Conversational AI Bot for US Financial Markets

A Project Report

submitted by

SALURU DURGA SANDEEP

in partial fulfilment of the requirements

for the award of the

DUAL DEGREE

(B.Tech. and M.Tech.)

with specialization in

DATA SCIENCE

DEPARTMENT OF MECHANICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

June 2021

CERTIFICATE

This is to certify that the project report titled **Coversational AI Bot for US Financial Markets**, submitted by **Saluru Durga Sandeep (ME16B125)**, to the Indian Institute of Technology Madras, for the award of the **Dual Degree**(B. Tech and M. Tech), is a bonafide record of the research work done by him under my supervision. The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: Chennai

Date: 30th June 2021

Dr. Nandan Sudarsanam
Project Guide
Associate Professor
Dept. of Management Studies
IIT Madras, 600 036

ACKNOWLEDGEMENTS

First of all, I would like to thank my guide Dr.Nandan Sudarsanam for providing me an opportunity to work with him and I am extremely grateful for his guidance, encouragement and support throughout the year. I appreciate that he would always make some time for this project, despite his very busy schedule. Being connected online on a regular basis helped the project progress smoothly and his insights on the project's approach were extremely helpful.

I would also like to extend my sincere gratitude towards the team at Quantel AI, the US based start-up especially Mr.Shyam that floated the problem statement and engaged with me regularly and the team at IIT Madras throughout the year. The regular discussions were very informative.

Last but not the least, I would like to thank my family for being by my side throughout this project and in very venture that I takeup in life.

ABSTRACT

KEYWORDS: Intent Classification, Entity Recognition, Natural Language Understanding(NLU), Paraphrase Generation, Conditional Random Field(CRF), Stock Market, Bot

A conversational AI Bot has mainly two components, one is Natural Language Understanding(NLU) and other is Dialogue management. In this dual degree project, we concentrate on building the NLU component, where different concepts and research studies of NLP is used. NLU component consists of two tasks, one is Entity recognition and other is Intent classification. Usually, these two tasks are interrelated. In this project, we do not have any benchmark dataset like other standard research problems. So we approach the problem statement from very basics i.e creating training dataset and build entity recognition, Intent Classification models.

In this project, we discuss on different training data creation techniques inspired from research papers on "Paraphrase Generation" i.e given a sentence, need to generate similar sentences. We will also discuss about different ways of improving the entity recognition task using pre-trained entity extractors, inspired from research on "Named Entity Recognition" task.Outcome of this paper is that we successfully generated around **1000 data samples** from just **80 samples** of initial data. We validate this data generation technique based on the Intent classification model performance. We will also discuss about challenges in this project i.e. lack of training data and lack of benchmark data

TABLE OF CONTENTS

Table of Contents (continued) **Page**

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Project Overview

A conversational AI Bot has mainly two components, one is **Natural Language Understanding(NLU)** and other is **Dialogue management**. In this project, we concentrate on building the NLU component, where different research studies of NLP is used. NLU component consists of two tasks, one is **Entity recognition** and other is **Intent classification**. Usually, these two tasks are interrelated. In this project, we do not have any benchmark dataset like other standard research problems. So we approach the problem statement from very basics i.e creating training dataset and build entity recognition, Intent Classification models.

In this project, we discuss on different training data creation techniques inspired from research papers on **"Paraphrase Generation"** i.e given a sentence, need to generate similar sentences. We will also discuss about different ways of improving the entity recognition task using pre-trained entity extractors, inspired from research on "Named Entity Recognition" task.Outcome of this paper is that we successfully generated around **1000 data samples** from just **80 samples** of initial data. We validate this data generation technique based on the Intent classification model performance. We will also discuss about challenges in this project i.e. **lack of training data and lack of benchmark data**

1.2 Functions of Chatbots in Financial Markets

Following are few main functions of a chatbot in financial markets.

(a) **Support :** More and more banks and brokers are deploying chatbots to reduce the costs of customer support. Many questions are typical and frequently asked, and canned responses with a touch of natural language processing (NLP) tools can significantly reduce the number of incidents, which require actual human interaction. For example, Query like "I want to place a buy order", here the chatbot analyses the query and ask appropriate questions like which company, how much quantity and type of order and other details required to place an order in the market. This can be made more interactive with the user using chatbot rather than just giving a form to fill the contents. It could be a real delighter for traders to do things on the go without the need to open a trading platform like close all positions, check the unrealized P&L etc.

(b) **Convert :** Most people prefer to engage with programs that are human-like, and this gives chatbot-style techniques a potentially useful role in interactive systems that need to elicit information from users, as long as that information is relatively straightforward and falls into predictable categories. After following up any queries a trader may have, they analyze aggregated queries and send out relevant appeals, news or opportunities. For example, if a trader requests the price of Litecoin, a logical step could be a personalized offer to open an account: "George, you recently asked me about Litecoin. Would you like to trade Litecoin, Bitcoin, Ether and 100+ more cryptocurrencies on our new multi-asset trading platform? Click here to find out more". **Personalization** ultimately produces **higher conversion** and **activation rates**.

(c) **Engage :** Chatbots give the ability to start small talk or engage in casual conversations between customers and businesses without human interaction. For example, Bot can send performance insight to inform the trader if he made it into the top-10 list of progressive traders according to this chatbot statistics, or if he was more profitable than X% of other broker's traders. This would give an overview of his performance in trading.

1.3 Examples

Chatbots have the ability to initiate and close sales, where a sale equals the placement of an order or a deposit. Let's see some of the examples

(a) **Notification :** The user sets an alert, for example, "Notify me when a Bitcoin is less than $7000". If this event occurs, the chatbot notifies the user adding a call to action "Would you like to trade the Bitcoin now?" In other words, any interest of the user to a financial instrument is followed by a call to action which facilitates a transaction.

(b) **Market Queries :** A second example can be how chatbots help discover new opportunities. For example, a trader may ask, "What are the top performing US stocks in the past 3 months?", the chatbot would then find such stocks and show them with an executive summary helping the user to make a trading decision.

(c) **Warnings :** Being connected to a trading platform, the chatbot can analyze the trader's exposure to the market and warn of potential threats, for example, "Annual Budget results are negative for AAAA sectors, which may cause drop in those sectors. Would you like to manage your positions now?". This way, the trader can be warned in such events automatically. Traditionally traders watch the news while trading to know the market events and sometimes the crucial market information can be missed but using a smart chatbot will make it easier for traders.

CHAPTER 2

Chatbot Components

2.1 NLU

NLU is the core component that interprets what users say at any given time and converts the language to structured inputs that system can further process. Since the chatbot is domain-specific, it must support so many features. NLU contains advanced machine learning algorithms to identify the user's intent and further matches them to the list of available intents the bot supports.

NLP Engine further has two components:

- **Intent Classifier :** In this task, we have some set of pre-defined intents(i.e sentence labels) like market_data, stock_data, portfolio_data and sector_data. Given the sentence, we need to classify it among the given 4 intents. This can be seen as **"Multi-Class Classification"** task. This is also known as **Sentence Level Labelling Task**.
- **Entity Extractor :** Entity extractor is what extracts key information from the user's query. This task is very similar to Parts-of-Speech Tagging task. Here, for each word we need to label them with the given entities(i.e labels). This is known as **"Word-Level Labelling Task"**. Refer to example in the figure[2.1]

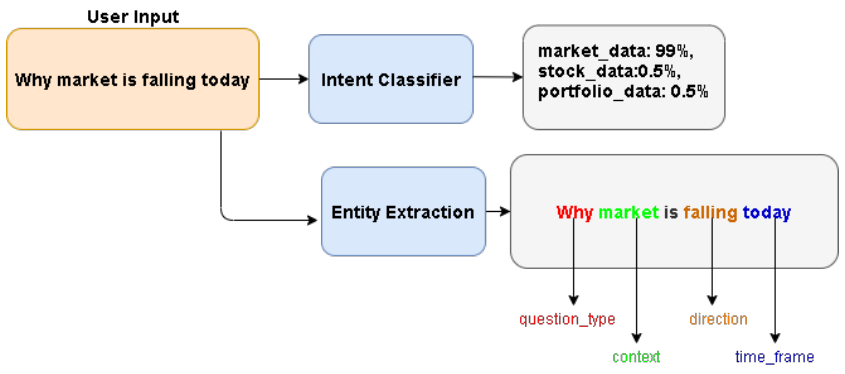


Fig. 2.1: NLU Example

2.2 Dialogue Management

It manages the actual context of the dialogue. For example, the user might say “He needs to order ice cream” and the bot might take the order. Then the user might say “Change it to coffee”, here the user refers to the order he has placed earlier, the bot must correctly interpret this and make changes to the order he has placed earlier before confirming with the user. Dialog management plugin enables us to do this.
Dialogue management further has following key plugins:

- **Feedback Mechanism :** Here the agent takes the feedback from user time to time to learn if the bot is doing fine with the conversation and the user is satisfied with the bot’s response. This reinforces the bot to learn from mistakes and corrects itself in future conversations.
- **Policy Learning :** Policy learning is a higher-level framework that teaches the bot to take more of happy paths during the conversation to improve overall end-user satisfaction.
  (a) Broadly it creates the network of happy paths and routes the conversation to end-user satisfaction.
  (b) The bot then tries to learn from the interactions and follows the interaction flow about the conversation it had with similar users in the past.

Note: As far as this dual degree project is considered. I worked on NLU component i.e Entity extraction and Intent classification.

CHAPTER 3

Literature Survey

3.1 Paraphrase Generation Task

3.1.1 Reformulating Unsupervised Style Transfer as Paraphrase Generation

Modern NLP defines the task of style transfer as modifying the style of a given sentence without appreciably changing its semantics, which implies that the outputs of style transfer systems should be paraphrases of their inputs.

In this paper **Kalpesh Krishna (2020)**, they reformulate **unsupervised style transfer** as a **paraphrase generation**. problem, and presented a simple methodology based on fine-tuning pre-trained language models on automatically generated paraphrase data. Despite its simplicity, this method significantly outperforms state-of-the-art style transfer systems on both human and automatic evaluations. This is an **unsupervised method (Style Transfer via Paraphrasing, or STRAP)** requires no parallel data between different styles and proceeds in **three simple stages**:

(a) Create **pseudo-parallel data** by feeding sentences from different styles through a **diverse paraphrase model**. Here we use pre-trained paraphrase model to generate pseudo-parallel training data

(b) Train style-specific inverse paraphrase models that convert these paraphrased sentences back into the original stylized sentences. This is intuitively to remove the style, so this is called **"Inverse Paraphrasing"**.

(c) Use the inverse paraphraser for a desired style to perform style transfer

Corpus that is used to train the model is **CDS (Corpus of Diverse Styles)** of 15M English sentences spanning 11 diverse styles. including the works of James Joyce, romantic poetry, tweets, and conversational speech.

3.1.2 Generating Syntactically Controlled Paraphrases without Using Annotated Parallel Pairs

In this paper **Kuan-Hao Huang (2021)**, demonstrated that it is possible to generate syntactically various paraphrases **without the need for annotated paraphrase pairs**. They proposed **Syntactically controlled Paraphrase Generator (SynPG)**, an encoder-decoder based model that learns to disentangle the semantics and the syntax of a sentence from a collection of unannotated texts. **SynPG** consists of a **semantic encoder, a syntactic encoder, and a decoder**. The semantic encoder considers the source sentence as a bag of words without ordering and learns a contextualized embedding containing only the semantic information. The syntactic encoder embeds the target parse into a contextualized embedding including only the syntactic information. Then, the decoder combines the two representations and generates a paraphrase sentence. The design of disentangling semantics and syntax enables SynPG to learn the association between words and parses and be trained by reconstructing the source sentence given its unordered words and its parse. Therefore, we do not require any annotated paraphrase pairs but **only unannotated texts** to **train SynPG**.Here, the model **uses Stanford NLP Parser internally** in the pre-processing step. Refer to the following sample example 3.1 from this paper **Kuan-Hao Huang (2021)**

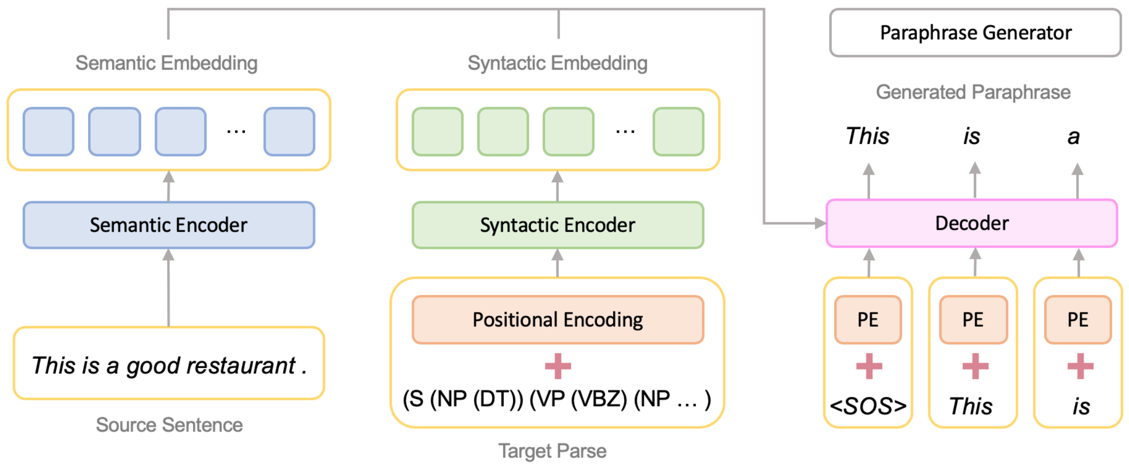


Fig. 3.1: Paraphrase generation with syntactic control

3.1.3 Synonymous Paraphrasing using Wordnet and Internet

In this paper **Igor A. Bolshakov (2004)**, proposed a method of synonymous paraphrasing of a text based on **WordNet synonymy** data and Internet statistics of stable **word combinations (collocations)**. Given a text, we look for words or expressions in it for which WordNet provides synonyms, and substitute them with such synonyms only if the latter form valid collocations with the surrounding words according to the statistics gathered from Internet. Evaluations of Collocation Statistics via Internet: our **goal** is to elaborate a **mechanism for assessing whether a word can be replaced with its synonym while keeping collocational cohesion of the text**, i.e., a means for deciding which synonyms of a given word can form good collocations with a word in the context. Simply, we can count the correct number of pages with a given collocation is between the two figures and consider the top 3 collocations for replacement in the original sentence.

3.1.4 Paraphrase Revisited with Neural Machine Translation

In the paper **Jonathan Mallinson (2017)**, we revisit **bilingual pivoting** in the context of neural machine translation and present a paraphrasing model based purely on neural networks called as **"ParaNet"**. **Bilingual pivoting method** is that two English strings e1 and e2 that translate to the same foreign string f can be assumed to have the same meaning. The method then **pivots over f** to extract **<e1,e2> as a pair of paraphrases**. Our model represents paraphrases in a continuous space, estimates the degree of semantic relatedness between text segments of arbitrary length, or generates candidate paraphrases for any source input. There are lot of **advantages** through this model like we can have dataset pairs which are used to train different machine translators. However **disadvantage** is these methods **have little to no control** over the preservation of the input meaning or **variation in the output surface form.**

3.1.5 Factorising Meaning and Form for Intent-Preserving Paraphrasing

In the paper **Tom Hosking (2021)**, they propose a method for generating paraphrases of English questions that retain the original intent but use a different surface form. The final model combines a careful choice of training objective with a **principled information bottleneck**, to induce a **latent encoding space** that disentangles meaning and form. Trained an encoder-decoder model to reconstruct a question from a paraphrase with the same meaning and an exemplar with the same surface form, leading to separated encoding spaces. We use a **Vector-Quantized Variational Autoencoder** to represent the surface form as a set of discrete latent variables, allowing us to use a classifier to select a different surface form at test time. Following figure 3.2 is the glimpse of an example flow in the model

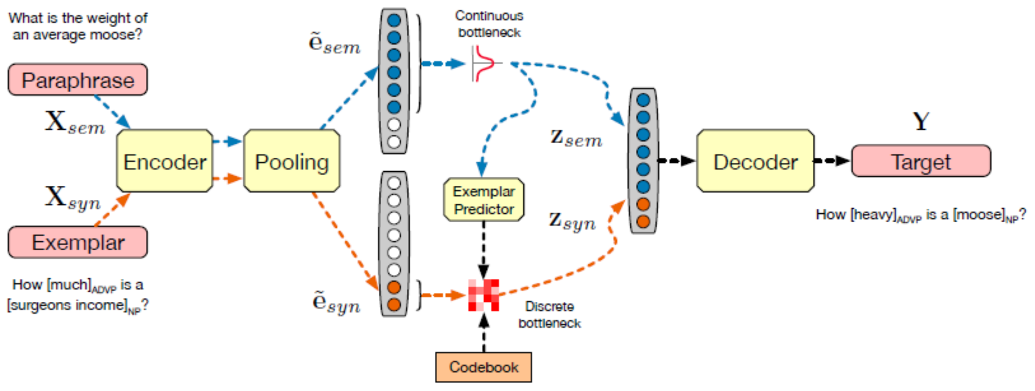


Fig. 3.2: Overview of the Separator Approach

3.2 NLU Task

3.2.1 CRF for Segmenting and Labeling Sequence Data

In this paper **John Lafferty (2001)**, first discussed about the limitations of Maximum Entropy Markov Model(MEMM)(from the paper A. McCallum (2000)) like **"Label Bias Problem"** i.e observations far away does not impact the early predictions, how to address these limitations using **Conditional Random Field Model** which uses undirected graphical model unlike **MEMM** which uses **directed graphical model**. In this paper John Lafferty (2001), we could clearly see the CRF Model performs better than the MEMM models in Parts-of-Speech Tagging task. In building our AI Bot, I used CRF Model as one of the entity extractor.In the later sections, we will discuss more about the pipeline of our AI Bot. In CRF Model, the output of current word depends on inputs and labels of its neighbours also. Refer to figure[3.3], for example, to predict the label of "falling" (true label is direction), our inputs are "falling", "is", "today", "no_intent", "timeframe".

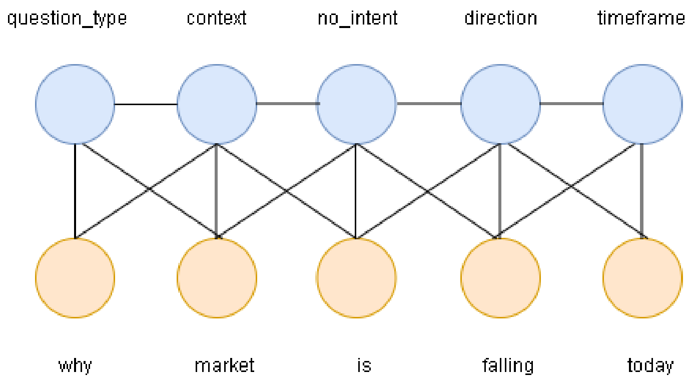


Fig. 3.3: Conditional Random Field(CRF) Architecture

3.2.2 Joint Intent Classification and Entity Extraction by CNN based CRF

In general, we try to build independent models for intent classification and entity extraction but in reality these two tasks has some dependency. In this paper **Puyang Xu (2013)**, the proposed architecture can be perceived as a neural network (NN) version of the **triangular CRF model (TriCRF)**, in which the intent label and the entity sequence are modeled jointly and their dependencies are exploited. For the Entity Extraction task, the features from the word sequence are automatically extracted using CNN, instead of predefined indicator functions like word-embeddings. Note that the features which are extracted through CNN layers are shared by the two tasks(i.e Intent Classification and Entity Extraction, refer to figure[3.4]). In this paper **M. Jeong**

(2008), TriCRF is proposed unlike the **linear chain CRF** which we discussed in section 3.2.

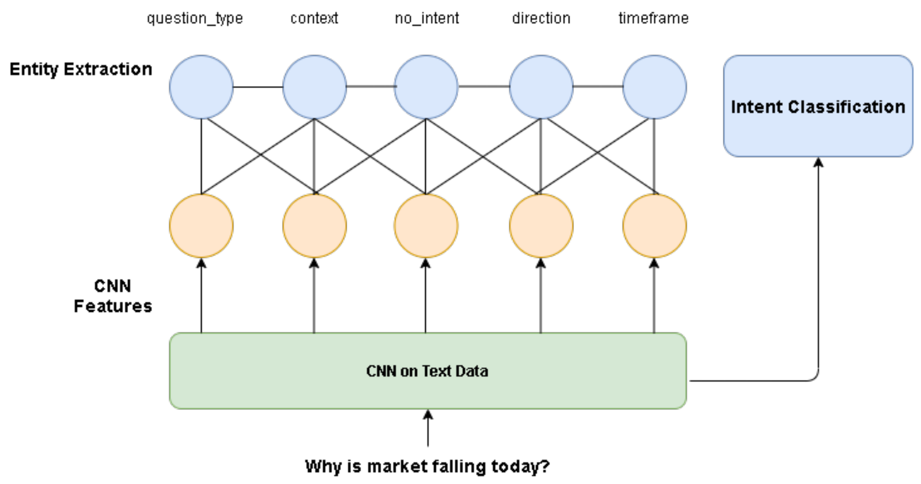


Fig. 3.4: Joint Intent Classification and Entity Extraction by CNN based CRF Architecture

3.2.3 Joint Intent Classification and Entity Extraction by Bi-LSTM

In this paper **Varghese Akson Sama (2020)**, the authors presented a **Simple LSTM - Bidirectional LSTM** in a joint model framework, for Intent Classification and Named Entity Recognition tasks. Both the **models** are **approached** as a **classification task**. This paper discuss the comparison of single models and joint models in the respective tasks, a data augmentation algorithm and how the joint model framework helped in learning a poor performing NER model in by adding learned weights from well performing Intent Classification model in their respective tasks. The experiment in the paper shows that there is **approximately 44% improvement** in performance of NER model when in joint model compared to when tested as independent model. Note that this architecture is very similar to figure [3.4] but only difference is inplace of extracting features using CNN, we use Bi-LSTM and hidden layers output is given as features to both Intent Classification and Entity Extraction. Refer to figure[3.5]

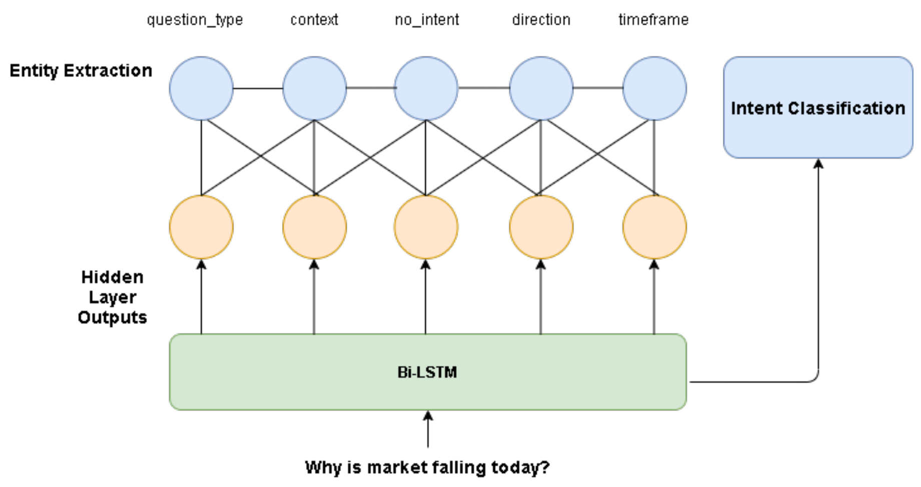


Fig. 3.5: Joint Intent Classification and Entity Extraction by Bi-LSTM

3.2.4 Joint Intent Classification and Entity Extraction by Sparse Attention Neural Networks

In this paper **Mingbo Ma (2017)**, authors proposed a jointly trained model for learning the two tasks simultaneously via **Long Short-Term Memory (LSTM) networks**. This model predicts the sentence-level category and the word-level label sequence from the stepwise output hidden representations of LSTM(from paper S. Hochreiter (1997)). We also introduce a novel mechanism of **"sparse attention"** to weigh words differently based on their semantic relevance to **sentence-level classification**. The overview of the architectures is as follows: We use CNN as a feature extraction tool to learn meaningful representation from both words and tags automatically. This representation is passed on to Multiple Input and Multiple Output LSTM architecture to generate hidden layer representation outputs for each word. This is used for word-level labelling. For the sentence-level task, we apply attention on top of the hidden layer output. Refer figure[3.6]

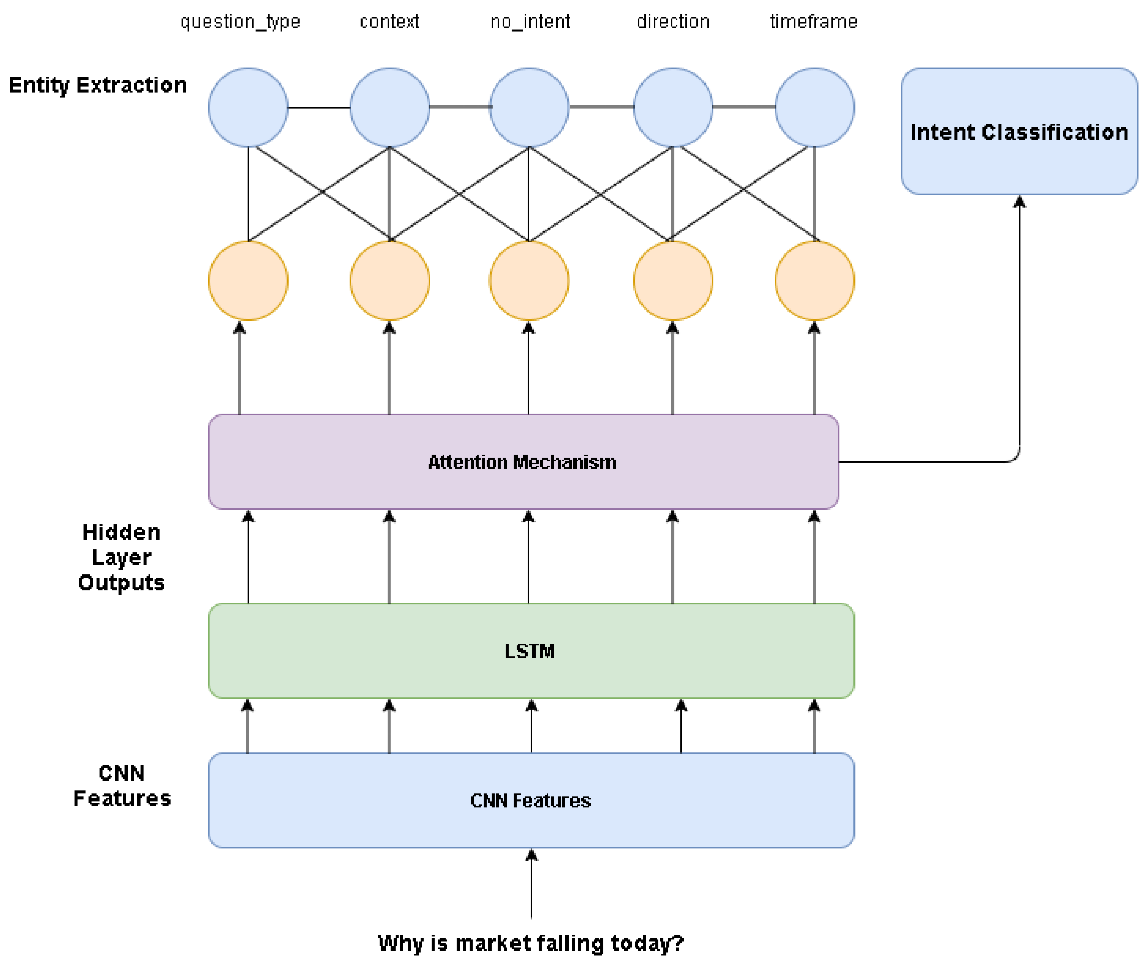


Fig. 3.6: Joint Intent Classification and Entity Extraction by sparse Attention Neural Networks

CHAPTER 4

Training Data Generation Results

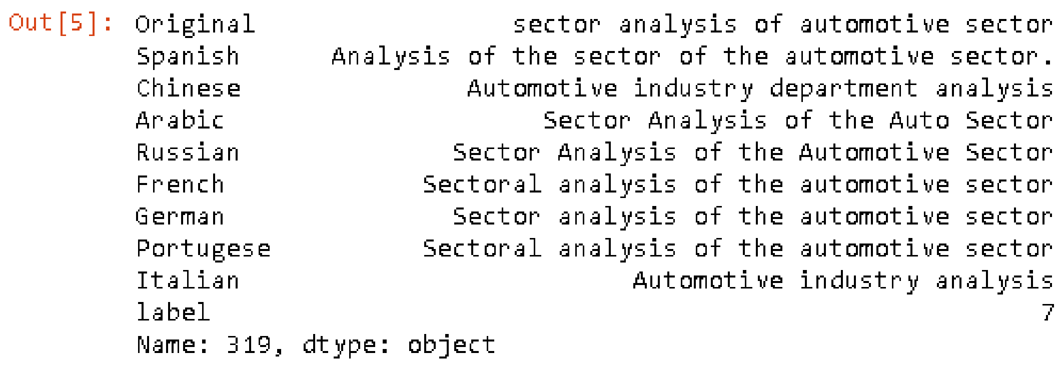
In Section 3.1, we have discussed about the literature work on paraphrase generation task. In this section, we will discuss about all the techniques used to create our training dataset. All the techniques applied are inspired from the literature review in section 3. Some techniques have small modifications to fine-tune to our task like domain specific and some techniques were directly applied from the literature, also shown significant results which we will be seeing in this section

4.1 Back Translation

This method has been inspired from the paper by **Jonathan Mallinson (2017)**, where the model uses pivoting language. Similarly, In this method I used different pivot languages means "**Multi-pivoting** approach. In this method there are two steps.

- Translate English sentence (Source language) to other languages like Spanish, French, Arabic, Chinese and so on.... Let's call this as "**pivot language**". We chose this pivot language based on the **good BLEU score**.
- Translate from pivot language to source language. This way, we can restore the Intent of the input sentence but structure of the sentence is different.

Here we use open-sourced state-of-the-art machine translations available online, such as **google translator**. Following figure 4.1 shows how back translation gives us the similar sentences given the input sentence.



```
Out[5]: Original                    sector analysis of automotive sector
        Spanish      Analysis of the sector of the automotive sector.
        Chinese                Automotive industry department analysis
        Arabic                      Sector Analysis of the Auto Sector
        Russian               Sector Analysis of the Automotive Sector
        French              Sectoral analysis of the automotive sector
        German                Sector analysis of the automotive sector
        Portugese           Sectoral analysis of the automotive sector
        Italian                           Automotive industry analysis
        label                                                        7
        Name: 319, dtype: object
```

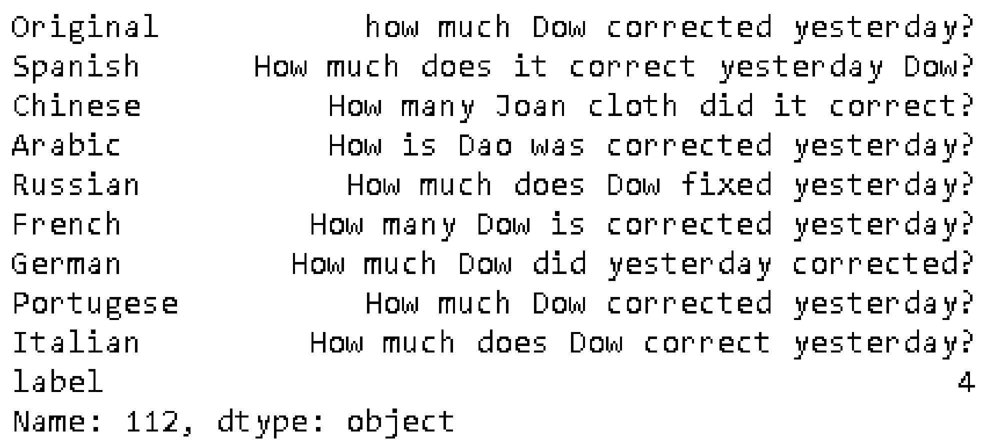
Fig. 4.1: Example of Back Translation. In this example, we have different ways of asking the input sentence "sector analysis of automotive sector"

Observations: From figure 4.1, we can observe the similar sentences with different words usage and different syntactic structure but the intent of the input sentence is preserved. For example, the word **"sector"** has been replaced with **"industry"** and **"department"**. This is a good example to show how back translation helps us in generating new data (paraphrase sentences). But there are sentences which are repeated, so we need **post-processing of generated sentences** to remove the same sentences.

4.1.1 Pros and Cons of Back Translation:

Advantages: One of the main advantage of this method is that we do not require any annotated data and also not required to train any neural machine translation models as we are using google translator which is trained on huge datasets by Google. This way we can leverage the best pre-trained models.

Disadvantages: There are **two disadvantages** in this method. **1)** we do not have control over the structure of the paraphrase sentence (i.e back translated sentence), **2)** We are not incorporating any domain knowledge into the translators, so there are high chances of generating sentences which are not in finance domain (refer to 4.2). This can be tackled using post-processing of the generated sentences. We will discuss about this in detail later.



```
Original              how much Dow corrected yesterday?
Spanish         How much does it correct yesterday Dow?
Chinese             How many Joan cloth did it correct?
Arabic              How is Dao was corrected yesterday?
Russian              How much does Dow fixed yesterday?
French             How many Dow is corrected yesterday?
German            How much Dow did yesterday corrected?
Portugese             How much Dow corrected yesterday?
Italian            How much does Dow correct yesterday?
label                                                 4
Name: 112, dtype: object
```

Fig. 4.2: Example of Back Translation. In this example, we can see that not all back translated sentences gives us required sentences i.e in financial domain

4.1.2 Sentence Similarity

One of the disadvantage of using back translation is that we have repeated sentences and completely unrelated sentences. To tackle these, we try post-processing technique called **"Sentence Similarity"** to remove the exactly same sentence and unrelated sentences. So first, we need to find different ways to find the representation of the input sentence and generated sentence. Then we use different similarity metrics to filter the sentences. Following figure4.3 is the post-processing pipeline.

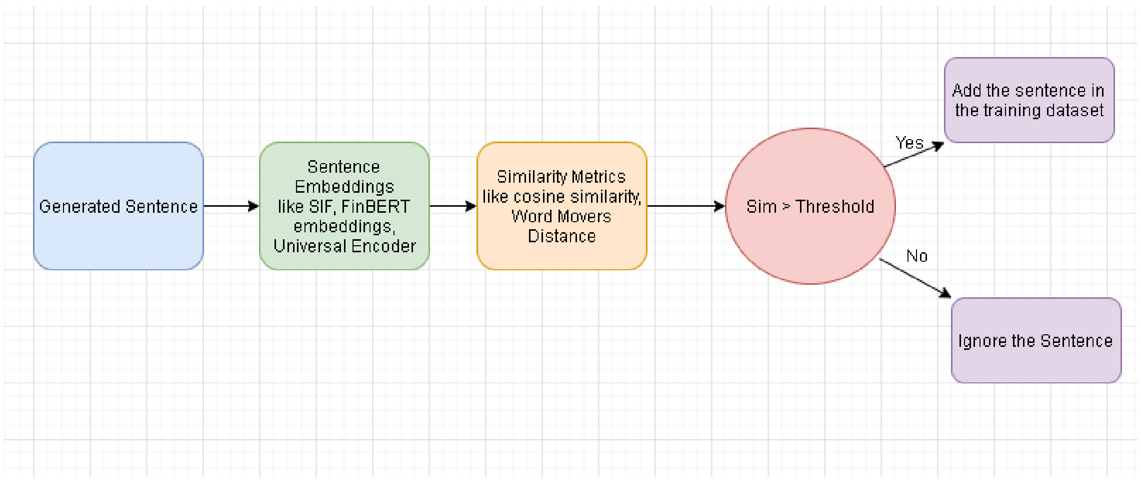

Generated Sentence
Sentence Embeddings like SIF, FinBERT embeddings, Universal Encoder
Similarity Metrics like cosine similarity, Word Movers Distance
Sim > Threshold
Yes
Add the sentence in the training dataset
No
Ignore the Sentence

Fig. 4.3: Post-Processing of Generated Sentences

Average of Word Embeddings

The easiest way to represent the sentence is by averaging the word representations (i.e word2vec, glove, fasttext,..) of words in the sentence. This method have lot of problems,

- Domain specific knowledge not been incorporated in the word embeddings, this can be tackled using word embeddings trained on finance data (refer to Sehrawat (2017))
- If there are lot of common words present in the both sentences so high similarity is always there between the two translated sentences.
- As this a simple average, even stopwords will have the **equal weightage**. So this dilutes the important words. This is **tackled using SIF** method.

Smooth Inverse Frequency(SIF)

SIF a weighting scheme to improve performance of sentence embeddings when compared to simple average of the word embeddings. When encoding a sentence, it is important to identify which words in the sentence are more significant. For example, if calculating the embedding of the sentence **"Is it automotive?"** the word **"Is"** does not add much meaning; looking for sentences or documents relating to the word "Us" will not yield any useful results for the original question. It's clear that **"automotive"** holds the most meaning in the question from a human standpoint, but how do you program a machine to identify that? SIF operates under the assumption that the most important words tend to also be used less frequently(Similar to tfidf). If you counted all the words in Wikipedia, the word "is" would most likely appear much more frequently than in "automotive". Weights of a word w are computed by a/(a + p(w)) where a is a parameter and p(w) is the word frequency of w, which can be estimated by scraping a large corpus. Then use **cosine similarity between input sentence and generated sentence**. there are few challenges using this method.

- Frequencies are computed from wikipedia data but we need to compute frequencies on financial data which computationally very expensive

Word Mover's Distance

Word mover's distance is a popular alternative to the simple average embedding similarity. The Word Mover's Distance uses the word embeddings of the words in two texts to measure the minimum amount that the words in one text need to **"travel"** in semantic space to reach the words of the other text. Word mover's distance is available in the popular Gensim library.

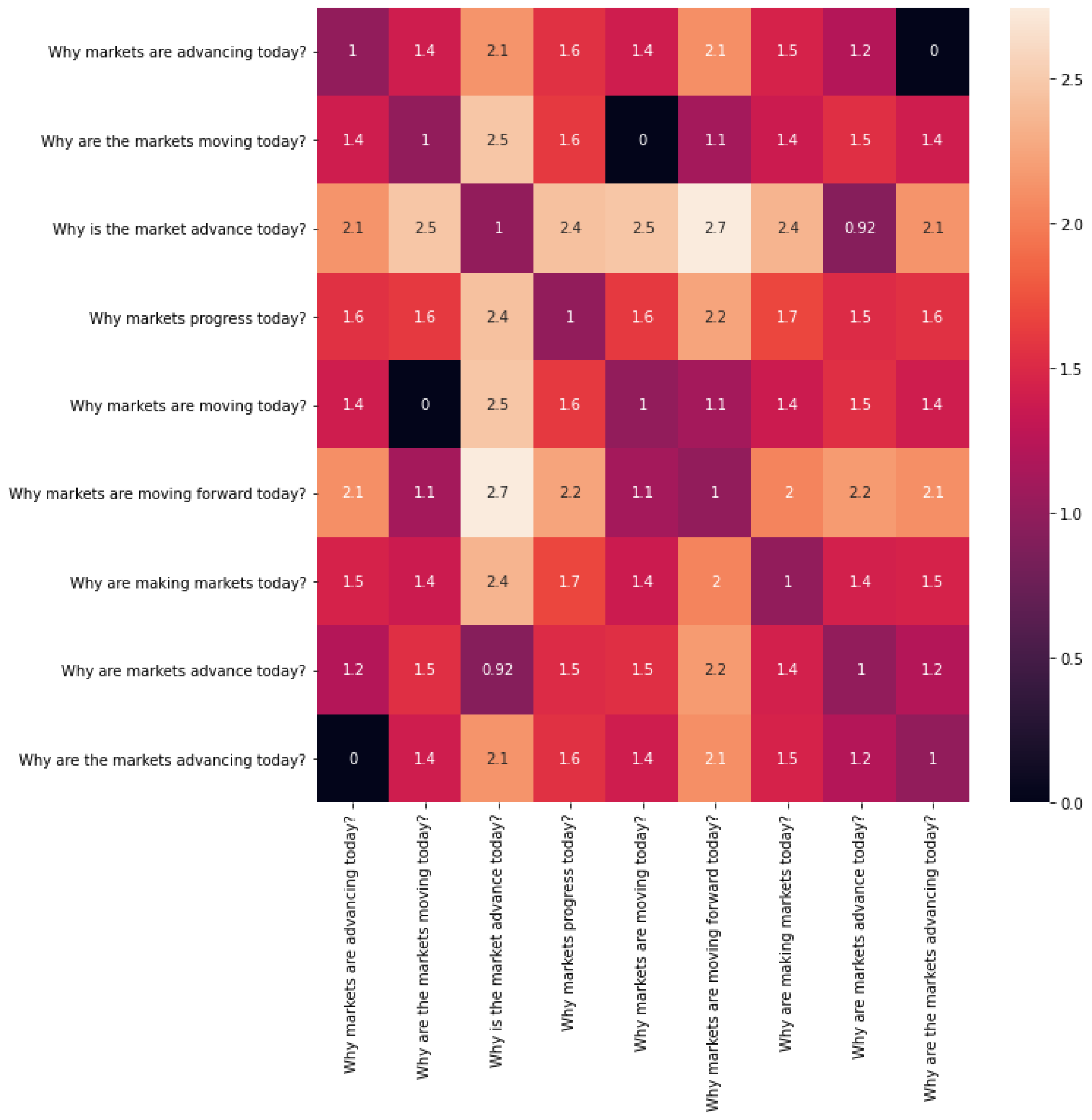


Fig. 4.4: Example of Word Movers Distance metrics

Observations: From the figure 4.4

- This is a **relative measurement**, so if given two sentences and one reference sentence, then asked if which of the two sentences is closer to the reference sentence. Then this method can be helpful. In absolute terms, this cannot be used i.e we cannot set a threshold to filter the generated sentences.

- This method can be used to identify the sentences which are exactly but different in the stopwords content.

Universal Sentence Encoder

This is inspired from the paper **Daniel Cer (2018)**, The models take as input English strings and produce as output a fixed dimensional embedding representation of the string. This is a transformer model which takes context into consideration. These models are famously used for sentence representation that will be used in downstream tasks like text summarizing, text similarity, sentimental analysis. Following figure**??** and figure4.6 shows how the completely unrelated sentences are having less similarity score and highly related sentences are having very high similarity score. Remember there are different exceptions where even this way of calculating similarity metric can also fail. For example, NASDAQ and full form of NASDAQ (i.e National Association of Securities Dealers Automated Quotations), then its similarity is less.

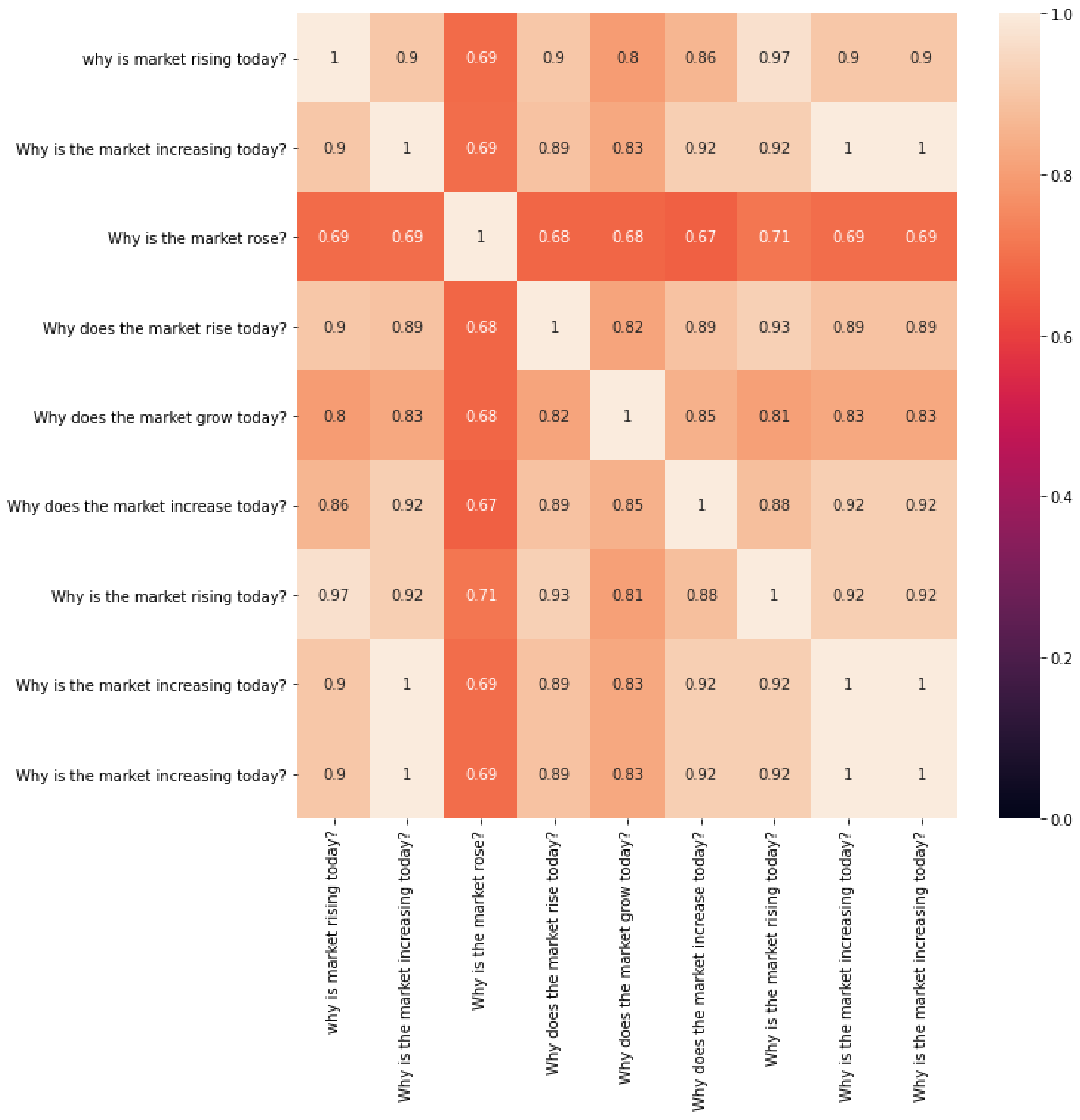


Fig. 4.5: Example 1 of Similarity using Universal Sentence Encoding.

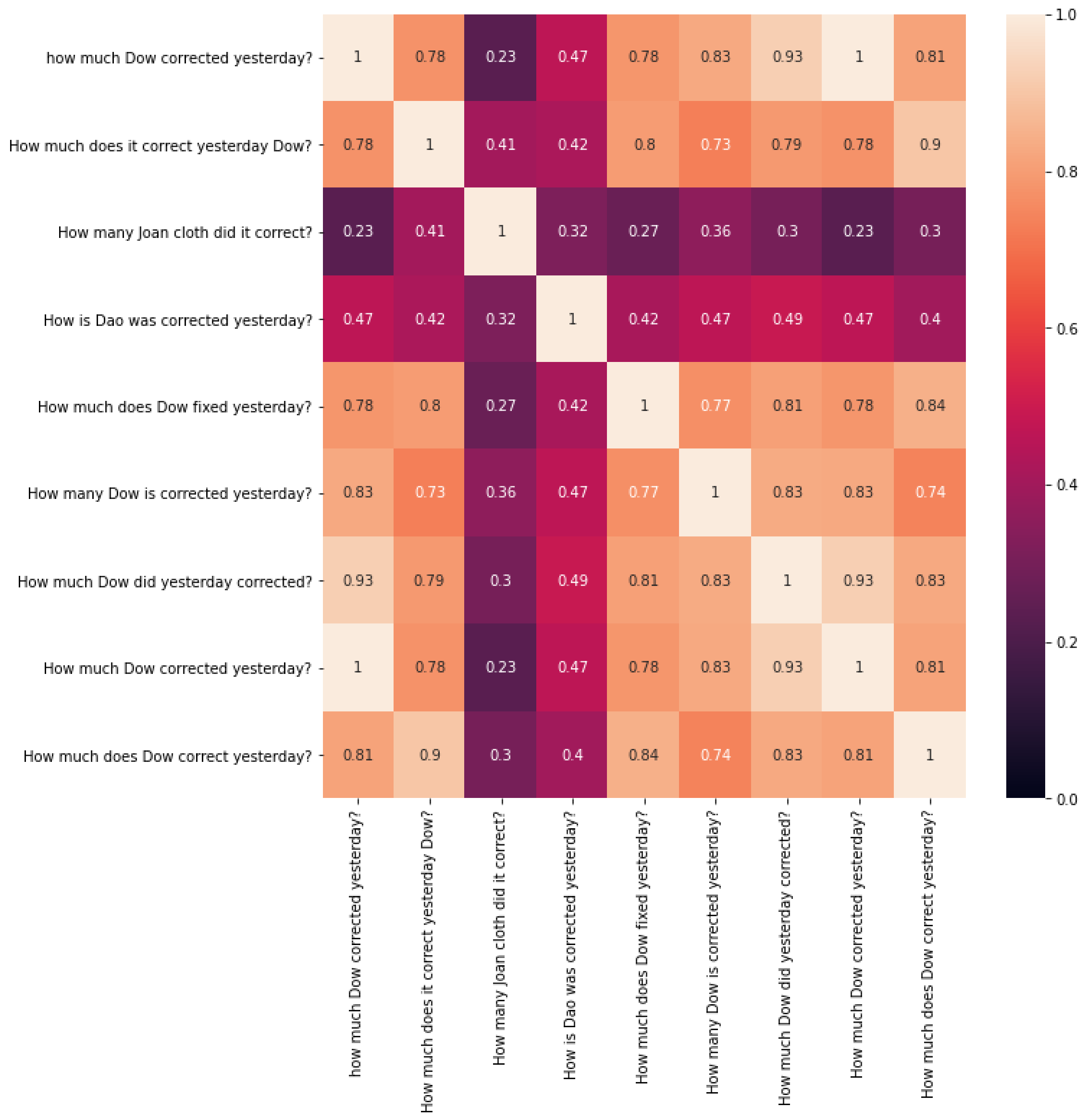


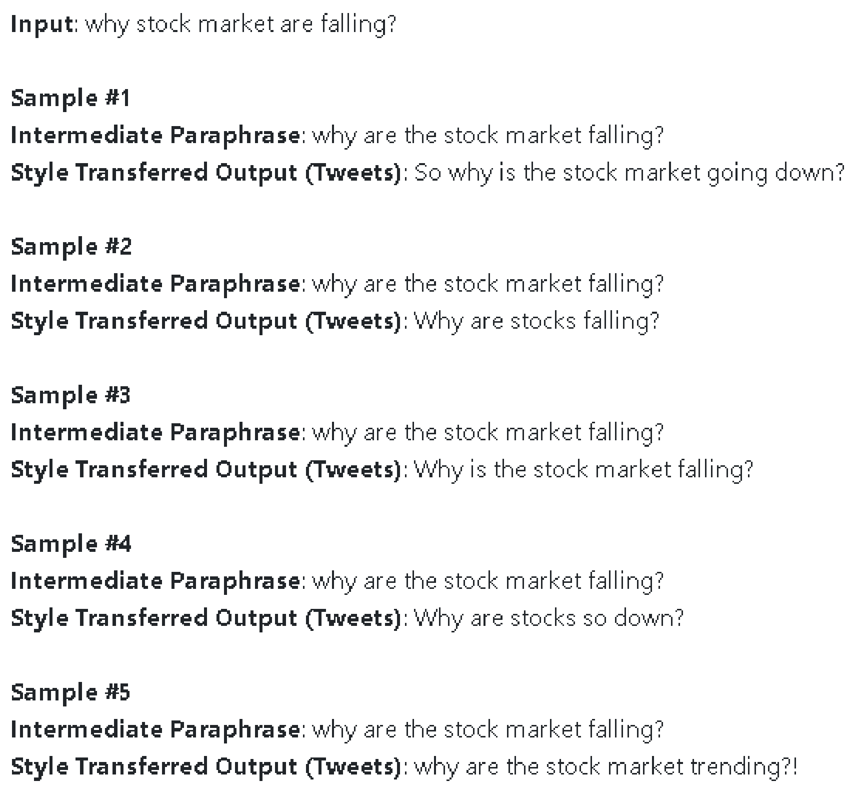
Fig. 4.6: Example 2 of Similarity using Universal Sentence Encoding

4.2 Synonymy and Antonym Replacement

This method is inspired from the paper **Igor A. Bolshakov (2004)**, as we know a sentence is called a paraphrase if two conditions are satisfied **1) Similar Intent, 2) Different Syntactic Structure**. Similar Intent can be achieved from many different ways. One commonly used method is to replace the words in the sentences with their corresponding synonym and antonyms. Now question can be asked, how we can get the synonym and antonym of a word? One simple way is from higher cosine similarity of two words(using word2vec, glove embeddings). But in this way, the domain knowledge is not incorporated in the method. So instead of general embeddings like word2vec, glove embeddings, we use domain specific embeddings which is trained 10K SEC filings (from the paper by **Sehrawat (2017)**).

4.3 Style Transfer on Finance Data

I was able to reproduce the model which is trained in the paper by **Kalpesh Krishna (2020)**. In this section, we will look at the outputs of this model on our financial data questions. From the below examples, we can observe that we can generate more paraphrases using this model.



Input: why stock market are falling?

Sample #1
Intermediate Paraphrase: why are the stock market falling?
Style Transferred Output (Tweets): So why is the stock market going down?

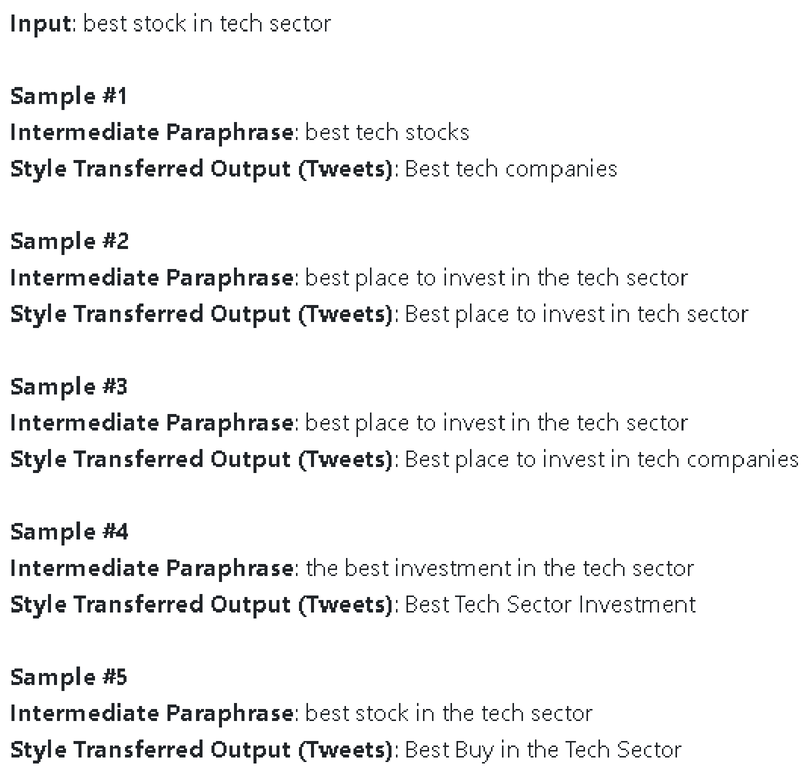
Sample #2
Intermediate Paraphrase: why are the stock market falling?
Style Transferred Output (Tweets): Why are stocks falling?

Sample #3
Intermediate Paraphrase: why are the stock market falling?
Style Transferred Output (Tweets): Why is the stock market falling?

Sample #4
Intermediate Paraphrase: why are the stock market falling?
Style Transferred Output (Tweets): Why are stocks so down?

Sample #5
Intermediate Paraphrase: why are the stock market falling?
Style Transferred Output (Tweets): why are the stock market trending?!

Fig. 4.7: Example 1 of Style Transfer Literature



Input: best stock in tech sector

Sample #1
Intermediate Paraphrase: best tech stocks
Style Transferred Output (Tweets): Best tech companies

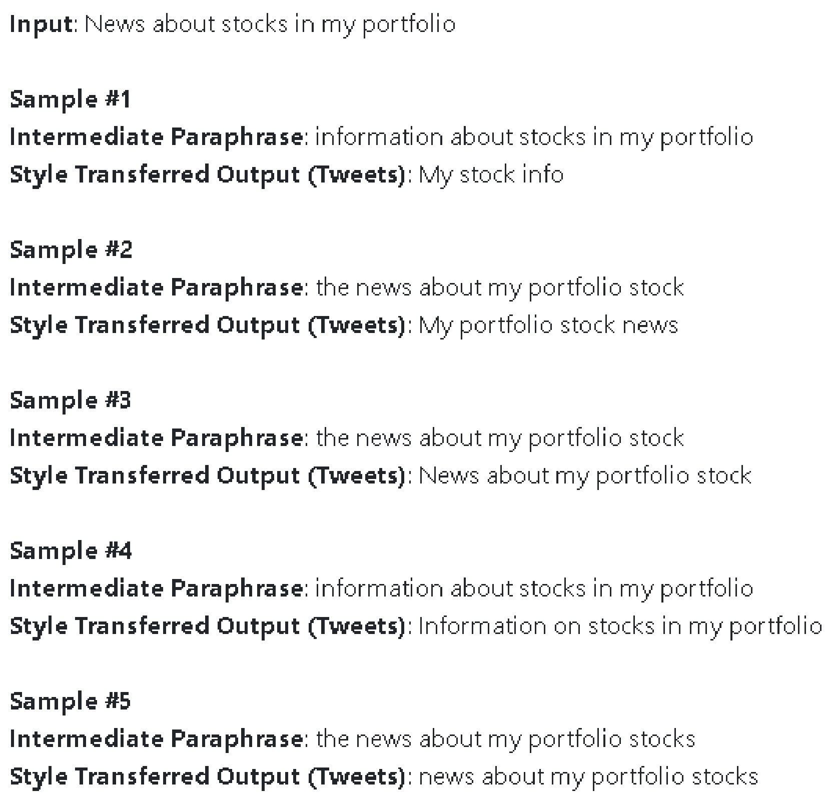
Sample #2
Intermediate Paraphrase: best place to invest in the tech sector
Style Transferred Output (Tweets): Best place to invest in tech sector

Sample #3
Intermediate Paraphrase: best place to invest in the tech sector
Style Transferred Output (Tweets): Best place to invest in tech companies

Sample #4
Intermediate Paraphrase: the best investment in the tech sector
Style Transferred Output (Tweets): Best Tech Sector Investment

Sample #5
Intermediate Paraphrase: best stock in the tech sector
Style Transferred Output (Tweets): Best Buy in the Tech Sector

Fig. 4.8: Example 2 of Style Transfer Literature



Input: News about stocks in my portfolio

Sample #1
Intermediate Paraphrase: information about stocks in my portfolio
Style Transferred Output (Tweets): My stock info

Sample #2
Intermediate Paraphrase: the news about my portfolio stock
Style Transferred Output (Tweets): My portfolio stock news

Sample #3
Intermediate Paraphrase: the news about my portfolio stock
Style Transferred Output (Tweets): News about my portfolio stock

Sample #4
Intermediate Paraphrase: information about stocks in my portfolio
Style Transferred Output (Tweets): Information on stocks in my portfolio

Sample #5
Intermediate Paraphrase: the news about my portfolio stocks
Style Transferred Output (Tweets): news about my portfolio stocks

Fig. 4.9: Example 3 of Style Transfer Literature

CHAPTER 5

NLU Pipeline

There are four stages to get the desired output as shown in figure[2.1]

- **Tokenization:** This splits the sentences into words called tokens.
- **Featurization:** In the case of Intent Classification, we are building a discriminative model, so we need features which differentiate between the intents. This is very crucial step in the NLU Pipeline.
- **Entity Extraction:** In this step, we try to extract as many entities as possible from the user-query. Here, I used different entity extractors which specialises in the given entities. Refer Section 5.4
- **Intent Classification:** This can be considered as "multi-class classification" task. We applied three classification models. 1) DIET Classifier(Transformer based Model), 2) Knearest Neighbors, 3) K-means clustering.

Following is the final pipeline for NLU task. We will discuss about the each component in the pipeline in detail.

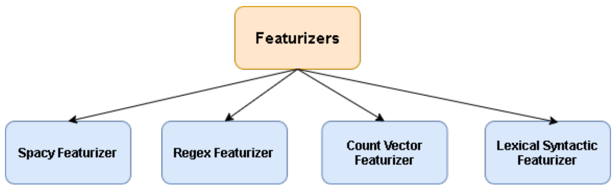


Fig. 5.1: NLU Pipeline

5.1 Tokenization

Tokenization is the task of splitting a text into meaningful segments, called tokens. Naive way of splitting a token is by using **"whitespace characters"**. This works well in most of the cases, but when there are sentences like "Hi, How are you?", In this case, tokenization outputs ["Hi,", "How", "are", "you?"]. So we use rule-based tokenizer available in Spacy. Procedure as follows: First, the raw text is split on whitespace characters, similar to text.split(' '). Then, the tokenizer processes the text from left to right. On each substring, it performs two checks:

(a) **Does the substring match a tokenizer exception rule?** For example, "don't" does not contain whitespace, but should be split into two tokens, "do" and "n't", while "U.K." should always remain one token.

(b) **Can a prefix, suffix or infix be split off?** For example punctuation like commas, periods, hyphens or quotes.

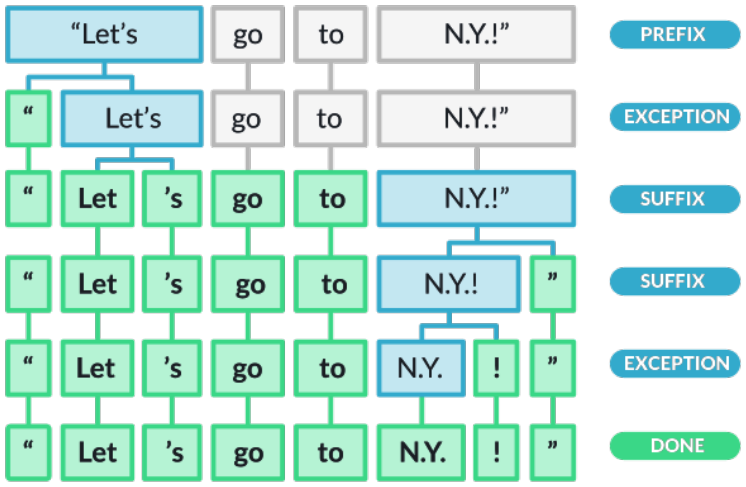


Fig. 5.2: Tokenization Example

If there's a match, the rule is applied and the tokenizer continues its loop, starting with the newly split substrings.

5.2 Featurizers

Feature is a machine readable representation that is used as input to a machine learning model. Common types of featurizers are:

- Bag of Words representation
- Labels(entity) or Tags(parts-of-speech)
- A numerical representation like a word embeddings

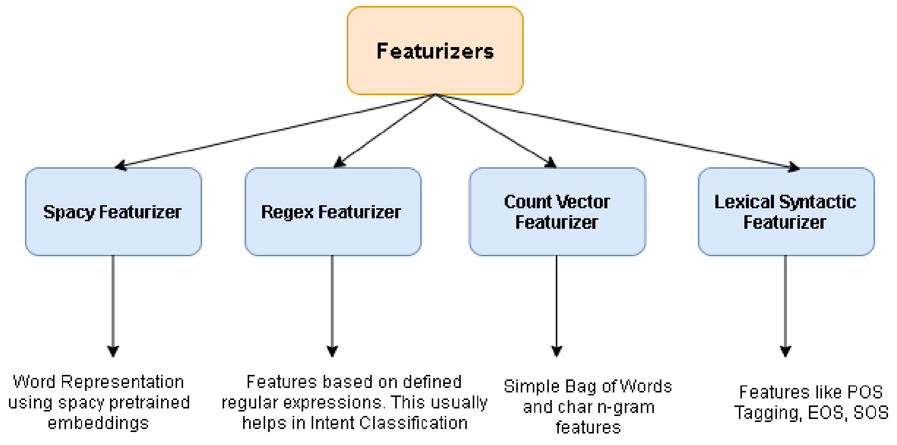

Featurizers
Spacy Featurizer
Regex Featurizer
Count Vector Featurizer
Lexical Syntactic Featurizer
Word Representation using spacy pretrained embeddings
Features based on defined regular expressions. This usually helps in Intent Classification
Simple Bag of Words and char n-gram features
Features like POS Tagging, EOS, SOS

Fig. 5.3: Featurizers

5.2.1 Spacy Featurizer

Word-embedding is a common way of incorporating general knowledge into the word representations. Spacy featurizer converts the sentence tokens into pre-trained word-embeddings. Note that word-embedding has both advantages and disadvantages.

Advantages :

- Better model performance with less training data required, this is because we are not training the embeddings from scratch.
- Faster training and iteration times

Disadvantages :

- We are limited to languages which have pre-trained word embeddings(In our project, this limitation is not an issue, we use english word-embeddings)
- Pre-trained word-embeddings do not cover domain specific words(We need to address this issue, as our application is in financial domain. One way is to train our supervised emeddings model but requires lot of training and computation)

5.2.2 Regex Featurizer

We can use regular expressions to improve intent classification by including the **RegexFeaturizer** component in the pipeline. When using the RegexFeaturizer, a regex does not act as a rule for classifying an intent. It only provides a feature that the intent classifier will use to learn patterns for intent classification. For example, we create few regular expressions then for each regex, a feature will be set marking whether this expression was found in the input, which will later be fed into intent classifier.

5.2.3 Lexical Syntactic Featurizer

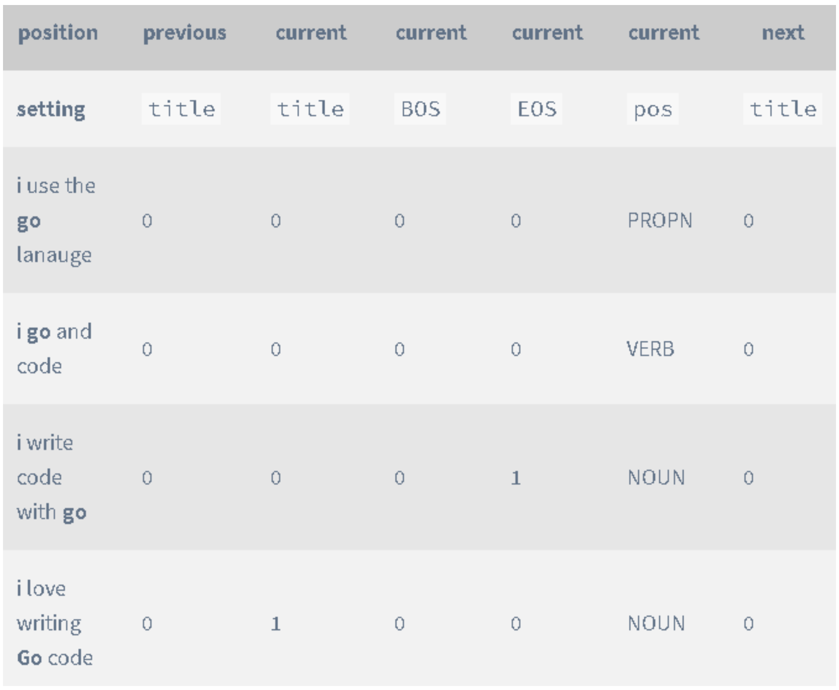
This featurizer extracts features from each of the tokens of the input like

- Is the token before written as a title?
- Is the current token written as a title?
- Is the current token at the beginning of the sentence?
- Is the current token at the end of the sentence?
- Parts-of-Speech of each tokens.

This featurizer is very important in many cases. For example, we are interested in detecting the programming languages as entities. In that case, the programming language **"Go"** can be very difficult to extract. Since the word **"go"** is most commonly used as a **verb**, there are many situations where a ML pipeline will have trouble correctly detecting it. If instead we knew up front that the word **"go"** was used as a **noun**, things might be a lot easier. Refer to figure [5.4]

5.2.4 CountVector Featurizer

This takes the input and outputs the standard bag of words(BOW) representation. This representations has some limitations like order of words in a sentence is not considered, sparse representation, and cannot handle spelling mistakes. To address some of this limitation we used n-gram model to take sequence of words into consideration and char n-gram model to handle spelling mistakes. For example, consider "running", if we mistakenly given input as "runnin", the BOW representation considers both these words as different. But in char 3-gram representation, running ->[run, unn, nni, nin, ing] and runnin -> [run, unn, nni, nin]. In this case almost all are matching.

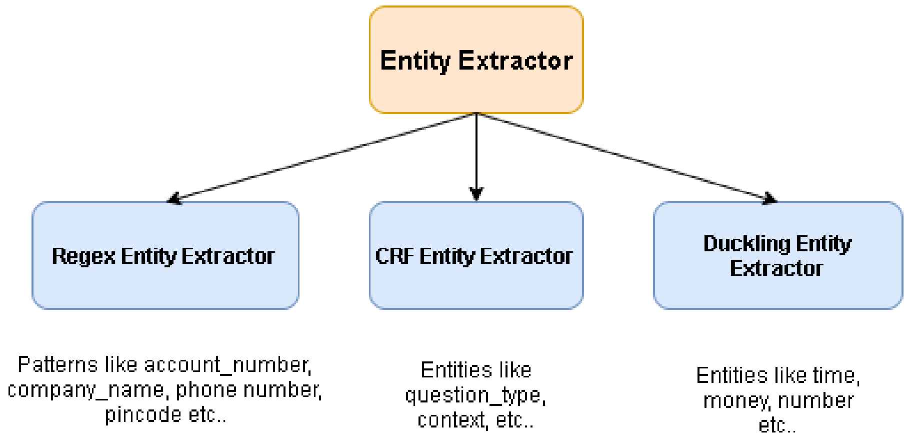


position	previous	current	current	current	current	next
setting	`title`	`title`	`BOS`	`EOS`	`pos`	`title`
i use the **go** lanauge	0	0	0	0	PROPN	0
i **go** and code	0	0	0	0	VERB	0
i write code with **go**	0	0	0	1	NOUN	0
i love writing **Go** code	0	1	0	0	NOUN	0

Fig. 5.4: Lexical Syntactic Featurizer Example

5.3 Entity Extractors

Following are the components in the entity extractors. We discuss in details in later subsections. Refer [5.5]



Entity Extractor
Regex Entity Extractor
CRF Entity Extractor
Duckling Entity Extractor
Patterns like account_number, company_name, phone number, pincode etc..
Entities like question_type, context, etc..
Entities like time, money, number etc..

Fig. 5.5: Entity Extractors

5.3.1 Regex Entity Extractor

If your entity has a deterministic structure. you can use regular expressions in our pipeline. For example, to extract entities like email, phone number, address, pincode and so on. For example, you could extract account numbers of 10-12 digits by including _d10,12 as regular expression. For entities like company name, person id, can be extracted using lookup table which uses regex matching algorithm. Refer to [5.6]

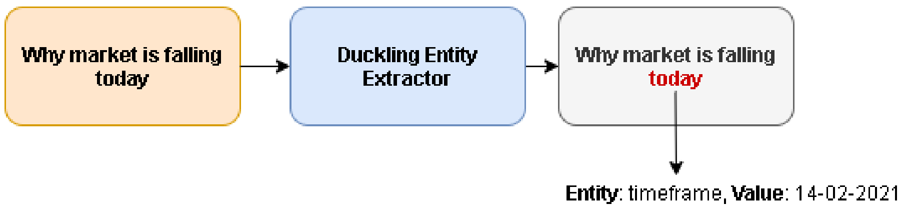

Why market is falling today
Duckling Entity Extractor
Why market is falling today
Entity: timeframe, Value: 14-02-2021

Fig. 5.6: RegexEntity Extractor Example

5.3.2 CRF Entity Extractor

For Entity Extraction, conditional random field (CRF)John Lafferty (2001) is a proven technique and has been used extensively. We discussed about CRF Modelling in section 3.1. Basically, CRF exploits the dependency with neighbouring words and its labels using undirected graphical model unlike MEMM where "Label Bias Problem" persists.

5.3.3 Duckling Entity Extractor

Duckling is an open-source entity extractor, built by Facebook. Here is the github link. This is used for extracting entities like time, money, number. This extractor significantly improved performace of the model, as duckling agnostic i.e it makes no assumption on the kind of data you want to extract, or the language. Following is the example [5.7]

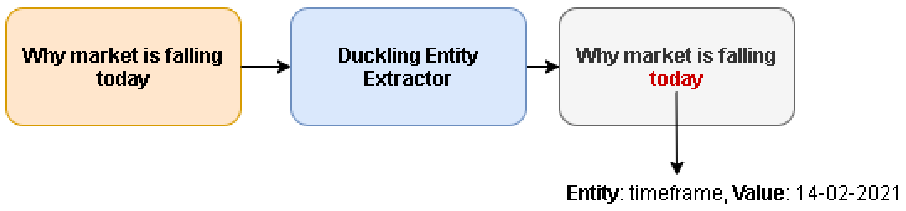

Why market is falling today
Duckling Entity Extractor
Why market is falling today
Entity: timeframe, Value: 14-02-2021

Fig. 5.7: Duckling Entity Extractors

5.4 Intent Classifiers

5.4.1 KMeans Clustering

Kmeans clustering is one of the famous **clustering technique**. This is **unsupervised** (i.e labels are not needed during clustering) and **non-parametric** method. This method is used intuitively when we need cluster of similar text, so this can be used in Intent Classification. We will apply this and analyze the results of Kmeans clustering in later sections. One of the main disadvantage of Kmeans clustering is we need to decide on K value, which is a hyper-parameter. This can be tackled using **elbow method**. There are other sophisticated clustering algorithms also.

5.4.2 KNearest Neighbours

The K-Nearest Neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm and also know **lazy learner**. KNN assumes that similar things exist in close proximity. In other words, similar things are near to each other. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness)— calculating the distance between points on a graph. There are other ways of calculating distance, and one way might be preferable depending on the problem we are solving. However, the straight-line distance (also called the Euclidean distance) is a popular and familiar choice. Since it is **non-parametric**, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). This is **memory intensive** algorithm i.e we need to store the training data during testing phase. This is computationally expensive but simple algorithm

5.4.3 DIET

DIET: Dual Intent and Entity Transformer is an architecture proposed in this paper Tanja Bunk (2020). Following is the schematic representation of DIET classifier

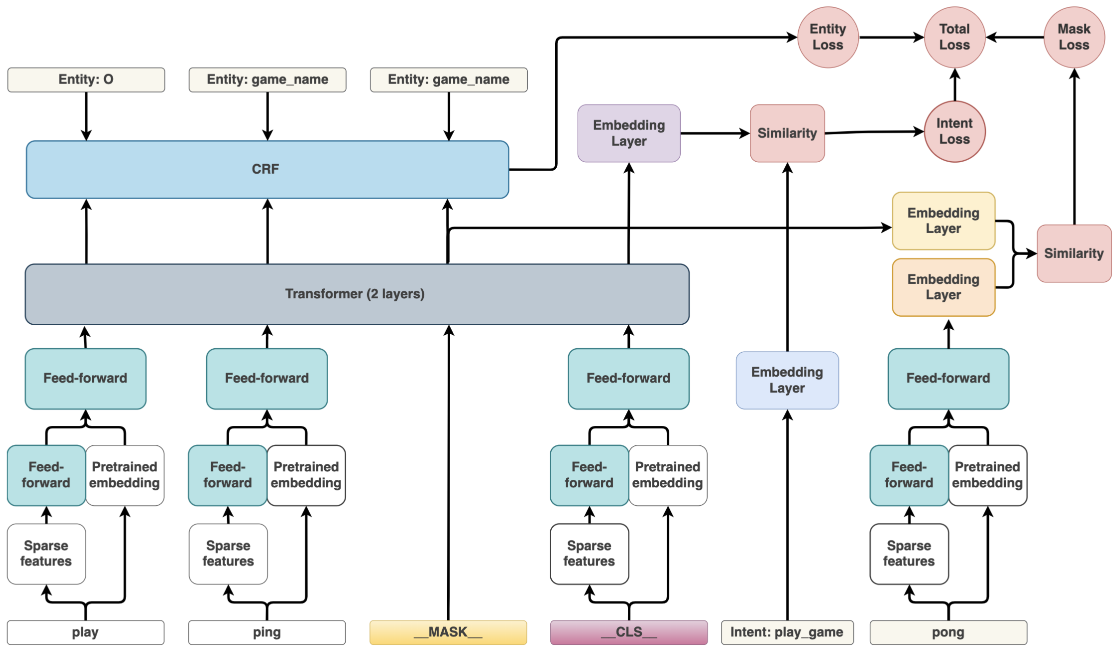


Fig. 5.8: A schematic representation of the DIET architecture. The phrase ”play ping pong” has the intent play game and entity game name with value ”ping pong”. Weights of the feed-forward layers are shared across tokens.Tanja Bunk (2020)

Consider an example , **Input: "Play Ping Pong"**. This belongs to the intent called **play_game**, this is true label of the given input. Entity should be extracted is game_name: "Ping Pong". Lets discuss the each component in the architecture[5.8] in more detail.

Component-1: Tokens

Consider the figure [5.9], In this the input token is **"play"**. There are two paths to it. Path-1 is pretrained embedding path, where we give the numeric representation of

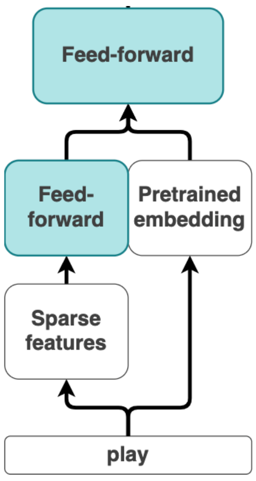


Fig. 5.9: Component-1: Tokens

the token(here "play") using word embeddings like Glove, Word2Vec, BERT. Path-2 is different from pretraining embedding. In this path, first we generate sparse features like one-hot encoding of the word **"play"**, Character n-gram representation like 1-gram, 2-gram and so on. This sparse features are concatenated and pass it on to the neural networks to generate dense features. At the end, we concatenate the feature from two paths and pass it on to the one more Feed-forward layer. Note we use dropout in the feed forward network to generalize better and avoid over-fitting. Finally, Output of this component is a vector of size d-dimension, where d is an hyperparameter to be tuned.

Component-2: Class Token

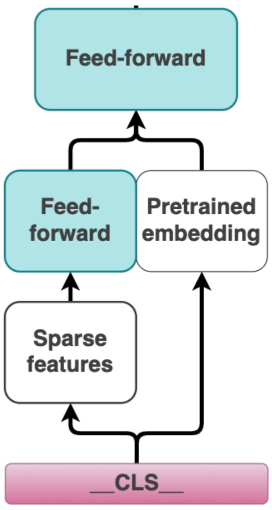


Fig. 5.10: Component-2: Class Token

Consider the figure [5.10], In this the **input** is the **whole sentence** i.e **"play Ping Pong"**. Here instead of taking each token in the input, we can take the whole sentence and summarize it into a numerical representation. Here also we have two paths just like Component-1. Path-1 is the pretrained embeddings, where we can average the word-embeddings of all the tokens or perform some sentence embedding techniques. Path-2 is very much similar to Component-1. Note that in the feed forward layers in Component-1 and Component-2 shares the weights.

Component-3: Mask

In DIET architecture, we not only learn the intent classification and entity extraction. We also try to mask few tokens and our goal is to predict the masked token, very similar to language model training. In this particular example, we are masking **"pong"**.

Component-4: Transformer Layer

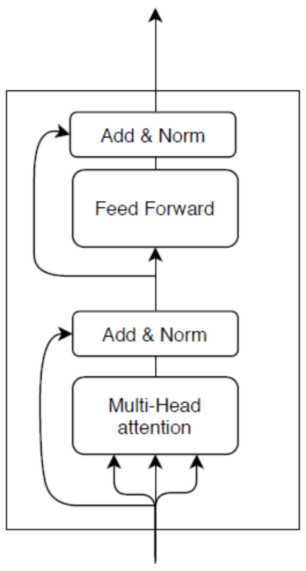


Fig. 5.11: Component-4: Transformer Layer, adopted from Alammar (2018)

In this paper Ashish Vaswani (2017), transformers have been introduced. In a simple way, Transformers can be thought of as alternate representation of the given inputs i.e if we input a size 512, we get an output of same dimension i.e 512. Here we use 2 layers of transformers. Note that the size of input is same as Output. The output of transfomer component will be sent to CRF (Conditional Random Field) for word-level labelling.

Loss Function

Total Loss = Entity Loss + Intent Loss + Mask Loss

DIET architecture has three different losses. They are:

(a) **Mask Loss:** From the Component-3, where we mask the token during training and try to predict the token, In our example, we masked **"Pong"**. So we consider a loss between predicted masked token and true masked token(i.e "pong")

(b) **Intent Loss:** From Component-2, where we represent the whole sentence into numeric vector. These two can be compared using similarity measures between Component-2 output and true intent(i.e play_game). This loss is called **Intent Loss**

(c) **Entity Loss:** As we know entity extraction task can be considered word-level labelling task. So we can have labelling loss for all the token labels which is called **Entity Loss**.

CHAPTER 6

Overview of Training Data

There are 9 intents and 8 entities in the overall NLU model. There are around 970 training samples in all the intents combined.

6.1 Entities

Entities are structured pieces of information inside a user message. For entity extraction to work, you need to either specify training data to train an ML model or you need to define regular expressions to extract entities based on a character pattern like email, phone number etc... Now lets have a look at the entities.

- **question_type:** This entity take values like why, which, when, etc..
- **context:** This entity take values like stock market, stocks, portfolio, sectors, etc...
- **context_name:** This entity take the company names.
- **asset_data:** This entity takes value like dividend, volume, price, Profit,Loss etc..
- **timeframe:** This entity captures the time from the user query.
- **direction:** This entity captures the direction like up, down, falling, crashing, etc.
- **relative_strength:** This entity captures the entities like good, better, best, bad, worse, etc..
- **action:** This entity captures the any action(i.e verb) in the query like "send me a notification", In this example "send" is the value of "action" entity.

Let's have a look at the intents

6.2 Intents

- **greet** This intent is for wishing by the user. This consists of terms like hi, hello, how are you?, etc.
- **deny** This intent is for rejecting reply or input from the user.
- **affirm** This intent is for acceptance reply or input from the user.
- **market_data** This intent is used if the user want to know the stock market overall. Some of the training examples look like "why stock market is falling today?", "what happened in the stock market yesterday?".
- **stock_data** This intent is very similar to market_data intent but only difference is here user want to know about the particular stock.
- **portfolio_data** This intent is used for queries regarding the portfolio of stocks.
- **sector_data** This intent is used for queries regarding sector information like pharma, technology, energy, etc.
- **ref_data** This intent is regarding queries like "which companies have their earnings announcement today?", "when will be the next dividend payout for AAPL?".

6.3 Balanced Dataset

Balancedness is data very important in multi-class classification. When it is imbalanced dataset, we need to perform sampling techniques. But in our case, our dataset is almost balanced. Refer to following figure 6.1.

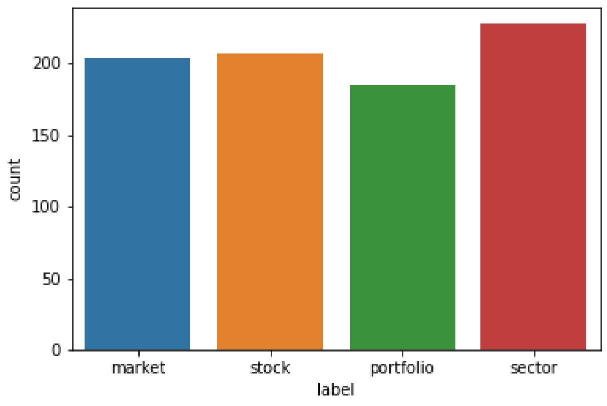


Fig. 6.1: Count plot of the four Intents

CHAPTER 7

Results

7.1 Validation Techniques

In this project, there are mainly **two challenges**. 1) **Lack of training dataset** to train NLU model, 2) There is **no benchmark dataset** to validate our different approaches. We **tackled** the lack of training dataset using different **data augmentations techniques** which are inspired from literature papers. But still we have a problem with validating our training data generation approaches. In this section, we will discuss **customized approach** I used to validate in this project.

We have initial dataset of size around 100 samples for 4 intents. **Motivation** of the following approach is **when we generate a sentence we want the generated sentence should have the same label as original sentence** i.e preserving the intent.

- we will split the dataset in the ratio of 60/40 i.e 60 samples in train data and 40 samples in test data. This is called **Out-of-sample testing**. Stratified splitting will give us 10 samples from each intent.
- We will use the train dataset to generate new training samples using different approaches like Back translation, Synonymy replacement and Style Transfer.
- Now we use an iterative method whether to include a generated sentence into data or not. Train KNN classifier on the train data(of size = 60). We chose K based on highest accuracy on the test data.
- Iterate over all the generated sentences. If generate sentence predicted label is same as original then add that sentence into train data and retrain the knn classifier, else store that sentence into a list(we will visit this later again)
- After iterating over all generated sentences, we will go through left over sentences whether they can be added into train data or not.
- **Stopping criteria** is if left over sentences are zero or no change in train data after a iteration.

7.2 KNN Algorithm for Intent Classification - Iterative Approach

Following table 7.1 are results obtained from following above iterative procedure to build the train data and tested on fixed test data. From table 7.1, we can clearly see that at **K value = 6**, we have **maximum test accuracy**.

K Value	Train Accuracy	Test Accuracy
2	0.95864	0.71176
3	0.97864	0.763489
4	0.93864	0.7322214
5	0.94864	0.798645
6	0.97038	0.82353
7	0.96556	0.77941
8	0.96368	0.76471
9	0.96403	0.76471
10	0.96054	0.76471
11	0.96424	0.76471
12	0.96255	0.79412
13	0.95865	0.79412
14	0.95507	0.77941
15	0.95408	0.76471
16	0.94416	0.75
17	0.94577	0.75
18	0.9457	0.73529
19	0.9429	0.72059
20	0.93808	0.72059

Table 7.1: Iterative Approach KNN results for different values of K

7.3 KNN Algorithm for Intent Classification - Standard Approach

Assumption: all the generated sentences from train dataset are considered in KNN algorithm i.e **generated sentences are having same Intents**.

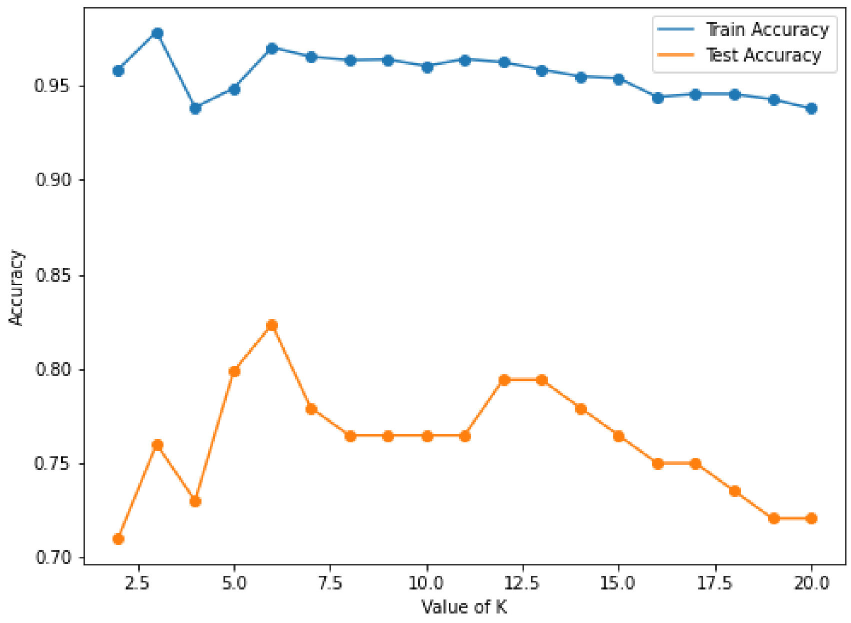


Fig. 7.1: Iterative Approach KNN results plot

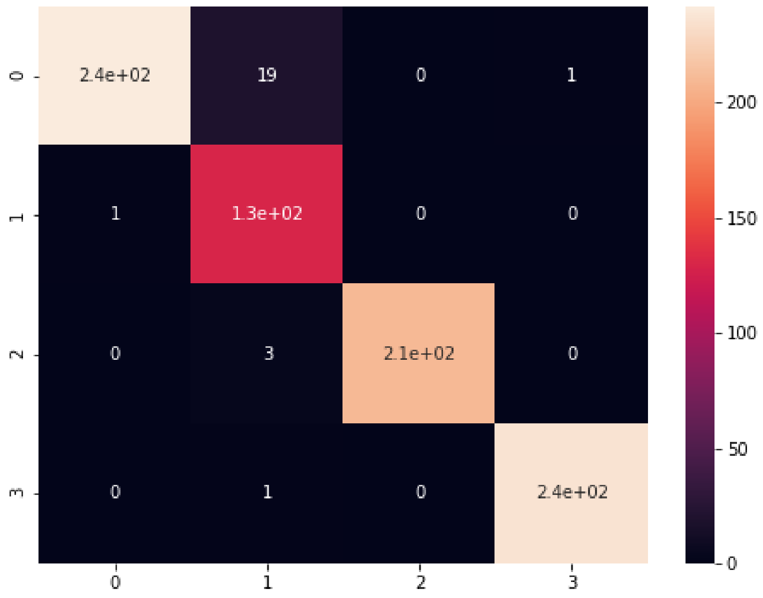


Fig. 7.2: KNN Train Confusion Matrix, K = 6

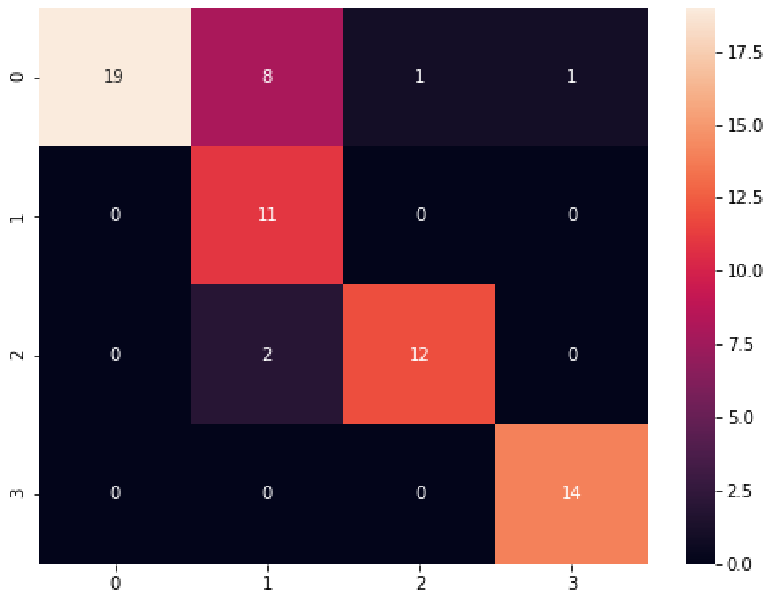


Fig. 7.3: KNN Test Confusion Matrix, K = 6

K Value	Train Accuracy	Test Accuracy
2	0.9733	0.82058
3	0.9818	0.80587
4	0.96238	0.80587
5	0.97573	0.83529
6	0.96117	0.83529
7	0.95874	0.82058
8	0.94175	0.82058
9	0.94782	0.82058
10	0.93811	0.77647
11	0.94417	0.79118
12	0.93932	0.77647
13	0.94053	0.77647
14	0.93325	0.79118
15	0.93325	0.77647
16	0.92476	0.76176
17	0.92597	0.76176
18	0.91626	0.76176
19	0.9199	0.74706
20	0.91748	0.73235

Table 7.2: Standard Approach KNN results for different values of K

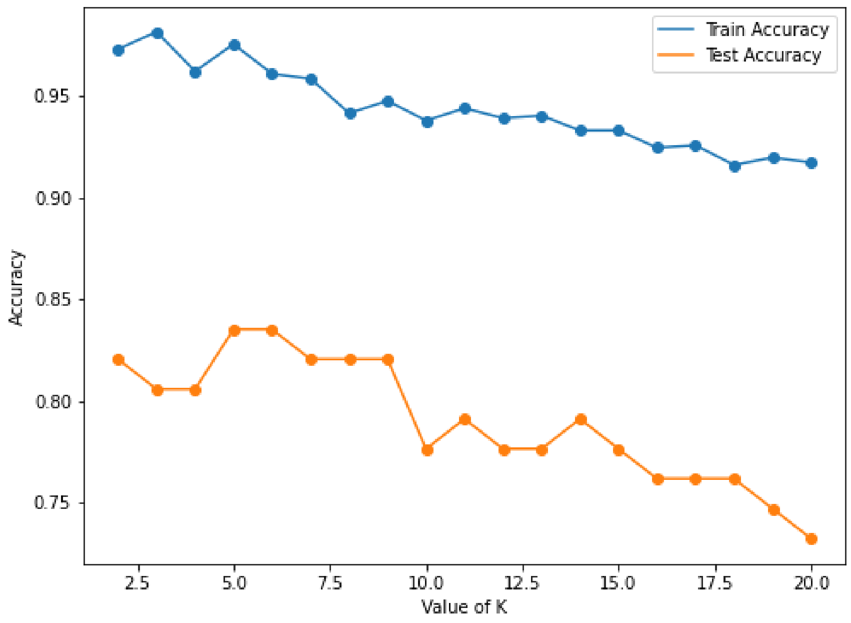


Fig. 7.4: Standard Approach KNN results plot

7.4 KMeans Clustering for Intent Classification

This section is about applying Kmeans clustering algorithm to make it a classification. Here we consider all the generated sentences in training dataset. Follow the procedure

- Fix value of K
- Apply KMeans clustering on train data to get K cluster centers. Each cluster can have combination of classes.
- Given a test datapoint, we find the which cluster it belongs to. The predicted label of its is majority class in that cluster.

Following table 7.3 has the results for varying K in Kmeans algorithm. At **K = 10**, we have **maximum text accuracy**.

Observation: From figures 7.7, 7.6, we clearly see that there is mis-classfication error is high for the label 0 and label 1. Most of the accuracy is lost only for those labels. This is very intuitive because, label 0 i.e market_data and label 1 i.e stock_data, these two intents are very close to each other. Only difference among them is

K Value	Train Accuracy	Test Accuracy
2	0.5182	0.48529
3	0.72573	0.63235
4	0.73058	0.67647
5	0.67961	0.66176
6	0.69417	0.66176
7	0.7051	0.70588
8	0.72209	0.64706
9	0.77306	0.76471
10	0.80097	0.79412
11	0.73058	0.70588
12	0.7318	0.64706

Table 7.3: KMeans results for different values of K

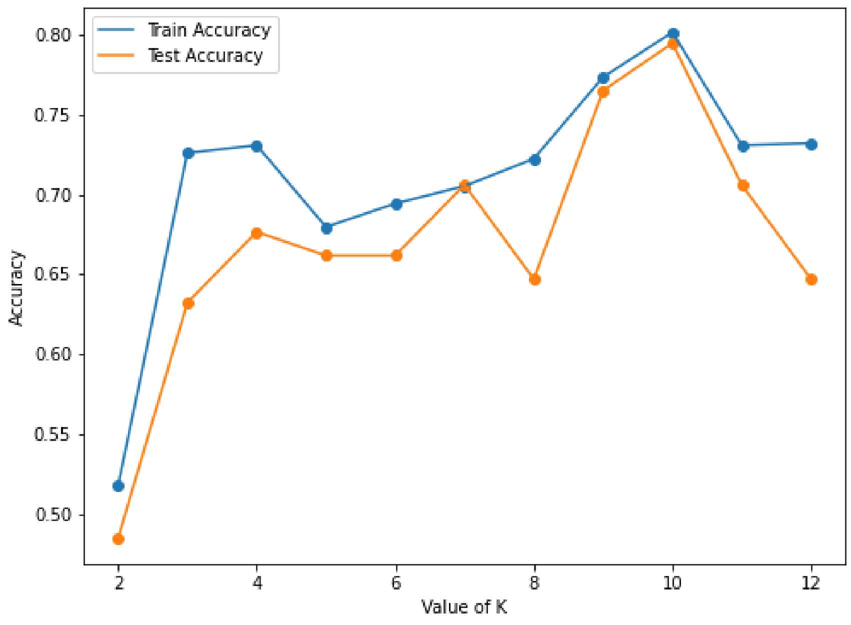


Fig. 7.5: Standard Approach KMeans results plot for different values of K

stock_data intent expect context_name entity i.e company name (stock name). So we can additionally pass our entity extraction results to classification and finalise the result based on entity extracted values. This is one of the reason entity extraction and intent classification are inter-related tasks.

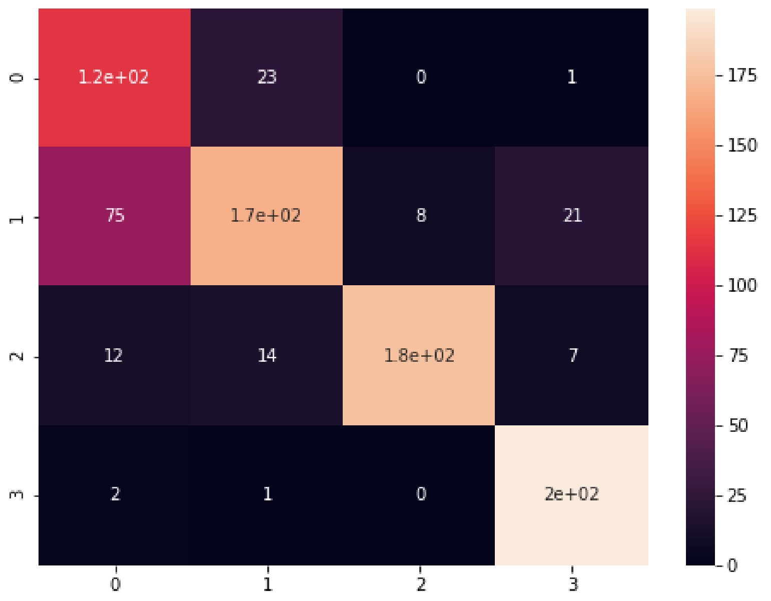


Fig. 7.6: KMeans Train Confusion Matrix, K = 10

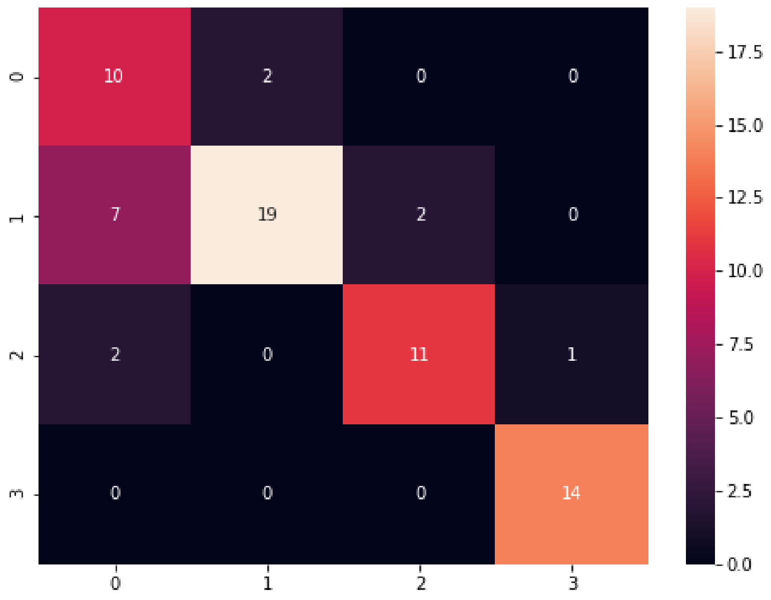


Fig. 7.7: KMeans Test Confusion Matrix, K = 10

CHAPTER 8

Conclusion and Future Work

8.1 Conclusion

In this project, we built the NLU component in chatbot from scratch. First, we discussed about various research methods to generate training dataset for training NLU component. We have briefly discussed about data generation using **Back translation, Synonymy replacement and Style Transfer**. We saw few examples in each methods. Among all the methods, Style Transfer is performing very impressive. We started with around **80-100** samples, after training data generation we have sample **size of 900**. In NLU we have two tasks, 1) Entity Extraction: For this task, As we know for entity extraction we need huge training data to get good performance. So we used pre-defined entity extractors like **duckling, regex, trained entity extractor using Conditional Random Field method**. 2) Intent classification: In this task, we discussed how iteratively we can filter the generated sentences. Here we mainly used **KNN classifier**, as this is non-parametric and very intuitive approach in this project. We performed some clustering using **KMeans** , also used as classification algorithm and visualized through **T-SNE approach**. We observed that how **market_data and stock_data are very similar** but we can differentiate these two intents using entities present i.e In the market_data class, we do not have context but in the stock_data class we have **context_name i.e company name**. From this we can say that entity recognition and intent classification are **inter-related tasks**. Finally, Performance of Intent classification is also dependent on the performance of entity recognition. Our best model performance is **train accuracy close to 96%** and **test accuracy around 83.5%** but there is huge difference between train and test accuracy in KNN algorithm. Incase of KMeans we have train and **test accuracy around 80%**, so **KMeans as classifier model** can be more **reliable than KNN model**.

8.2 Future Work

- In the paper Kalpesh Krishna (2020), we have lot of parameters to be tuned and highly sensitive to those parameters. So further we can work on analysing how the parameters are affecting the output.
- We built entity recognition model and Intent classification model separately. But these two tasks highly inter-dependent. So inspired from the paper Tanja Bunk (2020), we can built a single model. But DIET is a transformer model so it requires huge data to train. There are other literature availble to train joint models for both tasks.
- We used embeddings from Sehrawat (2017), this is trained on just SEC filings. But we need embeddings which are closer to conversations, i.e we can refer the paper Matthew Henderson (2020) for building domain-specific embeddings and building own pre-trained model. But this requires lot of computational resources.

REFERENCES

1. **A. McCallum, F. P., D. Freitag** (2000). Maximum entropy markov models for information extraction and segmentation,. *ICML*, 591–598.

2. **Alammar, J.** (2018). The illustrated bert, elmo, and co. (how nlp cracked transfer learning),.

3. **Ashish Vaswani, N. P. J. U. L. J.-A. N. G. L. K. I. P., Noam Shazeer** (2017). Attention is all you need,.

4. **Daniel Cer, S.-y. K. N. H. N. L. R. S. J. N. C. M. G.-C. S. Y. C. T. Y.-H. S. B. S. R. K., Yinfei Yang** (2018). Universal sentence encoding.

5. **Igor A. Bolshakov, A. G.** (2004). Synonymous paraphrasing using wordnet and internet.

6. **John Lafferty, F. C. P., Andrew McCallum** (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data,. *ICML*.

7. **Jonathan Mallinson, M. L., Rico Sennrich** (2017). Paraphrasing revisited with neural machine translation.

8. **Kalpesh Krishna, M. I., John Wieting** (2020). Reformulating unsupervised style transfer as paraphrase generation.

9. **Kuan-Hao Huang, K.-W. C.** (2021). Generating syntactically controlled paraphrases without using annotated parallel pairs.

10. **M. Jeong, G. L.** (2008). Triangular-chain conditional random fields,. *IEEE Transactions on Audio, Speech, and Language Processing*, 1287–1302.

11. **Matthew Henderson, N. M. P.-H. S. T.-H. W. I. V., Iñigo Casanueva** (2020). Convert: Efficient and accurate conversational representations from transformers,. *Computation and Language*.

12. **Mingbo Ma, L. H. B. X. B. Z., Kai Zhao** (2017). Jointly trained sequential labeling and classification by sparse attention neural networks,.

13. **Puyang Xu, R. S.** (2013). Convolutional neural network based triangular crf for joint intent detection and slot filling,. *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop*.

14. **S. Hochreiter, J. S.** (1997). Long short-term memory,. *Cambridge, MA, USA*.

15. **Sehrawat, S.** (2017). Learning word embeddings from 10-k filings for financial nlp tasks.

16. **Tanja Bunk, V. V. A. N., Daksh Varshneya** (2020). Diet: Lightweight language understanding for dialogue systems,. *Rasa*.

17. **Tom Hosking, M. L.** (2021). Factorising meaning and form for intent-preserving paraphrasing.

18. **Varghese Akson Sama, Y. V. K. B. G. N., Sarang Salehaa** (2020). Bidirectional lstm joint model for intent classification and named entity recognition in natural language understanding,. *International Journal of Hybrid Intelligent Systems,*, 13–23.