Topic Detection for Customer Support Queries

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Bachelor thesis

Degree Program in Business Information Technology
or Computer Applications
spring, 2023

Liam Huyberechts

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Author Liam Huyberechts
Subject Topic Detection for customer support queries
Supervisors Tommi Lahti

Abstract
Year 2023

ABSTRACT

This study investigates the efficacy and utilization of artificial intelligence in the customer service sector by implementing topic detection for customer support queries to identify underlying topics within past questions.

Most importantly, findings of the efficacy of such a system and the prerequisite work needed to implement it are reflected in the thesis. As well as the potential improvements and issues which could arise from its implementation, namely, the need to collect large amounts of quality data.

With reference to books, scientific papers, and documentation – the findings and implementation of a topic detection system are presented, along with the technical understanding and potential influence topic detection can have with regards to customer support.

Finally, the development results display the testing methodology and metrics used when assessing the use of an implementation of topic detection for customer support queries.

Keywords Natural Language Processing, Artificial Intelligence, Customer Support, Topic Modelling
Pages 36 pages and appendices 6 pages

Contents

List of Abbreviations

AI	Artificial Intelligence
CSV	Comma Separated Value
CTM	Correlated Topic Model
LDA	Latent Dirichlet allocation
LSA	Latent Semantic Analysis
ML	Machine Learning
MLOPS	Machine Learning Operations
NLP	Natural Language Processing
NMF	Non-negative Matrix Factorization
SVD	Singular Value Decomposition

List of Figures

Annexes

Annex 1 Material management plan

Annex 2 Diary

Annex 3 Bitext data sample

Annex 4 Data Collection File

Annex 5 Model Creation and training

Annex 6 Model Testing and Results

1 Introduction

With the advent of easy to use and widely accessible tools for the creation of applications and the business opportunities that come out of that, an often-overlooked aspect of is the quality of the customer support within these businesses, which can often be the differentiator between low and high customer satisfaction. AI is set to play a large role in the development of better and more accessible customer support solutions.

Artificial Intelligence has seen a major explosion in popularity and implementation in recent years. Due to both technological progress and funding, major projects across the tech space have seen milestones reached at increasing intervals. These projects hope to influence and improve the experiences had with large sectors of the technology industry, as well as day-to-day products available.

The complexity of customer support structures and costs in resources and time related to them has led to a large number of companies outsourcing this service to regions which offer it at more competitive costs, which is the most important aspect of outsourcing (Zulaykho et al., 2022, p. 1571). At the same time, it has led to increasing developments in automated customer support services such as chatbots.

As the prevalence of artificial intelligence increases in use cases throughout the customer journey, from advertising to after-sale support. Utilizing this technology has become increasingly important in relation to efficiency within a business environment and the customer support sector specifically.

This thesis aims to answer the questions: “How to implement an AI topic model for identifying specific topics within a set of documents?” and “What is the impact of topic detection with regard to responding to customer support queries?”

2 Customer support and artificial intelligence

This section will dive deeper into what customer support is, how it differs from customer service, and how artificial intelligence has influenced this aspect of the customer experience and its impact on efficiency. Up until this point, only customer support has been mentioned. Customer support and customer service are often used interchangeably. Moving forward in this chapter, customer support will be discussed in the context of being a substructure or smaller part within customer service.

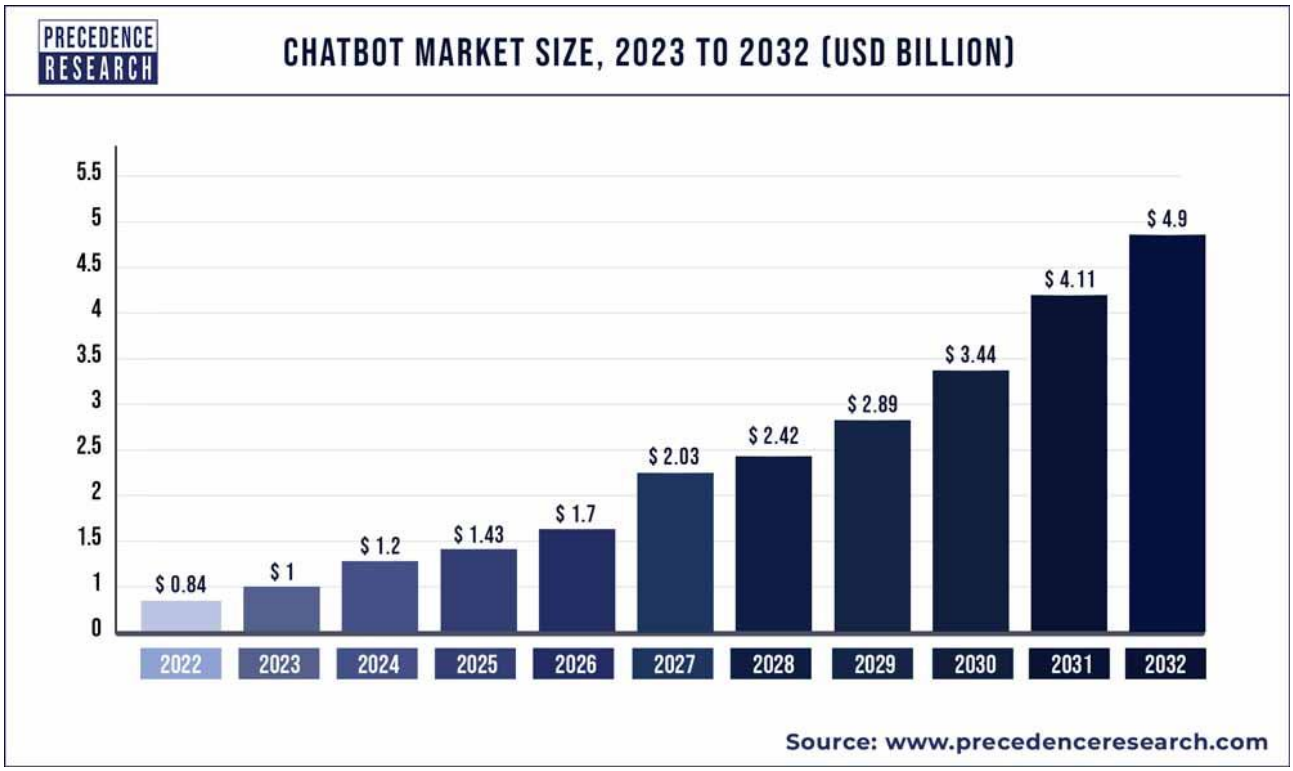
2.1 Customer support

There is a distinct difference between customer support and customer service. A customer support employee will generally deal with customer after a product has been sold, meaning they are directly responsible for the use and operation of the product, while a customer service rep would aid or assist a customer in the procurement of a product. The role of customer service is to benefit the customer experience while customer support is merely transactional (Chambers, 2022).

Customer support is a section within customer service and an important aspect of customer satisfaction. Among many other things, the internet has vastly expanded the avenues for customer and businesses interactions. The main avenues for customers to communicate with a company are: in-person interactions, phone calls, emails, and engagement through social media (Leonard, 2018).

With this increase in the ways in which a customer can communicate with customer support there are many more opportunities for AI to be used in these support channels. The use of AI in the form of chatbots alone is rising rapidly with a global market size of USD 0.84 billion in 2022 and expected to grow to around 4.9 billion in 2032 (*Chatbot Market Size To Hold USD 4.9 Billion By 2032*, n.d.).

Figure 1 Chatbot market size



Note. (*Chatbot Market Size To Hold USD 4.9 Billion By 2032*, n.d.)

2.2 Artificial Intelligence

This section will briefly cover the history and progression of Artificial Intelligence in computing.

2.2.1 A Brief History of Artificial Intelligence

The question of true intelligence has been asked throughout history. Artificial intelligence can be defined as the study and development of computer systems that can copy intelligent human behavior While the debate about whether or not true human intelligence can be applied to something artificial is still ongoing to this day, scientists past and present have tried to understand and implement it in computers. According to Russel and Norvig, there are four approaches that have been followed: Thinking Humanly, Thinking Rationally, Acting Human, and Acting Rationally (Russell & Norvig, 2009, p. 2).

The birth of AI itself can be linked back to classical philosophers who pondered over the systems of the human mind and to what extent our knowledge is our own. The term Artificial Intelligence, however, was only officially coined at a conference in Dartmouth College in 1956 (Lewis, 2014). Heinlein & Kaplan describe the history and progression of AI in four seasons: AI Spring as The Birth of AI, AI Summer and Winter as The Ups And Downs of AI, and the AI Fall as The Harvest of AI.(Haenlein & Kaplan, 2019).

2.2.2 Natural Language Processing

Natural language processing (NLP) is a part of AI that deals with the similarities between human natural language and a computer's ability to understand it. NLP combines computational linguistics – modelling human language with rules – and AI to better process and 'understand' human language (*What Is Natural Language Processing?*, n.d.).

As humans we can look at a word, phrase, or document and understand its underlying meaning based on the context it is portrayed in. Computers on the other hand do not have the capacity to formulate these contextual clues without being programmed. This is the problem that NLP tackles.

NLP lies in the space between human language and computer understanding of that language. This space can be traversed in both ways, for example, one way would entail the computer understanding something that human language represents such as a command or question. The other way would be the creation of a certain piece of language by the computer such as the summarization of a document into its own text. These two different 'flows' are called Natural Language Understanding (NLU) and Natural Language Generation (NLG) respectively (*NLP vs. NLU vs. NLG*, 2020).

2.2.3 Supervised learning versus unsupervised learning

The two major types of approaches to machine learning models are supervised learning models and unsupervised learning models. With each approach having its own strengths and weaknesses, it is important to understand which one to pick for any machine learning project.

Supervised machine learning takes both labelled input and output in order to train. This data is prepared by someone such as a data scientist before the training begins. This type of machine learning model is often used to classify new and unseen data and predict outcomes based on the labelled data it 'learned' from (Seldon, 2022).

Unsupervised machine learning on the other hand takes in unlabelled data and immediately tries to interpret or group any trends to 'understand' the data. This allows the model to immediately take in large amounts of data without being prepared and is often used in understanding the relationships between different sets of data such as recommendations for music (Seldon, 2022).

2.2.4 What is topic modelling?

Topic modelling is a machine learning technique that utilizes unsupervised learning for scanning and categorizing topics within documents and phrases by analyzing them for patterns to show an underlying characteristic of the document or phrase in question.

Most machine learning approaches will have many different methods of implementation. Topic modelling is no different and can be classified broadly into two main categories: probabilistic approaches and non-probabilistic approaches.

The probabilistic approach to topic modelling is based on the idea that each document in a corpus or set of documents is a mix of different topics where each topic is a distribution of words – meaning each word in a document will relate to a specific topic. In the same way each document is made of a distribution of topics.

Latent Dirichlet Allocation (LDA) is the most popular probabilistic approach for topic modelling and is the approach taken in this project. LDA assumes that each text is a combination of set latent topics and a topic is a distribution of words (Jelodar et al., 2019).

A Correlated Topic Model (CTM) varies from LDA in that it allows the number of topics to be unbounded rather than being set or fixed. Meaning each the topics are not necessarily independent of each other. A CTM models a document's words from a mixture model. These

mixture components are shared by all the documents in the set of documents. A CTM allows multiple topics to be exhibited in each document with different proportions(Blei & Lafferty, 2007).

These are two probabilistic topic models which differ from non-probabilistic models in that they do not base their methods on matrix factorization techniques. Matrix factorization is a filtering method to identify relationships between ‘user’ and ‘entity (Chen, 2020). Much like probabilistic topic models, there are various non-probabilistic approaches such as Non-negative Matrix Factorization (NMF), Singular Value Decomposition (SVD), and Latent Semantic Analysis (LSA).

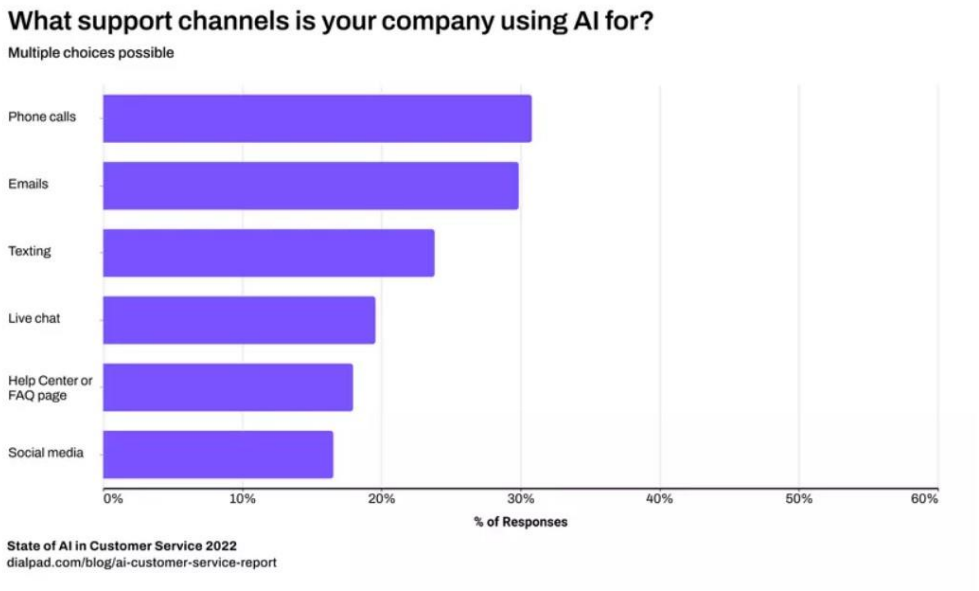
2.3 Artificial Intelligence within the customer support sector

A large aspect of customer service comes from the utility of after-sale customer support. Throughout the years, the internet has become a more important part of this sector of business operations. Improvements in technology and computing power have allowed artificial intelligence systems and applications to become more integrated into the process of business. In many industries, artificial intelligence has become a major part of the customer journey when it comes to their decisions, whether that is visible or not. From conceptualization of products to marketing, it will be difficult to find a company which has not directly or indirectly used AI somewhere in their business pipeline.

The involvement of AI in the customer support sector, specifically AI chatbots, has seen a rise in popularity since 2018 (Piercy, 2021). In the near future, as more of the population will have easy access to social media and gain more trust in the use of AI within customer support, being able to correctly analyse and categorize customer support questions and data will become an invaluable resource to staying on top of any potential problems a company may have (*The State of AI in Customer Service Report 2022*, n.d.).

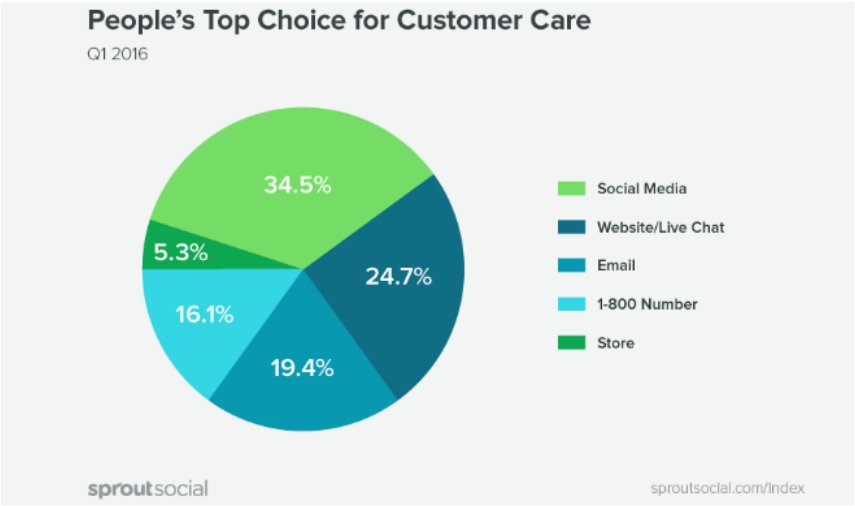
According to customer service agents, as of 2022, AI is mainly being used for phone calls and emails, as seen in Figure 2 Customer support channels making use of AI.

Figure 2 Customer support channels making use of AI



In Figure 3 Customer support channels 2016, over 30% of people preferred social media as a choice for customer support. Which was an increase of 18% since the year prior (*90% of Social Media Users Reach Out to Retailers | Porch Group Media*, n.d.).

Figure 3 Customer support channels 2016



This disconnect between the channels in which AI is utilized and the preferred channel which people seek is not necessarily indicative of a lagging customer support industry, but possibly just that AI for phone calls require more resources to operate.

3 Topic modelling and its use cases

This section will cover the potential use cases of topic detection within the customer support sector that are available as well as the intended use case of the practical part of the project which is to be implemented.

3.1 Redirection

The current most common use of AI within customer support is the redirection of customer inquiries to the correct customer support agent (*The State of AI in Customer Service Report 2022*, n.d.). Topic modelling allows for the questions to be categorized into a specific topic and then forwarded to the correct person who handles that particular type of request. This can be seen as a type of 'understanding' of the customer question.

One potential example of a use case of this method would be to create a topic detection system within a bank's customer support structure to redirect questions coming from a customer; with the idea of aiding the process of customer support staff in answering those questions, by which a bank could increase the speed and efficiency of responses to customer support queries coming from various sources.

3.2 Data Labelling

Having access to properly labelled data is a major part in supervised machine learning which is previously explained in chapter 2.2.3. An alternative use case for a topic model is to label incoming data. Once the model has been trained and tested sufficiently to output the accurate and desired topic, it could label this raw data for later use on different language models (Russell & Norvig, 2009, p. 695).

For example, if the bank is confident in its trained topic model and need a large amount of labelled customer support questions for a supervised learning language model, the topic model could be

incorporated into that pipeline to automate what would normally be an intensive and tedious task.

3.3 Topic analysis and keyword extraction

The main implementation of this thesis project is to use topic modelling for analysis and keyword extraction of a large document of past customer support questions. This means that out of all the questions asked, the major themes and topics will be gathered by the topic model and displayed for easy analysis.

By using topic detection techniques, it is possible to automatically identify the topics and subtopics that are discussed in a given document. This can provide understanding of the main ideas and concepts conveyed in that document, and can also be useful for categorizing and organizing large amounts of text data.

With relation to keyword extraction, this process involves identifying the most important or relevant words or phrases that are used in a text. This can be used in various ways, such as search engine optimization, content analysis, and information retrieval. By using topic detection, it is possible to identify the key topics and subtopics in a text, and then retrieve the most important keywords or phrases associated with each topic.

The implementation of topic detection in this thesis will serve as an outline for both topic analysis and keyword extraction for the identification of words and topics in customer support questions. This implementation serves to be replicable with minimal resources and access to data.

4 Methodology

This section will explore the methods and management of the practical aspect of this project. It will also explain some of the reasoning for boundaries and limitations set on the use case.

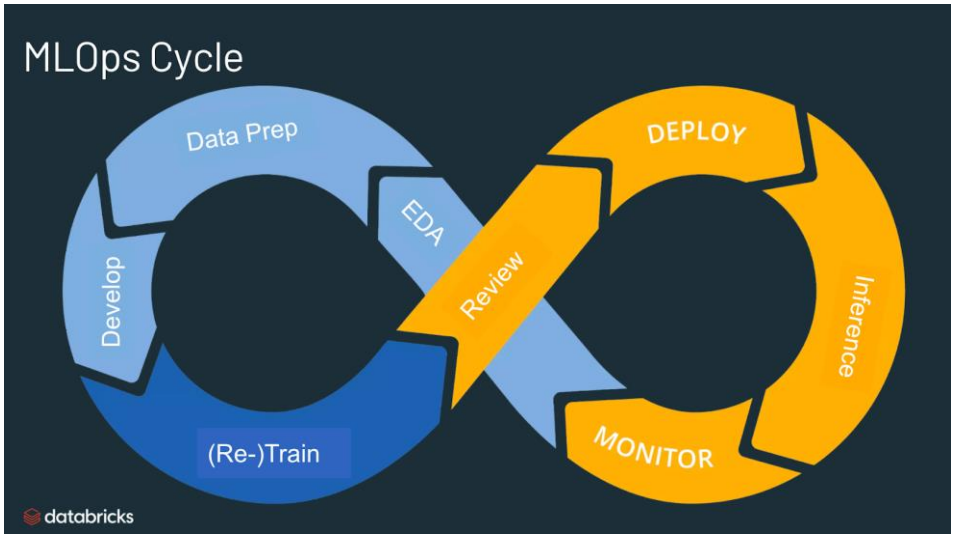
4.1 Project management

Project management refers to the planning, organizing, and coordination done in order to create and maintain a certain project or application to ensure the correct handling of resources and data.

4.1.1 Machine Learning Operations

Many machine learning projects have a set way of approaching development and operations leading into production. One of those ways is MLOPs which employ general standard functions of the machine learning pipeline. This allows all parts of the development and operations teams to work within set boundaries and follow certain guidelines.

Figure 4 MLOPS cycle



This project will work on a smaller scale than general MLOPS projects in production. Only the left side of the MLOPS cycle will be utilized since after development the process is mainly involved with the production and deployment aspect of machine learning programs.

Starting at Exploratory Data Analysis (EDA), data is iteratively found or created and then cleaned, edited, or visualized for the data prep of the project. The data preparation stage seeks to further transform the data by removing duplicates and expose any important features. After this, the model is trained and implemented in the develop stage(*What Is MLOps?*, n.d.).

4.1.2 Diary

Keeping a project diary is an effective way to track the progress made during a project and documenting all the work done throughout the project. For the sake of this thesis, a project diary will be kept primarily to show the processes taken when installing and configuring the development and testing environment. This will ensure that any variables can be kept to a minimum when trying to replicate the work done during the project. The content of the diary can be found in Annex 1.

4.2 Data collection

Many machine learning methods and projects employ a large amount of data to support the development of the specific use case. However, the advent of widely popular and open-source language models such as those provided by libraries such as Gensim allow for projects to be created on pretrained models where models have already been fed the appropriate data. The use of such an existing model allows the developer to not need exorbitant amounts of data to train and retrain their model, but rather make use of the tools available from those libraries.

This being said, this project will still make use of publicly available data of customer support queries from https://blog.bitext.com/free-customer-support-dataset to allow for the training and testing of the program – a sample of this data can be found in Annex 2. This is a free dataset specifically designed for customer support chatbots, meaning it is labelled and formatted in a very accessible way for the training of machine learning models. That being said, it will serve as a good testing bed for unsupervised learning and later testing since the separate labelled data can be used separately for the training of the model to compare differences and test efficacy of the model with the data.

4.3 Limitations and boundaries

The vast number of techniques available to developers in the AI field allow for near endless possibilities when it comes to developing a machine learning program. As mentioned in chapter 2.2.3, there many differences between the two major approaches of machine learning models and what each is most suitable for. As well as this, the scope of a machine learning project can vary greatly, such as how high level does one approach a machine learning model ranging from doing all the mathematics from scratch in code to using a service which has its own premade model.

One of the main limitations of working with LDA is the model will assume that each document is a collection of topics, and each topic is a collection of words. This leads to the model being mostly unable to detect correlation between topics in the document (Blei & Lafferty, 2007). This means a document or set of documents about a specific topic can often be mistaken for a broader topic.

Having a clear goal is an important part of machine learning applications as the scope of a project can easily get out of control with the intricacies of language models themselves.

The main focus of this project is the practical implementation of topic detection for a document or a set of documents to attain the overall topics from the customer support questions. The boundaries set for this project keep that objective in mind. Therefore, alongside the shortened MLOPs cycle mentioned above, there is a set goal this project hopes to achieve with the self-trained language model. Foremostly, to understand and train an LDA language model using publicly available customer support questions, then using more questions from the same dataset which was not used in the training to extract the major topics from the questions, and finally to display and analyze the accuracy of the language model based on various parameters such as the labels which already exist on the data.

There are several options for existing LDA models within different python libraries that can be used, the most popular libraries are: Gensim, Scikit-learn, PyLDAvis, and SpaCy.

While the understanding of LDA as a use case will be explored and is important to the use case in this thesis, the model used will be the premade LDA model of the Gensim library.

5 Technology

This section will cover the technologies used in aiding the development and implementation of a topic detection system for this specific use case.

5.1 Environment

In order to maintain a consistent environment within the prototype to allow for the replicability of the project there will be a set installation of Linux with the use of VirtualBox and vagrant.

Linux is a collection of open-source operating systems based on Unix. Developers have long favoured Linux for its transparency and replicability for development of projects. For the project, Ubuntu will be used.

In order to utilize Linux on a windows pc for the project a virtualization software will be employed to standardize the project. Virtualisation software simulates the hardware functionality of the intended operating system within the computer. The virtualization software will be VirtualBox v6. 1.24.

Managing virtual machines is a tedious task when various different kinds of configurations and installations are required. Vagrant an open-source tool used to provision and configure virtual development environments in a reproducible way. The use of vagrant easily allows certain environment variables and settings to be stored in a simple configuration file, it also gives the optionality to easily back up work and progress on a project.

Python is a general-purpose programming language with a large variety of AI and machine learning modules used to assist in the development of projects. It is one of the most widely used programming languages when it comes to machine learning and AI.

5.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation is a generative statistical model used to identify similar groups of data in an unobserved way. In simple terms, LDA looks at common or repeating aspects of data to find similarities between different pieces of data. For example, for topic modelling, it will look at repeating words or phrases within a document to extract the general idea of that document based on how frequently those words will appear. It will then show the probability that those words or phrases have at being the topic of the document.

The general process of topic detection with LDA is as follows:

1. Pre-processing: The initial step is to pre-process the text data by removing stop words, punctuation, and other non-relevant information. The remaining text is then tokenized, stemmed, lemmatized, and converted into a bag of words representation.

2. Pick the number of topics: The next step is to select the number of topics that the algorithm should identify in the text data. This is typically done using trial and error or through more advanced techniques like perplexity score or topic coherence.

3. Model training: While training, LDA learns the distribution of words within each topic and the distribution of topics within each document.

4. Topic interpretation: Once the LDA model has been trained, the topics can be interpreted by examining the most probable words in each topic. This helps to assign a label or name to each topic that summarizes its content.

Tokenization is the process of separating the sentences into individual words or tokens. This means each word in the sentence will be separated and made lowercase. This process allows for the machine learning models to interpret each word in the sentence by converting it to the correct machine-readable format.

Stop words are some of the most common and inconsequential words within a certain phrase or sentence. Words such as ”the”, ”it”, or ”and” have very little impact on the topic of a specific phrase or question, on top of this, these words occur disproportionately within any set of phrases or questions. Removal of this list of stop words from the text will reduce the noise in the text, in turn making the detection more accurate.

Stemming is the process of simplifying the tokenized words further. Much like removing stop words from the entire sentence, stemming focuses on single words. Aiming to remove unnecessary suffixes and prefixes from words and converting them to the ‘root’ word. Words like creative, creating, created, and creates would all be stemmed into the word ‘create’.

Lemmatization is a more advanced technique of formatting words and pays more attention to the context of the word itself. While stemming will only remove a word’s suffix or prefix, lemmatizing will convert the word to its actual ‘root’ word with regard to the context. Words such as better and best will be converted to the word ‘good’, while stemming could not deal with this type of word.

6 Topic detection development

The focus of this section is the technical set up of the environment which the program will be developed and the implementation of the technology mentioned in Chapter 5 Technology to form the use case.

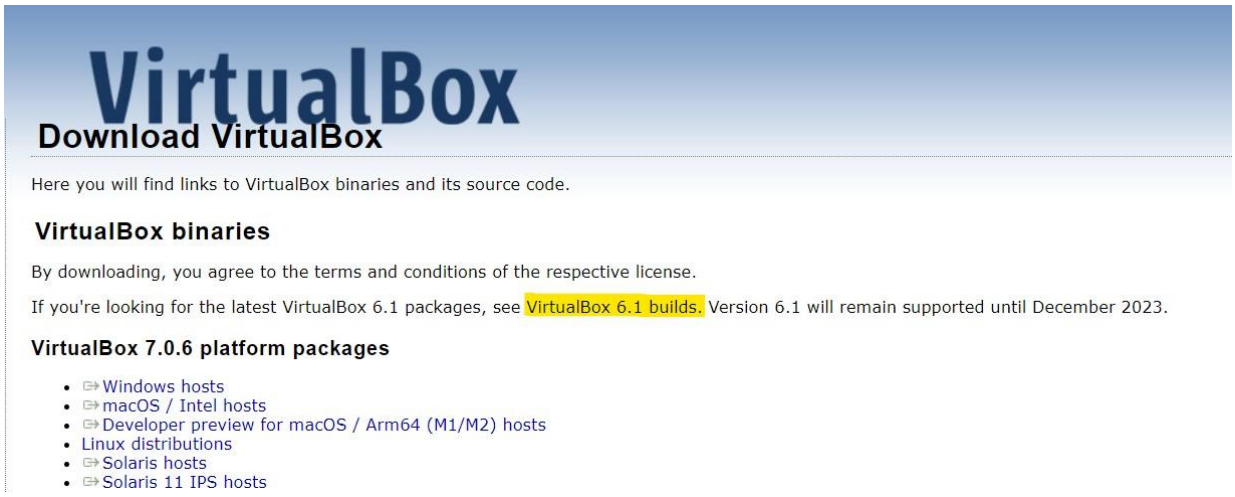
6.1 Software installation

The software installations for the use of this project involves VirtualBox, Vagrant, Python, and the accompanying packages for python as laid out below.

6.1.1 VirtualBox

The installation of VirtualBox allows for the simple use of virtual machines on the host computer.

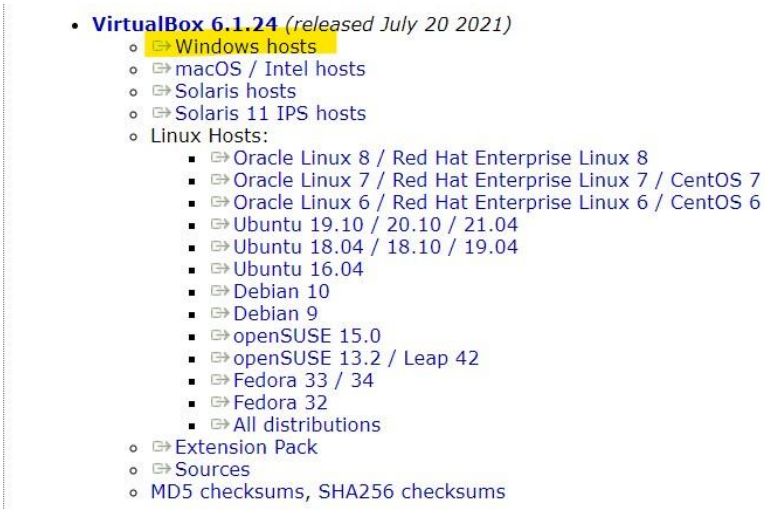
Figure 5 VirtualBox Downloads Page



Note. (*Downloads – Oracle VM VirtualBox*, n.d.)

For the installation of VirtualBox version 6.1.24, the downloads page for VirtualBox is visited. As seen in Figure 5 VirtualBox Downloads Page visiting the VirtualBox 6.1 builds page through the link highlighted in yellow, the correct version of the software can be found.

Figure 6 VirtualBox Version 6.1 Downloads Page



Note. (*Download_Old_Builds_6_1 – Oracle VM VirtualBox*, n.d.)

Follow the default installation steps for VirtualBox on windows.

6.1.2 Vagrant

The installation for Vagrant 2.2.18, like VirtualBox, is simple. Firstly, the correct version of the vagrant installer for the host platform must be downloaded and installed. The default procedures are to be followed for the installation and vagrant will automatically be added to the path of the machine. To ensure that multiple hypervisors can be used on the system without error, the host machine must ensure that Hyper-V is not enabled. This can be turned off with the following command in PowerShell using elevated privileges:

```
Disable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V-All
```

It is also possible to do this through the windows settings by navigating to ‘Apps and Features’, selecting ‘Turn windows Features on or off’, and unselecting Hyper-V(*Installing Vagrant | Vagrant | HashiCorp Developer*, n.d.)

6.1.3 Python

Python 3.9.13 is used in the development of this project. The installation of python is done using the vagrant file in the diary in Annex 1.

This project relies heavily on the Gensim library. Gensim is an open source and free Python library used for process unstructured text using unsupervised machine learning algorithms (*Gensim*, n.d.).

6.2 LDA Model

With the installation of the software necessary to develop the LDA topic model using Gensim. In order to make human language understandable to the model, it must first be trained on a set of unstructured data.

The LDA topic model comes from the Gensim package for python.

6.2.1 Training Data

The data from Bitext, as laid out in chapter 4.2, contains some extra information which is not necessary for the training of the model. It will therefore need to be pre-processed. The information of this dataset does provide some interesting information about the questions provided, such as the category and intent of the question (in this case 'utterance'). This extra information would be incredibly useful for a supervised learning language model. However, for the implementation of this project, only the utterance/question is needed. For the training of the model, one could separate the differentiate the questions based on the category.

Figure 7 Bitext dataset structure

	A	B	C	D
1	flags	utterance	category	intent
2	BM	I have problems with canceling an order	ORDER	cancel_order
3	BIM	how can I find information about canceling orders?	ORDER	cancel_order
4	B	I need help with canceling the last order	ORDER	cancel_order
5	BIP	could you help me cancelling the last order I made?	ORDER	cancel_order
6	B	problem with cancelling an order I made	ORDER	cancel_order

First the training data for the is transformed into a list of separate strings. It is loaded from the csv file and set as a single long list of all the questions as seen in the figure below.

Figure 8 read_data function python file

```
import csv
import random

random.seed(10)

def read_data():
    questions_list = []
    with open('Bitext_Sample_Customer_Service_Training_Dataset.csv', 'r') as file:
        reader = csv.reader(file)
        next(reader)  # skip the first row
        for row in reader:
            questions_list.append(row[1])
    random.shuffle(questions_list)
    return questions_list[0:int(len(questions_list)*0.90)]
```

The function ‘read_data’ in Figure 8 will return the list of the first 90% of questions in the file as a single list of strings. The output will look as follows: ['I have problems with cancelling an order', 'how can I find information about cancelling orders?', 'I need help with cancelling the last order', ...]. Before the data is returned, it is randomly shuffled in order to obtain an even distribution of the data.

With this data formatted a list of strings, it can now be further processed for the language model to understand.

6.2.2 Pre-processing

In order for the LDA model to be able to ’understand’ the data, which is inputted, various pre-processing steps need to be taken. This includes tokenizing the text, removing stop words, lemmatizing the words, and creating a bag-of-words representation of the text.

This is done with a premade tokenizer from Gensim within the simple_preprocess function.

Figure 9 Gensim import and simple_preprocess import.

```
import gensim
from gensim.utils import simple_preprocess
from nltk.corpus import stopwords
```

In Figure 9 Gensim import and simple_preprocess import. are the imports for the Gensim library and the simple_preprocess function necessary for the pre-processing of the data while the stopwords from nltk.corpus contains the necessary stopwords mostly used in NLP projects.

Figure 10 Pre-process function

```
lemmatizer = WordNetLemmatizer()

# Function to preprocess each question
def preprocess(text):
    result = []
    # Tokenize the text into individual words
    for token in gensim.utils.simple_preprocess(text):
        # Remove stop words and words with length less than 3 token length
        if token not in stopwords and len(token) > 3:
            token = lemmatizer.lemmatize(token)
            result.append(token)
    # print(result)
    return result
```

In Figure 10 Pre-process function shows the preprocess function which takes in a single string as the text as a parameter and returns the processed result which is a list of tokens/words. Firstly, result list is initialized. Then each word is tokenized with the simple_preprocess function from the gensim library itself, after which the length of each token (or word) is checked whether it is not in the stop words and that the length is greater than three letters, if so, it is lemmatized using the NLTK lematizer and added to the result list. Finally, the result list is returned. For example, if the input is a list of strings similar to ”I want help setting another shipping address up”, the output of

this function will return a list of tokens/words such as [”set”, ”ship”, ”address”] since the words ’I’, ’want’, ’help’, ’another’, and ’up’ are removed since they are either part of the stop words or less than 3 letters in length.

Figure 11 processed_questions

```
# Preprocess each question in the list
processed_questions = [preprocess(question) for question in training_data]
```

Line 32 in Figure 11 processed_questions represents the variable for processed questions where each string in the list of training questions is pre-processed using the aforementioned function.

Lastly before the model can be trained, the words need to be changed into a corpus object. A corpus object is a collection of words represented as a bag-of-words vector. A bag-of-words is a representation of text data as a collection of word counts, where the order of words is ignored. In a bag-of-words representation, a document is represented as a vector of word frequencies, where each element of the vector represents a unique word in the vocabulary and the value of the element represents the frequency of that word in the document. Alongside this corpus, a dictionary must be created to map each unique word to a unique ID.

Figure 12 Dictionary and Bag-of-Words

```
# Create a dictionary of all words in the questions
dictionary = gensim.corpora.Dictionary(processed_questions)

# Convert each question into a bag-of-words representation
bow_corpus = [dictionary.doc2bow(question) for question in
processed_questions]
```

In Figure 12 Dictionary and Bag-of-Words on line 35, the dictionary variable uses the gensim.corpora.Dictionary module to map each word to an integer ID into a dictionary-like object. Line 38 creates the corpus object from each question.

6.2.3 Model training

Figure 13 LDA Model Training

```
lda_model = gensim.models.LdaModel(corpus=bow_corpus, num_topics=8,
id2word=dictionary)
```

Training the base model involves specifying certain parameters in the genism.models.Lda_Model model, as seen above in Figure 13 LDA Model Training. The first parameter is the corpus that will be used in the training of the model, here the bow_corpus is used. The second parameter is num_topics, this is to specify the number of latent topics to be extracted from the training corpus. Lastly the id2word parameter is used to map the word IDs to the correct words to determine the vocabulary size, as well as for printing the topics.

7 Development results

This section will cover the testing environment, criteria, metrics, and results for the testing of the practical application of this project.

7.1 Description of testing environment

The criteria for testing an LDA topic model involves the accuracy of its representation of underlying topics of in a set of documents or texts. In the case of this thesis, the set of texts is a long list of questions about customer support which come from the Bitext dataset seen in chapter 4.2 and chapter 6.2.1.

7.1.1 Testing Methods for LDA

There are several ways to evaluate the performance of a topic model such as measuring the perplexity, measuring the coherence, and visualising the model.

The perplexity of a model measures how well the topic model predicts the held-out test data – hold-out test data being the part of the data set which was not used in the training of the model itself, generally the lower the perplexity score the better the model.

The coherence measures the interpretability of the topics which are generated. This means the measure shows how similar the top words of each topic are compared to the topic which the model ‘chose’ for those words, this can help assess how meaningful the identified topics are.

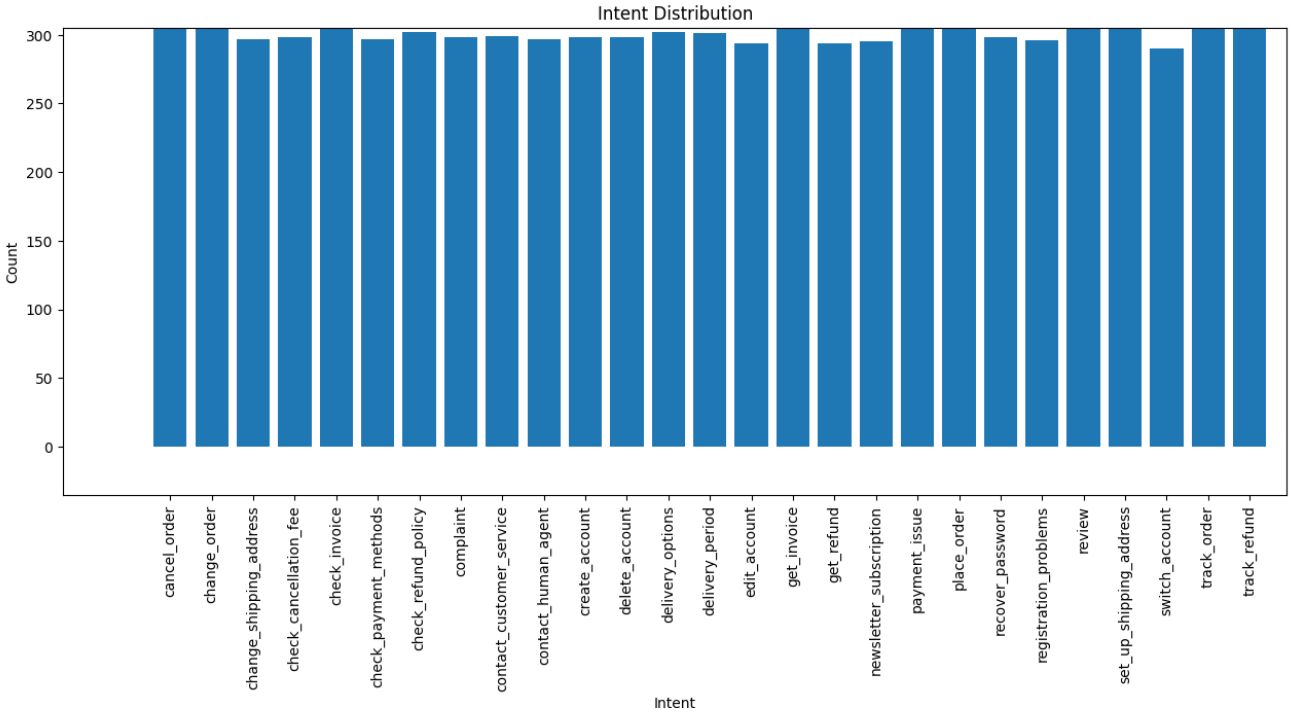
Visualising the results of the LDA topic model can convey, in a more readable way, the topics and their relationship to the documents. There are many different ways to visualise these results: word clouds, topic distributions, topic networks, and many others. Visualising the outputs allow the results

7.1.2 Training Data versus Testing Data

The data used for training and testing has been split where 90% of the data will be used for training and the remaining 10% will be used in the training. With a total of 8174 entries, 7357 will be used for training and 817 for testing purposes. Analysing this data provides some interesting insights about the data itself which may serve useful to the analysis of the model's training.

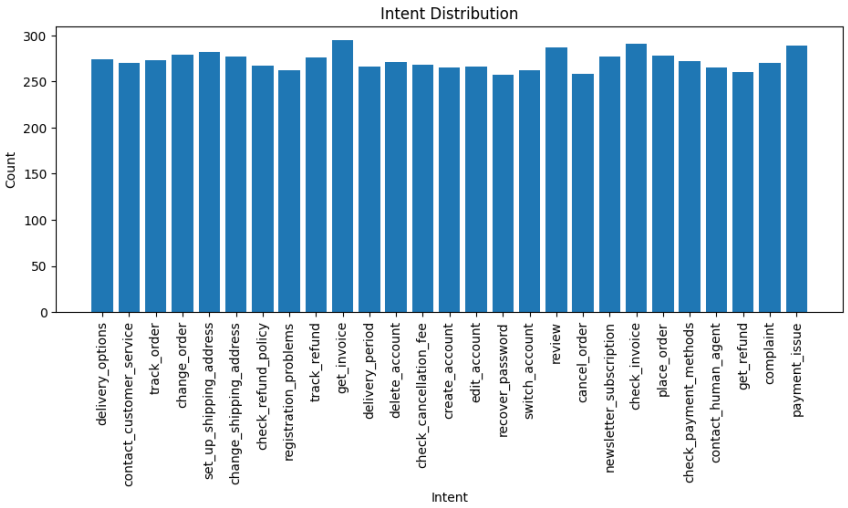
Looking at the distribution of labels across all the data using the total count of each label we see that each label occurs around the same number of times, this is shown in Figure 14 Distribution of Topics in All Data.

Figure 14 Distribution of Topics in All Data



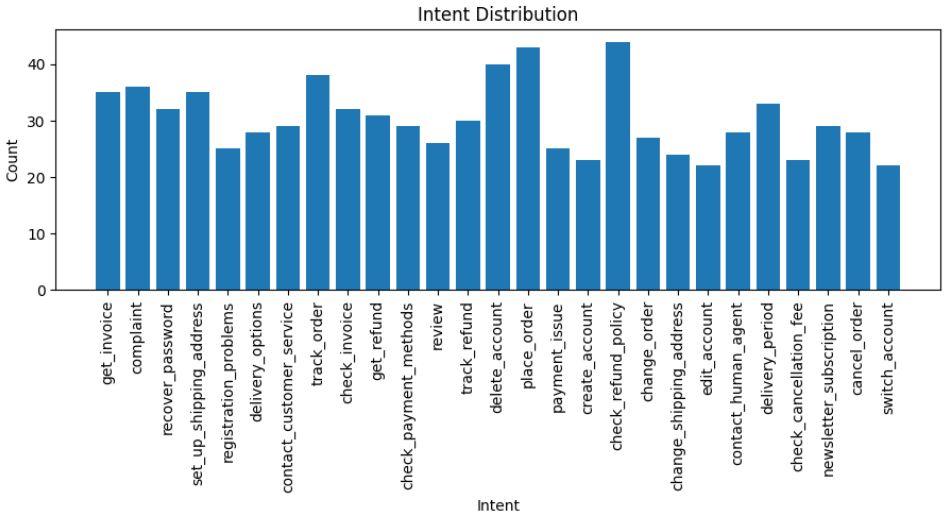
Looking at the same graph made for the training data in Figure 15 Intent Distribution for training data there is an almost even distribution of the intents with a lower count for each.

Figure 15 Intent Distribution for training data



This is especially visible in the testing data in Figure 16 Intent Distribution of testing data where each intent occurs less than 50 times.

Figure 16 Intent Distribution of testing data



It is important to have this data be distributed equally among the training data to ensure not one intent is favoured in the training of them model.

7.2 Testing metrics

The testing for this project will underline the performance in a simple and readable way which could show the efficacy of the trained model using several steps. Each step will be shown and explained with the corresponding output further analysed in chapter 7.3.

7.2.1 Default Parameters for trained model

The model will be trained with the default parameters as shown in chapter Model training The output of topics from the model itself with the initial training data will be compared to the included labelled topics in the dataset. This will provide a baseline for measuring the competency of the

7.2.2 Iteratively Adjusting Parameters for training

In order to further improve the model's accuracy, the parameters: 'num_topics', 'chunksize', and 'passes' will be iteratively changed, and the resulting perplexity and coherence will be monitored in order to obtain the most suitable parameters.

Typically, a lower perplexity score is favourable for a better model, however, this is not a set rule and often depends on the data used for the training.

7.2.3 Single Category Comparison

The final test for the model will try to emulate as best as possible to see how accurately the model can predict a certain topic from the test data. Only the questions with a single category will be passed into the model to test the topics it will provide, using this output it will be easier to compare the results to the actual intent and analyse whether or not that output has a correlation to the topic predicted.

7.3 Results

In this subchapter, the outputs and results of the metrics explained in chapter 7.2 will be shown and analysed according to the numbering structure of the chapter.

7.3.1 Outputs for default parameters of trained model

The model training as seen in chapter 6.2.3 with only the default parameters using the training data alone, the output of the model itself can be analysed as follows:

```
print_default_topics = lda_model.print_topics(num_topics=10,num_words=10)
print(print_default_topics)
```

The method ‘print_topics’ is used on the ‘lda_model’ variable which has the LDA model stored. The parameters ‘num_topics’ and ‘num_words’ represent the number of topics and number of words respectfully that are to be displayed. This will output a list of ten topics, each topic will be made up of ten words. It is important to note that the ten topics will not be actual words but rather just indexed from 0 to 9 as according to the model a topic is only a distribution of words. So, in this case the words itself will serve as the topic. The output of this method will give us a list of tuples where each topic index is the first item of each tuple, followed by a string of all the words with their confidence score.

```
[(0, '0.137*"order" + 0.109*"check" + 0.096*"payment" + 0.067*"methods" +
0.060*"made" + 0.040*"cancellation" + 0.035*"available" + 0.030*"cancel" +
0.030*"options" + 0.030*"last"'), (1, '0.091*"check" + 0.068*"refund" +
0.049*"agent" + 0.046*"file" + 0.040*"arrive" + 0.038*"complaint" +
0.034*"contact" + 0.033*"assistance" + 0.030*"delivery" +
0.027*"anything"'), …]
```

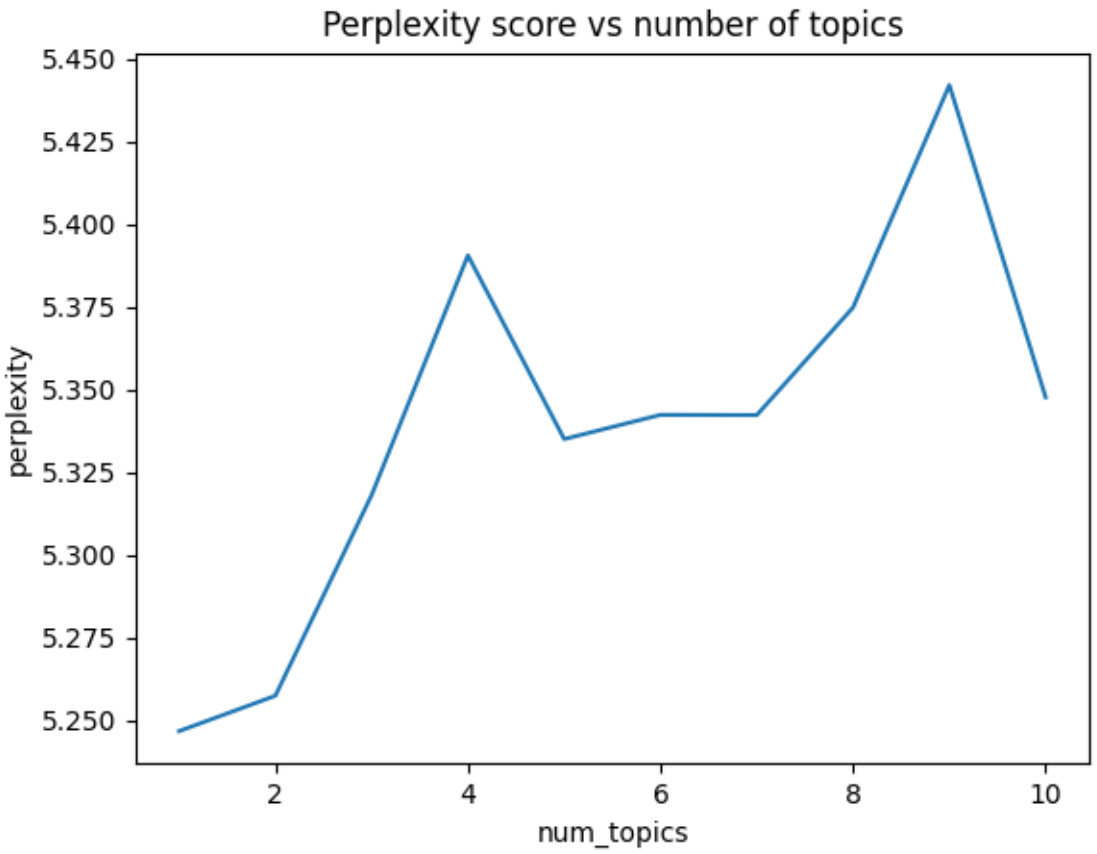
Here the output of the first three items of the list is shown with topic ‘0’ having the top words “order” and “check” with 0.137 and 0.109 confidence score respectively. While topic ‘1’ has the words “check” and “refund” with a combined confidence of only about 0.15. This gives already us some interesting insights about the contents of the questions themselves and might not inspire much confidence in the model itself.

7.3.2 Iterative Adjustments of parameters

When changing parameters for the training of the model: Figure 17 Perplexity score against the number of topics, Figure 18 Perplexity score against the chunk size, and Figure 19 Perplexity score against the number of passes shows the outputs of the perplexity score for the change in number of topics, chunk size, and passes respectively.

An initial overview of the graphs may incite confidence that certain values are much more favourable for each parameter, however, the scale of change of the perplexity score is relatively small. This may be due to the small size of the training data.

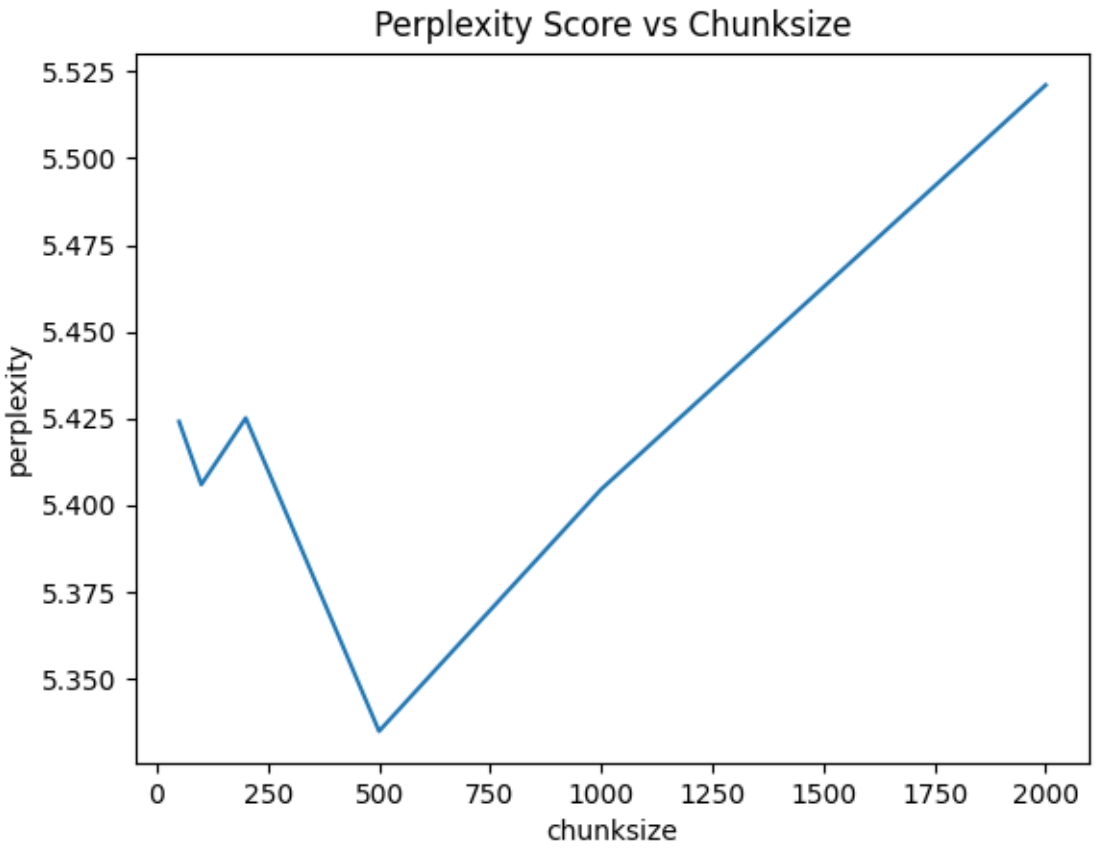
Figure 17 Perplexity score against the number of topics



In Figure 17 Perplexity score against the number of topics, one topic provides the lowest perplexity score with subsequent scores going back and forth. This may once again be due to the

amount of training data. Since the tests following this try to evaluate the model in a more readable way, the number of topics chosen for the final model is five, as it gave the lowest score for a reasonable number of topics to analyse.

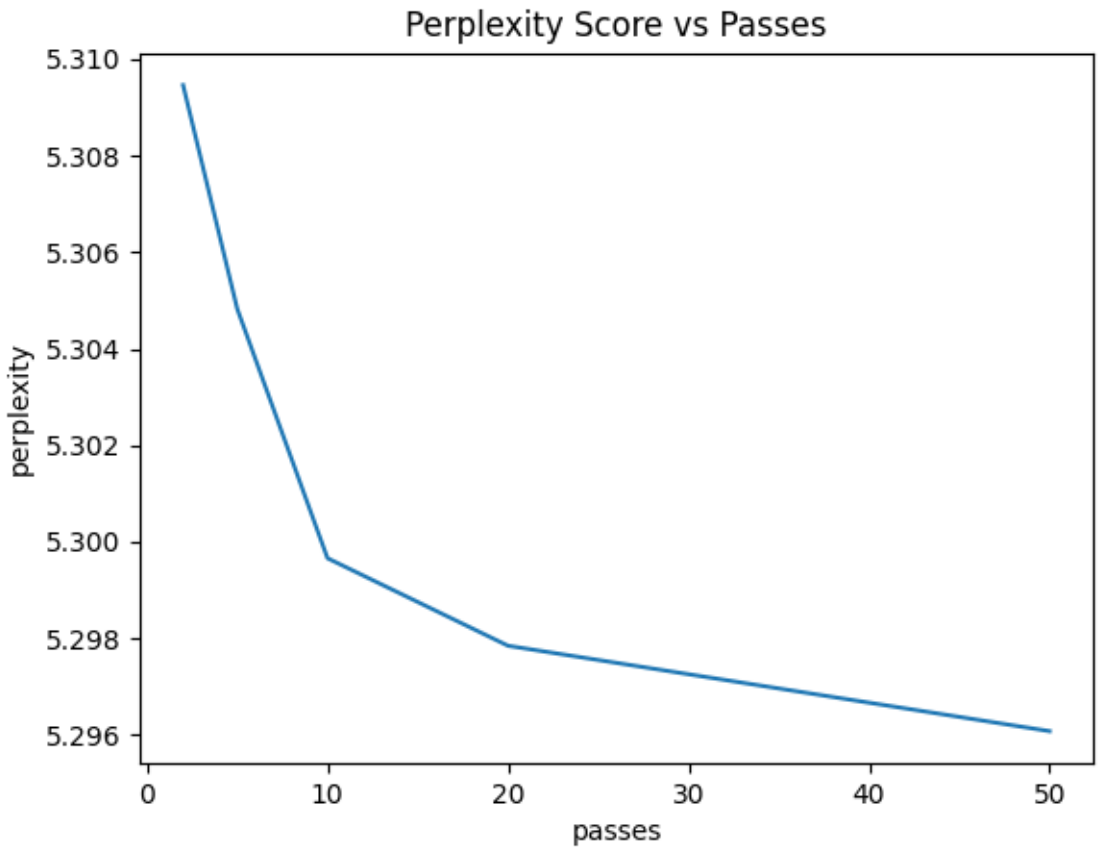
Figure 18 Perplexity score against the chunk size



The outputs in Figure 18 Perplexity score against the chunk size is similar to the values of Figure 17 Perplexity score against the number of topics as they show a similar up and down trend for different values of chunk size. Following a chunk size of 500 there is a clear up trend in the perplexity score indicating a worse model. This again might be due to the small dataset.

Moving forward, a chunk size of 500 will be implemented in the training of the model.

Figure 19 Perplexity score against the number of passes



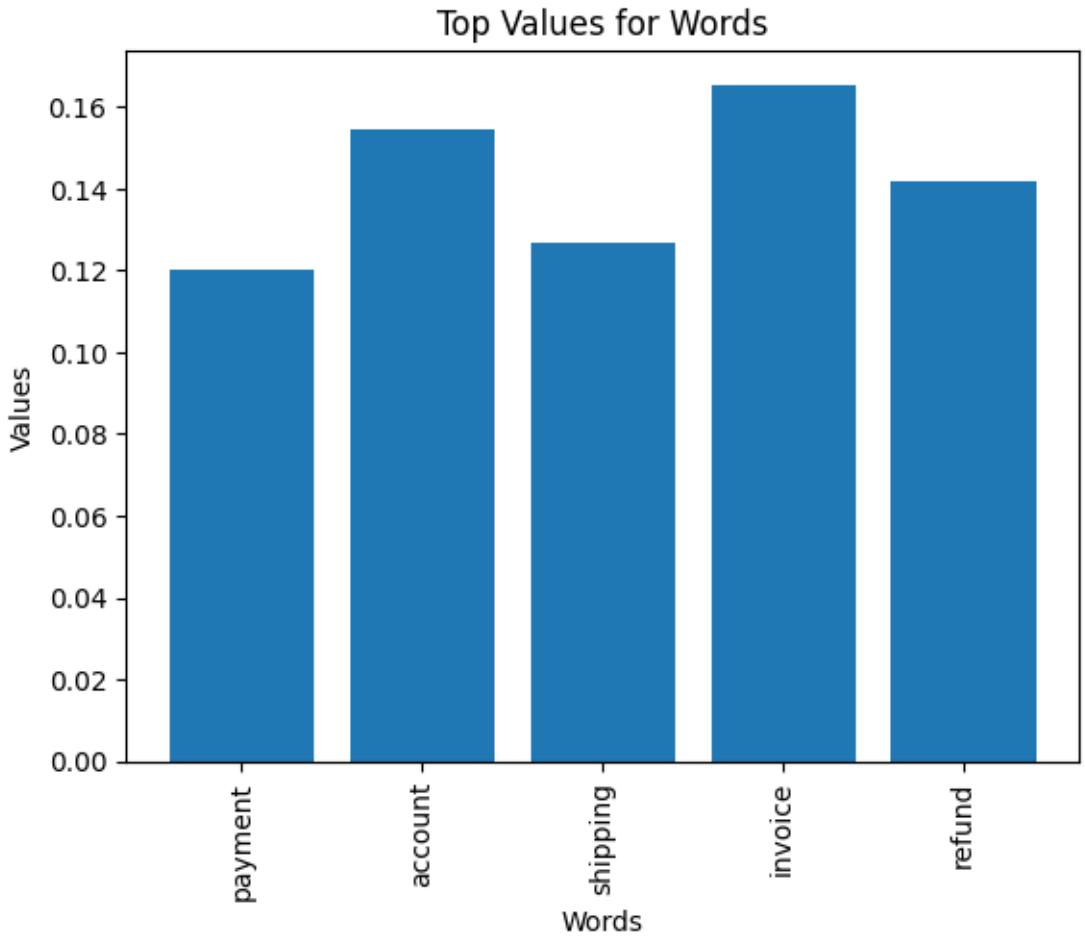
Contrary to the previous tests, Figure 19 Perplexity score against the number of passes shows a clear down trend as the number of passes increased. Flattening out as the values approach higher numbers. This parameter has a large impact on the amount of time it takes to train the model so moving forward the training of the model will have a value of 50 passes.

7.3.3 Single Category Comparison result

When testing the model for one category, the test data is split to only contain the data which has the “REFUND” value for the category column as seen in Figure 7 Bitext dataset structure.

Using this data, it is possible to gain more human-readable insights into the model’s performance compared to simply looking at the perplexity score.

Figure 20 Top words for topics



For this test, the model identified five different topics, in Figure 20 Top words for topics, it is possible to see the top words related to each of those topics (as the actual topics aren't verbosely stated but inferred through the topic words).

This bar graph shows that for topics identified, only one of the five has the top word refund, which may indicate a poor model. It is important to view this data with a grain of salt, viewing only the top word for each topic may suggest something apart from what the model is actually putting out. When looking at the top five words for each identified topic:

```
[(0.12020849, 'payment'), (0.08475516, 'delivery'), (0.0677974, 'option'),
(0.052612323, 'information'), (0.040747087, 'method')]
[(0.1545826, 'account'), (0.058210902, 'user'), (0.04838664, 'assistance'),
(0.04739705, 'order'), (0.03381329, 'agent')]
```

```
[(0.12697662, 'shipping'), (0.089318454, 'service'), (0.08325439, 'customer'),
(0.07520747, 'change'), (0.04661864, 'item')]
[(0.16540629, 'invoice'), (0.10392865, 'last'), (0.102127336, 'order'),
(0.04995759, 'track'), (0.04821235, 'registration')]
[(0.1418642, 'refund'), (0.056253534, 'newsletter'), (0.050475724, 'money'),
(0.05039674, 'status'), (0.050283507, 'would')]
```

It seems for some of the identified there is very little correlation between the words presented and what one might typically associate with words regarding refunds, words such as ‘shipping’ and ‘account’. On the other hand, words such as ‘payment’, ‘invoice’, and ‘refund’ might suggest that the data which was passed in has something to do with refunds.

It is important to note that out of the test data, only 90 questions had the category ‘REFUND’, creating a small dataset to test on. Since the model was trained on data from all categories and then tested on only this small section of data, it’s possible the model is trying to find topics which do not exist in the test dataset.

8 Summary and recommendations

In summary the thesis presents an analysis of both customer support/customer service and artificial intelligence, as well as the role artificial intelligence has played in customer service over the last few years in chapter 2: Customer support and artificial intelligence.

The information in chapter 3 Topic modelling and its use cases, chapter 4 Methodology, and chapter 5 Technology - details the use cases, methodology, and technology respectively which is used in the development of a topic detection system using latent Dirichlet allocation in python.

Chapter 6 Topic detection development shows the implementation of the system using customer support questions found online. Explaining the start-to-finish process of implementation including the development environment, data pre-processing, and model training.

Finally, chapter 7 Development results explains the different testing methods and metrics used to assess the model and interprets the results of the implementation of the model developed.

As for recommendations for this project both looking back and moving forward. If somebody was to replicate what I have done it is very important to understand the language which is used in training and testing of the model. A large aspect of training an LDA model depends on the quality of data collected. Another important aspect as repeatedly shown in the testing is the size of the training data. Having a much larger good quality dataset would drastically increase the accuracy of the model considering the means in which LDA works.

During this thesis a large portion of what is covered was unfamiliar to me. Throughout the research and development of the project I was able to learn about many new and different things in the field of AI specifically which has opened my eyes further to the possibilities for use cases with this technology. My lack of understanding at the start of this project was a large contributor to the simplicity of the outcome of this project. That being said, I believe that what I have learned is valuable in and around working life.

References

90% of Social Media Users Reach Out to Retailers | Porch Group Media. (n.d.). Retrieved 5 March 2023, from https://porchgroupmedia.com/blog/90-percent-social-media-users-reach-out-retailers-why-social-media-your-secret-weapon/

Blei, D. M., & Lafferty, J. D. (2007). A correlated topic model of Science. *The Annals of Applied Statistics*, *1*(1), 17–35. https://doi.org/10.1214/07-AOAS114

Chambers, S. (2022, June 21). *Customer Service vs. Customer Support: Explained*. https://www.helpscout.com/blog/customer-service-vs-customer-support/

Chatbot Market Size To Hold USD 4.9 Billion By 2032. (n.d.). Retrieved 6 March 2023, from https://www.precedenceresearch.com/chatbot-market

Chen, D. (2020, July 9). *Recommendation System—Matrix Factorization*. Medium. https://towardsdatascience.com/recommendation-system-matrix-factorization-d61978660b4b

Download_Old_Builds_6_1 – Oracle VM VirtualBox. (n.d.). Retrieved 6 March 2023, from https://www.virtualbox.org/wiki/Download_Old_Builds_6_1

Downloads – Oracle VM VirtualBox. (n.d.). Retrieved 6 March 2023, from https://www.virtualbox.org/wiki/Downloads

Gensim: Topic modelling for humans. (n.d.). Retrieved 18 February 2023, from https://radimrehurek.com/gensim/

Haenlein, M., & Kaplan, A. (2019). A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence. *California Management Review*, *61*(4), 5–14. https://doi.org/10.1177/0008125619864925

Installing Vagrant | Vagrant | HashiCorp Developer. (n.d.). Installing Vagrant | Vagrant | HashiCorp Developer. Retrieved 7 March 2023, from https://developer.hashicorp.com/vagrant/docs/v2.2.18/installation

Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., & Zhao, L. (2019). Latent Dirichlet allocation (LDA) and topic modeling: Models, applications, a survey. *Multimedia Tools and Applications*, *78*(11), 15169–15211. https://doi.org/10.1007/s11042-018-6894-4

Leonard, J. (2018, September 14). *What Are The 4 Best Avenues for Communicating With Customers?* Business 2 Community. https://www.business2community.com/communications/what-are-the-4-best-avenues-for-communicating-with-customers-02117712

Lewis, T. (2014, December 4). *A Brief History of Artificial Intelligence*. Livescience.Com. https://www.livescience.com/49007-history-of-artificial-intelligence.html

NLP vs. NLU vs. NLG: The differences between three natural language processing concepts. (2020, November 12). Watson Blog. https://www.ibm.com/blogs/watson/2020/11/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/

Piercy, A. (2021, September 27). *AI-Powered Customer Service: What to Watch for in 2022*. Hitachi Solutions. https://global.hitachi-solutions.com/blog/ai-customer-service-trends/

Russell, S. J., & Norvig, P. (2009). *Artificial Intelligence A Modern Approach* (Third Edition).

Seldon. (2022, September 16). *Supervised vs Unsupervised Learning Explained*. Seldon. https://www.seldon.io/supervised-vs-unsupervised-learning-explained

The State of AI in Customer Service Report 2022. (n.d.). Dialpad. Retrieved 14 February 2023, from https://www.dialpad.com/blog/ai-customer-service-report/

What is MLOps? (n.d.). Databricks. Retrieved 6 February 2023, from https://www.databricks.com/glossary/mlops

What is Natural Language Processing? | IBM. (n.d.). Retrieved 5 February 2023, from https://www.ibm.com/topics/natural-language-processing

Zulaykho, K., Nazira, A., & Abdulaziz, E. (2022). *MEASURING GLOBAL OUTSOURCING AND ITS EFFECT ON FIRMS PRODUCTIVITY*.

Annex 1: Material management plan

During the development project, a diary is kept in Annex 2: Diary, in which technical information about the project is collected. This information contains information with regard to development for the thesis. The diary is stored on drive C of the author's computer and is continuously backed up and synced to the personal OneDrive of the author. The diary is kept at station C for at least one year after the completion of the thesis.

During the development of the project there are no meetings of importance with regard to the development of the project and therefore no minutes available.

Information is collected on the success of the completed project in images of results and outcomes which are stored on the drive C of the author’s computer and are also backed up to the personal OneDrive of the author. These results are to be stored for at least one year after the completion of the thesis.

Annex 2: Diary

Installation of VirtualBox

1. Once the VirtualBox download is complete, open the setup file to start the installation process.

2. You will see the VirtualBox setup wizard. Click "Next" to continue.

3. All the default settings can be chosen for the installation.

4. Once the installation is complete, click "Finish" to exit the setup wizard.

Running the Virtual Environment

Vagrant File:

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.hostname = 'data-science'
  config.vm.synced_folder "Thesis_Ubuntu/", "/home/vagrant/Thesis_Ubuntu/"

  config.vm.provider "virtualbox" do |vb|
    # Display the VirtualBox GUI when booting the machine
    vb.gui = true
      vb.name = "TOPIC DEV"
  end
      config.vm.provision "shell", path: "script.sh"
end
```

This vagrant file will provision the ubuntu virtual machine using virtual box and name it ‘TOPIC DEV’. It will then run the accompanying shell script to update packages and install python.

Shell Script

```
apt-get update
apt-get -y install python3
```

Annex 3: Bitext Data Sample

flags	utterance	category	intent
BM	I have problems with canceling an order	ORDER	cancel_order
BIM	how can I find information about canceling orders?	ORDER	cancel_order
B	I need help with canceling the last order	ORDER	cancel_order
BIP	could you help me cancelling the last order I made?	ORDER	cancel_order
B	problem with cancelling an order I made	ORDER	cancel_order
BI	can you help me canceling my last order?	ORDER	cancel_order
BE	I do not know how to cancel the last order I have made	ORDER	cancel_order
BM	problems with canceling my orders	ORDER	cancel_order
BM	I have problems with cancelling the last order I have made	ORDER	cancel_order
BIMP	could you give me information about order cancellations?	ORDER	cancel_order
B	I need help canceling the last order I have made	ORDER	cancel_order
B	I need help with cancelling the order I made	ORDER	cancel_order
BE	I do not want the order	ORDER	cancel_order
BIP	how could I find information about cancelling an order?	ORDER	cancel_order
BIMP	would you give me information about order cancellations?	ORDER	cancel_order
BM	I have problems with cancelling my order	ORDER	cancel_order
BI	how to cancel an order?	ORDER	cancel_order
BM	I have problems with cancelling an order I have made	ORDER	cancel_order
B	assistance with cancelling the order I made	ORDER	cancel_order
BI	can you give me information about canceling an order?	ORDER	cancel_order
BM	problems with canceling orders	ORDER	cancel_order
BE	I do not know how I can cancel the last order I made	ORDER	cancel_order
BP	I would like to cancel the order I made	ORDER	cancel_order
B	I need help canceling an order	ORDER	cancel_order
BIP	how could I get information about cancelling an order?	ORDER	cancel_order
B	I need help cancelling an order I made	ORDER	cancel_order
BM	I have problems with cancelling orders	ORDER	cancel_order
B	I have a question about canceling an order	ORDER	cancel_order
BM	problem with cancelling orders	ORDER	cancel_order

Annex 4: Data Collection file: read_data.py

```
import csv
import random

random.seed(10)

def read_data():
    questions_list = []
    with open('Bitext_Sample_Customer_Service_Training_Dataset.csv', 'r') as file:
        reader = csv.reader(file)
        next(reader)  # skip the first row
        for row in reader:
            questions_list.append(row[1])
    random.shuffle(questions_list)
    return questions_list[0:int(len(questions_list)*0.90)]

def read_test_data():
    questions_list = []
    with open('Bitext_Sample_Customer_Service_Training_Dataset.csv', 'r') as file:
        reader = csv.reader(file)
        next(reader)  # skip the first row
        for row in reader:
            questions_list.append(row[1])
    random.shuffle(questions_list)
    print(len(questions_list[int(len(questions_list)*0.90):-1]))
    return questions_list[int(len(questions_list)*0.90):-1]

def category_test_data():
    questions_list = []
    with open('Bitext_Sample_Customer_Service_Training_Dataset.csv', 'r') as file:
        reader = csv.reader(file)
        next(reader)  # skip the first row
        for row in reader:
            if row[2] == "REFUND":
                questions_list.append(row[1])
    random.shuffle(questions_list)
    print(len(questions_list[int(len(questions_list)*0.90):-1]))
    return questions_list[int(len(questions_list)*0.90):-1]
```

Annex 5 : Model Creation and Training

```
from read_data import read_data, read_test_data
import gensim
from gensim.utils import simple_preprocess
from gensim import corpora
import matplotlib.pyplot as plt

import nltk
from nltk.corpus import stopwords

stopwords = stopwords.words('english') + ["help", "could", "need", "want", "item", "address",
"know"]

from nltk.stem import WordNetLemmatizer

training_data = read_data()
testing_data = read_test_data()

lemmatizer = WordNetLemmatizer()

# Function to preprocess each question
def preprocess(text):
    result = []
    # Tokenize the text into individual words
    for token in gensim.utils.simple_preprocess(text):
        # Remove stop words and words with length less than 3 token length
        if token not in stopwords and len(token) > 3:
            token = lemmatizer.lemmatize(token)
            result.append(token)
    # print(result)
    return result

# Preprocess each question in the list
processed_questions = [preprocess(question) for question in training_data]

# Create a dictionary of all words in the questions
dictionary = gensim.corpora.Dictionary(processed_questions)

# Convert each question into a bag-of-words representation
bow_corpus = [dictionary.doc2bow(question) for question in processed_questions]


# Train the LDA model on the bag-of-words corpus
lda_model = gensim.models.LdaModel(corpus=bow_corpus, num_topics=8, id2word=dictionary)

lda_model.save("models\lda_model")
```

Annex 6: Model Testing and results

```
topics_and_score = []

for idx, topic in lda_model.print_topics(num_topics=-1, num_words=2):
    # print(idx, topic)
    topic_words = topic.split(" + ")
    top_2_words = " + ".join(topic_words[:1])
    top_2_words = top_2_words.split("*")
    topics_and_score.append(top_2_words)
    # print(f"Topic {idx}: {top_2_words}")

# print("topics and score: ", topics_and_score)

sorted_list = sorted(topics_and_score, key=lambda x: float(x[0]), reverse=True)

# print(sorted_list)

labels = [item[1] for item in sorted_list]
values = [float(item[0]) for item in sorted_list]

# Plot the bar chart
plt.bar(labels, values)

# Add axis labels and a title
plt.xlabel("Labels")
plt.ylabel("Values")
plt.title("Sorted List Bar Chart")

# Display the chart
plt.show()

processed_testing_data = [preprocess(question) for question in testing_data]

# Convert each question into a bag-of-words representation
bow_doc = [dictionary.doc2bow(question) for question in processed_testing_data]
# print(bow_doc)
# lda_model = gensim.models.LdaModel(bow_corpus, num_topics=10, id2word=dictionary,
passes=5)

from gensim import models

lda_model = models.ldamodel.LdaModel.load("models\lda_model")

# Get the topics and their corresponding words
topics = lda_model.show_topics(num_topics=2, formatted=False)
print_topics = lda_model.print_topics(num_topics=-1, num_words=3)
for topic in print_topics:
    print(topic)

# print(topics)

# Print each topic and its corresponding words
for topic in topics:
    topic_id = topic[0]
    topic_words = [word[0] for word in topic[1]]
    # print(f"Topic {topic_id}: {' '.join(topic_words)}")

test_data_topics = lda_model.get_document_topics(bow=bow_doc)
test_data_topics = lda_model.top_topics(corpus=bow_doc)
```