nature of the

platform. Twitter has a 280-character limit for posts, which means that tweets are typically shorter and more concise than other types of text, such as news articles or blog posts. This can make it difficult for NER systems to extract named entities, as they may not have enough context to accurately identify and classify them.

Additionally, Twitter has its own language and conventions, such as hashtags, emojis, and shortened words, which can make it difficult for NER systems to accurately process and understand the text.

Another difficulty is that there are many new named entities and unusual surface shapes within the user-generated content. For instance, "black mamba," the name of a poisonous snake, is a morph coined by Kobe Bryant for his ferocity in basketball games. Such morphs and tokens are extremely difficult to identify and categorize.

2.3.1 Training Dataset: Tweetbank and Tweetner7

For this task, we focus on two different datasets: Tweetner7 [8] and Tweetbank [9]. Both datasets contains only tweet collected in the last decade.

Tweetbank contains 3,550 labeled anonymous English tweets annotated in Universal Dependencies. The guidelines used to label the dataset are the CoNLL 2003 guidelines[7]. The targets present are the standard four-class CoNLL 2003 both to help annotator and to have more instances per class to improve the learning of NER models. The named entities are:

- **PER**: persons (e.g., Joe Biden, joe biden, Ben, 50 Cent, Jesus)
- **ORG**: organizations (e.g., Stanford University, stanford, IBM, Black Lives Matter, WHO, Boston Red Sox, Science Magazine, NYT)
- **LOC**: locations (e.g., United States, usa, China, Boston, Bay Area, CA, MT Washington)
- **MISC**: named entities which do not belong to the previous three. (e.g., Chinese, chinese, World Cup 2002, Democrat, Just Do It, Top 10, Titanic, The Shining, All You Need Is Love)

In the paper [9] it is also reported the inter-annotator agreement (IAA) measure between three annotators. It is useful to set a maximum f1-score achievable from the model. The IAA is the token-level pairwise F1 score calculated without the O label.

Label	Quantity	F1
PER	777	84.6
LOC	317	74.4
ORG	541	71.9
MISC	519	50.9
Overall	2,154	70.7

Dataset	Train	Dev	Test
Tweets	1,639	710	1,201
Tokens	24,753	11,742	19112
tokens per tweet	15.1	16.6	15.9
BerTweet-baseline $f_1 score$ 73.71			

Table 2.3: Tweetbank statistic. Left table for IAA measure. Rigth table split statistics.

[7] https://www.clips.uantwerpen.be/conll2003/ner/

In Table 2.3 is clear that PER, LOC, and ORG have higher F1 agreement than MISC. Therefore, we expect MISC to have worse results than the other classes. On the right it is also displayed the split performed over Tweetbank. In the table, the BerTweet baseline is reported from the Tweetbank paper [2][8].

Tweetner7 [8] has been developed to analyzed the short-term degradation of NER models over time. Indeed, The collected tweets (from September 2019 to August 2021) are filtered to get weekly trending topics. The dataset contains annotations labeled by three annotator where the annotation with an agreement of 1/3 were discarded. The annotation with 2/3 manually were validate (roughly half of the instances). The named entities are:

- **PERSON**: persons (e.g., Joe Biden, joe biden, Ben, 50 Cent, Jesus)
- **LOCATION**: locations (e.g., United States)
- **GROUP**: names of groups (e.g. Nirvana, San Diego, Padres)
- **EVENT**: names of events (e.g. Christmas, Super Bowl)
- **PRODUCT**: names of products (e.g. Iphone).
- **CREATIVE WORK**: names of creative works (e.g. Bohemian Rapsody. The work should be created by a human
- **CORPORATION**: names of corporation (e.g. Google)

In Table 2.4 are reported the training, validation and test datasets statistics. The tag that it is most present in the dataset is "person" with 7028 tokens. Instead, the other tags contain roughly half of the tokens.

The baseline results reported in the paper for Tweetner7 [8] are shown in the table 2.5. The best performing model is the one composed of Bertweet-Large with 66.7 and 62.2 for micro and macro $f_1 score$ respectively.

[8]It is the second highest reported but the first is trained on another dataset

Label	Train all 2020 + 2021	Valid 2021	Test 2021
person	**7,028**	**283**	**2,712**
group	3,555	227	1,516
event	3,210	131	1,097
product	2,776	111	972
corporation	2,602	102	900
creative work	2,351	74	731
location	1,956	72	716
all	24,484	1,000	8,644
Number of Tweets	7,111	310	2,807

Table 2.4: Tweetner7 target count for train test and validation set

Model	Micro F1	Macro F1
BerTweet-large	**66.7**	**62.2**
BerTweet-base	65.5	61
RoBERTa-large	66	61.8
RoBERTa-base	65.2	61
Bert-large	63	59
Bert-base	62.2	57.6

Table 2.5: Tweetner7 baselines reported in literature

2.3.2 Model Architecture: Word Embedding, Bi-LSTM, Attention and CRF

The NER is a classic sequence labelling problem where you get as input a sequence of tokens $\mathbf{w} = (w_1, w_2, \ldots, w_k)$ and the output requested is a sequence of tokens labels $\mathbf{y} = y_1, y_2, \ldots, y_k$. Figure 2.21 shows the overall structure of the implemented model.

In Step D.1, the input text is preprocessed and tokenized in order to be processed by the model. In Step 2, multiple embeddings are obtained for each token. These embeddings include character-level sub-word information, pretrained transformer word embeddings (BerTweet [2])), and Part of Speech (POS) embeddings. Each embedding layer converts the token into a dense feature vector, where the vector represents the projection of the word into a continuous vector space [10]. These feature vectors are used to represent the words in a more comprehensive manner.

In Step 3, the final dense vector for each token is obtained by concatenating the character, word, and POS embeddings. Specifically, given character embedding $e_i^c \in \mathbb{R}^{d_c}$, word embedding $e_i^w \in \mathbb{R}^{d_w}$, and POS embedding $e_i^p \in \mathbb{R}^{d_p}$, the final vector is formulated as $X_i = e_i^p \bigoplus e_i^w \bigoplus e_i^c$, where $i \in 1{,}2, \ldots, k$ and $\bigoplus$ represents the concatenation operation.

In Step 4, the information extracted from each token is analyzed using a bidirectional LSTM (Long Short-Term Memory). This type of network consists of a forward LSTM that extracts past information and a backward LSTM that captures future information in the sequence. The final output for each token is the concatenation of the hidden states, i.e. $y_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$. This structure allows the hidden states to capture both historical and future context information. Another option that was considered was to use a stacked Bi-LSTM [11].

In Step 5, a Multi Head Attention layer is added on top of the Bi-LSTM. This layer focuses on relevant hidden states to extract as much information as possible from the text. In Step 6, this extracted information is pre-processed by a neural network that performs dimensionality reduction. Finally, a Conditional Random Field (CRF) is used to label each token based on the correlation between the current label and the neighboring labels, effectively imposing a structure on the predictions. This process helps to ensure that the labels are coherent and meaningful

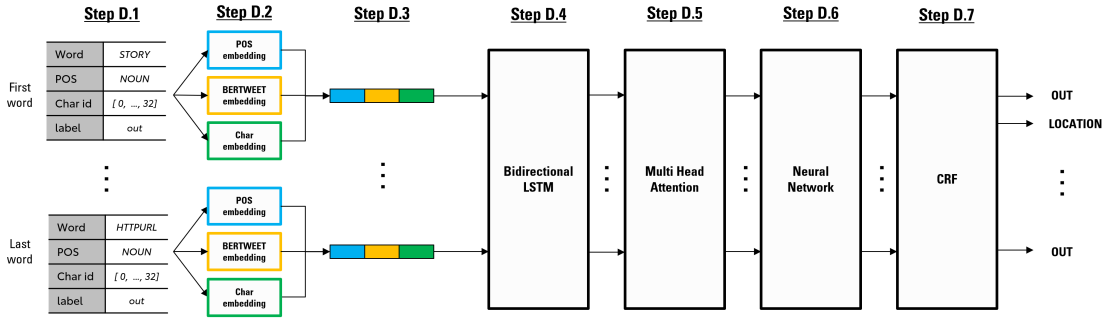


Figure 2.21: NER architecture step-by-step process

In summary:

- **Step 1 - Prepare the data**: Process the sentence and then tokenize it.
- **Step 2 - Extract Embeddings**: Extract several information from the token.
- **Step 3 - Concatenate**: Concatenate the 3 embeddings to get one dense vector for each token.
- **Step 4 - Bi-LSTM**: Extract the historical and future relations between tokens.
- **Step 5 - Multi head attention**: Focus attention on the most important relationships.
- **Step 6 - Neural Network**: Process all the information encoded so far
- **Step 7 - Conditional Random Field**: label each token considering also the immediate neighbors

2.3.3 Step D.1: Pre-Processing of the data

This step is crucial to generate the correct input to the model. Since the model exploits different embeddings, we need to design the pre-processing phase to contain all the input without loss of information. Fig. 2.22 is the step-by-step procedure to transform raw text as input to be processed by the model.

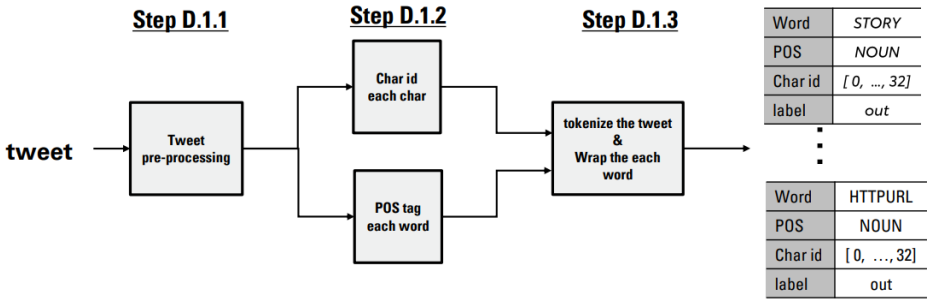


Figure 2.22: Step D.1: Pre-processing pipeline to prepare model input

The first step (step D.1.1) is the same of Sentiment Analysis, deeply explained in Section 2.2.2.

The second step can be run in parallel and it is used to extract unique Id useful to define the embeddings i.e. simply lookup-table. For the char id, we exploit the UTF-8 encoding from 0 to 255. We manually add three more ids to define the start, the ending of the word and the padding: 257, 258, 256 respectively. For the POS ids we use the hugging face model[9] based on the paper [12]. Also in this case, we add two more ids to the already present tags[10]: 17 and 18 for the start and end of the sentence.

[9]https://huggingface.co/TweebankNLP/bertweet-tb2_ewt-pos-tagging

[10]TweebankNLP/bertweet-tb2_ewt-pos-tagging/blob/main/config.json

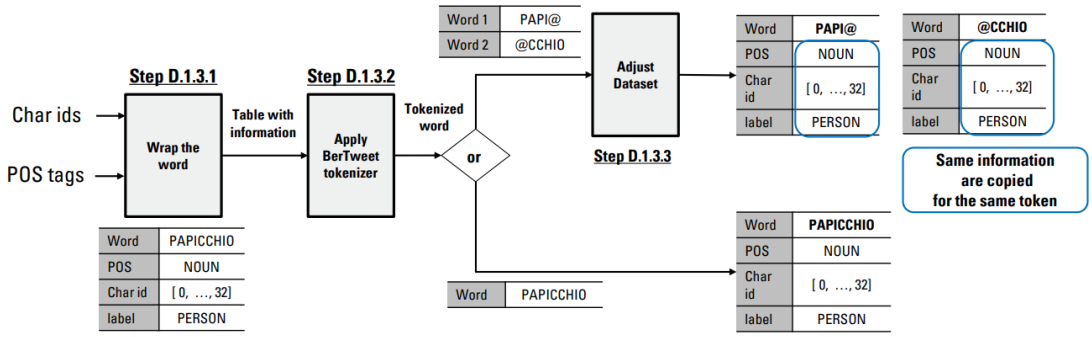


Figure 2.23: Step D.1.3: software design pipeline to address tokenization

To obtain the best results when using BerTweet to extract word embeddings, it is necessary to use the same tokenizer for both training and prediction. This is because the tokenizer will split rare words into smaller sub-words, which helps to better capture their meaning. For instance, the word "annoyingly" might be decomposed into "annoying" and "ly". Sub-word tokenization allows the model to have a reasonable vocabulary size while being able to learn meaningful context-independent representations.

Using a tokenizer that decomposes rare words into smaller sub-words can cause problems when working with other types of information in the dataset. To address this issue, a software pipeline (as shown in Fig. 2.23) can be used to first wrap the word with all the desired information, and then copy this information onto the split words. This helps to ensure that all relevant information is preserved and properly associated with the decomposed words.

2.3.4 Step D.2. & Step D.3: Extract and concatenate embeddings

After the preparation of the input, the following step is dedicated to the extraction of the feature. From each word the model extracts three different embeddings:

1. **POS embedding**: Feed the model the grammatical structure. The parameters learn which structure contains the specific target.

2. **BerTweet embedding**: Extract context word representation. The parameters are not trained.

3. **Char embedding**: Detect knowledge from pair, triple, etc. . . of characters. The parameters learn how to deal with misspelled words.

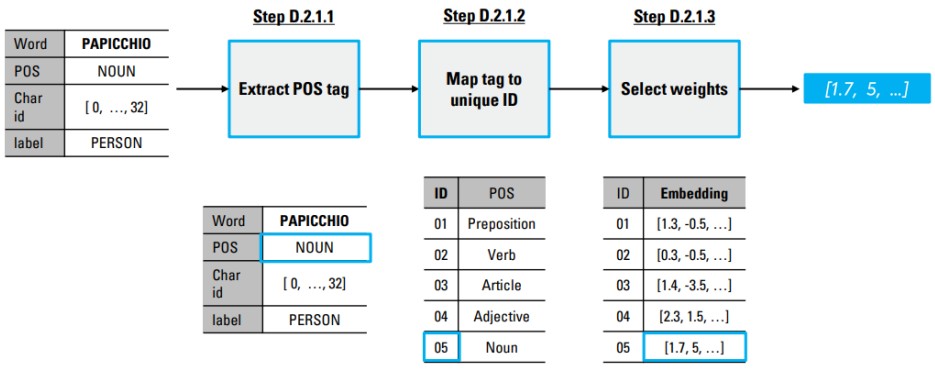


Figure 2.24: Step D.2.1: POS embedding feature extraction

The process of extracting features from the Part of Speech (POS) tag is shown in Fig. 2.24. After extracting the POS tag from the input, a lookup table is used to obtain the POS embedding vectors based on their unique ID. These vectors are randomly initialized and contain the model's knowledge of each POS tag after training. The goal of using POS embeddings is to provide the model with information about the grammatical structure of the input, so that it does not have to learn this information from scratch. This can help the model to better understand the meaning and context of the input text.

Figure 2.25: Step D.2.3: Char embedding feature extraction

Similarly to the process for extracting POS embeddings, character embeddings (Fig.2.25)are also obtained by using a lookup table based on the unique ID of each character. However, an additional step of 1D convolution[11] is applied as a final step. This allows the model to handle misspelled words by learning how to combine different characters in order to obtain the correct label. The parameters within the kernel try to learn how to "adjust" misspelled words in order to produce the desired output. This can be especially useful when dealing with noisy or unstructured input text.

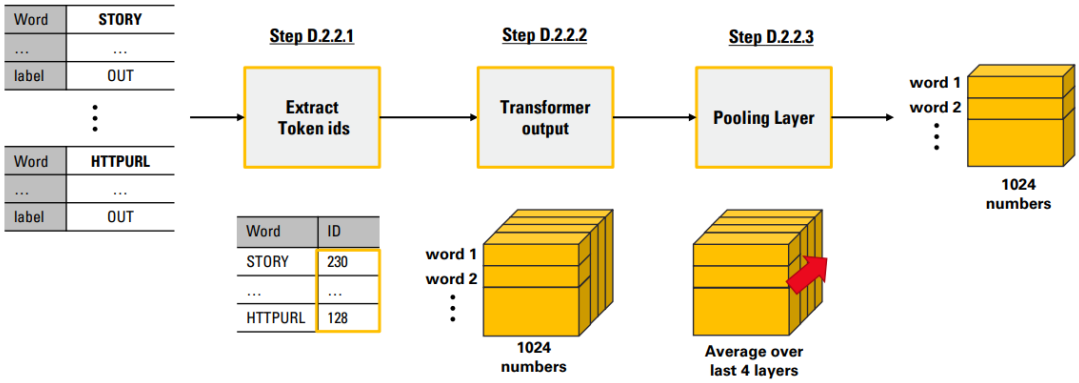


Figure 2.26: Step D.2.2: Word embedding feature extraction

The word embedding is based on BerTweet [2]. This is the only feature extraction context based. Indeed, the transformer is able to extract the word embeddings based on its sentence. This is possible thanks to the attention mechanism. The

[11]One dimension because it slides only to the right (the kernel size is the width)

process to extract word embeddings (Fig. 2.26) takes as input the entire sentence (difference from previous embeddings). The last four layers of the transformers are averaged together. It has been found that this is the best option for Named Entity Recognition (NER). The transformer's parameters are not trained to prevent overfitting to the dataset, as the goal is to keep the model general and able to perform well on a different dataset.

2.3.5 Step D.3-D.7: Classification Head

So far in the model, we have extracted word-level features that are context-aware, but we have not yet encoded the relationships between these features. Steps D.4 and D.5 (as shown in Fig. 2.27) are used to address this by encoding their relationships.

To avoid overfitting, a dropout layer is applied before the input is passed to the bidirectional LSTM. This layer analyzes the feature embeddings (the words) and tries to learn how they are connected. The output of the LSTM is a matrix with shape [number of words, 515x2], where the number of columns is the concatenation of the two LSTM hidden states (each with a size of 512). The multi-head attention layer focuses the model on the most important parts of the sentence for the labeling process. For example, if most of the tags follow a specific grammatical structure, the attention layer can detect this pattern. Another dropout layer is applied at the end to avoid overfitting the dataset by reducing the number of learned connections.

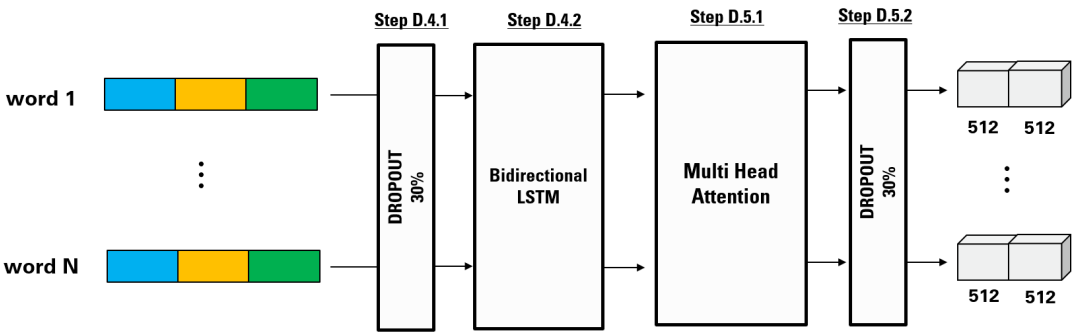


Figure 2.27: Step D.4 & D.5: Bidirectional LSTM and Multi Head Attention

The word-level features are converted to numbers using the different word embeddings. The relationships between these features are encoded by the bidirectional LSTM and attention layers. At this point, the model's output needs to be analyzed and converted back to labels (as shown in Fig. 2.28). The input to this process is the output of the attention layer, which is a matrix of size [number of words, 1024]. This matrix is first transposed, and then the dimension is reduced from 1024 to the current number of tags using a neural network (step D.6). The output of this step is then analyzed by a linear Conditional Random Field (CRF), which labels each word based not only on the output of step D.6 but also on the previous and next predictions. This helps to ensure that the labels are coherent and meaningful

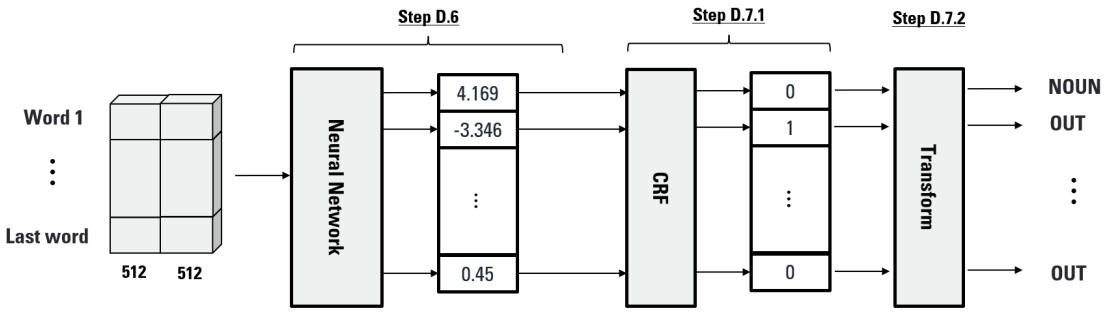


Figure 2.28: Step D.6 & D.7: Neural Network and CRF

2.3.6 Ablation Study: All the steps are really necessary?

Before training a model, it is common to conduct ablation studies to understand the contribution of each component of the model to its overall performance. In an ablation study, a specific component of the model is removed or "ablated" to see how the model's performance changes without that component. This can help identify which components are important for the model's performance and which may be unnecessary or redundant. It is important to conduct ablation studies before training the model, as the results can inform the final model architecture and hyperparameter selection. After the model is trained, it is not useful to conduct ablation studies, as any changes to the model architecture or hyperparameters would require retraining the model. The NER model contains 11 structural hyperparameters:

1. ***apply_pos:*** Whether to apply the POS embedding or not.
2. *pos_embedding_dim:* Dimension assigned to learn the POS attribute.
3. ***apply_char:*** Whether to apply CHAR embedding or not.
4. *char_embedding_dim:* Dimension assigned to learn the CHAR attribute.
5. *char_output_channel:* Number of filter for each kernel. More filters means more possible pattern learnt.
6. *char_kernel_sizes:* Filter size. Bigger sizes means learning connection with more letters.
7. ***apply_transformer:*** Whether to apply Transformer embedding or not.
8. *lstm_hidden_size:* Dimension assigned to learn the connection between words.
9. ***lstm_num_layers:*** Whether to stack 2 LSTM or not.
10. ***apply_attention:*** Whether to apply Attention or not.
11. ***apply_special_tag:*** Whether to apply START and END tag or not.

Considering two values for each of the numerical hyperparameters, we would need to train 2048 models, one for each possible combination of the values. However, since this is not practical, we have selected 6 specific hyperparameters that have a significant impact on the model structure and will only consider these in our ablation studies, reducing the number of combinations to 64.

We are faced with the challenge of analyzing 64 different trainings in order to select a general model with high performance and low memory size. This can be a

daunting task, as there are many potential combinations of hyperparameters to consider. One solution to this problem is to use TensorBoard logs to help us analyze the data. TensorBoard is a visualization tool that allows us to view and analyze training data in order to identify trends and patterns. By using TensorBoard logs, we can approximately analyze the entire hyperparameter analysis, focusing specifically on structural hyperparameters.

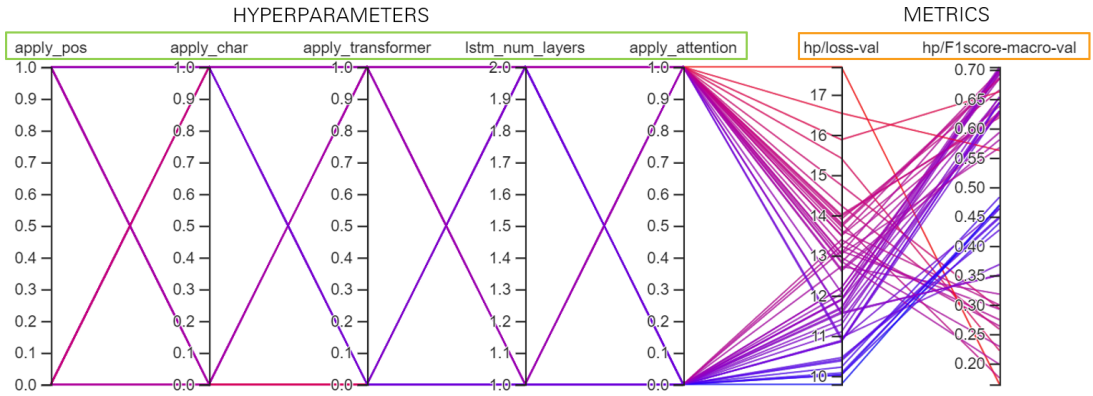


Figure 2.29: TensorBoard plot of the 64 training combination.

The figure 2.29 displays the results of the 64 trainings on the TensorBoard plot. The plot shows the hyperparameters used in each training in green columns and the metric used to evaluate the trainings, which is the validation loss and the average F1-score on the validation dataset, in an orange column. The plot can be read from left to right, with each line representing one training combination and the corresponding hyperparameters and metrics. The hyperparameters used are categorical and are represented by a 1 if they were used in the training or a 0 if they were not. The model was trained for a maximum of 3 epochs in order to reduce the training time.

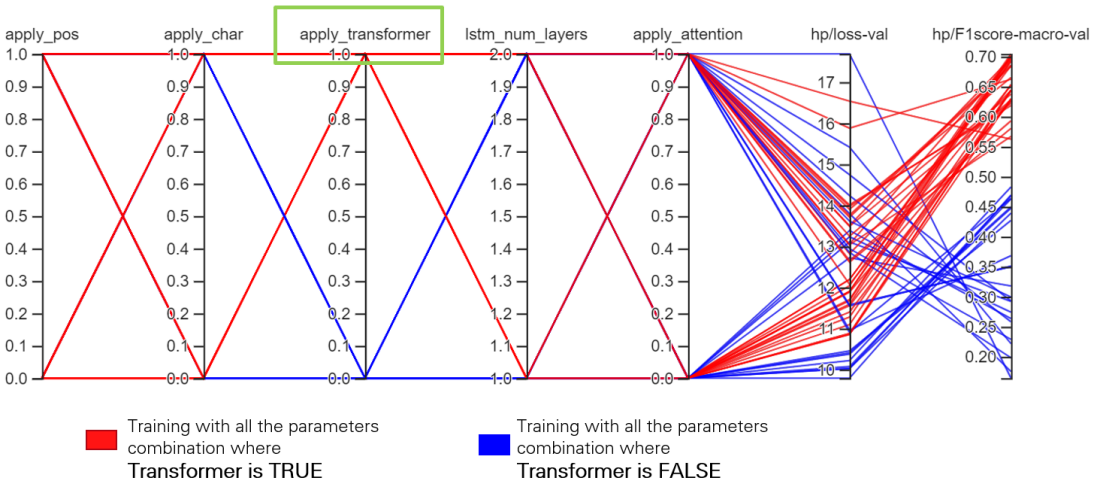


Figure 2.30: TensorBoard plot with focus on whether to apply BerTweet for word embedding or not.

The analysis was not fully reported, but in the figure 2.30 the initial step is shown. This visualization is slightly different because it focuses on the use of BerTweet. The red lines represent combinations where the transformer is used, while the blue lines represent combinations where the transformer is not used. It is clear from the plot that when the transformer is used, the macro F1-score increases by about 0.2 points, indicating that the transformer is crucial for the model. The next steps of the analysis were conducted only on the 32 combinations where the transformer is used. The final set of hyperparameters is chosen based on low validation loss, high F1-score, and, if multiple options met these criteria, the combination with the lower memory footprint is selected.

The best combination of hyperparameters is reported in the table 2.6. We can notice how the POS embedding resulted to be not beneficial to the task. There are several potential reasons for this. One possibility is that the POS embeddings are not providing enough useful information to the NER model, and are therefore adding unnecessary noise or confusion. A Another possibility is that the POS embeddings are conflicting with the other features being used by the NER model, leading to redundancy which negatively affect the model's performance. The multi-head attention is also removed from the model. One reason could be because the Bi-LSTM layer is already sufficiently capturing the context and dependencies in the input data, making the attention layer unnecessary. Alternatively, the attention layer may not be training effectively due to insufficient data or insufficient training time, resulting in its inability to focus on the most relevant parts of the input data. The final architecture is reported in Fig. 2.31.

hyperparameter	purpose	result
apply_pos	Whether to apply the POS embedding or not	NOT
apply_char	Whether to apply CHAR embedding or not	YES
apply_transformer	Whether to apply Transformer embedding or not	YES
lstm_num_layers	Whether to stack 2 LSTM or not	NO
apply_attention	Whether to apply Attention or not	NO
apply_special_tag	Whether to apply START and END tag or not	YES

Table 2.6: Ablation study best hyperparameter combination

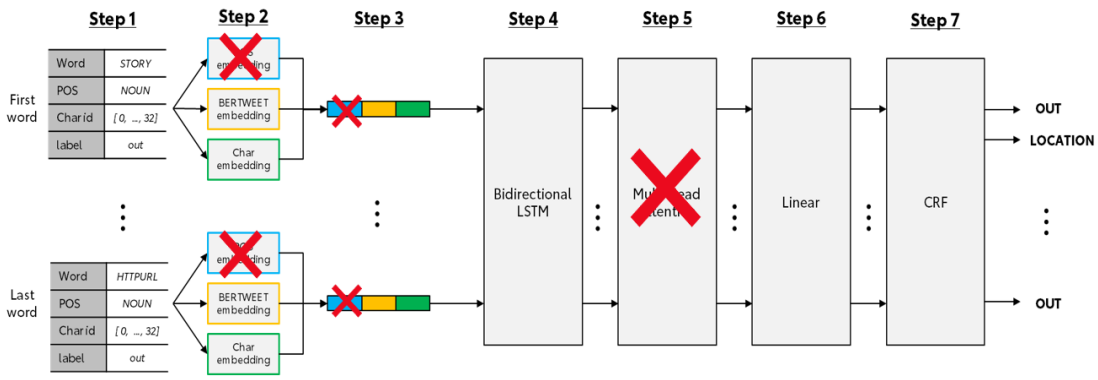


Figure 2.31: NER model after ablation study

2.3.7 Training Phase and Results

To evaluate the effectiveness of our sentiment analysis model, we compared its performance to that of a human benchmark, using the Human Level Performance analysis method on a set of collected tweets. However, this approach is not practical for named entity recognition (NER) because creating a comprehensive dataset would be too costly. As a result, we trained the NER model on two distinct datasets, which are described in more detail in Section 2.3.1. The final NER predictions are a combination of the output from both of these models.

The code section below specifies the hyperparameters for training a model. These hyperparameters can be grouped into several categories: Training, POS embedding, Char Embedding, BerTweet embedding, Bi-LSTM, and multi-head attention. The "apply_special_tag" parameter in the first section is used to mark the start and end of a sentence, and the "char_max_word_len" parameter specifies the maximum length of a word that will not be truncated. This is useful in the Char Embedding section because it allows for padding to be applied to a batch of words to ensure that they all have the same length. If a word is too long, the resulting matrix will contain mostly padding IDs, which can negatively affect the model's performance. In the Char Embedding section, the "char_kernel_size" parameters define the number of filters to be learned for convolutions over three and six characters. The BerTweet hyperparameters are taken from its paper [2], and the POS embedding and multi-head attention layers are not used.

```
# Training hyperparameters
batch_size_train=32,
batch_size_val=32,
lr=1e-3,
apply_special_tag=True,
# POS EMBEDDING
apply_pos=False,
pos_vocab_size=19,
pos_embedding_dim=128,
# CHAR EMBEDDING
apply_char=True,
char_vocab_size=259,
char_embedding_dim=128,
char_padding_idx=256,
char_max_word_len=50,
char_output_channel=30,
char_kernel_sizes=[3, 6],
# BERTWEET EMBEDDING
apply_transformer=True,
```

```
transformer_embedding_dim=1024,
transformer_embedding_max_sentence=512,
# Bi-LSTM
lstm_hidden_size=512,
lstm_num_layers=1,
# APPLY ATTENTION
apply_attention=False
```

In this model, there are 361 million (see table 2.7 for more details) total parameters, with 355 million of those parameters belonging to the BerTweet component and the rest associated with the rest of the model. However, only 6.6 million of these parameters are trained, as the transformer component is not being trained. This is done in order to allow the system to take advantage of the knowledge and abilities of the pre-trained transformer model, which has learned from a large amount of general-purpose text data, rather than starting from scratch on the NER task. Additionally, not specializing the pre-trained transformer on the NER datasets, allows all of the models in the Voice of Customer (VoC) tool to share the same backbone, avoiding any potential shifts in data or concepts between the collected tweets and the dataset used for training.

Name	Params	Trained
char_embedding	33.2 K	YES
charCNN	34.6 K	YES
bertweet_embeddings	355 M	NO
bi-LSTM	6.5 M	YES
Linear	8.2 K	YES
CRF	80	YES
Trainable params	6.6 M	
Non-Trainable params	355 M	
Total Params	361 M	
Total estimated model params size	1.4 K MB	

Table 2.7: NER model parameters: trained versus untrained

To assess the performances we use the F1-score. The F1 score is a measure of a model's performance that combines precision and recall, see section 2.2.3 for more details. There are three main variations: micro F1, macro F1, and weighted F1. Micro F1 calculates overall performance across all classes. Macro F1 calculates the average performance for each individual class. Weighted F1 assigns weights to each class and calculates the weighted average performance. Since the number of tokens per class can vary a lot, as shown in the datasets section 2.3.1, the macro F1 score

is used as evaluation metric. This because it can provide a balanced evaluation of the model's performance across all entity classes. In NER tasks, it is important to not only identify the named entities correctly, but also to do so consistently across all classes. The macro F1 score takes into account the performance of each individual class and averages these scores, allowing the model's overall performance to be evaluated in a balanced way. In contrast, the micro F1 score can be useful for tasks with a large number of classes, but it may not provide as balanced an evaluation of the model's performance, as the overall scores may be influenced more by the performance on the more frequently occurring classes. The weighted F1 score, on the other hand, may not provide as balanced an evaluation as the macro F1 score, as the performance of the less important or easier classes may be given less weight.

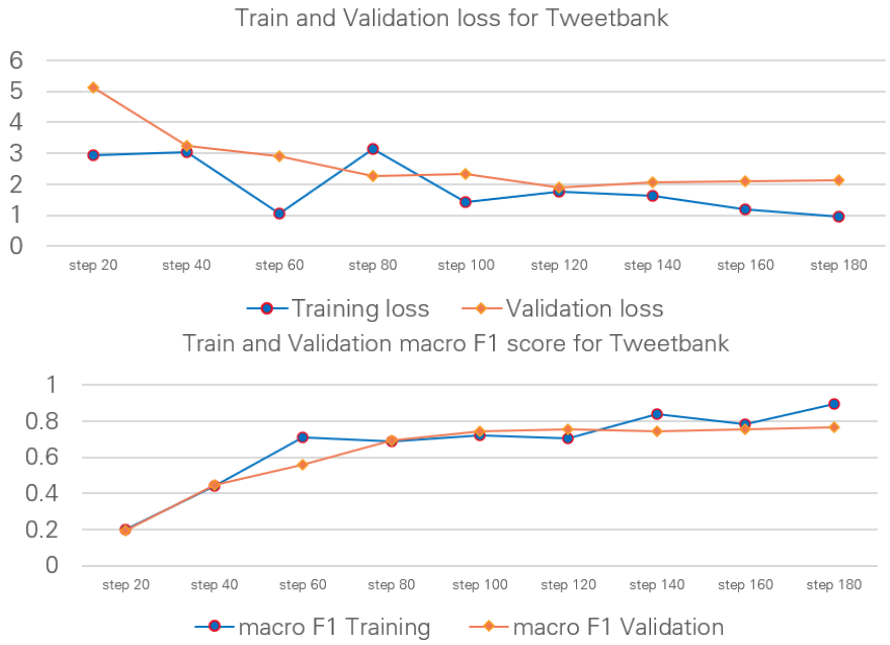


Figure 2.32: Tweetbank NER model performance metrics: Training and validation loss and F1 score. 1,639 tweets for training vs 710 for validation

In figures 2.33 and 2.32, the training results for the two different models. The model trained on Tweetbank appears to converge more quickly and has a lower validation loss compared to the model trained on Tweetner. This could be due to a number of factors. One possibility is that Tweetbank is a simpler dataset with fewer labels, making it easier for the model to learn from. On the other hand, Tweetner has more labels, which may make it a more challenging dataset for the model to learn from. Both models also have some fluctuations in their

training loss, which could potentially be reduced by increasing the batch size during training. By increasing the batch size, the model is able to see more examples at once, which can help it get a more accurate estimate of the loss. This can in turn help stabilize the training process and reduce the fluctuations. In addition, it's worth noting that the models also have different macro F1 scores. For the Tweetbank model, the F1 score increases significantly from the beginning to the end of training, starting from a low value of 0.2 and ending around 0.88. On the other hand, the F1 score for the Tweetner model remains relatively stable, with values between 0.55 and 0.8. This suggests that the model trained on Tweetbank is able to improve its performance more significantly over the course of training, while the model trained on Tweetner may have reached a plateau in its performance.

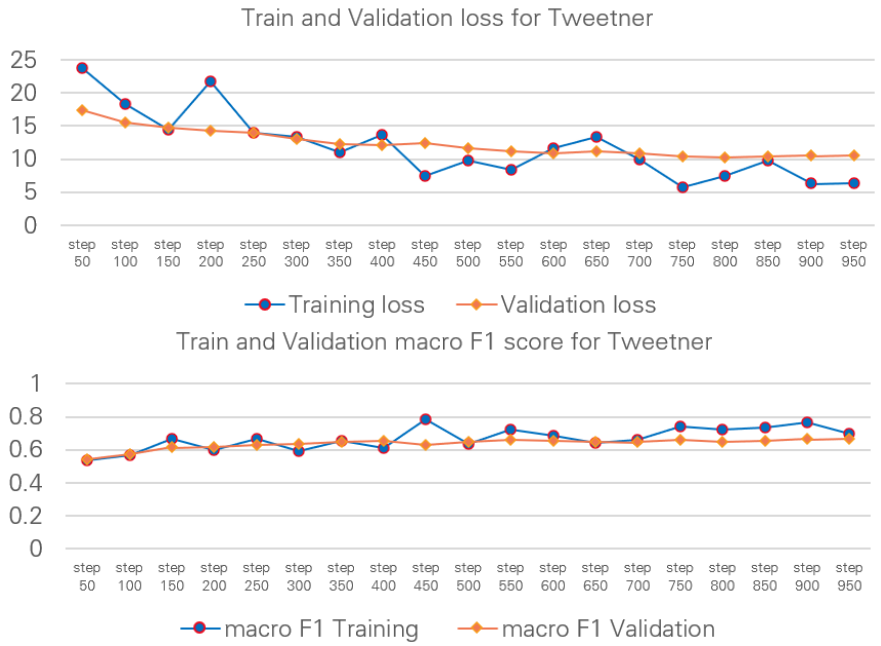


Figure 2.33: Tweetner NER model performance metrics: Training and validation loss and F1 score. 7,111 tweets for training vs 310 for validation

After the training procedure, the trained model are tested to assess the performances on unseen data. It's important to note that the test phase should only be performed once, after the model has been fully trained and tuned. This is because evaluating the model on the test set gives you an unbiased estimate of its performance on unseen data. If you evaluate the model on the test set multiple times, or if you use the test set to tune the model, you may end up with a overly optimistic estimate of the model's performance. In both cases, the trained models

not only closed the gap with the baseline but they also surpassed them as shown in figure 2.34. We can safely put them in production.

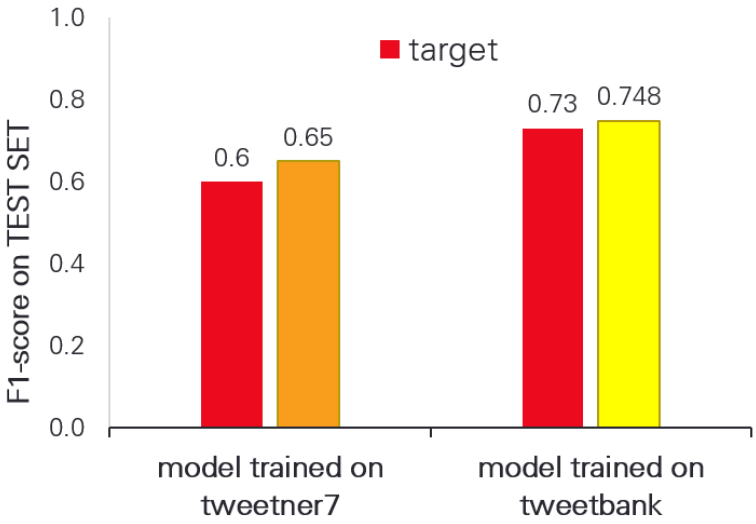


Figure 2.34: NER test results compared with baseline. Tweetbank with 1,201 tweets and Tweetner with 2,807 tweets as test set

The final version of the Voice of Customer (VoC) tool will be a combination of the predictions made by two different models. Ideally, the best solution would be to create a custom dataset with entities that are specific and relevant to the company. However, this can be a time-consuming and expensive process, so the decision has been made to use a combination of the predictions made by the two models instead. When the models make predictions for the same entity, the final prediction will be a combination of the two. For example, if one model predicts that the entity "organization" from a tweet is "Toyota Motor" and the other model predicts only "Toyota", the final prediction will be the shorter of the two, in this case "Toyota". This approach will help ensure that the final prediction is as accurate and relevant as possible. It's worth noting that this approach may not work in all cases, and it may be necessary to use additional methods to combine the predictions of the two models. However, for many entities, this simple approach should be sufficient to produce reliable and useful predictions.

Chapter 3

Use case and conclusion

3.1 Use case

The following paragraph describes a hypothetical scenario in which the implemented tool for Voice of the Customer (VoC) is used. Please note that the data and the analysis are obtained with the tool but the main target is fictitious. This use case is provided to help readers better understand the capabilities and potential uses of the VoC tool.

Imagine that the sales team has noticed a 5% drop in sales of Battery Electric Vehicles (BEVs) in Europe in December 2022. In order to identify the cause of this decrease and where it is occurring, the team is conducting a thorough analysis, considering all possible explanations. There is currently no clear reason for the decline in sales, so the team is not ruling out any potential causes and is actively trying to determine what might be contributing to the decrease. For this reason they ask to use the VoC tool.

We are using the Toyota Business Practice (TBP) procedure to analyze the decline in sales in Europe along with the VoC tool. We collect over 10,000 tweets about Battery Electric Vehicles (BEVs) in Europe with the first step, but this data alone is not sufficient to fully understand the situation. In Figure 3.1, we are performing a trend analysis for each individual country. From the data, we can see that Spain, Belgium, Germany, and Italy have not shown much interest in this trend in December 222. While this information, along with marketing data, may help us understanding the reason for the decline in sales, it is not enough on its own. For example, we don't know if people are discussing BEVs positively or negatively, or what specific topics they are discussing. Therefore, our analysis is not yet complete.

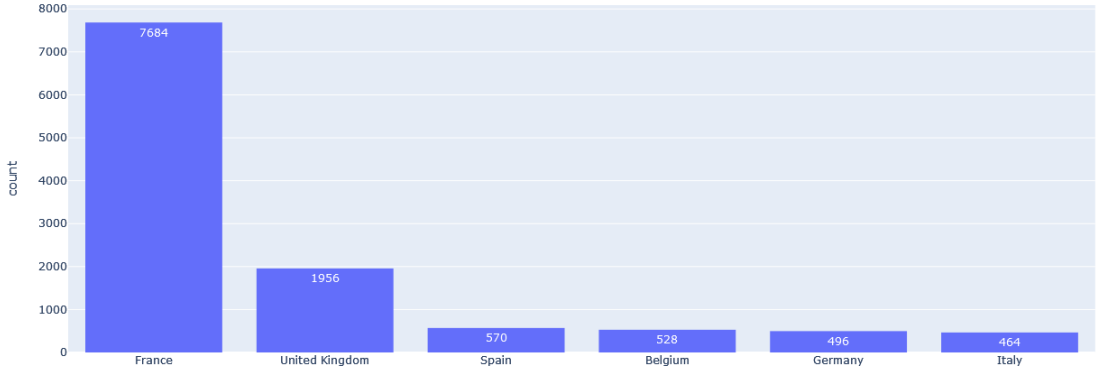


Figure 3.1: Step A: Analyzing the trend of "Battery Electric Vehicle" mentions on Twitter in December 2022, with data in the top six European countries based tweet volume.

As part of step B of the Voice of the Customer (VoC) tool, we are conducting sentiment analysis on the collected tweets to gain further insight into the situation. Upon analyzing the data shown in figure 3.2, we have discovered that a significant number of the negative tweets are coming from France. This information is important because it helps us to identify a potential source of the decrease in sales. By understanding what is driving negative sentiment towards Battery Electric Vehicles (BEVs) in France, we can work to address any issues and improve our sales in the region. There could be a variety of factors that are contributing to the negative sentiment towards Battery Electric Vehicles (BEVs) in France, such as the availability and accessibility of charging infrastructure or consumer concerns about the performance and range of BEVs. It would be necessary to conduct further analysis and research in order to identify the specific causes of the behavior in France.

In the previous step, we identified negative tweets about Battery Electric Vehicles (BEVs) coming from France, but we are not yet sure where to focus our attention. As a result, the next step in the Toyota Business Practice (TBP) procedure is to identify and prioritize the problem based on its level of importance, level of urgency, and level of expansion. To do this, we are using Named Entity Recognition (NER) on the negative tweets about BEVs tweeted from France. Figure 3.3 shows some of the tweets about BEVs tweeted from France in December 2023, with the predicted product in bold on the left and the predicted person in bold on the right. From the analysis, we can see that the model is detecting LITHIUM as a product in the tweets, and upon manual analysis, we see that people are complaining about the

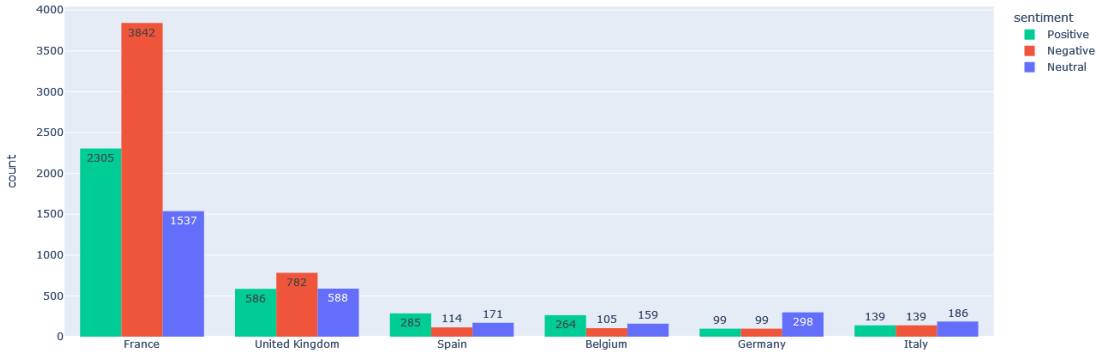


Figure 3.2: Step B: Performing sentiment analysis for "Battery Electric Vehicle" mentions on Twitter in December 2022 with data in the top six European countries based tweet volume.

weight of the battery, a water reaction, and the battery's autonomy. In addition, one person worth noting is Siwinskis. This family in Florida is having a difficult time after receiving a large quote to replace the batteries in their second-hand electric vehicle. The quote was more than they paid for their used 2014 Ford Focus Electric. This tweet became a trend in France.

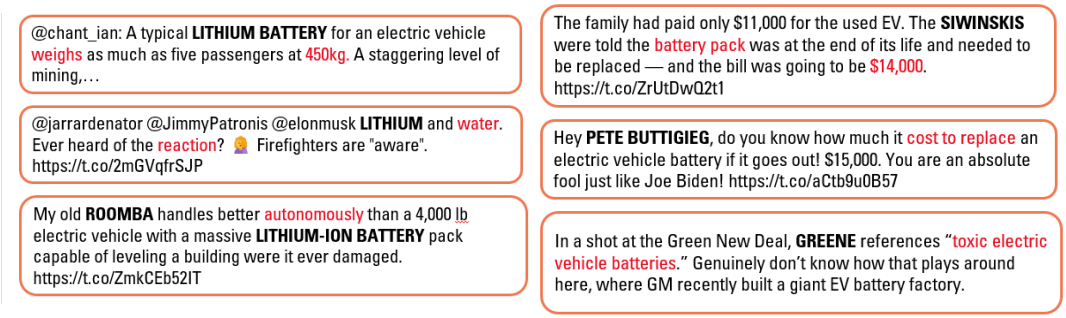


Figure 3.3: Step D: Applying NER to negative tweets about "Battery Electric Vehicle" tweeted from France in December 2023. The predicted product is in bold on the left and the predicted person is in bold on the right.

To summarize, we began our analysis with a small detail about a decrease in sales of Battery Electric Vehicles (BEVs) in Europe in December 2022. However, using the implemented Voice of the Customer (VoC) tool, we were able to conduct various analyses and identify potential root causes of the problem, such as the trend tweet of Siwinskis and customer complaints about an explosion caused by a

reaction between water and lithium. It's worth noting that we were able to collect and analyze tweets in less than 10 minutes, with about 7 minutes spent scraping the data from the Twitter API and less than 3 minutes for sentiment analysis and Named Entity Recognition (NER) performed on about 10,000 tweets exploiting GPU computation.

3.2 Conclusion

This thesis presents the VoC tool which has been developed and deployed for conducting analysis in real time. The tool consists of four macro steps that can be run in parallel and are flexible. The decision-making flow may differ from the one presented, depending on the specific needs and goals of the organization. Two architectures have been proposed and developed one for Sentiment Analysis and one for Named Entity Recognition (NER). In the case of Sentiment Analysis, we use Human Level Performance to demonstrate Data & Conceptual shift and we propose one possible process to create manually label dataset. For NER, we propose a method for processing the data to leverage different word embeddings and a way to conduct ablation studies using TensorBoard. The tool has been recognized for its contributions to the field and it will be a valuable resource for conducting VoC analysis in the future.

In this thesis, we explored also the intersection of Machine Learning Operations (MLOps) and Toyota Business Practice (TBP). Through our research, we identified several key areas where MLOps principles can be applied to TBP, including data management, model deployment, and system monitoring. The developed tool demonstrates how these principles can be applied in a real-world scenario.

Our research shows that by incorporating MLOps principles into TBP, organizations can improve the efficiency and effectiveness of their machine learning initiatives. By automating and streamlining processes, organizations can reduce the time and resources required for model development and deployment, and ensure that models are performing at their best. Additionally, the use of MLOps principles can help organizations to better understand the performance of their models and identify areas for improvement.

Bibliography

[1] Nianwen Xue. «Steven Bird, Evan Klein and Edward Loper. Natural Language Processing with Python. O'Reilly Media, Inc.2009. ISBN: 978-0-596-51649-9.» In: *Natural Language Engineering* 17.3 (2011), pp. 419–424. DOI: 10.1017/S1351324910000306 (cit. on p. 16).

[2] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. «BERTweet: A pre-trained language model for English Tweets». In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 9–14. DOI: 10.18653/v1/2020.emnlp-demos.2. URL: https://aclanthology.org/2020.emnlp-demos.2 (cit. on pp. 17, 34, 36, 41, 49).

[3] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. DOI: 10.48550/ARXIV.1907.11692. URL: https://arxiv.org/abs/1907.11692 (cit. on pp. 17, 24).

[4] Alexis Conneau et al. «Unsupervised Cross-lingual Representation Learning at Scale». In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. URL: https://aclanthology.org/2020.acl-main.747 (cit. on p. 17).

[5] Alexis Conneau et al. *Unsupervised Cross-lingual Representation Learning at Scale*. 2019. DOI: 10.48550/ARXIV.1911.02116. URL: https://arxiv.org/abs/1911.02116 (cit. on p. 24).

[6] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. *Parameter-Efficient Transfer Learning for NLP*. 2019. DOI: 10.48550/ARXIV.1902.00751. URL: https://arxiv.org/abs/1902.00751 (cit. on p. 24).

[7] Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. «AdapterFusion: Non-Destructive Task Composition for Transfer Learning». In: (2020). DOI: 10.48550/ARXIV.2005.00247. URL: https://arxiv.org/abs/2005.00247 (cit. on p. 24).

[8] Asahi Ushio, Leonardo Neves, Vitor Silva, Francesco Barbieri, and Jose Camacho-Collados. *Named Entity Recognition in Twitter: A Dataset and Analysis on Short-Term Temporal Shifts*. 2022. DOI: 10.48550/ARXIV.2210.03797. URL: https://arxiv.org/abs/2210.03797 (cit. on pp. 33, 34).

[9] Hang Jiang, Yining Hua, Doug Beeferman, and Deb Roy. *Annotating the Tweebank Corpus on Named Entity Recognition and Building NLP Models for Social Media Analysis*. 2022. DOI: 10.48550/ARXIV.2201.07281. URL: https://arxiv.org/abs/2201.07281 (cit. on p. 33).

[10] Fan Zhang. «A hybrid structured deep neural network with Word2Vec for construction accident causes classification». In: *International Journal of Construction Management* 22.6 (2022), pp. 1120–1140. DOI: 10.1080/15623599.2019.1683692. eprint: https://doi.org/10.1080/15623599.2019.1683692. URL: https://doi.org/10.1080/15623599.2019.1683692 (cit. on p. 36).

[11] Muzamil Hussain Syed and Sun-Tae Chung. «MenuNER: Domain-adapted BERT based NER approach for a domain with limited dataset and its application to food menu domain». In: *Applied Sciences* 11.13 (2021), p. 6007 (cit. on p. 36).

[12] Hang Jiang, Yining Hua, Doug Beeferman, and Deb Roy. «Annotating the Tweebank Corpus on Named Entity Recognition and Building NLP Models for Social Media Analysis». In: *In Proceedings of the 13th Language Resources and Evaluation Conference (LREC)* (2022) (cit. on p. 38).