Master thesis

Business Analytics

An approach to map textual questions to a response out of a predefined response set.

Bob Mes (2650287)

August 2021

First reader:
Dr. M. Cochez

Second reader:
Drs. F. den Hengst

Company supervisor:
J. Schouten

Vrije Universiteit Amsterdam
Faculty of Sciences
De Boelelaan 1081
1081 HV Amsterdam

DutchChannels
Data team
Mozartlaan 27D
1217 CM Hilversum

Preface

This thesis is written to fulfill the Master Business Analytics at the Vrije Universiteit (VU) Amsterdam. The research was conducted from February 2021 to August 2021 at DutchChannels, which provided access to customer service data and offered the opportunity to write this thesis.

I want to thank Jesse Schouten for the excellent guidance, support, and feedback offered throughout the internship.

Also, I want to thank Aron Peters, who unfortunately could not guide me through the whole internship. However, the guidance and feedback in the first months were valuable.

I would also like to thank Michael Cochez for being the first supervisor and feedback throughout the internship. Also, I would like to thank Floris den Hengst for being the second supervisor and reading the report.

Bob Mes
Amsterdam, 13-8-2021

Abstract

In the last years, consulting companies have used machine learning to support their clients in making decisions. Especially in the area of Marketing and the Internet of Things, models are needed that can deal with many data, unstructured data, and text. DutchChannels also want to support their customers via Chatbot. This study proposes a model to answer questions in a well-defined domain, the Subscription Video On Demand (SVOD) Customer Service (CS) domain. Therefore a model should be developed that maps a textual question to a predefined response.

The dataset in this research is gathered from the DutchChannels customer service. This data comes from two different channels: New Faith Network and withLove. Besides English also Dutch, Norwegian, and Swedish are present in the data. To create one dataset, all questions are translated to English, and the two channels are merged. The nature of the gathered data is sources like mail, Facebook, and a portal. Before this dataset was workable, some data preparation should be done. To solve this problem as a multi-class classification problem, all agent responses were matched with one of the predefined responses. This was done using word2vec and the soft cosine similarity score. After, evaluation we choose only to keep the top scores resulting in a final data set of size 16,905 with an estimated error (in the matching) of 3.5% and a margin of error of 3.8% with 95% confidence.

Two different kinds of classification models are tested. First, multi-class classification models. These models are trained to classify the type of question. Then with this classification, the corresponding response to that type is returned to the customer. For this model, the textual question would first be processed by an NLP pipeline. The text would undergo some transformations like spellchecking, stopword removal, and lemmatization. A feature vector would be made with a TF-IDF score for uni- and bigrams and 20 LDA topics with the processed text.

The second classification models that are tested are the binary classification models. These models get as input the word embeddings of the textual question and the word embedding of the standard responses (one for one). The question and standard response will be encoded to a vector by the same network. With these encodings, a matching score will be calculated. The standard response with the highest matching score will be returned to the customer.

The end result yields a model that can map a textual question to a response. Overall, the random forest had the best performance with an accuracy of 0.55 and a recall@5 of 0.82. Of the binary classification models, the BiLSTM + attention network performed the best with an accuracy of 0.46 and a recall@5 of 0.71.

Contents

1 Introduction

In the last years, consulting companies have used machine learning to support their clients in making decisions. Especially in the area of Marketing and the Internet of Things, models need to handle a lot of data, unstructured data, and text. Experts have estimated that eighty to ninety percent of data in any organization is unstructured, and this amount is growing significantly. Therefore, techniques that can deal with unstructured data like text need to be developed and improved. In this study, we propose a model to answer questions in a well-defined domain, the Subscription Video On Demand (SVOD) Customer Service (CS) domain. A chatbot could, for example, help run a helpdesk, react to social media messages, or help choose from a series of products.

1.1 Chatbots

A chatbot is a conversational agent that can interact with humans using natural language. Chatbots are being developed for multiple domains and different reasons. Chatbots are developed to answer questions, imitate someone's writing style, and do many other things. Chatbots can be classified into two types: retrieval-based and generative-based. As the name suggests, a retrieval-based chatbot will retrieve the utterance the chatbot will give. This utterance is retrieved from a predefined set of utterances. Because this chatbot retrieves its utterances from a predefined set, it is suitable for a closed-domain chatbot. A generative chatbot generates the utterances itself. This is an exciting field of study because this process, on the surface, will be more like that of the human mind.

1.2 DutchChannels

DutchChannels is a Subscription Video On Demand (SVOD) company that creates SVOD channels for niche audiences. The two largest channels are New Faith Network (NFN) and WithLove (WL), and the company also has some smaller Channels like TheaterThuis. New Faith Network is a Channel for Christians that offers the largest selection of Christian films, series, and own productions in the eight countries in which they are active. WithLove is a Channel for people that love romantic feel-good films and series and is currently active in 4 countries with many more planned in 2021. DutchChannels was established in 2017 and is growing fast. The company saw its overall subscriber base grow by more than 400% in 2020. Such fast growth also brings challenges with it. One can expect a significant rise in the number of questions asked to Customer Service (CS). The company can do several things to keep the three crucial Key Performance Indicators (KPIs) response time, resolution time, and customer satisfaction at reasonable levels. The two most obvious ones are expanding its already large CS team or creating a chatbot (that reduces pressure on CS). The advantages of creating a chatbot are that it is cost-effective and "works" 24 hours a day, seven days a week. Enabling and helping customers find their answers or point them in the right direction at any time.

1.3 Research goals

The main objective of this study is to map a question to the right response out of a predefined set of responses. When the mapping is done to high quality, many questions could be answered without the interference of a human. The research question of this paper is:

How do classification models perform when mapping a textual question to a predefined set of 46 answers from the subscription video-on-demand domain?

In this research, different types of classification models will be tested.

- Multi-class classification
- Binary classification (point-wise ranking)

With multi-class classification, the textual question will be mapped to a class, and the chatbot will return the corresponding response of that class. The models that will be tested for multi-class classification are K Nearest Neighbors (KNN), Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), and XGBoost (XGB).

With binary classification, all the predefined responses will get a score (in combination with the question), and the chatbot will return the response with the highest score. For binary classification, the dual encoder architecture is tested. We tried to enhance the performance of this architecture by adding different layers like BiLSTM, CNN, self-attention, and 1-max pooling layer.

1.4 Report structure

The report is structured as follows. Chapter 2 is a literature study on chatbots. First, the different types of chatbots are described. Hereafter some insight is given to different techniques used to match context and responses. In chapter 3, the data that is used in this research is analyzed. Also, the cleaning procedure of this data is discussed in this chapter. In chapter 4, the experimental setup of this research is given. The implementation, preprocessing, cross-validation procedure, and performance metrics are exhaustively discussed. Then in chapter 5, all the models that are tested are described. In chapter 6, the results of the models on the test set are investigated. Hereafter in chapter 7, the value of implementing two of these models for DutchChannels is discussed. In the last two chapters, the conclusion and discussion are given.

2 Literature study

As we study the literature around chatbots, we note a wide variety of chatbots and chatbot techniques. In section 2.1 we will discuss the different chatbot types. Retrieval-based and generative-based models are described in sections 2.2 and 2.3. In section 2.4 a combination of these two models is reviewed (called hybrid models).

2.1 Chatbot types

There are multiple ways to categorize chatbots [2]. A common way to divide chatbots is on how they produce their answer [61] [12] and on what domain they work [2]. In figure 1 a rough categorization using the above mentioned characteristics is shown. In this figure, chatbots are categorized into four different sections. The type of domain the chatbot operates (open or closed) and how the chatbot gives his/her answer (retrieval or generative). Open-domain chatbots are chatbots that operate on an open domain. This means that it can answer a variety of different topics. On the other hand, close domain chatbots only answer questions of a specific domain. Therefore, a chatbot that can answer a wide variety of questions like the weather, the price of stock x, did Ajax win the Eredivisie in 2020, etc., is classified as an open domain chatbot. A chatbot that can only answer questions about the weather (for example) is classified as a close domain chatbot. Hence, the chatbot that is made in this research will fall under the closed domain chatbots.

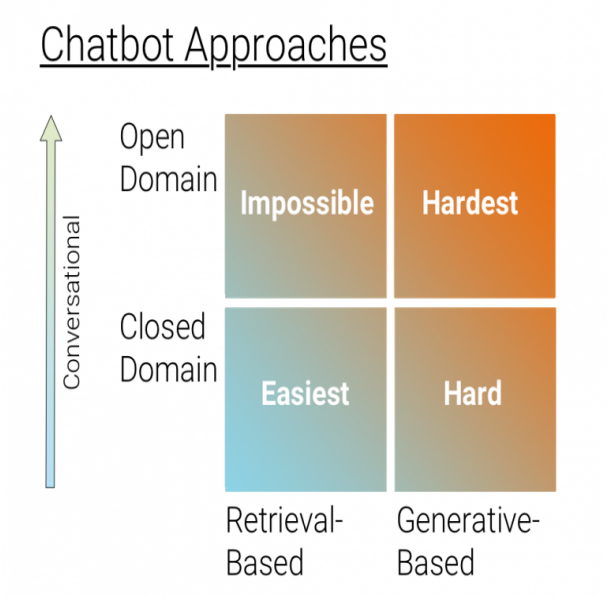

Figure 1: Types of chatbots, from [23]

The next split is done on how the chatbot creates its answers. This can be done either retrieval-based or generative-based. With retrieval-based, the chatbot retrieves its answer from a (knowledge) database. Whereas generative chatbots generate their answers fully by themselves. In figure 1 we can see that chatbots can be categorized into the following groups:

1. Open-domain retrieval-based chatbots Chatbots in this domain receive their answers from

a fixed set and give answers in an open domain. Hence, this set should cover an enormous set of questions (any possible question you can think of). Alternatively, it could use a search engine in the background to obtain their response set.

2. Open-domain generative-based chatbots This kind of chatbot again works on an open domain and hence should answer all questions. However, generative chatbots can, in theory, generate every answer that is possible. To do this, the chatbot should be able to perform the same intellectual tasks humans successfully. Although this area is heavily researched, we are far from completing this task.

3. Closed-domain generative-based chatbots Here the chatbot generates their answers on a fixed domain of questions. Because of the generative character of the chatbot, it can handle questions presented in the underlying dataset and new questions. This type of chatbot is suited for more human-like interaction and can have its own "personality." Nevertheless, there are also some downsides to this kind of chatbot. Generating answers increase the complexity of the problem by a lot. These answers can have grammatical and spelling errors, and the chatbot often learns a global/standard response like "I do not know" to a lot of questions [52] [12].

4. Closed-domain retrieval-based chatbots In this area, the chatbot is made to answer questions on a specific domain. This means it is not designed to answer every question possible but rather a fixed/predefined number of questions. Therefore, this problem can be solved (to a certain level) with today's techniques and machines. This is how most companies solve/create their chatbot.

Lately, also a new type of chatbot has risen. The hybrid chatbot. This chatbot is a combination of retrieval and generative-based chatbot.

2.2 Retrieval-based methods

A retrieval-based system should pick the most relevant response of a fixed set of responses. In table 1 an example of how a retrieval-based should work is given. Here it is given a set of two responses, and it should pick the first one to return. Matching two textual segments is a challenging problem as the system needs to model the segments, as well as their relations [62]. We will divide retrieval-based systems into two groups: vector space models and matching models.

	Context
C	I would like to **unsubscribe** please.
	Response
✓	Thank you for your message, we regret to read that you wish to **cancel** your **subscription**. You can do so by logging in via this website: www.newfaithnetwork.com/login. After you have logged in, at the top of your screen, select 'Account' > 'Membership' > 'Update' > 'Cancel'.
✗	Thank you for bringing this to our attention. We are working hard to solve this issue and advise you to try it again later. Meanwhile we advise you to use the Google Chrome internet browser on your device. You can enjoy New Faith Network here: www.newfaithnetwork.com/login. Our apologies for the inconvenience.

Table 1: Example of response selection (taken from DC dataset)

2.2.1 Vector space model

Vector space models are typically considered simple. Hence it is often used as a baseline. Here a mathematical model is used to represent the text documents. This is done via Term Frequency-Inverse Document Frequency (TF-IDF) [45]. This is a statistical method to calculate the importance of a word in a document regarding its importance in a corpus. It presumes that a word is important in a document when it appears a lot in that given document and is less frequent in the whole corpus.

$$TF - IDF(w_{i,j}) = TF(w_{i,j}) * log\left(\frac{N}{DF(w_{i,j})}\right) \tag{1}$$

Here TF denotes Term Frequency which is the number of occurrences of the word in a document, DF stands for Document Frequency which represents the number of documents that contains the word, and N is the total number of documents in the corpus. TF-IDF is widely used in Information Retrieval and still represents a strong baseline for multiple domains like keyphrase extraction [26]. [35] also used TF-IDF as a baseline for their research. All the utterances (of the dialogue) are concatenated and represented as a vector with TF-IDF scores (for all the words). This TF-IDF vector is also created for all the candidate responses. Then it hypothesizes that the "good" response is most similar to the context in terms of word frequency. To obtain the similarity between two documents, the computed the cosine similarity.

$$cos\theta = \frac{d_1 * d_2}{||d_1|| * ||d_2||} \tag{2}$$

Where d_1 and d_2 are vectors for documents one and two (that need to be compared). A limitation of this model is that the words in one document should exactly match the words in the other document. This means two different words with roughly the same meaning will be processed as different words. Hence, they do not positively influence the similarity score where in reality, it should have.

Before any of the above calculations are done with text, it usually first undergoes different prepossessing steps. [29] and [58] stress the importance of preprocessing in text mining techniques and applications. Some of the different preprocessing steps are:

- Tokenization: identifying individual tokens (e.g., word, punctuation) within a sentence.
- Part-of-speech tagging: the process of marking up a word as corresponding to a particular part of speech
- Stop-word removal: the process of removing certain words. This is done because stop-words, by definition, are meaningless words that have low discrimination power [33].
- Stemming: A technique to bring different alterations of a word back to its base word [47]. The four common types of stemming are table lookup approach, successor variety, N-gram stemmers, and Affix removal stemmers. [29]
- Lemmatization: A technique to bring different alterations of a word back to its base word. With this process, a word is transformed back to its dictionary form (called lemma). [47]
- N-grams: stands for the number of tokens used as a feature. Where 1-grams (unigrams) are features consisting of 1 token, 2-grams (bigrams) are features consisting of two tokens, and so on. With N-gram models, the probability of a certain token following a sequence of tokens is estimated.

2.2.2 End-to-end matching methods

Here models are discussed that compute a matching score between the context and candidate response. In outlines, these models look the same as the vector space in that they compute a matching/similarity score between two vectors. However, the matching models consist of two networks (encoders) that encode the context and response into a vector (instead of TF-IDF). These models can backpropagate the result through the network (hence it learns), whereas the vector space model is static (it cannot learn).

The input to models like these is usually word embeddings. Word embeddings are vectors that represent a word [37]. These are the latent representations of the corresponding words, and arithmetic operations are possible between words with these vectors. In equation 3 an example is given. When adding the embeddings king and woman to each other, one would get the queen vector.

$$embedding(king) + embedding(woman) = embedding(queen) \tag{3}$$

There are different word embeddings techniques like word2vec from [37] or glove embeddings [42]. Word2Vec uses a combination of the continuous bag of words (CBOW) and skip-gram model. Both of these are shallow neural networks. CBOW will predict the current word based on the context (words surrounding the current word), where the skip-gram model will predict the surrounding words given the current word. This process is shown in figure 2. Then the hidden representations of CBOW or the skip-gram model can be used as word embeddings (the projection layer in the figure).

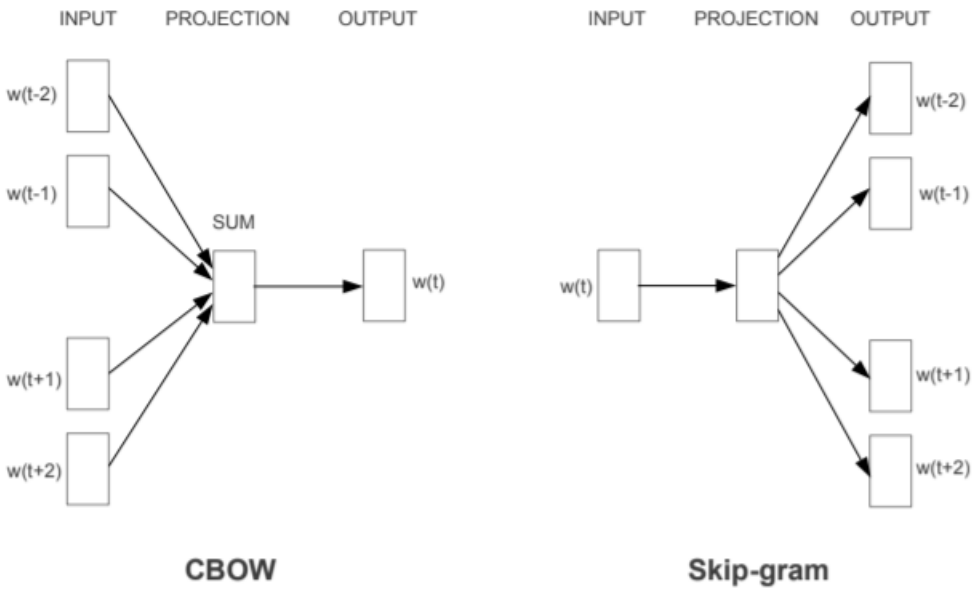

Figure 2: CBOW and Skip-gram, from [37]

GloVe, however, uses matrix factorization techniques on the word-context matrix. A large matrix is constructed with rows representing a word and columns representing a context. The

value in row i column j then stands for the number of times we see this word in this context. This "big" matrix is then factorized to yield a lower-dimensional matrix, where each row now is a word embedding.

Matching models can be divided into single turn matching models and multi-turn matching models. Single-turn models match the response only once with the whole context or with only the last utterance. In comparison, multi-turn models match the response with each utterance (in the context) separately and then aggregate these scores into a final score. These categories both will be discussed in more detail in the following two sections.

End-to-end single-turn matching models The first end-to-end single-turn matching model was the dual encoder of [34]. Here, they represent the context and candidate responses using an embedding layer (that is finetuned during training). They use a siamese network consisting of two RNNs with tied weights to produces encodings (vectors) for the context and response. With these two vectors, the matching score is calculated in the following way:

$$match(\mathbf{c}, \mathbf{r}) = \mathbf{c}^T \mathbf{M} \mathbf{r} \tag{4}$$

This approach is also used by [8] and has been widely used as a scoring model in information retrieval and question answering [18]. Here $\mathbf{c} \in \mathbb{R}^d$ vector representation of the context, $\mathbf{r} \in \mathbb{R}^d$ that of the response, and $\mathbf{M} \in \mathbb{R}^{dxd}$ is a matrix that is learned by the model. This can be thought of as a generative approach. Given some input response, we generate a context vector with the product of $\hat{\mathbf{c}} = \mathbf{M}\mathbf{r}$. When $\hat{\mathbf{c}}$ and $\mathbf{c}$ are similar, the dot product between the two vectors will give a high score. [34] evaluated the dual encoder by feeding it a list of candidate responses (of size 10). Hence, one context is compared with ten responses. After this comparing the responses are ranked accordingly to the binary probability given by the model. On this ranking, they calculated recall 1, 2, and 5.

This is a framework has been the basis of multiple single-turn matching models. [28] tried replacing the "normal" RNN layer with a CNN or Bidirectional LSTM layer. [55] not only changed layer types but also changed the loss function from binary cross-entropy to a hinge loss function utilizing cosine similarity. Also, they researched adding a CNN or attention layer on top of a Bidirectional LSTM layer. [60] proposed a system called MV-LSTM. This allows the matching to be based on the positional sentence representations. In [10] they replaced equation 4 with a cross product between vectors c and r connected to a fully connected layer which "predicts" the similarity score.

End-to-end multi-turn matching models [67] also exploited the word level similarity between the context and responses. Two different similarity scores are computed in this model, and the model is trained to minimize two losses. Both word and sequence level similarities are computed. This model is called Sequential Marching Network (SMN).[63] improved the SMN by matching the context with the response on multiple levels of similarity. Deep Attention Matching Network (DAM) is an extension of the SMN. DAM is created by [68] and is entirely based on the attention mechanism [4]. With this, they want to eliminate the limitations of recurrent neural networks.

2.3 Generative-based models

Generative chatbots are like humans in that they generate responses word by word (however, how they do this generation is not necessarily human-like). At first, they process the incoming text, then generate a response to this text. This field is extensively researched in the last years and applied in many different domains like machine translation, image captioning, etc. There are two different architectures of generative models: modular and end-to-end.

2.3.1 Modular methods

A modular architecture is composed of 4 key components:

1. Natural Language Understanding (NLU): Parses the incoming question (text) into predefined semantic slots.
2. Dialogue State Tracker (DST): It processes the dialogue history and together with the (new) information of the NLU, it will output the current dialogue state.
3. Dialogue Policy Learning (DPL): Uses the current state to choose the next action
4. Natural Language Generation (NLG): The NLG will generate a response based on the chosen action.

This architecture has two main limitations [66]. One is the credit assignment problem, where the user's feedback is hard to propagate back through the architecture. The second problem is the interdependence of the components. The input of a component is the output of another component; hence when one component changes, the whole system changes. Nevertheless, because the components are often made individually, an improvement of a component does not necessarily mean improving the whole system. When one module is adapted to a new environment, all the modules should be adapted accordingly to ensure global optimization. Adapting components to a new environment requires significant human effort. With the rise of neural networks, an increasing interest came in joining the multiple components in one module. This resulted in the end-to-end architecture.

2.3.2 End-to-end generative methods

End-to-end models, as the name already says, do everything in one model. This kind of model is widely adopted in many domains is it allows to alleviate the limitations of the modular architectures. Therefore, less human interaction is needed. Also, when the whole model is differentiable, the model can be optimized through backpropagation [21]. This is the main advantage of end-to-end models over modular ones.

End-to-end generative models are trained on a large human-to-human conversation using deep learning networks to learn to generate human-like text [48]. The encoder-decoder model is one of the most used and powerful models for dialogue generation [15], [59]. [53] introduced the first framework of the encoder-decoder model called sequence-to-sequence. The idea behind it is to map the input sequence to the output sequence. This "simple" framework has impressive performances in different NLP tasks, machine translation [65], dialogue generation [39], image captioning [44], etc. However, generating responses is a difficult task. Because of the problems with handling long conversations due to the memory issues of LSTM and RNN [5] utterances generated by this framework were mostly short, and general [59]; [52].

Many extensions and improvements have been introduced to the encoder-decoder framework. An important extension is the use of the attention mechanism [4], [36] what enables the framework to better handle long sequences [56].

2.4 Hybrid models

Generative and retrieval models can also be combined into hybrid models. They are combining the advantages of the two different approaches. For example, retrieval-based models often give precise but dull answers, whereas generative-based models tend to give fluent but meaningless responses. [51] propose a combined model that retrieves a response. With the original (input)

text, this response is fed into a generator that returns a generated response. The retrieved and generated response are then ranked using a re-ranker. In the case of [51], they used a Gradient Boosting Decision Tree (GBDT) using several high-level features like term similarity, Topic similarity, etc., as re-ranker.

3 Exploratory Data Analysis

In this section, customer service data is analyzed from a customer service platform called FreshDesk. Freshdesk enables a company to interact with the customer in many ways, like email, a portal, or Facebook. This section provides details regarding the primary datasets and data cleaning procedure used during this research.

3.1 Raw data

The raw data collected from the Freshdesk Application Programming Interface (API) consists of two files. One file contains the customers' first question, and another contains the conversations between the customer and agent. With conversations meaning all the responses, the agent and customer send to each other. The file with the first question will be referred to as a ticket file (ticket is is a term in Freshdesk). The other file will be called the conversation file. Both files will be further described in the following sub-sections.

3.1.1 Ticket file

The ticket file consists of the first question the customer asks the agent. Some other interesting data that this file contains are type and subtype. This data is filled in manually by the agents that treat the ticket. The columns of the raw ticket file are given in table 2.

Column type	**Columns**
String	subject, description_text, due_by, fr_due_by, fwd_emails, relpy_cc_emails, type, tags, cc_emails, to_emails cf_technical, cf_payments, cf_filmseries_related, cf_device, cf_brand, cf_other, cf_refund, cf_cancellation, cf_screenshot, cf_acc_and_subs
Boolean	fr_escalated, is_escalated, spam
ID (key)	email_config_id, id, group_id, product_id, requester_id, responder_id, status, priority, source, tweet_id
Datetime	created_at, updated_at

Table 2: Ticket dataset and columns

As one can see in table 2 subtype is not present. This is because the subtype is distributed over different columns. All columns where the name start with "cf" is a subtype column. The different types (including a short description) that are present in the data are:

1. Account and subscription: This contains questions about their account, such as 'how do I cancel' or 'how do I become a member.'
2. Collaboration: Are messages of people that want to collaborate with DutchChannels.
3. Film/series related: Questions about the content of the platform.
4. Payments: Payment related questions like why the payment failed or different kinds of payment methods
5. Technical: Questions about the technical part. These mostly question about how to watch on tv and malfunctions. Also, questions like how to use subtitles belong to this type.
6. Other: All other types are in this category. This also contains Facebook comments.

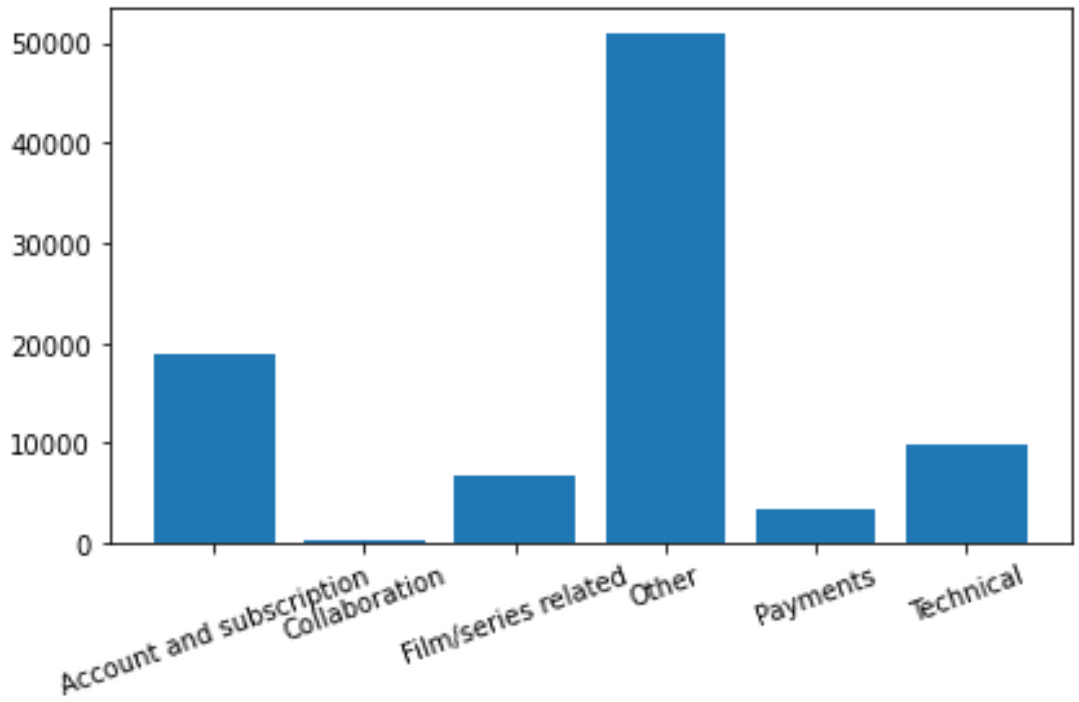

Figure 3: Distribution of the raw ticket dataset

The data is gathered from different sources. The most important sources are email, portal, and Facebook. Before cleaning, the ticket file consists out of 93.161 rows. The data collected comes from the two different platforms (NFN and WL) and consists of 4 different languages. The distribution of the raw data is given in table 3.

NFN		WL	
Dutch	31,182	Dutch	19,116
Swedish	5,740	Swedish	1,303
Norwegian	4,872	Norwegian	567
English	20,558		
Total	66,470	**Total**	22,021

Table 3: Distribution of the ticket dataset (before cleaning)

3.1.2 Conversation file

This file consists of all the utterances that are done on a ticket. This means all the answers an agent (customer service employee) gives on the ticket and all the responses CS gets from the customer. Because of this, the file has significantly more rows than the ticket file. The conversation data consists of 157.370 rows. To this data, no metadata is added by the agents. The raw ticket file is shown in table 4.

Column type	Columns
String	body_text, from_email, bcc_email, support_email, cc_emails, to_emails
Boolean	private, incoming
ID (key)	id, user_id, ticket_id, source
Datetime	created_at, updated_at

Table 4: Conversation dataset and columns

3.2 Data cleaning

Before one can work with the obtained data, much cleaning is required. In the next section, the process of cleaning the ticket and conversation file is described.

3.2.1 Cleaning ticket data

The data has to be cleaned in multiple ways. In the ticket file, questions from all kinds of platforms are collected. Because of this also Facebook comments (under posts of DutchChannels) are present in the data. These are just comments (no questions) and hence do not carry any value for training the chatbot. After deleting these comments, the leftover part is merged with the conversation file (with an inner join). This way, all the tickets for which there is no response are deleted. The distribution of the tickets over the channels and countries is shown in table 5.

NFN		WL	
Dutch	12,448	Dutch	8,060
Swedish	3,303	Swedish	1,009
Norwegian	3,067	Norwegian	432
English	7,818		
Total	28,967	**Total**	9,813

Table 5: Distribution of the ticket dataset (after cleaning)

In total, 38,780 tickets with responses of agents are collected. However, this does not mean that the final data set has 38,780 question-answer pairs. Still, many of these "first" questions are not questions but comments or other things (like a question to cooperate). Also, in the conversation following a ticket, customers sometimes ask another question, which means that one ticket with its conversation can have multiple "interesting" question-answer pairs available for the final data set.

3.2.2 Cleaning conversation data

All the utterances of the agents are matched with the predefined responses. This is done so that the chatbot can give general/standard responses to question, and this way, the problem can be transformed into a classification problem. The matching is done in the following way. First, a document vector was created for all the utterances and standard responses. This document vector was obtained via TF-IDF, and word2vec [37]. A textual question is transformed into a vector using the TF-IDF technique. The final text vector was obtained by multiplying the TF-IDF scores with the corresponding word2vec embeddings and adding the resulting vectors to create one vector of length 300. Then the soft cosine similarity score is computed between the

utterance and all the standard responses. The soft cosine similarity is a lot like the "regular" cosine similarity score (described in section 2.2.1).

$$\text{Soft Cosine}(a,b) = \frac{\sum\sum_{i,j=1}^{N} a_{ij}b_{ij}}{\sqrt{\sum\sum_{i,j=1}^{N} s_{ij}a_{ij}a_{ij}}\sqrt{\sum\sum_{i,j=1}^{N} s_{ij}b_{ij}b_{ij}}} \tag{5}$$

Where $s_{ij} = sim(f_i, f_j)$, here f_i and f_j are features corresponding to the basis vectors and $sim(.)$ is a similarity measure. When there is no correlation between f_i and f_j this equation is the same as the "normal" cosine similarity formula 4.

The idea behind equation 5 is to compute the cosine similarity score of the text vectors projected on a non-orthogonal basis, where the angle between two basis vectors is derived from embeddings vectors of the corresponding words [50]. Because of this, text vectors with different words but with (roughly) the same meaning will get higher similarity scores than with using "normal" cosine similarity. To use word2vec and hence the technique mentioned above, the languages other than English are translated to English. This is done via the Microsoft translator.

To evaluate the matchings made with the technique above, bins are made on the similarity score. For the different combinations of country and channel, these bins are evaluated. These bins are too big to evaluate every question. Hence (random) samples out of the bins are taken and evaluated. For all combinations except WL-NO, samples of size 60 were taken. For WL-NO, samples of size 20 were taken. In table 6 the results of the evaluation are given. The margin of errors are calculated with the following equation:

$$z * \sqrt{p(1-p)} / \sqrt{\frac{(N-1)*n}{N-n}} \tag{6}$$

Where p is the sample proportion, z the z-score associated with a level of confidence, n the sample size, N the population size.

Channel - Country	Bin size	Sample size	Margin of error (95% confidence)
NFN - NL	1,825	60	12.5%
NFN - NO	503	60	11.9%
NFN - SE	459	60	11.9%
NFN - EN	1,071	60	12.5%
WL - NL	1,222	60	12.4%
WL - NO	59	20	18%
WL - SE	147	60	9.8%

Table 6: Bin and sample sizes with their margin of error

In figures 4 and 5, the percentage wrongly matched utterances per bin are given.

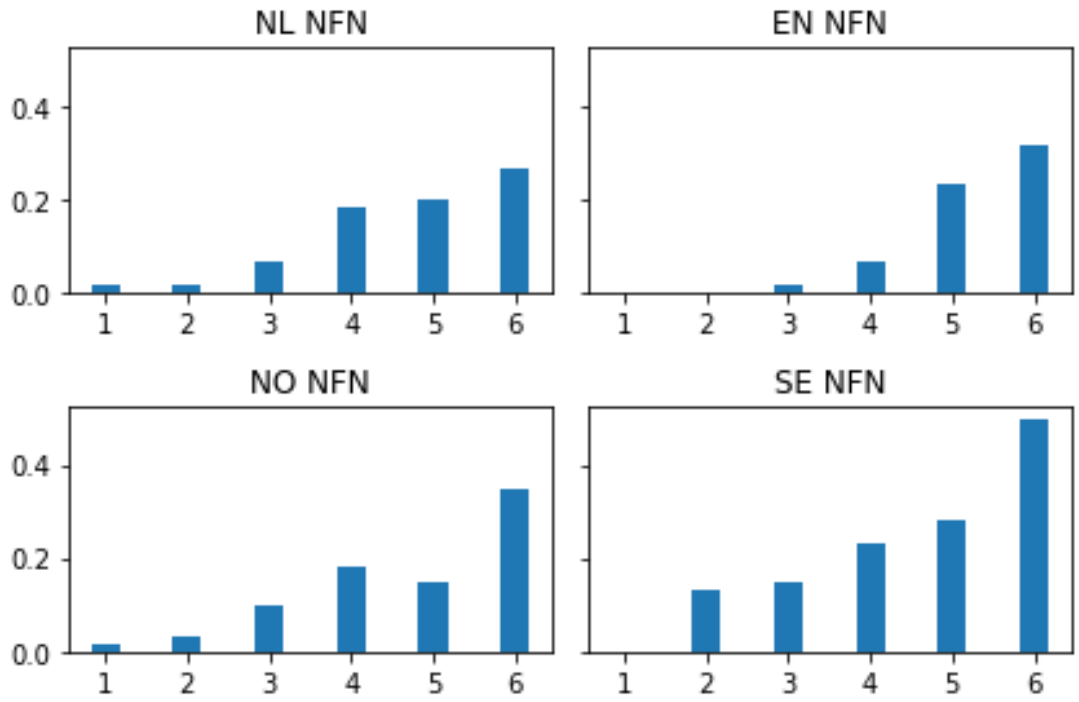

Figure 4: Percentage wrong in NFN bins (1 is "best" bin)

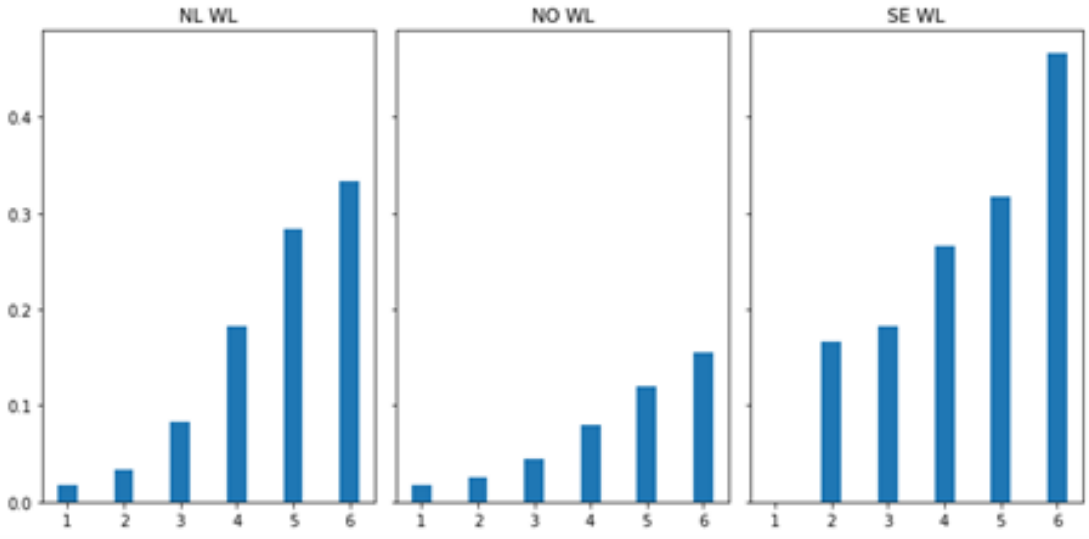

Figure 5: Percentage wrong in WL bins (1 is "best" bin)

It is, of course, essential for the final performance of the chatbot to have a small number of wrongly matched data present. As can be seen in the two figures is that the matching went pretty well in the "best" bins but pretty quickly made too many errors. In table 7 the bins that

are kept in the final data set are shown. We choose to keep the two best bins for all channel country combinations to have sufficient data present from all combinations in our final dataset. When the third-best bin was below 5% error rate, this bin is also added to the final dataset.

Channel - Country	Bins taken
NFN - NL	[1, 2, 3]
NFN - NO	[1, 2, 3]
NFN - SE	[1, 2]
NFN - EN	[1, 2, 3]
WL - NL	[1, 2, 3]
WL - NO	[1, 2]
WL - SE	[1, 2]

Table 7: The bins that are chosen for the final dataset

This resulted in a dataset of 17.543 matched utterances. This contains an estimated percentage of wrongly matched responses of roughly 3.5% with a margin of error of 3.01%. This margin of error has a 95% confidence. Some manual manipulation is done after testing models resulting in a final set of 16.905 records.

3.3 Data Analysis on final dataset

In tables 8 and 9 the distribution of the five least and most occurring question types are given. As one can see, the dataset is highly imbalanced, with one type (cancel subscription) by far the biggest one. The distribution of all question types are given in table 22 in appendix A. After the top 3 most occurring types, there is a big gap to the other questions types. After this, there is a standard decrease in the size of the types, with the least occurring type (app available) having a size of only 15 rows. Figure 6 also confirm the highly imbalanced nature of our data.

Question type	Number of occurrences	Percentage of dataset
App available	15	0.0886%
Devices compatible	15	0.0886%
What is	15	0.0886%
Refund after trial period	16	0.0945%
Why subscription cancelled	17	0.101%

Table 8: 5 least occurring question types

Question type	Bins taken	Percentage of dataset
Cancel subscription	5675	33.55%
Technical issues	2727	16.12%
Account details	2726	16.12%
Watch tv	597	3.53%
Login details	515	3.04%

Table 9: 5 most occurring question types

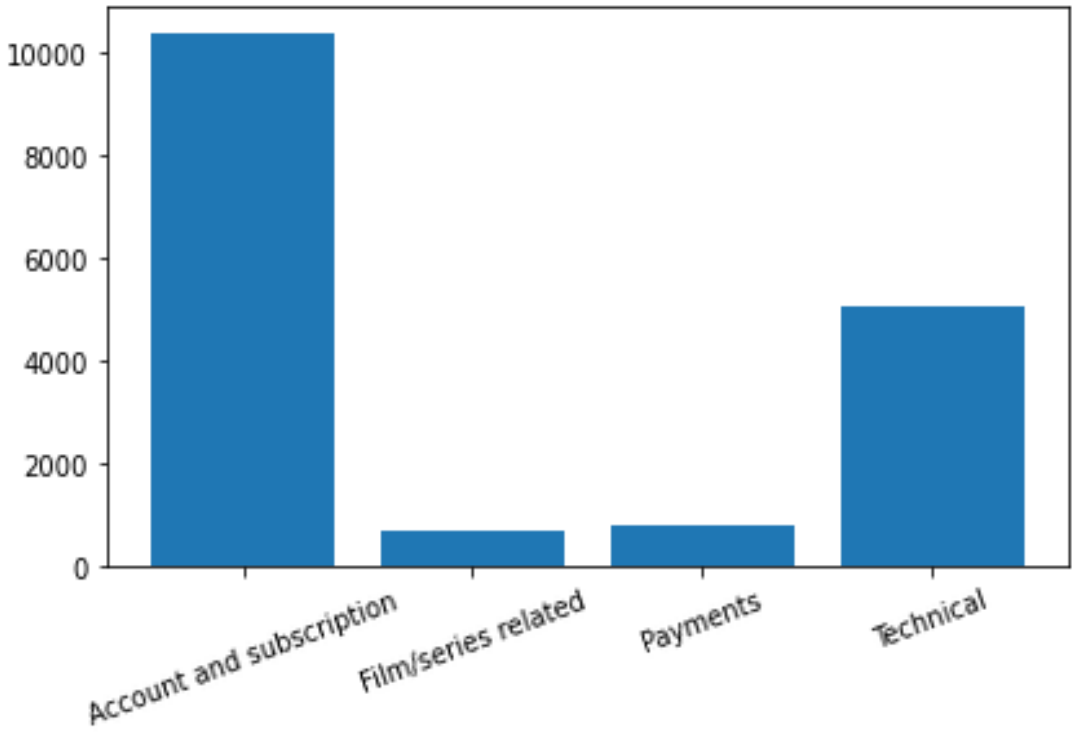

Figure 6: Distribution of the cleaned ticket dataset

To get a feeling of what kind of words are used in the questions, in figure 7 the most common unigrams are given. In figure 22 in appendix B the most common bigrams are shown [1].

[1] This is after lemmatization and stopword removal

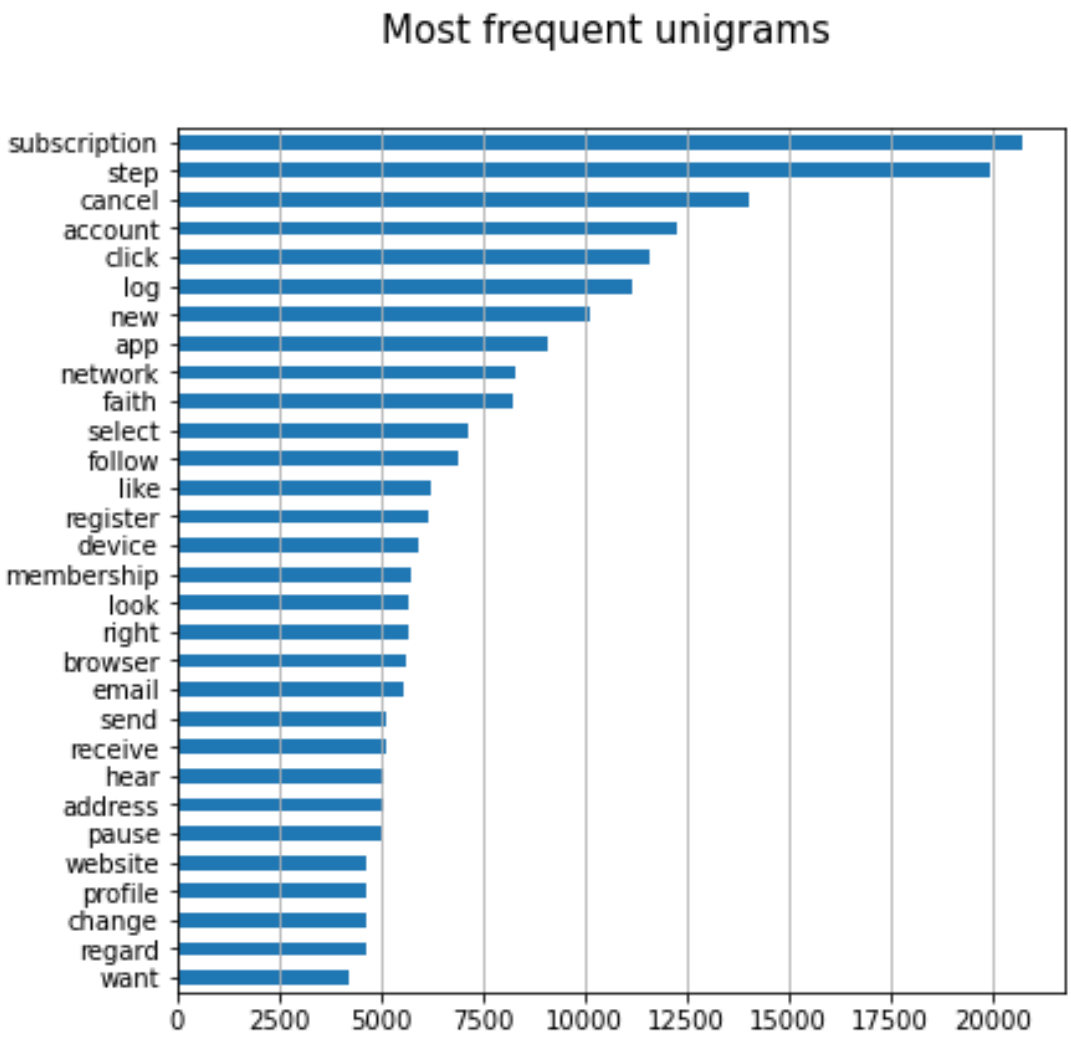

Figure 7: Most commen unigrams present in questions

One can see that the most occurring unigrams and bigrams correlates with the most occurring question type (cancel subscription) with unigrams like subscription and cancel and bigrams like cancel subscription. The unigrams and bigrams fit the first expectation one has with the most occurring words in this domain.

In figure 8 the distribution of the number of words is given. As one can see, all the histograms are (highly) left-skewed. This was also expected from the data as the customer asks questions succinctly. The boxplot (right figure) shows the same result but nicely shows that the data still contains some outliers (long questions). These outliers will not be deleted from the data as we assume that some customers also can ask long questions to the chatbot. In appendix E the same figure (as figure 8) are shown but then for the number of characters and sentences. These figures confirm the situation of figure 8.

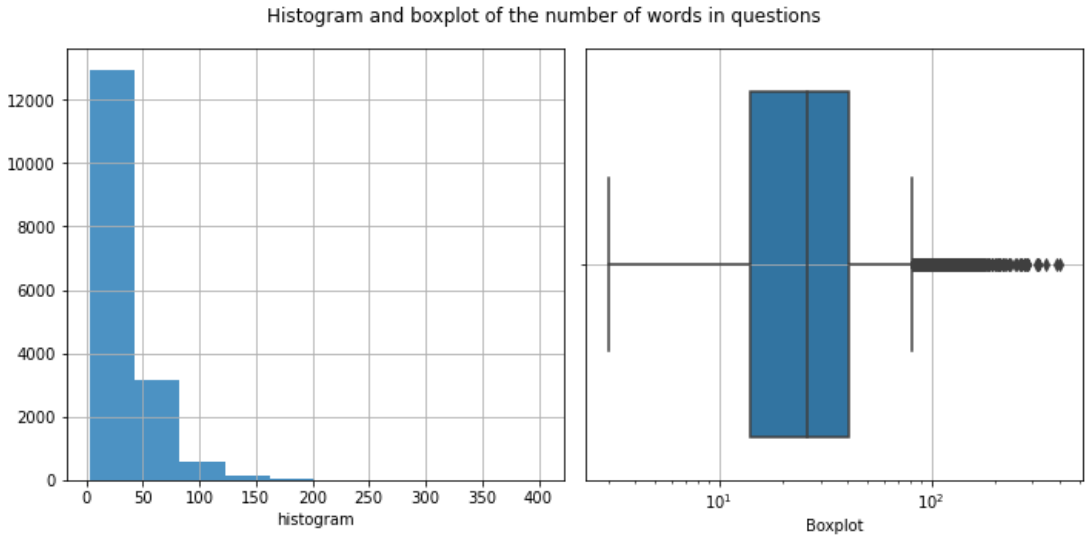

Figure 8: Histogram and boxplot of word counts

4 Experimental Setup

The main focus of the experiments is to propose a model that retrieves the correct answer for a question. Figure 9 describes the workflow used in this research.

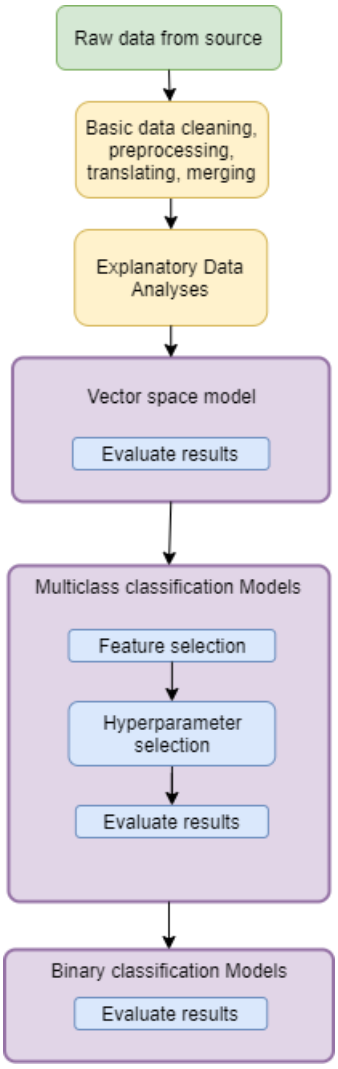

Figure 9: Workflow used in research

First, some minor preprocessing steps are performed on the data, after which an explanatory data analysis is conducted. Hereafter the project undergoes an experimental phase. The first step in the experimental phase is to create some baselines and produces results with these baselines. Hereafter, the multi-class classification models will be made. These models will classify an incoming question to one of the created classes. Then the chatbot gives the corresponding answer back. At last, the binary classification models are tested and created. These models will match an incoming question with all the candidate responses.

4.1 Implementation of the two kinds of classification models

In this section, the implementations of the two different kinds of researched classification models are discussed.

4.1.1 Multi-class classification models

In figure 10 an overview of the implementation of the multiclass models is given. The first step is data preprocessing. This is done with the pipeline described in section 4.2.

In the second step, the features are created from the processed text. These features include TF-IDF scores and LDA topics. More about these features are described in section 5.5.1.

In the multiclass classification model step, a model will classify what kind of question the user asks.

Finally, the corresponding answers of the predicted class are returned to the user.

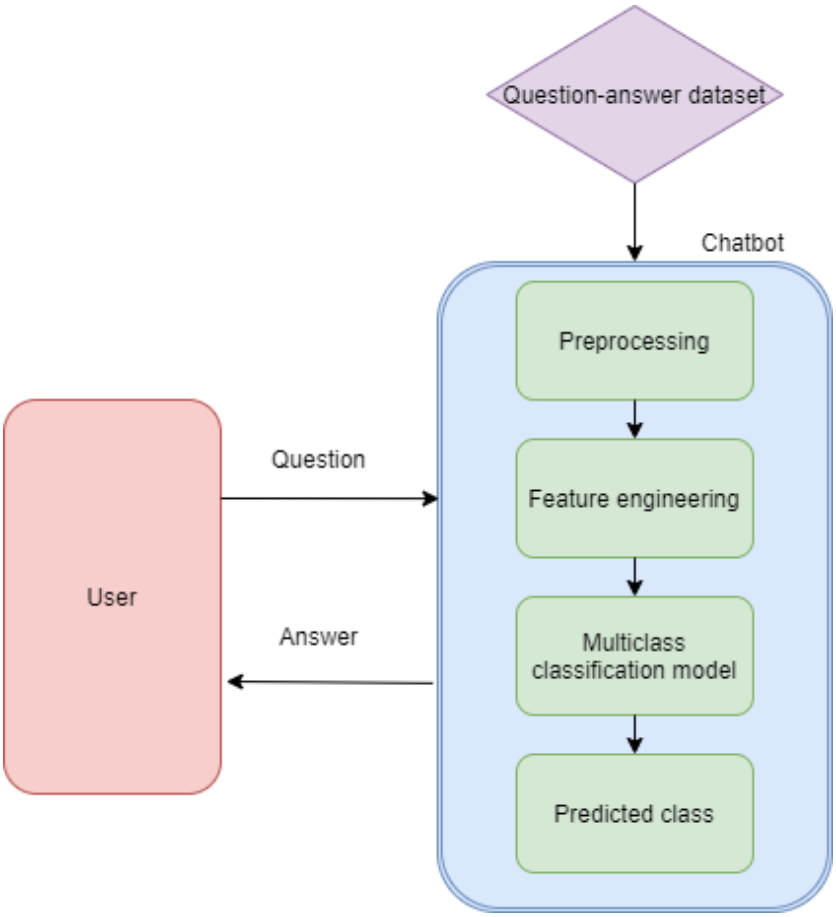

Figure 10: Implementation multi-class classification models

4.1.2 Binary classification models

In figure 11 an overview of the implementation of the binary classification models is given. The first step is converting the words to the corresponding word embeddings.

In the binary class classification model step, a model will score all the predefined answers. These represent how good this answer is for the question.

Finally, the predefined answers are ranked on the scores, and the answer with the highest score is returned to the user.

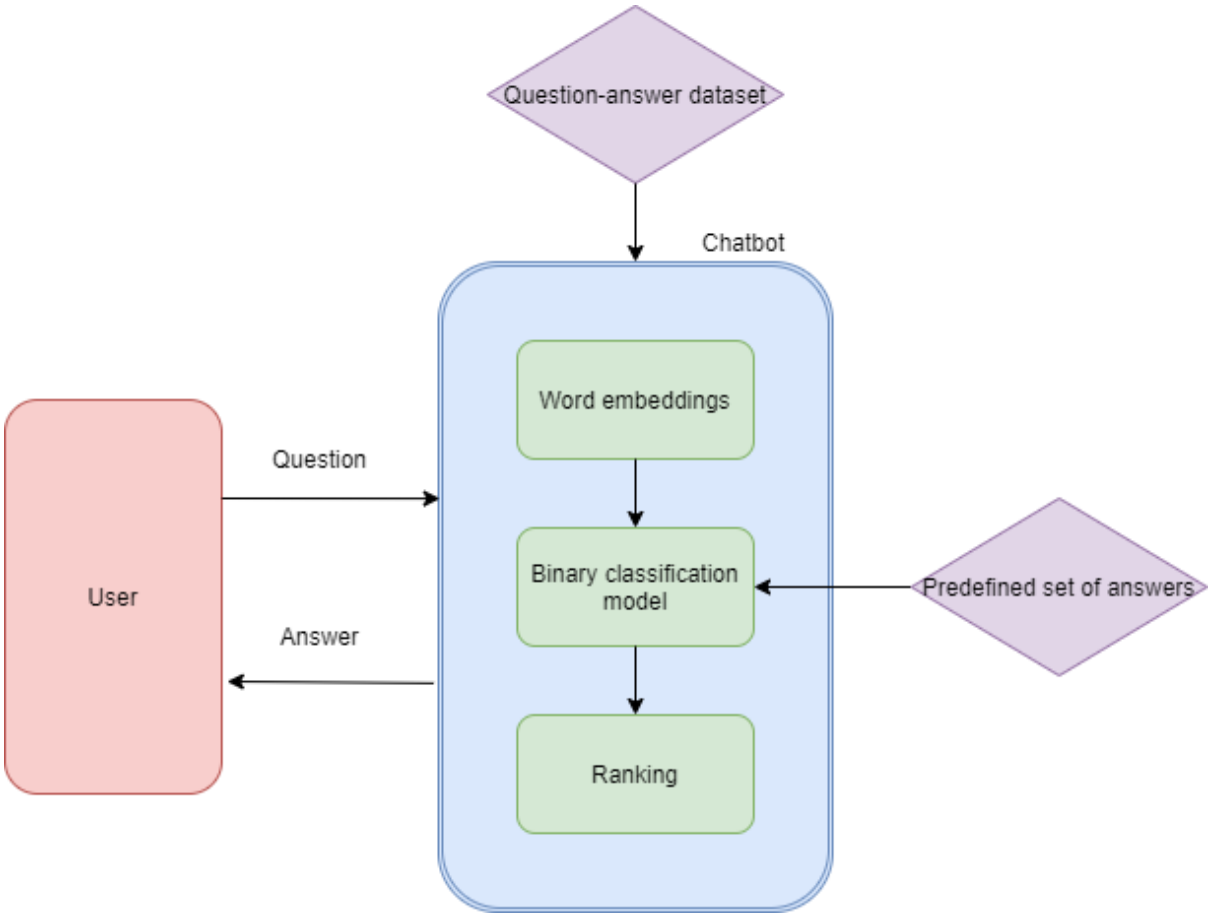

Figure 11: Implementation binary classification models

4.2 Preprocessing

To enhance the performance of the models, the raw text will first undergo an NLP pipeline shown in figure 12. The transformations are done on the right side (blue boxes) on the example question "Hello, why is my subscription of New Faith Network cencelled?".

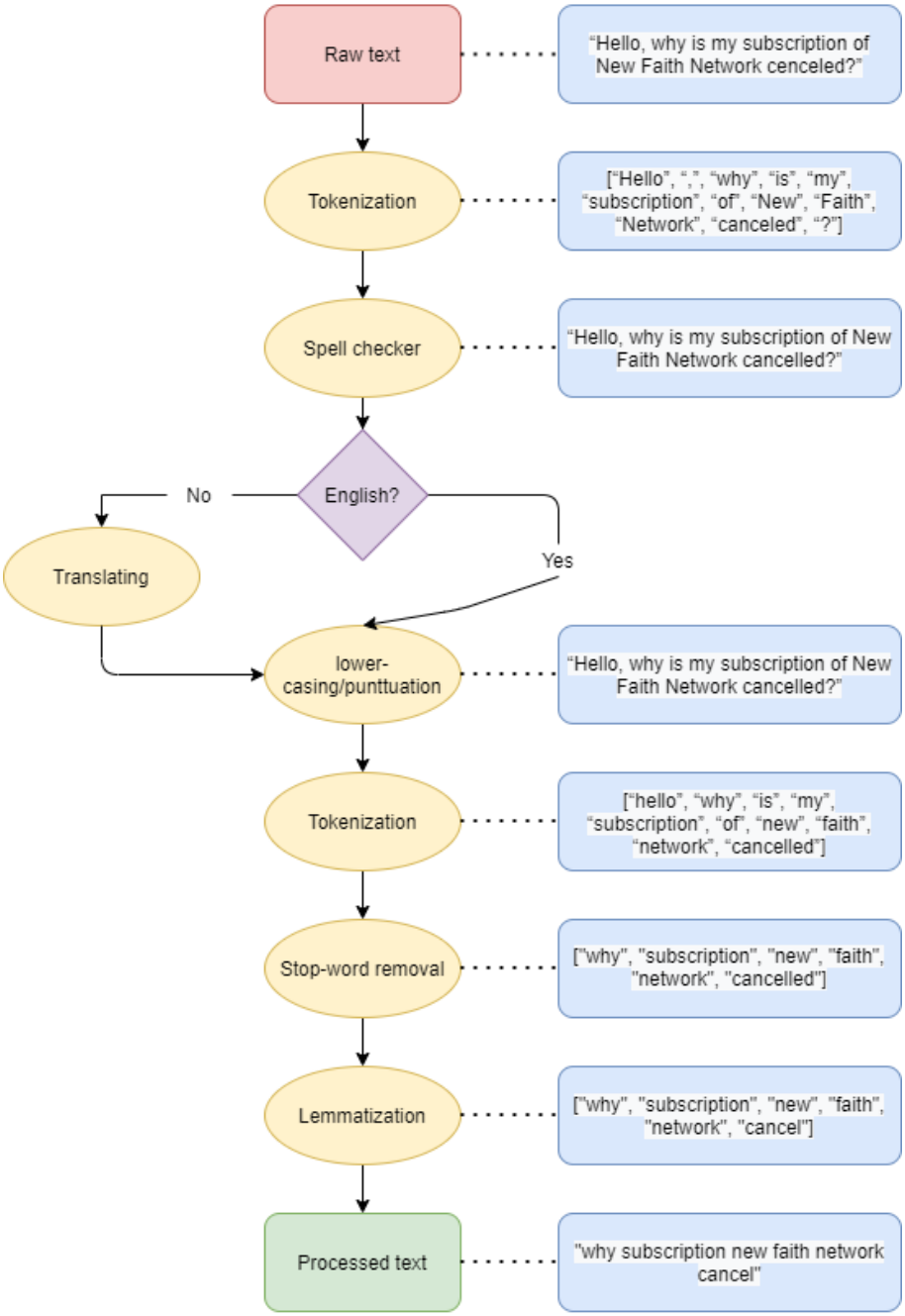

Figure 12: Preprocessing pipeline of text

First, all the texts will be broken down into components. In this research, this is done by separating the words on white space or punctuation. Then a spell checker is used to check

and corrected misspelled words. The Hunspell [2] spell checker is used in this research. This is a widely adopted spell checker used by LibreOffice, OpenOffice.org, Mozilla Firefox 3 Thunderbird, Google Chrome. After spelling, the tokens are concatenated and sent to a translator if it is not English. The translator used in the research is the Microsoft Translator. All the capital letters and punctuation is preserved because the Translator has a better performance with these present in the text. After the translator step or check, the text is lowercased and punctuation is removed. Then, the text is tokenized again. Hereafter all the stopwords are removed from the text. This is done because stopwords, by definition, are meaningless words that have low discrimination power [33]. At last, the leftover words are lemmatized.

4.3 Cross validation procedure

First, the dataset is split into a train and a test set. Table 10 gives the sizes of these two sets. This is done in a stratified manner. This means that the distribution (of the target variable) is kept the same in both the training and test set as in the original dataset.

Dataset	Size (n)	Size (%)
Train	12,685	75%
Test	4,229	25%

Table 10: Splitting procedure for dataset

After splitting the test set, a cross-validation scheme was used to evaluate the models and hyperparameters. The splitting of the training and validation sets was stratified to ensure that all classes are present in the train and validation set. Furthermore, a 3-fold cross-validation scheme is used. Because of the highly imbalanced nature of the dataset oversampling is used to balance this. In the cross-validation scheme, only the training set is oversampled, and the validation set is not. This is done because we want the performance of the validation set to imitate the test set's performance. Hence, the distribution of the validation set should not be altered (so it is as close as possible to the distribution of the test set). The final model will be trained on the whole train set where all classes are oversampled to the exact size of the biggest class. A drawback of this is that the model could overfit the examples that are sampled many times.

4.4 Performance metric

The fact that two different kinds of models will be compared gives an exciting dynamic to the problem. Because of this, both kinds of models will be evaluated with metrics usually used in both fields. However, some of these metrics have a lot in common (even share the same name).

4.4.1 Classification metrics

Four of the most used classification metrics will be used. The formulas of these metrics are given below. In these formulas, TP stands for true positive, TN for true negative, FN for false negative, and FP for false positive.

Accuracy proportion of rightly classified predicitions over all predictions.

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{7}$$

[2] http://hunspell.github.io/

Recall proportion of real relevant instances that where rightly classified as relevant.

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

Precision proportion of classified relevant instances that where really relevant.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

F1-score the harmonic mean of precision and recall.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{10}$$

4.4.2 Ranking metrics

Because this problem has one ground truth (instead of multiple ground truths), some standard metrics like Precision@k and mean average precision (MAP) are not used.

Recall@k: The proportion of true labels (ground truths) present in top-k recommendations.

$$Recall@k = \frac{\#\text{ of true labels captured in top k}}{\#\text{ of true labels}} \tag{11}$$

MRR (mean reciprocal rank): The average reciprocal of the ranks of the first accurate label for all contexts.

$$MRR = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{1}{rank_i} \tag{12}$$

Here $|C|$ are the number of contexts, and $rank_i$ is the rank of the first found true label for context i^{th}.

5 Models

In this section, the models that are researched are discussed. Also, the implementation and the tuning of the models will be given.

5.1 Baseline models

Three different baseline models are used in this research. A random model, max class model, and a bag of words model. In this section, the three models will be described.

5.2 Random model

This model, together with the most occurring model, will not use any information of the textual question. This model gives a random response back from the candidate responses. Hence, the chance that this model gives the correct responses to a question is

$$\frac{1}{length(candidate\,responses)}$$

5.3 Max class model

This model always predicts the most occurring class. The ranking version of this model will consistently rank the most occurring class as one, then invariably the second most occurring class as two, and so on.

5.4 Bag of words model

This model is more enhanced as the random model and is a frequently used baseline in NLP [10] [35] [25] [19]. In section 2.2.1 this model is described in further detail.

5.5 Classification models

Here the different classification models are discussed. Besides a description and implementation also some results of the hyperparameter search will be given.

5.5.1 Features

Features should be made for the multiclass classification models. In this research, two kinds of features are made (and concatenated):

- TF-IDF features
- LDA features (topic modeling)

For all the uni- and bigrams, scores are calculated. However, not all these scores are input to our models. Because the dimension of our input vector would be too large, we conducted a chi-square test to reduce the dimension of the input vector. With this test, we test the independence of the target variable with the feature (one for one). When the p-value of this test is below our α we reject the null hypothesis and say that there is a dependence between the target variable and the feature. One drawback of this method is that it tests the dependence on a single feature. However, a combination of features could have valuable predicting power. In our research, we chose an α of 0.025. This resulted in 678 TF-IDF features.

To this TF-IDF vector, we will add 20 topic features. These topic features are created with Latent Dirichlet Allocation (LDA) [7]. LDA is a probabilistic model used to discover "topics" that describe the entire corpus. It models each document as a mixture of K topics. Where K is a predefined number of topics, each of these K topics is a mixture of terms present in the corpus.

The idea behind adding these 20 topics is that with these topics, filtered words (by the chi-square test) will still influence the prediction. Adding these 20 topic features to the 678 TF-IDF features results in a final feature vector of 698 features.

5.5.2 Hyperparameter search

For all models, a random search as proposed in [6] is done over a predefined grid. Random search is chosen because it is easy to carry out (and hence implement) and is more efficient than grid search. Also did a random search found better models in most cases than grid search comparing the experiments of [31] and [6]. Suggesting that random search has sufficient power to find the "best" model or one of the best models when doing enough searches.

The sci-kit learn implementation of the random search was used to perform the hyperparameter search. For all models, 50 random searches are conducted.

5.5.3 K nearest neighbours

Description K nearest neighbours [54] is a type of instance-based learning or non-generalizing learning. This algorithm does not attempt to "learn" the underlying probability distribution with a general internal model. However, it stores the training data. Then classification is computed from a simple majority vote of the k closest (training) data points.

Implementation The sci-kit learn implementation in Python was used to obtain model results for k nearest neighbors.

Hyperparameter selection K nearest neighbors is a straightforward model with a small number of hyperparameters. The number of closest neighbors that are considered (k) is an essential hyperparameter and is highly dependent on the problem. Besides k, two other hyperparameters were investigated. The weight each neighbor gets in the "voting" and how the distance of the neighbors to the point of interest is calculated. In table 11 the hyperparameter grid of the random search is given.

Hyperparameter	range
Number of neighbours	[1, 3, 5, ... , 49]
Weights	[uniform, distance]
Metric	[euclidean, manhattan, minkowski]

Table 11: Hyperparameter grid for k nearest neighbour

The best model found in the random search had an f1-score of 0.503 and the following hyperparameters:

- Number of neighbours: 9
- Weights: Distance
- Metric: Euclidean

5.5.4 Logistic regression

Description Logistic regression [41] is a classification model, although his name suggests otherwise. The probabilities describing the outcome of a single point are modeled using a logistic function. Typically, it performs a binary classification. However, it can also perform multi-class classification. This can be done by minimizing the multinominal loss across the entire probability distribution or fitting a binary problem for each label and then using the one-vs-rest (OvR) scheme.

ImplementationThe sci-kit learn implementation in Python was used to obtain model results for linear regression.

Hyperparameter selection Three different hyperparameters are investigated for the logistic regression. Different ways of penalizing a wrong matched are tried. Also, the regularization parameter C and algorithm used to optimize (solver) are tested. C is the inverse regularization strength. Thus a higher value specifies less regularization. The solvers and other hyperparameters that are tried are given in table 12. Using the solvers newton-cg and lbfgs and minimizing the multinominal loss (instead of using one-vs-rest), the model learns a true multinomial logistic model [3]. This indicates that the probability estimates should be calibrated then the one-vs-rest scheme.

Hyperparameter	range
Penalty	[none, L1, L2]
C	np.logspace(-4, 4, 20)
Solver	[newton-cg, lbfgs, liblinear]

Table 12: Hyperparameter grid for logistic regression

The best model found in the random search had an f1-score of 0.451 and the following hyperparameters:

- Penalty: L2
- C: 29.764
- Solver: Newton-cg

5.5.5 Random forest

Description Because random forest [11] uses decision trees, the decision tree will be highlighted first. A decision tree uses the entire dataset to make a prediction. Decision trees consist out of root, decision, and leaf nodes. In the root and decision nodes, the splits in the tree are done. The root node creates the first split in the tree, the splits that follow are made in the decision nodes. What split is done in the nodes is determined with the greedy algorithm. When no more splits made can be achieved, the decision tree has come to an end. The last nodes of a decision tree are called leaf nodes, and here the prediction is generated. When the problem is a classification problem, the prediction will be the majority vote of the data points present in the leaf node as shown in 13. Random forest will create several random subsets (of the data and the features), after which different decision trees can be fitted to the different subsets. The predictions of the several decision trees will be combined, after which this combination will produce the prediction of the random forest.

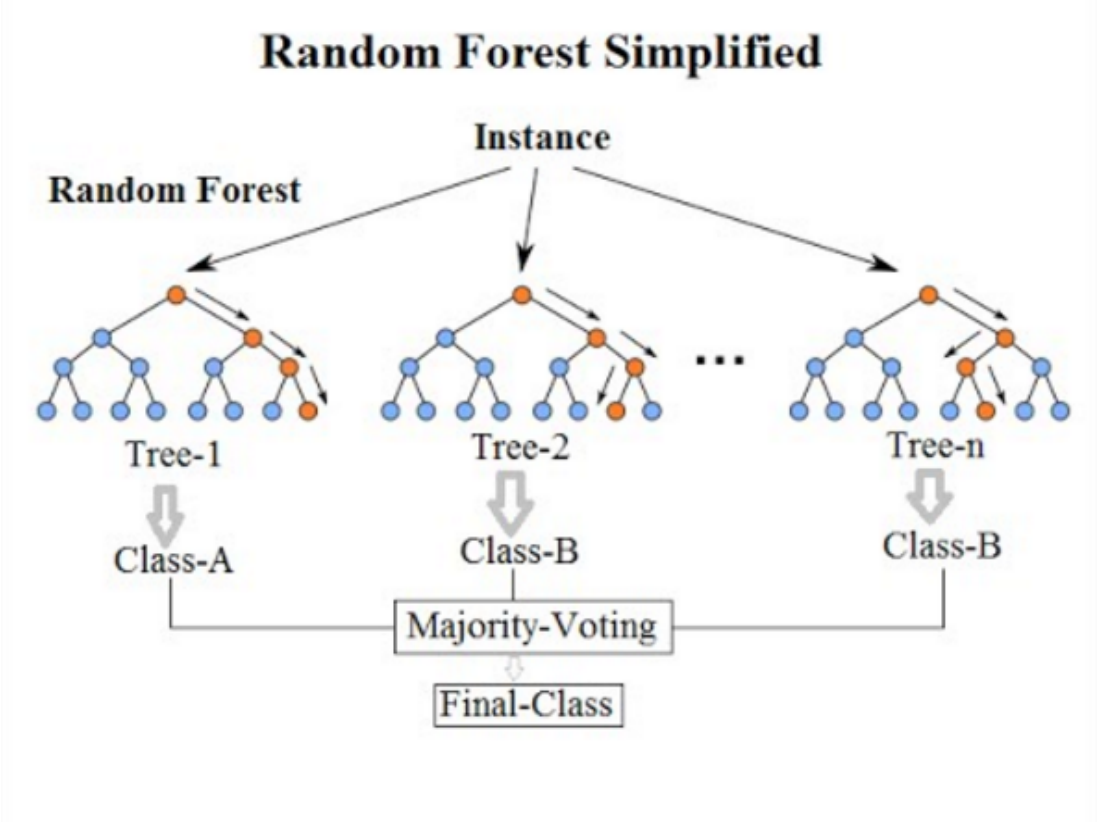

Figure 13: Random forest classifier, from [27]

Implementation: The sci-kit learn implementation in Python was used to obtain model results for the random forest.

Hyperparameter selection Up to six different hyperparameters are investigated for the random forest. The hyperparameters mentioned in [43] that can be tuned with the sklearn implementation are tested. We also added the maximum depth of the trees and the number of samples required to split an internal node. In table 12 the grid of the random search is given.

The best model found in the random search had an f1-score of 0.551 and the following hyperparameters:

- Number of estimators: 800
- Max features: Auto
- Max depth: 60
- Min samples split: 10
- Min samples leaf depth: 2
- Bootstrap: False

Hyperparameter	range
Number of estimators	[200, 400, 600, ..., 2000]
Max features	[auto, sqrt]
Max depth	[10, 20, 30, ..., 110, None]
Min samples split	[2, 5, 10]
Min samples leaf	[1, 2, 4]
Bootstrap	[True, False]

Table 13: Hyperparameter grid for random forest

5.5.6 Support vector machine

Description Support vector machine [9] performs classification by finding the hyperplane that differentiates the two classes "well." To find the "best" hyperplane support vector machine will "look" for the hyperplane whose distance to the nearest observation (maximum margin) is the largest.

The abovementioned hyperplane is linear. However, not all problems can be solved by linear decision boundaries. To be able to "predict" nonlinear decision boundaries, SVMs use the kernel trick. Via the kernel trick (a mathematical transformation), the data is projected in a higher dimension. In this dimension, the "best" hyperplane is searched (as described above and hence linear). When projecting the final decision boundaries back on the original space (lower dimension), it becomes nonlinear [16].

Implementation The sci-kit learn implementation in Python was used to obtain model results for the support vector machine.

Hyperparameter selection The three hyperparameters that are tested are C, Gamma, and Kernel. The kernel stands for the transformation of the "original" data to a higher dimension. In which one hopes the data is better predictable. Gamma is the coefficient used in the three kernels that are tested. The larger gamma is, the more closely the model will try to fit the training data. C is a regularization parameter of the model. The strength of the regularization is inversely proportional to C. In table 14 the hyperparameter grid is given.

Hyperparameter	range
C	[0.01, 0.1, 1, 10, 100]
Gamma	[1, 0.1, 0.01, 0.001, scale]
Kernel	[rbf, poly, sigmoid]

Table 14: Hyperparameter grid for support vector machine

The best model found in the random search had an f1-score of 0.511 and the following hyperparameters:

- C: 10
- Gamma: 1
- Kernel: rbf

5.5.7 XGBoost

Description The way XGBoost (Extreme Gradient Boosting) [13] is a gradient boosting algorithm designed to be highly scalable with also good performance [3]. Boosting (in machine learning) stands for the family of models that uses "weak" learners to produce a final "strong" learner. These learners are built sequentially such that each subsequent learner aims to reduce the errors of the previous learner. Hence, the learner that grows next in the sequence will learn from an updated version of the residuals. XGBoost enhances the gradient boosting algorithm in six different ways. It has the option to penalize through both L1 and L2 regularization. They are smartly handling sparse data. The use of the weighted quantile sketch algorithm, because this can effectively handle weighted data. It uses cache awareness, out-of-core computing, and block structure of parallel learning from the computing side.

Implementation The xgboost[4] implementation in Python was used to obtain model results for the support vector machine. This implementation is easy to use with the sci-kit learn package.

Hyperparameter selection In table 15 the hyperparameters that are tuned for the xgboost algorithm in this research are given. The number of estimators is the number of weak learners that are built. Learning rate stands for the shrinkage size used after each boosting step. This makes the boosting process more conservative. The subsample is the proportion of the training data that is sampled for growing a new learner. Max depth is the maximum depth of a learner (tree). Colsample by tree is the proportion of features (columns) to be sampled for growing a new learner. Min child weight stands for the minimum instances needed in a child. When a partition step results in a child with fewer instances than min child weight, the building process will give up further partitioning.

Hyperparameter	range
Number of estimators	int(150, 1000)
Learning rate	uniform(0.01, 0.6)
Subsample	uniform(0.3, 0.9)
Max depth	[3, 4, 5, 6, 7, 8, 9]
Colsample by tree	uniform(0.5, 0.9)
Min child weight	[1, 2, 3, 4]

Table 15: Hyperparameter grid for XGBoost

The best model found in the random search had an f1-score of 0.539 and the following hyperparameters:

- Number of estimators: 165
- Learning rate: 0.079
- Subsample: 0.708
- Max depth: 4
- Colsample by tree: 0.753
- Min child weight: 1

[3] List with XGBoost winning solutions on Kaggle: https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions

[4] https://xgboost.readthedocs.io/en/latest/index.html

5.6 Binary classification models

In this section, the models that are tested are discussed. Also, the hyperparameters that are used are described. For these models, no hyperparameter tuning is done. We choose to use the same hyperparameters of the corresponding papers.

5.6.1 Data transformation

Before the models can be trained, the data should be transformed. This is because the models expect binary labels. In figure 14, the flow of this transformation is shown. First, we add the column label to the original dataset (16,000 rows). This column will be filled with 1's because we assume that the matching (between questions and answers) is good. Then we will upsample all the classes that are below 500 to 500. This is done to let the model see a significant number of examples for all classes. We did not just copy-paste the data but matched the question randomly with an answer of the same class. Hence, creating "new" examples instead of just adding the same examples. As last negative sampling is performed to create negative samples (samples with label 0). Negative sampling has, in general, a positive impact on the models' performance [38]. For negative sampling, we randomly sampled an answer from the dataset. However, we concluded all the "right" answers from the set we sample from. Because of this, all the negative sampled matchings are 100% negative. We negative sampled the whole data set 3 times. Creating a ratio of positive-negative samples of 1:3. In table 16 the distribution and the absolute number of positive and negative samples are given.

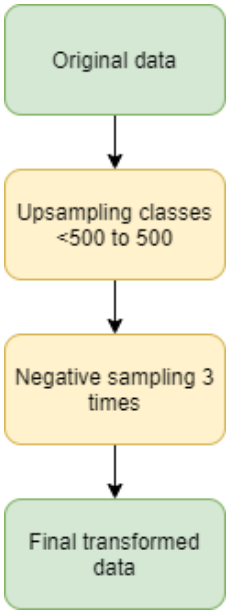

Figure 14: Flow of the transformation of the dataset

5.6.2 Global parameters

All the matching systems are implemented with Keras backed by Tensorflow [1]. Furthermore, the models were trained on a single NVIDIA GeForce GTX 1650 Ti GPU. Also, for all models,

	size (n)	size (%)
Positive samples	29,842	25%
Negative samples	89,526	75%
Total	119,368	100%

Table 16: Distribution final train data for binary classification models

the embedding layer is initialized with the Glove embeddings (of dimension 300) [42]. In training, all the models will finetune this embedding layer. We limited the size of both the context and the response to 160 words.

Further learning hyperparameters used by all models are:

- number of epochs: 50
- learning rate: 0.001
- Optimize: Adam
- Recurrent layer: 1

5.6.3 Dual Encoder LSTM

Description The dual encoder of [34] is already partly discussed in section 2.2.2. Here a more precise description of the system will be given. In figure 15 a global representation of the network is given. First, the embeddings of the context (question) and response (one of the candidate responses) are fed in the network. This embedding layer is part of the network and is learned in the training process. Both inputs will then be encoded in a vector of size d. In the figure, it seems that two different recurrent networks perform the encodings. However, this is a siamese neural network that shares the weights between them. Hence, both the context and response are encoded by the same recurrent network as in [34] [35]. Also, this approach got better results than using two different networks in experiments conducted by [55]. In equations 13, 14, and 15 the dual encoder in (simplified) mathematical expressions are given. Here f stands for the neural network that will encode the input. In our experiment, we choose to use an LSTM architecture instead of an RNN architecture. This is done because LSTMs are designed to solve the vanishing gradient problem of vanilla RNN [24]. Therefore LSTMs are more suited for working with long texts than vanilla RNNs [22]. Also, in the experiments of [35] LSTM outperformed RNN.

[24]

$$c = f(\text{Context}) \tag{13}$$

$$r = f(\text{Response}) \tag{14}$$

$$p(y = 1|c, r) = \sigma(c^T M r + b) \tag{15}$$

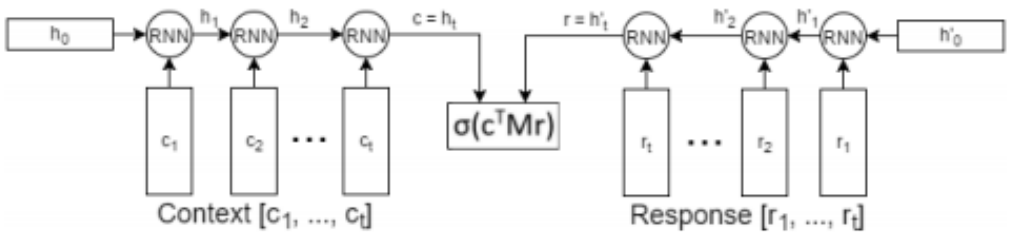

Figure 15: Diagram of the dual encoder, from [35]

Hyperparameters The number of hidden (LSTM) units were set to 300 as in [10]. In training this model, a batch size of 256 is used. These hyperparameters result in architecture given in equation 16. Here the output dimension of the layer is given between brackets.

$$Input(160, 300) \rightarrow LSTMlayer(300) \tag{16}$$

5.6.4 Dual Encoder Bidirectional LSTM

Description Normal direction LSTM suffers from not utilizing the information from future input (tokens/text). Bidirectional LSTM [46] counters this problem by processing a sequence on two directions (forward direction, backward direction). For these two different directions, it also outputs two independent sequences of LSTM output vectors. The output of the Bidirectional LSTM is the concatenation of the two output vectors of both directions (as is shown in equation 17).

$$h_t = \overrightarrow{h_t} || \overleftarrow{h_t} \tag{17}$$

Hyperparameters To keep the weights (LSTM units) the same between the LSTM and BiLSTM models, we set the hidden units to 150 for both directions. Resulting in 300 units (the same as the LSTM). In training this model, a batch size of 64 is used. This is done because of the limitations of the GPU used. These hyperparameters result in architecture given in equation 18. Here the output dimension of the layer is given between brackets.

$$Input(160, 300) \rightarrow BiLSTMlayer(300) \tag{18}$$

5.6.5 Dual Encoder Bidirectional LSTM and CNN

Description The QA-LSTM/CNN proposed in [55] inspires this model. In figure 16 the architecture of the system in [55] is given. On top of the bidirectional LSTM layer, a CNN layer is built. While CNN's original use was for computer vision [32], have they also been successfully applied to the field of NLP [30]. In this architecture also a 1-max pooling layer is applied to the convolutions. This will result in an encoding vector with the same dimension as the number of filters in the convolution layer. "The intuition of this structure is, instead of evenly considering the lexical information of each token as the previous subsection, we emphasize on certain parts of the answer, such that QA-LSTM/CNN can more effectively differentiate the ground truths

and incorrect answers." [55]. The one change that is done in this research is that we changed the cosine in figure 16 with the score function of the original dual encoder in [35] (equation 15).

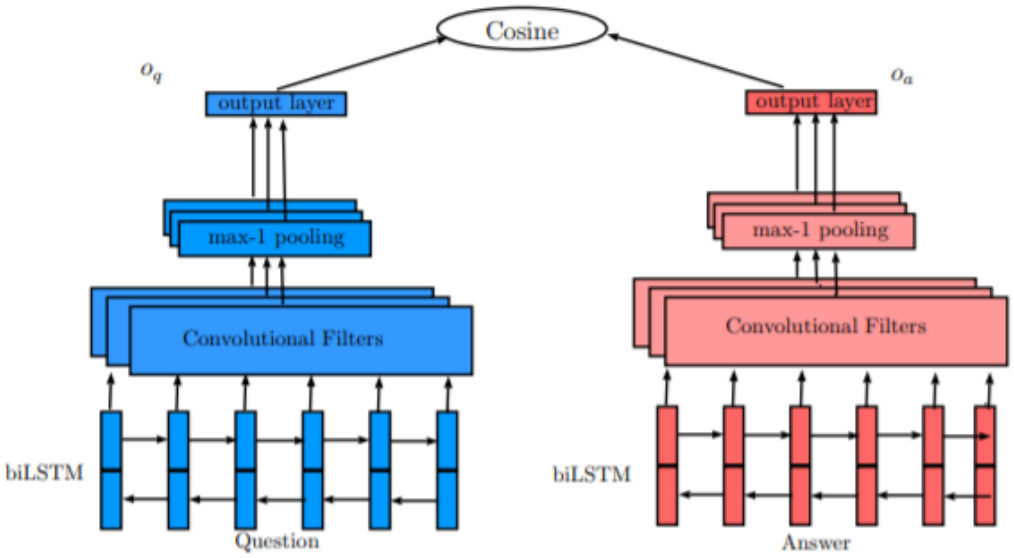

Figure 16: QA-LSTM/CNN, from [55]

Hyperparameters For the BiLSTM again, 150 units in both directions are used. The filter with of the CNN is set to 2 as [55] saw no improvement when increasing. The number of filters is set to 1000 as they also saw no significant improvement in increasing this number to 2000 or 4000 in [55]. In training this model, a batch size of 64 is used. This is done because of the limitations of the GPU used. These hyperparameters result in architecture given in equation 19. Here the output dimension of the layer is given between brackets.

$$\begin{aligned} Input(160, 300) \rightarrow BiLSTMlayer(160, 300) \rightarrow \\ CNNlayer(160, 300, 1000) \rightarrow \\ Max - pooling \rightarrow Output(1000) \end{aligned} \tag{19}$$

5.6.6 Dual Encoder Bidirectional LSTM and Self-Attention

Description Self-attention [64], also known as intra-attention, is a mechanism to transform word embeddings to contextualized embeddings. Where word embeddings only contain information about the word, contextualized embeddings incoorporates also information about the context (sequence) in which the word is used. In figure 17, a visual representation of self-attention mechanism is given. In equations 20, 21, and 22 the equations of the self-attention mechanism are given [5]. Where x_i is the word embedding of token i (in the sentence, W_k, W_q,

[5]These equations are tailored to figure 17, hence when equation 21 is done in different order (like $w'_{ij} = q_i^T k_j$) softmax should be applied row wise and equation 22 should be $y_i = \sum_j w_{ij} v_j$

and W_v are matrices that are learned by the model. With these matrice the query (q_i), key (k_i), and value (v_i) vectors are made per token embedding. Multiplying the transposed query matrix with the key matrix result in the attention score matrix ($w^{'}$). Applying the softmax column wise (so that the columns sum to 1) result in the attention weight matrix (w_i). Via equation 22 the final contextualized embeddings are obtained.

Self-attention has been successfully applied to multiple tasks in NLP. Examples are abstractive summarization [40], machine reading [14], and learning task-independent sentence representations and language understanding [49]. Also, is it the core idea of the multi-head self-attention of the Transformer [57]. Because of these successful implementations, we also opted to test self-attention in the dual encoder architecture. We used the same architecture as the dual encoder Bidirectional LSTM and CNN (described in section 5.6.5). In this experiment, the CNN layer is replaced with a self-attention layer. This layer outputs the same dimension as it gets as input. A max-pooling layer is applied over the output of the self-attention layer (in the time dimension).

$$\begin{aligned} k_i &= W_k x_i \\ q_i &= W_q x_i \\ v_i &= W_v x_i \end{aligned} \tag{20}$$

$$\begin{aligned} w^{'}_{ij} &= k_i^T q_j \\ w_{ij} &= softmax(w^{'}_{ij}), \text{ softmax applied column wise} \end{aligned} \tag{21}$$

$$y_i = \sum_j w_{ji} v_j \tag{22}$$

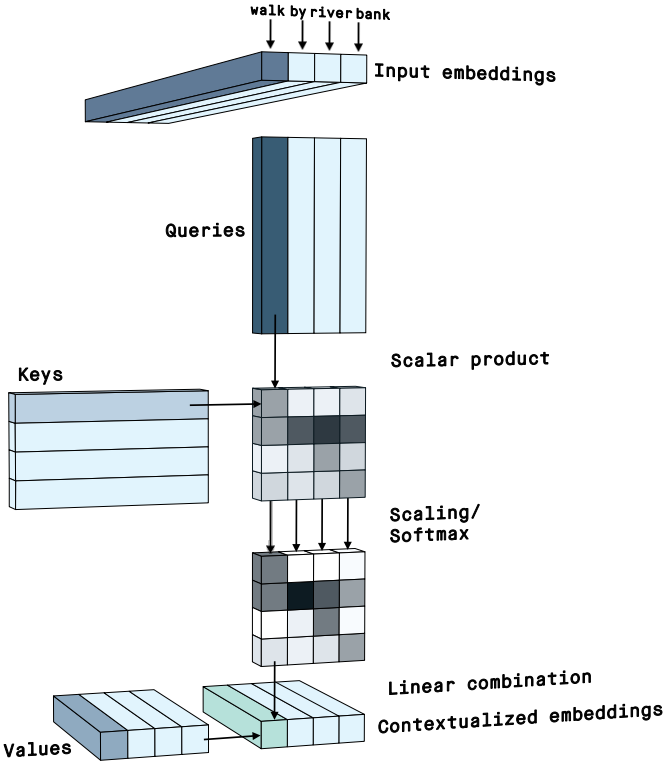

Figure 17: Self-attention mechanism, from [20]

$$\begin{aligned} Input(160, 300) \rightarrow BiLSTMlayer(160, 300) \rightarrow \\ Self-attentionlayer(160, 300) \rightarrow \\ Max-pooling \rightarrow Output(300) \end{aligned} \tag{23}$$

Hyperparameters For the BiLSTM, again, 150 units in both directions are used. In training this model, a batch size of 64 is used. This is done because of the limitations of the GPU used. These hyperparameters result in architecture given in equation 23. Here the output dimension of the layer is given between brackets.

6 Results

In this section, the result of the models on the test set will be investigated. First, we will do this via the performance metrics described in section 4.4. However, the certainty of the predictions will also be discussed as this can be of important information on the workings of the models. We want a model that gives good predictions but is "certain" about the predictions when it predicts right.

6.1 Performance of the models

The models are judged based on the classification and ranking metrics that are described in section 4.4. In figures 18, 19, and 20 the classification metrics of the models on the held-out test set are given. For precision and f1-score, the weighted average is given. The weighted average recall is not shown because this is the same as the accuracy in the multi-class case. In appendix C the results of the figures are given in table form and in appendix F the above mentioned metrics are given per class. The figures show that random forest stands out with the best performance on all metrics except precision. This implicates that the random forest (on weighted average) predicts too many positives (for classes) where it should not be positive. However, the accuracy of the random forest is by far the largest, which implies that the random forest finds the largest proportion of the actual positives. Also, the f1-score (combination of recall and precision) is the highest for the random forest. Furthermore, almost all models outperform the baseline used in this research. The max class baseline outperforms only the logistic regression. In appendix C the classification metrics are given per class for all models except baseline models. What notices is that all models perform well on the classes that have the most examples. This was also to be expected.

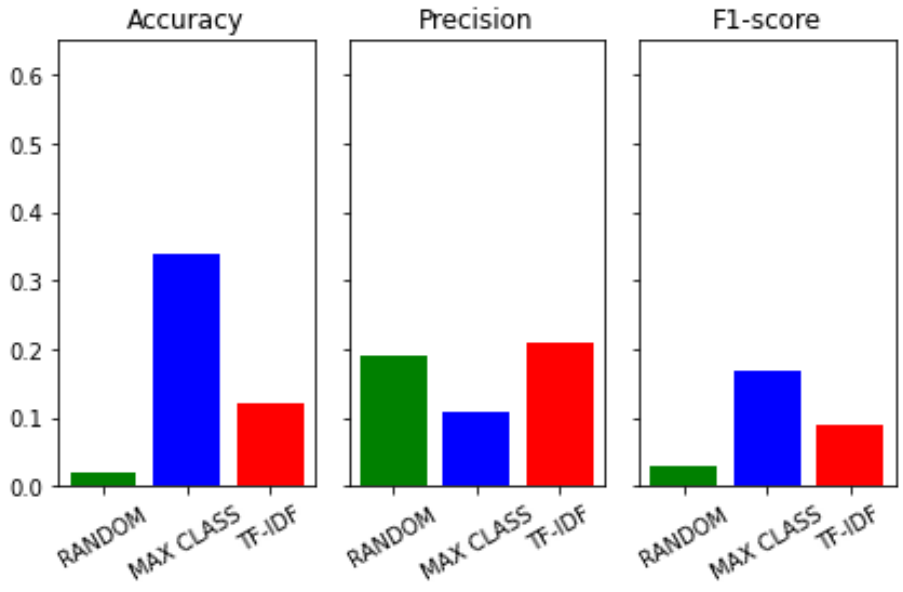

Figure 18: Classification results of the baseline models

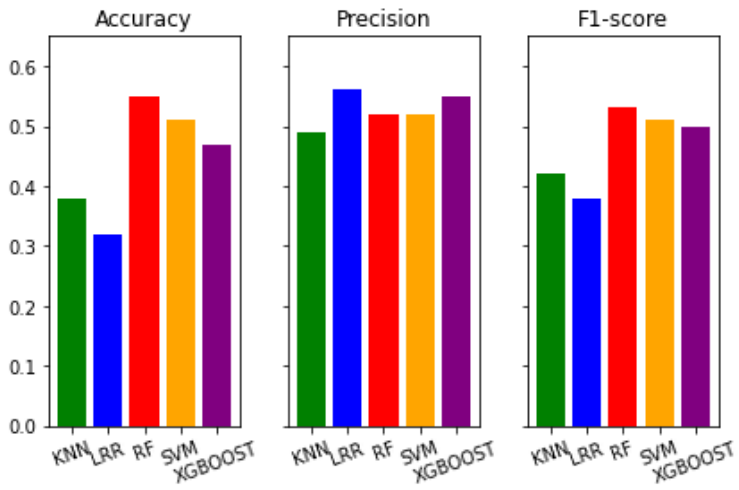

Figure 19: Classification results of the multi-class models

When only looking at the binary models in figure 20 we see the same pattern as by the classification models. Also, here one model (BiLSTM + attention) is the best on all metrics except precision, which means that compared to the other models, this model predicts more times a class true when in reality it is false.

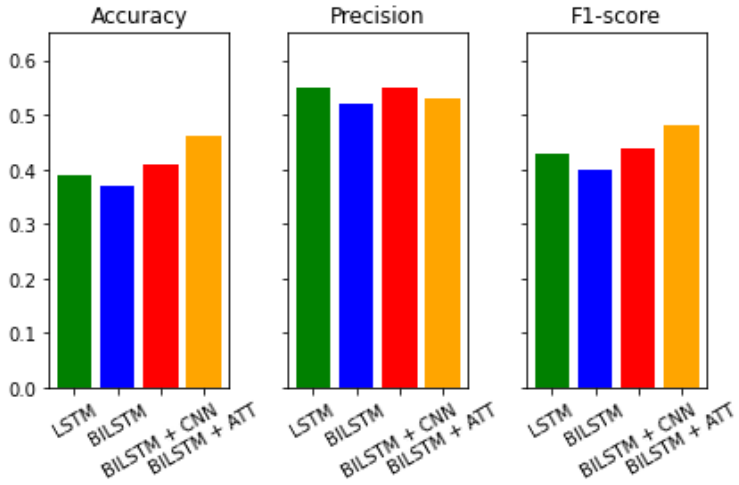

Figure 20: Classification results of the binary models

The scores on the ranking metrics are given in table 23. In appendix D the ranking results are given in figures. Random forest and support vector machines are the two models that perform the best on the ranking metrics. Of the matching models also again the BiLSTM + Attention model performs the best (on all the metrics). However, the max class baseline performs well on these metrics. Only random forest, support vector machine, and XGBoost performed better than max class baseline on all metrics. Of the binary models, no model performs better than

the max class baseline. The BiLSTM + Attention model is the only one that can "compete" with recall@5 and recall@10 almost as high and MRR significantly higher than the max class baseline.

Model	recall@5	recall@10	MRR
Random baseline	0.11	0.22	0.07
Max class baseline	0.72	0.84	0.50
TF-IDF baseline	0.35	0.49	0.21
KNN	0.58	0.60	0.49
LR	0.72	0.86	0.50
RF	**0.82**	0.90	**0.67**
SVM	**0.82**	**0.91**	0.66
XGBoost	0.78	0.88	0.61
LSTM	0.66	0.70	0.51
BiLSTM	0.64	0.78	0.50
BiLSTM + CNN	0.57	0.68	0.50
BILSTM + Attention	0.71	0.82	0.57

Table 17: Results of the models on the ranking metrics

6.2 Certainty of predictions

Here the "certainty" of the random forest and BiLSTM + Attention are discussed. This is done by comparing the probability/match score of the predicted class and the difference between the predicted class and the second-highest class on good and bad classified instances. In figure 21 density plots of these scores are given for random forest and BiLSTM + Attention model. Note that the probability (random forest) value and match score (BiLSTM + Attention) should not be compared. One is a probability, and summing them all (for all classes) will result in 1. The match score, in theory, summing them all could result in 0 or 46. When looking at the probability of the predicted class of random forest (upper left figure), one notices that the probabilities are higher for the true label. The difference figure (upper right) is even more visible between good and bad predictions. At the bottom of the figure, the same two figures are given for the BiLSTM + attention model. This model, in general, predicts pretty high scores, and there seems not to be a clear difference between the good and badly predicted groups.

The Mann-Whitney U rank test is performed to test if the two groups are different from each other. All four hypothesis tests is a one-sided test and an α of 0.05 is used. In table 18 the p-value of these tests is given. All p-values are so low that it is almost zero. Hence all null hypotheses are rejected, which means that the max probabilities/scores of the "bad" prediction group are significantly smaller than that of the good prediction group. This is also the same for the difference between the highest and second-highest probability/score.

Model	Max probability/score	Difference highest and second highest
RF	0.00	0.00
BILSTM + Attention	0.00	0.00

Table 18: P-value Mann-Whitney U rank test

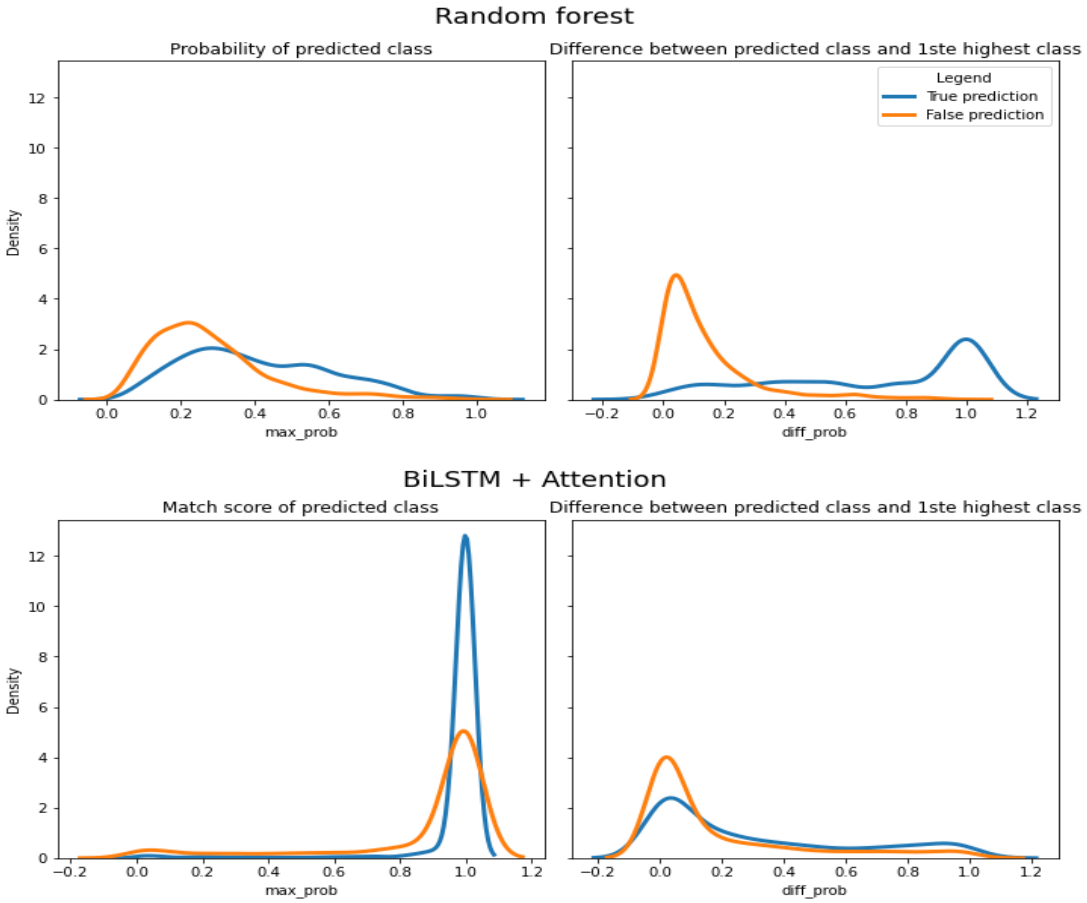

Figure 21: Certainty of predictions

7 Value of implementation for DutchChannels

In this section, the value of implementing two of the tested models is discussed. The overall best model (random forest) and best binary model (BiLSTM + Attention) will be reviewed. The value of the implementation will be measured in the percentage of questions it solves. Also, a (rough) prediction of the absolute number of questions the model could have solved from 12-07-2021 until 08-08-2021 (covering four weeks) will be discussed. In table 33 in appendix G the distribution of the tickets of period 12-07-2021 until 08-08-2021 is given.

The percentage of questions rightly answered is the same as the accuracy given in section 6.1, which means that the random forest solves roughly around 55% of the question in the test set and BiLSTM + Attention around 46%. However, not all questions are solved by the given answer. For multiple answers, the input of an agent is needed. For example, resending an activation mail or checking if a refund is possible. Of the 46 predefined answers, 12 need some agent interaction on them. In this evaluation, the answered proportion of all answers (the 46) and directly solving answers ($46 - 12 = 34$) are reviewed. In table 19 the size and proportion of the directly solving answers are given. Both models are close to each other in terms of absolute number and percentage than for all answers.

Model	size (n)	size (%)
RF	1402	33.15%
BILSTM + Attention	1327	30.00%

Table 19: Directly solved questions

In table 20 the accuracy for all and directly solving answers are given for both models per general type. It can be seen that the Film/series related typed questions almost always will not need any agent interaction. However, the other classes need some agent interaction. Furthermore, the BiLSTM + Attention network seems to have much trouble answering payment related questions.

	Random forest		BiLSTM + Att	
type	All	directly solved	All	directly solved
Account and subscription	0.60	0.44	0.50	0.41
Film/series related	0.55	0.54	0.55	0.55
Payments	0.40	0.1	0.16	0.12
Technical	0.46	0.08	0.39	0.1

Table 20: Percentage questions answered and solved (without needing agent) per general type.

In table 21 a prediction of the absolute number of tickets solved by the models is given. Note that in this prediction, it is the case that all customers that contact the CS will first try via the chatbot. In reality, these values would be significantly lower. Random forest will directly solve around 781 tickets in the period, and it will help speed up handling about 1200 tickets. BiLSTM + Attention network will directly solve 756 tickets and will help speed up handling around 1000 tickets. These are substantial numbers on a total of 2,188 incoming tickets. In table 33 the distribution of the tickets in the researched period are given.

	Random forest		BiLSTM + Att	
type	All	directly solved	All	directly solved
Account and subscription	795	583	662.5	543.25
Film/series related	150.15	147.42	150.15	150.15
Payments	80.4	20.1	32.16	24.12
Technical	70.1	30.8	150.15	38.5
Total	1202.65	781.32	999.96	756.02

Table 21: Questions answered and solved (without needing agent) per general type.

8 Conclusion

This report shows the performance of different chatbot implementations on a specific domain. Several models were tested to "answer" a textual question. These models are to be divided into two groups. A multiclass classification group that maps a textual question to a class and gives the corresponding response of that class. Furthermore, the binary classification group ranks a textual question with all the possible responses (and gives the response with the highest score). The following research question fueled this research:

How do classification models perform when mapping a textual question to a predefined set of 46 answers from the subscription video on demand domain?

The models were evaluated on two different kinds of metrics (classification and ranking). On the classification metrics, the random forest produced the best results for all the models tested. It got the highest Accuracy (0.55) and F1-score (0.53). The precision was 0.52%. This is 0.04 percent point below the highest precision of the models (gotten by logistic regression). Also, all models except logistic regression outperformed the baselines on all classification metrics used in this research.

On the ranking metrics, support vector machine and random forest performed equally. Support vector machine got a recall@10 of 0.91, the highest recall@10 of all models, where random forest got the highest MRR (0.67). On recall@5, random forest and support vector machine had the same score of 0.82, also the highest. Only the above-mentioned (two) models performed better than the max class baseline on all ranking metrics.

The best binary model is the BiLSTM + attention model. However, this model was significantly worse than random forest. However, it performed reasonably well with an accuracy of 0.46, precision of 0.53, and F1-score 0.48. Also, on the ranking metrics, this model performed well with a recall@5 of 0.71, recall@10 of 0.82, and MRR of 0.57.

When DutchChannels would decide to use the random forest as their chatbot, a substantial number of tickets could be directly solved by the chatbot. Also, on several tickets, the chatbot could help speed up the process (like asking for information). Of the 2,188 incoming tickets (of the period 12-07-2021 until 08-08-2021), an estimated 782 tickets could be directly solved and could help speed up the process of 421 tickets. However, note that these absolute values are based on the assumption that every customer would use the chatbot before contacting customer service. In reality, these values would be significantly lower.

Some future work could be researching the value of transfer learning. This can be done in two different ways. Via tweaking a state-of-the-art model like BERT [17] and performing the classification with this tweaked BERT model. However, with the pointwise ranking models, one can also use an entirely different dataset. The models used in this research for pointwise ranking (neural networks) need many data. Using a different (large) dataset, these models can learn the semantics of matching questions with answers. We would expect that the performance of these models would significantly improve because these models are notorious for needing many data.

Another exciting area to enhance the performance is ensemble learning. In this research, ensemble methods like random forest and XGBoost are used. However, one could also research an ensemble of the models themselves. In [28] they showed that for the pointwise ranking, this could be useful. Their best classifier was an ensemble of 11 LSTMs, 7 BiLSTMs, and 10 CNNs.

What also could be interesting to enhance the models' performance is inserting a follow-up question to the model. This would implicate that when a model is wrong or not certain enough, he would ask the customer a question. This could provide the model with the needed information to make a "good" classification and improve the performance. This question could be a default question corresponding to a class/answer or an intelligent question that a model learns to ask.

9 Discussion

The report's goal was to propose a model that gives the correct response to as many questions as possible on the subscription video-on-demand domain—using the data available of the DutchChannels Customer service. Given the results on the test set, the proposed model in the report could be useful in answering textual questions. However, since the data is gathered from the mail, a customer service portal, and Facebook and not from chats, the results are not expected to be accurate. This is because we expect the customer to interact differently on chat than on the gathered resources.

Also, the distribution of the question (types) could differ from the "real" world. In the matching step (data preparation), this distribution is altered from the original distribution of gathered questions. Besides this alteration, one could also argue that the distribution differs from the year or event happening. An example will be if DutchChannels adds a feature or a "popular" movie to its collection. We then can expect that a lot more questions about this feature or movie would be asked. Also, altering the distribution of the tickets and so influencing the final performance of the chatbot.

In the mentioned matching step responses of agents are matched with one of the predefined responses. However, this matching is not perfect, which means there is some implicit error rate in our model. This will influence the performance of our model and the accuracy of the results on the test set.

In this research, two different kinds of classification models are investigated. When implementing one of the models, one should also consider the models' speed and "freedom." Random forest (best model) is a multiclass classification model. This model has the best results and also answers a question almost directly. However, this model pays in "freedom" when DutchChannels wants to remove or add a predefined answer to the set. This model should be entirely retrained. This retraining could result in a drop-off (or enhancement) of its performance. A binary model will be a bit slower and would pay a price in performance. Nonetheless, this model is more flexible in adding new predefined answers. In this case, one could still use the original network and give matching scores to all the predefined answers (plus the newly added one).

Also, adding a feedback loop will be different for the different kinds of models. With feedback loop meaning that the chatbot will check if the answer was helpful. Only positive (correctly) classified responses can be added to the data for the multiclass classification models. Hence, the model cannot learn from its mistakes. A binary classification model, however, can learn from its mistakes. Here all predictions (good or bad) can be used for (re-)training the model. This could mean that in time and hence collecting enough data; the binary classification models could outperform the multiclass classification models.

This research is focused on answering as many textual questions as possible. Hence, it is tailored to keeping as many questions away from the agents (customer service). However, there is also a client-side perspective one should consider before deploying a chatbot. The negative influence of a "bad" response (false positives) on the customer experience instead of sending them directly to an agent. One could influence this by setting a certainty threshold. In section 6.2 the certainty of the models is discussed. The results obtained in this section suggest that it should be possible (reduce false positives without reducing true positives too much). With this threshold, we could also tackle the questions we do not have an answer for. The assumption is then made that all predefined responses get a low probability/score and fall below the threshold. These unable to answer questions would then be sent to an agent.

References

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)

[2] Adamopoulou, E., Moussiades, L.: Chatbots: History, technology, and applications. Machine Learning with Applications **2**, 100006 (2020)

[3] Anzai, Y.: Pattern recognition and machine learning. Elsevier (2012)

[4] Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)

[5] Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks **5**(2), 157–166 (1994)

[6] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012)

[7] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. the Journal of machine Learning research **3**, 993–1022 (2003)

[8] Bordes, A., Weston, J., Usunier, N.: Open question answering with weakly supervised embedding models. In: Joint European conference on machine learning and knowledge discovery in databases. pp. 165–180. Springer (2014)

[9] Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on Computational learning theory. pp. 144–152 (1992)

[10] Boussaha, B.E.A.: Response selection for end-to-end retrieval-based dialogue systems. Ph.D. thesis, Nantes (2019)

[11] Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)

[12] Chen, H., Liu, X., Yin, D., Tang, J.: A survey on dialogue systems: Recent advances and new frontiers. Acm Sigkdd Explorations Newsletter **19**(2), 25–35 (2017)

[13] Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. pp. 785–794 (2016)

[14] Cheng, J., Dong, L., Lapata, M.: Long short-term memory-networks for machine reading. arXiv preprint arXiv:1601.06733 (2016)

[15] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)

[16] Cristianini, N., Shawe-Taylor, J., et al.: An introduction to support vector machines and other kernel-based learning methods. Cambridge university press (2000)

[17] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[18] Echihabi, A., Marcu, D.: A noisy-channel approach to question answering. Tech. rep., UNIVERSITY OF SOUTHERN CALIFORNIA MARINA DEL REY INFORMATION SCIENCES INST (2003)

[19] Eck, M., Vogel, S., Waibel, A.: Low cost portability for statistical machine translation based on n-gram frequency and tf-idf. In: International Workshop on Spoken Language Translation (IWSLT) 2005 (2005)

[20] Futrzynski, R.: Getting meaning from text: self-attention step-by-step video (2020), `https://peltarion.com/blog/data-science/self-attention-video`, visited on 2021-08-09

[21] Gao, J., Galley, M., Li, L.: Neural approaches to conversational ai. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 1371–1374 (2018)

[22] Gers, F.A., Schraudolph, N.N., Schmidhuber, J.: Learning precise timing with lstm recurrent networks. Journal of machine learning research **3**(Aug), 115–143 (2002)

[23] Hill, C.: Enterprise chatbots: What we know, and how we'll act

[24] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[25] Hofmann, K., Tsagkias, M., Meij, E., Rijke, M., Hofmann, K., Tsagkias, E.: A comparative study of features for keyphrase extraction in scientific literature (2009)

[26] Hofmann, K., Tsagkias, M., Meij, E., Rijke, M.: A comparative study of features for keyphrase extraction in scientific literature. In: Proceedings of the 18th ACM conference on information and knowledge management, Hong Kong, China (2009)

[27] Jagannath, V.: Random forest template for tibco spotfire (2020), `https://community.tibco.com/wiki/random-forest-template-tibco-spotfire`, visited on 2021-08-09

[28] Kadlec, R., Schmid, M., Kleindienst, J.: Improved deep learning baselines for ubuntu corpus dialogs. arXiv preprint arXiv:1510.03753 (2015)

[29] Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J., Nithya, M., Kannan, S., Gurusamy, V.: Preprocessing techniques for text mining. International Journal of Computer Science & Communication Networks **5**(1), 7–16 (2014)

[30] Kim, Y.: Convolutional neural networks for sentence classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (08 2014). https://doi.org/10.3115/v1/D14-1181

[31] Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: Proceedings of the 24th international conference on Machine learning. pp. 473–480 (2007)

[32] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[33] Lo, R.T.W., He, B., Ounis, I.: Automatically building a stopword list for an information retrieval system. In: Journal on Digital Information Management: Special Issue on the 5th Dutch-Belgian Information Retrieval Workshop (DIR). vol. 5, pp. 17–24 (2005)

[34] Lowe, R., Pow, N., Serban, I., Charlin, L., Pineau, J.: Incorporating unstructured textual knowledge sources into neural dialogue systems. In: Neural information processing systems workshop on machine learning for spoken language understanding (2015)

[35] Lowe, R., Pow, N., Serban, I., Pineau, J.: The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. arXiv preprint arXiv:1506.08909 (2015)

[36] Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)

[37] Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)

[38] Nugmanova, A., Smirnov, A., Lavrentyeva, G., Chernykh, I.: Strategy of the negative sampling for training retrieval-based dialogue systems. In: 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops). pp. 844–848. IEEE (2019)

[39] Pandey, G., Contractor, D., Kumar, V., Joshi, S.: Exemplar encoder-decoder for neural conversation generation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1329–1338 (2018)

[40] Paulus, R., Xiong, C., Socher, R.: A deep reinforced model for abstractive summarization. arXiv preprint arXiv:1705.04304 (2017)

[41] Peng, C.Y.J., Lee, K.L., Ingersoll, G.M.: An introduction to logistic regression analysis and reporting. The journal of educational research **96**(1), 3–14 (2002)

[42] Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). pp. 1532–1543 (2014)

[43] Probst, P., Wright, M.N., Boulesteix, A.L.: Hyperparameters and tuning strategies for random forest. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **9**(3), e1301 (2019)

[44] Rennie, S.J., Marcheret, E., Mroueh, Y., Ross, J., Goel, V.: Self-critical sequence training for image captioning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7008–7024 (2017)

[45] Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Communications of the ACM **18**(11), 613–620 (1975)

[46] Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE transactions on Signal Processing **45**(11), 2673–2681 (1997)

[47] Schütze, H., Manning, C.D., Raghavan, P.: Introduction to information retrieval, vol. 39. Cambridge University Press Cambridge (2008)

[48] Shang, L., Lu, Z., Li, H.: Neural responding machine for short-text conversation. arXiv preprint arXiv:1503.02364 (2015)

[49] Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., Zhang, C.: Disan: Directional self-attention network for rnn/cnn-free language understanding. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)

[50] Sidorov, G., Gelbukh, A., Gómez-Adorno, H., Pinto, D.: Soft similarity and soft cosine measure: Similarity of features in vector space model. Computación y Sistemas **18**(3), 491–504 (2014)

[51] Song, Y., Yan, R., Li, C.T., Nie, J.Y., Zhang, M., Zhao, D.: An ensemble of retrieval-based and generation-based human-computer conversation systems. (2018)

[52] Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.Y., Gao, J., Dolan, B.: A neural network approach to context-sensitive generation of conversational responses. arXiv preprint arXiv:1506.06714 (2015)

[53] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in neural information processing systems. pp. 3104–3112 (2014)

[54] Tam, V., Santoso, A., Setiono, R.: A comparative study of centroid-based, neighborhood-based and statistical approaches for effective document categorization. In: Object recognition supported by user interaction for service robots. vol. 4, pp. 235–238. IEEE (2002)

[55] Tan, M., Santos, C.d., Xiang, B., Zhou, B.: Lstm-based deep learning models for non-factoid answer selection. arXiv preprint arXiv:1511.04108 (2015)

[56] Tiancheng, Z.: Learning to converse with latent actions. Ph.D. thesis, PhD dissertation, Carnegie Mellon University. 25, 27, 35 (2019)

[57] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)

[58] Vijayarani, S., Ilamathi, M.J., Nithya, M., et al.: Preprocessing techniques for text mining-an overview. International Journal of Computer Science & Communication Networks **5**(1), 7–16 (2015)

[59] Vinyals, O., Le, Q.: A neural conversational model. arXiv preprint arXiv:1506.05869 (2015)

[60] Wan, S., Lan, Y., Guo, J., Xu, J., Pang, L., Cheng, X.: A deep architecture for semantic matching with multiple positional sentence representations. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 30 (2016)

[61] Wang, B., Cao, H.: A summary of research on intelligent dialogue systems. In: Journal of Physics: Conference Series. vol. 1651, p. 012020. IOP Publishing (2020)

[62] Wang, M., Lu, Z., Li, H., Liu, Q.: Syntax-based deep matching of short texts. arXiv preprint arXiv:1503.02427 (2015)

[63] Wu, Y., Wu, W., Xing, C., Zhou, M., Li, Z.: Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. arXiv preprint arXiv:1612.01627 (2016)

[64] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: International conference on machine learning. pp. 2048–2057. PMLR (2015)

[65] Zhang, X., Su, J., Qin, Y., Liu, Y., Ji, R., Wang, H.: Asynchronous bidirectional decoding for neural machine translation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)

[66] Zhao, T., Eskenazi, M.: Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. arXiv preprint arXiv:1606.02560 (2016)

[67] Zhou, X., Dong, D., Wu, H., Zhao, S., Yu, D., Tian, H., Liu, X., Yan, R.: Multi-view response selection for human-computer conversation. In: Proceedings of the 2016 conference on empirical methods in natural language processing. pp. 372–381 (2016)

[68] Zhou, X., Li, L., Dong, D., Liu, Y., Chen, Y., Zhao, W.X., Yu, D., Wu, H.: Multi-turn response selection for chatbots with deep attention matching network. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1118–1127 (2018)

Appendices

A The 46 predefined classes/responses

Table 22 in appendix A shows all the question 46 different question types. Furthermore, it also gives the absolute size and proportion of these question types before the train test split.

ID (label)	type	question$_type$	size (n)	size(%)
0	Account and subscription	Account - no subscription	140	0.828
1	Account and subscription	Account details	2726	16.117
2	Technical	App available	15	0.089
3	Technical	App malfunction - refer to website	216	1.277
4	Technical	App update	231	1.366
5	Film/series related	Availability countries	45	0.266
6	Technical	Browser	157	0.928
7	Account and subscription	Cancel subscription	5675	33.552
8	Account and subscription	Cancelled subscription - confirmation	311	1.839
9	Account and subscription	Cannot log in - username already in use	118	0.698
10	Payments	Change payment details	107	0.633
11	Technical	Change size of subtitles	24	0.142
12	Technical	Chromecast - icon missing	69	0.408
13	Technical	Chromecast - issue is being worked on	214	1.265
14	Technical	Chromecast - subtitles	131	0.775
15	Technical	Continue watching - how to	29	0.171
16	Technical	Continue watching - titles do not disappear	20	0.118
17	Technical	Devices compatible	15	0.089
18	Technical	Devices simultaneously	33	0.195
19	Account and subscription	Double subscription on one account	100	0.591
20	Film/series related	Heartland	481	2.844
21	Account and subscription	How do I become a member	77	0.455
22	Account and subscription	Login details	515	3.045
23	Account and subscription	Only trailers	505	2.986
24	Account and subscription	Password lost	75	0.443
25	Account and subscription	Pause subscription	40	0.236
26	Payments	Payment Methods	20	0.118
27	Payments	Payment failed	138	0.816
28	Payments	Payment of selected subscription	94	0.556
29	Account and subscription	Price increase - informing why	42	0.248
30	Payments	Refund - email reminder not received	43	0.254
31	Payments	Refund after trial period	16	0.095
32	Payments	Refund global	354	2.093
33	Account and subscription	Resend password reset email	88	0.520
34	Account and subscription	Send activation email	43	0.254
35	Technical	Smart TV app - issues	20	0.118
36	Account and subscription	Subscription costs	82	0.485
37	Technical	Technical issues	2727	16.123
38	Technical	Use google chromecast	82	0.485
39	Technical	Use subtitles	72	0.426
40	Account and subscription	Voucher code	113	0.668
41	Technical	Watch tv	597	3.530
42	Film/series related	What is	15	0.089
43	Account and subscription	Why subscription cancelled	17	0.101
44	Technical	Working on a solution	117	0.692
45	Film/series related	content statement	165	0.976

Table 22: Distribution of all question types

B Most frequent bigrams

The appendix consists out of figure 22 that shows the most common bigrams present in the questions.

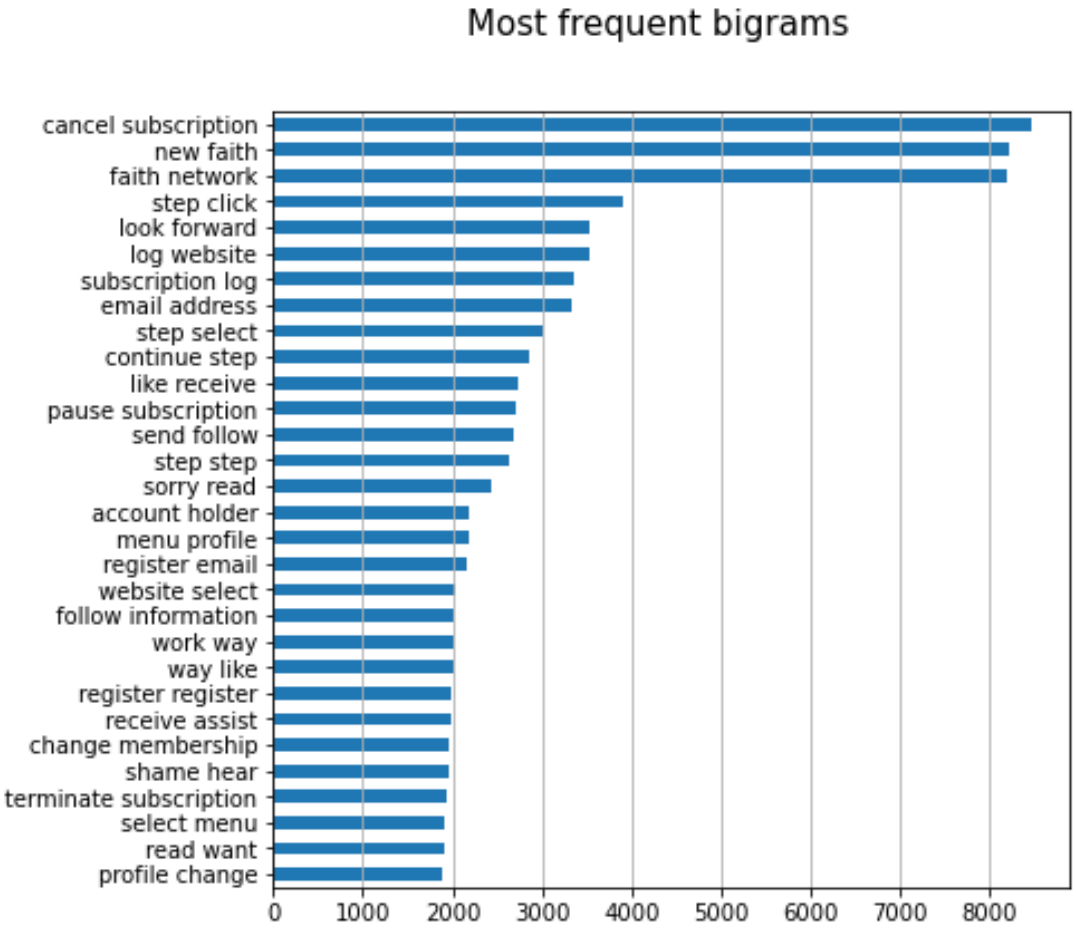

Figure 22: Most commen bigrams present in questions

C Classification results

The tables in this section give the classification results per type for the multiclass and binary classification models.

Model	Accuracy	Precision	F1-score
Random baseline	0.02	0.19	0.03
Max class baseline	0.34	0.11	0.17
TF-IDF baseline	0.12	0.21	0.09
KNN	0.38	0.49	0.42
LR	0.32	**0.56**	0.38
RF	**0.55**	0.52	**0.53**
SVM	0.51	0.52	0.51
XGBoost	0.47	0.55	0.50
LSTM	0.39	0.55	0.43
BiLSTM	0.37	0.52	0.40
BiLSTM + CNN	0.41	0.55	0.44
BILSTM + Attention	0.46	0.53	0.48

Table 23: Results of the models on the classification metrics

D Ranking results

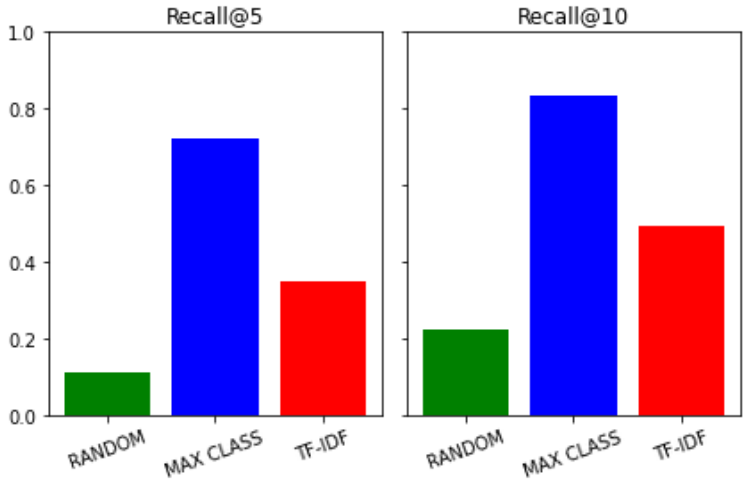

Figure 23: Ranking results baseline models

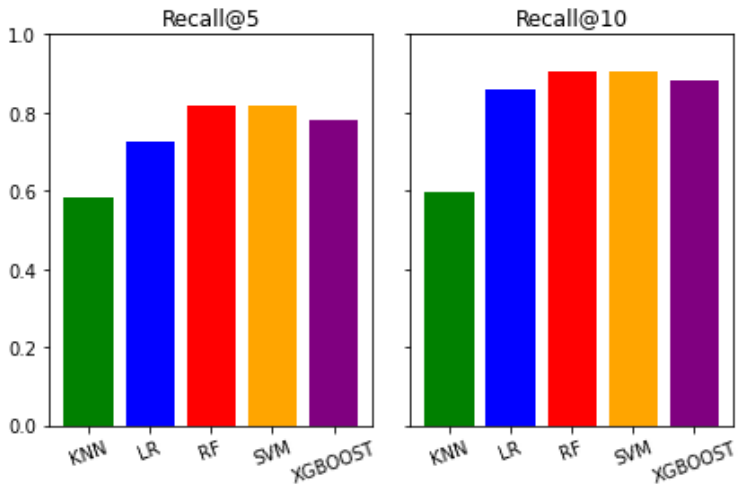

Figure 24: Ranking results multi-class classification models

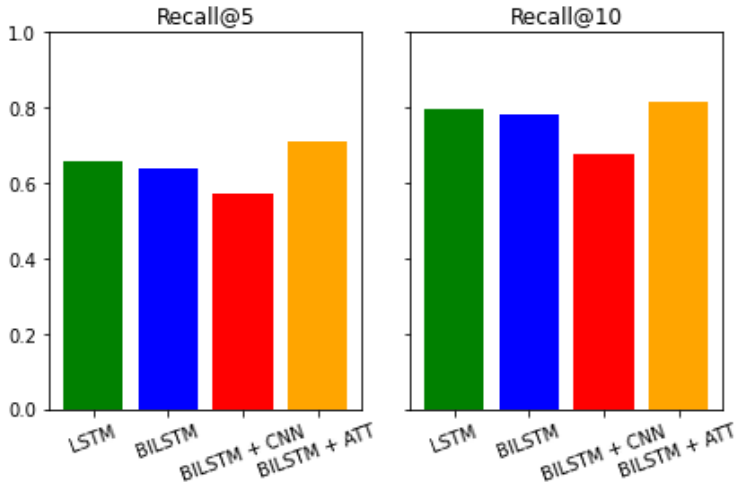

Figure 25: Ranking results binary classification models

E Character and sentence counts of the questions

Figures 26 and 27 in this appendix give insight in the number of characters and the number of sentences that are presented in the questions.

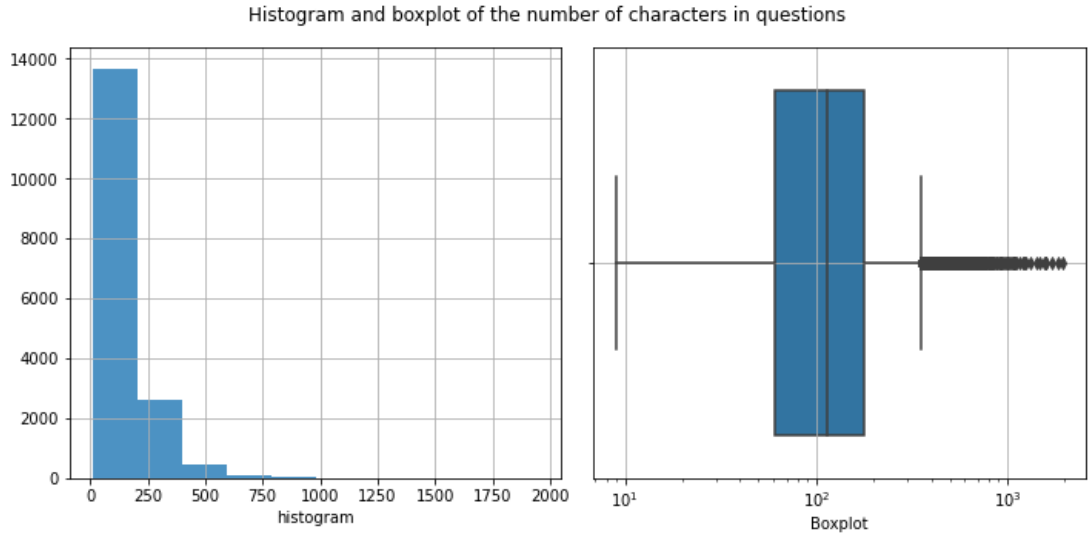

Figure 26: Histogram and boxplot of character counts

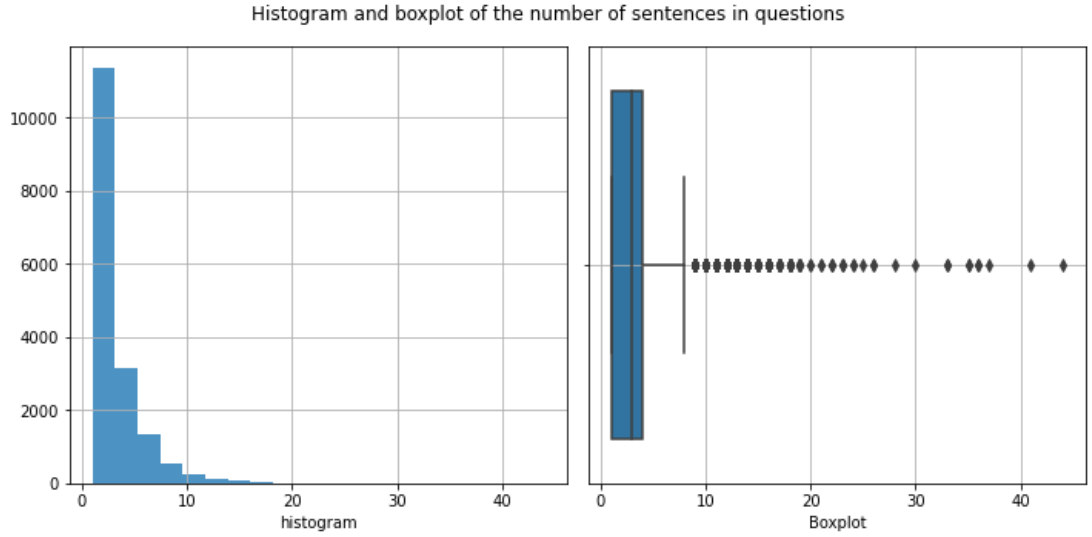

Figure 27: Histogram and boxplot of sentence counts

F Classification results per class

In this appendix the accuracy, precision, recall, and f1-scores of the models are given. These metrics are given per class.

Class	precision	recall	f1-score	support
Account - no subscription	0.03	0.06	0.04	35
Account details	0.43	0.28	0.34	682
App available	0.33	0.25	0.29	4
App malfunction - refer to website	0.12	0.19	0.15	54
App update	0.06	0.10	0.07	58
Availability countries	0.00	0.00	0.00	11
Browser	0.02	0.05	0.03	39
Cancel subscription	0.84	0.60	0.70	1419
Cancelled subscription - confirmation	0.15	0.33	0.21	78
Cannot log in - username already in use	0.03	0.07	0.05	30
Change payment details	0.47	0.67	0.55	27
Change size of subtitles	0.00	0.00	0.00	6
Chromecast - icon missing	0.00	0.00	0.00	17
Chromecast - issue is being worked on	0.01	0.02	0.01	54
Chromecast - subtitles	0.16	0.24	0.19	33
Continue watching - how to	0.00	0.00	0.00	7
Continue watching - titles do not disappear	0.33	0.20	0.25	5
Devices compatible	0.00	0.00	0.00	4
Devices simultaneously	0.33	0.12	0.18	8
Double subscription on one account	0.00	0.00	0.00	25
Heartland	0.55	0.70	0.62	120
How do I become a member	0.03	0.11	0.05	19
Login details	0.11	0.19	0.14	129
Only trailers	0.19	0.32	0.23	126
Password lost	0.00	0.00	0.00	19
Pause subscription	0.23	0.30	0.26	10
Payment Methods	0.67	0.40	0.50	5
Payment failed	0.01	0.03	0.02	35
Payment of selected subscription	0.00	0.00	0.00	23
Price increase - informing why	0.20	0.10	0.13	10
Refund - email reminder not received	0.00	0.00	0.00	11
Refund after trial period	0.00	0.00	0.00	4
Refund global	0.30	0.47	0.37	89
Resend password reset email	0.00	0.00	0.00	22
Send activation email	0.06	0.09	0.07	11
Smart TV app - issues	0.00	0.00	0.00	5
Subscription costs	0.03	0.05	0.03	20
Technical issues	0.51	0.30	0.38	682
Use google chromecast	0.06	0.15	0.09	20
Use subtitles	0.09	0.11	0.10	18
Voucher code	0.36	0.46	0.41	28
Watch tv	0.35	0.46	0.39	149
What is	0.00	0.00	0.00	4
Why subscription cancelled	0.00	0.00	0.00	4
Working on a solution	0.01	0.03	0.02	29
content statement	0.02	0.02	0.02	41

Table 24: Classification result KNN

Class	precision	recall	f1-score	support
Account - no subscription	0.04	0.14	0.07	35
Account details	0.53	0.18	0.26	682
App available	0.25	0.25	0.25	4
App malfunction - refer to website	0.17	0.30	0.22	54
App update	0.07	0.21	0.10	58
Availability countries	0.06	0.09	0.07	11
Browser	0.05	0.15	0.08	39
Cancel subscription	0.90	0.49	0.63	1419
Cancelled subscription - confirmation	0.18	0.38	0.25	78
Cannot log in - username already in use	0.04	0.10	0.05	30
Change payment details	0.40	0.74	0.52	27
Change size of subtitles	0.50	0.17	0.25	6
Chromecast - icon missing	0.10	0.29	0.15	17
Chromecast - issue is being worked on	0.11	0.19	0.14	54
Chromecast - subtitles	0.13	0.30	0.19	33
Continue watching - how to	0.09	0.29	0.14	7
Continue watching - titles do not disappear	0.00	0.00	0.00	5
Devices compatible	0.00	0.00	0.00	4
Devices simultaneously	0.29	0.25	0.27	8
Double subscription on one account	0.02	0.08	0.03	25
Heartland	0.60	0.72	0.65	120
How do I become a member	0.01	0.05	0.01	19
Login details	0.23	0.22	0.22	129
Only trailers	0.22	0.21	0.22	126
Password lost	0.02	0.11	0.04	19
Pause subscription	0.20	0.40	0.27	10
Payment Methods	0.00	0.00	0.00	5
Payment failed	0.05	0.14	0.07	35
Payment of selected subscription	0.00	0.00	0.00	23
Price increase - informing why	0.13	0.40	0.20	10
Refund - email reminder not received	0.00	0.00	0.00	11
Refund after trial period	0.00	0.00	0.00	4
Refund global	0.44	0.51	0.47	89
Resend password reset email	0.05	0.14	0.07	22
Send activation email	0.13	0.36	0.20	11
Smart TV app - issues	0.20	0.20	0.20	5
Subscription costs	0.03	0.20	0.06	20
Technical issues	0.62	0.16	0.25	682
Use google chromecast	0.06	0.25	0.09	20
Use subtitles	0.05	0.17	0.08	18
Voucher code	0.37	0.61	0.46	28
Watch tv	0.49	0.41	0.45	149
What is	0.00	0.00	0.00	4
Why subscription cancelled	0.00	0.00	0.00	4
Working on a solution	0.03	0.21	0.06	29
content statement	0.09	0.17	0.12	41

Table 25: Classification result logistic regression

Class	precision	recall	f1-score	support
Account - no subscription	0.14	0.03	0.05	35
Account details	0.45	0.54	0.49	682
App available	0.45	0.54	0.49	4
App malfunction - refer to website	0.21 0.09	0.13	54	
App update	0.11	0.07	0.09	58
Availability countries	0.00	0.00	0.00	11
Browser	0.00	0.00	0.00	39
Cancel subscription	0.80	0.79	0.80	1419
Cancelled subscription - confirmation	0.67	0.28	0.40	78
Cannot log in - username already in use	0.00	0.00	0.00	30
Change payment details	0.56	0.70	0.62	27
Change size of subtitles	1.00	0.33	0.50	6
Chromecast - icon missing	0.04	0.06	0.05	17
Chromecast - issue is being worked on	0.17	0.04	0.06	54
Chromecast - subtitles	0.14	0.03	0.05	33
Continue watching - how to	0.00	0.00	0.00	7
Continue watching - titles do not disappear	0.00	0.00	0.00	5
Devices compatible	0.00	0.00	0.00	4
Devices simultaneously	0.40	0.25	0.31	8
Double subscription on one account	0.00	0.00	0.00	25
Heartland	0.61	0.78	0.69	120
How do I become a member	0.00	0.00	0.00	19
Login details	0.30	0.19	0.24	129
Only trailers	0.37	0.28	0.32	126
Password lost	0.00	0.00	0.00	19
Pause subscription	0.45	0.50	0.48	10
Payment Methods	0.40	0.40	0.40	5
Payment failed	0.00	0.00	0.00	35
Payment of selected subscription	0.00	0.00	0.00	23
Price increase - informing why	0.14	0.30	0.19	10
Refund - email reminder not received	0.00	0.00	0.00	11
Refund after trial period	0.00	0.00	0.00	4
Refund global	0.61	0.63	0.62	89
Resend password reset email	0.00	0.00	0.00	22
Send activation email	0.33	0.27	0.30	11
Smart TV app - issues	0.00	0.00	0.00	5
Subscription costs	0.19	0.15	0.17	20
Technical issues	0.51	0.67	0.58	682
Use google chromecast	0.11	0.20	0.14	20
Use subtitles	0.18	0.17	0.17	18
Voucher code	0.39	0.54	0.45	28
Watch tv	0.44	0.50	0.47	149
What is	0.00	0.00	0.00	4
Why subscription cancelled	0.00	0.00	0.00	4
Working on a solution	0.00	0.00	0.00	29
content statement	0.09	0.05	0.06	41

Table 26: Classification result random forest

Class	precision	recall	f1-score	support
Account - no subscription	0.18	0.09	0.12	35
Account details	0.46	0.58	0.51	682
App available	1.00	0.25	0.40	4
App malfunction - refer to website	0.38	0.22	0.28	54
App update	0.10	0.10	0.10	58
Availability countries	0.00	0.00	0.00	11
Browser	0.00	0.00	0.00	39
Cancel subscription	0.79	0.77	0.78	1419
Cancelled subscription - confirmation	0.32	0.24	0.28	78
Cannot log in - username already in use	0.00	0.00	0.00	30
Change payment details	0.54	0.48	0.51	27
Change size of subtitles	1.00	0.17	0.29	6
Chromecast - icon missing	0.20	0.12	0.15	17
Chromecast - issue is being worked on	0.16	0.13	0.14	54
Chromecast - subtitles	0.50	0.15	0.23	33
Continue watching - how to	0.11	0.14	0.12	7
Continue watching - titles do not disappear	0.50	0.20	0.29	5
Devices compatible	0.00	0.00	0.00	4
Devices simultaneously	0.60	0.38	0.46	8
Double subscription on one account	0.10	0.04	0.06	25
Heartland	0.62	0.67	0.64	120
How do I become a member	0.00	0.00	0.00	19
Login details	0.25	0.26	0.26	129
Only trailers	0.32	0.35	0.34	126
Password lost	0.00	0.00	0.00	19
Pause subscription	0.50	0.30	0.37	10
Payment Methods	0.00	0.00	0.00	5
Payment failed	0.17	0.06	0.09	35
Payment of selected subscription	0.00	0.00	0.00	23
Price increase - informing why	0.05	0.10	0.07	10
Refund - email reminder not received	0.00	0.00	0.00	11
Refund after trial period	0.00	0.00	0.00	4
Refund global	0.53	0.47	0.50	89
Resend password reset email	0.00	0.00	0.00	22
Send activation email	0.50	0.09	0.15	11
Smart TV app - issues	0.00	0.00	0.00	5
Subscription costs	0.00	0.00	0.00	20
Technical issues	0.52	0.61	0.56	682
Use google chromecast	0.10	0.15	0.12	20
Use subtitles	0.30	0.17	0.21	18
Voucher code	0.38	0.36	0.37	28
Watch tv	0.55	0.47	0.51	149
What is	0.00	0.00	0.00	4
Why subscription cancelled	0.00	0.00	0.00	4
Working on a solution	0.03	0.03	0.03	29
content statement	0.07	0.05	0.06	41

Table 27: Classification result Support Vector Machine

Class	precision	recall	f1-score	support
Account - no subscription	0.08	0.14	0.10	35
Account details	0.5	0.35	0.42	682
App available	0.33	0.25	0.29	4
App malfunction - refer to website	0.20	0.26	0.23	54
App update	0.09	0.17	0.12	58
Availability countries	0.07	0.09	0.08	11
Browser	0.04	0.08	0.05	39
Cancel subscription	0.86	0.73	0.79	1419
Cancelled subscription - confirmation	0.31	0.35	0.33	78
Cannot log in - username already in use	0.02	0.03	0.02	30
Change payment details	0.44	0.70	0.54	27
Change size of subtitles	0.67	0.33	0.44	6
Chromecast - icon missing	0.03	0.06	0.04	17
Chromecast - issue is being worked on	0.10	0.17	0.13	54
Chromecast - subtitles	0.22	0.42	0.29	33
Continue watching - how to	0.11	0.14	0.12	7
Continue watching - titles do not disappear	0.00	0.00	0.00	5
Devices compatible	0.00	0.00	0.00	4
Devices simultaneously	0.29	0.25	0.27	8
Double subscription on one account	0.17	0.16	0.17	25
Heartland	0.56	0.78	0.65	120
How do I become a member	0.04	0.11	0.05	19
Login details	0.25	0.31	0.28	129
Only trailers	0.30	0.37	0.33	126
Password lost	0.00	0.00	0.00	19
Pause subscription	0.21	0.40	0.28	10
Payment Methods	0.29	0.40	0.33	5
Payment failed	0.06	0.09	0.07	35
Payment of selected subscription	0.03	0.04	0.04	23
Price increase - informing why	0.11	0.30	0.16	10
Refund - email reminder not received	0.00	0.00	0.00	11
Refund after trial period	0.00	0.00	0.00	4
Refund global	0.51	0.62	0.56	89
Resend password reset email	0.00	0.00	0.00	22
Send activation email	0.15	0.27	0.19	11
Smart TV app - issues	0.00	0.00	0.00	5
Subscription costs	0.07	0.15	0.10	20
Technical issues	0.59	0.33	0.42	682
Use google chromecast	0.09	0.20	0.12	20
Use subtitles	0.08	0.17	0.11	18
Voucher code	0.37	0.57	0.45	28
Watch tv	0.40	0.57	0.47	149
What is	0.00	0.00	0.00	4
Why subscription cancelled	0.00	0.00	0.00	4
Working on a solution	0.02	0.03	0.02	29
content statement	0.10	0.17	0.13	41

Table 28: Classification result XGBoost

Class	precision	recall	f1-score	support
Account - no subscription	0.00	0.00	0.00	35
Account details	0.53	0.18	0.27	682
App available	0.00	0.00	0.00	4
App malfunction - refer to website	0.12	0.33	0.18	54
App update	0.07	0.24	0.11	58
Availability countries	0.30	0.27	0.29	11
Browser	0.03	0.15	0.05	39
Cancel subscription	0.88	0.75	0.81	1419
Cancelled subscription - confirmation	0.04	0.06	0.05	78
Cannot log in - username already in use	0.04	0.20	0.07	30
Change payment details	0.45	0.52	0.48	27
Change size of subtitles	0.00	0.00	0.00	6
Chromecast - icon missing	0.18	0.18	0.18	17
Chromecast - issue is being worked on	0.06	0.07	0.06	54
Chromecast - subtitles	0.07	0.30	0.11	33
Continue watching - how to	0.00	0.00	0.00	7
Continue watching - titles do not disappear	1.00	0.20	0.33	5
Devices compatible	0.00	0.00	0.00	4
Devices simultaneously	0.40	0.50	0.44	8
Double subscription on one account	0.02	0.16	0.04	25
Heartland	0.62	0.60	0.61	120
How do I become a member	0.02	0.05	0.02	19
Login details	0.27	0.07	0.11	129
Only trailers	0.21	0.17	0.19	126
Password lost	0.04	0.05	0.04	19
Pause subscription	0.50	0.30	0.37	10
Payment Methods	0.00	0.00	0.00	5
Payment failed	0.06	0.06	0.06	35
Payment of selected subscription	0.03	0.17	0.05	23
Price increase - informing why	0.29	0.20	0.24	10
Refund - email reminder not received	0.07	0.36	0.12	11
Refund after trial period	0.00	0.00	0.00	4
Refund global	0.37	0.33	0.35	89
Resend password reset email	0.04	0.05	0.04	22
Send activation email	0.07	0.18	0.11	11
Smart TV app - issues	0.00	0.00	0.00	5
Subscription costs	0.00	0.00	0.00	20
Technical issues	0.58	0.23	0.33	682
Use google chromecast	0.00	0.00	0.00	20
Use subtitles	0.11	0.06	0.07	18
Voucher code	0.18	0.39	0.25	28
Watch tv	0.50	0.18	0.27	149
What is	0.00	0.00	0.00	4
Why subscription cancelled	0.00	0.00	0.00	4
Working on a solution	0.03	0.07	0.04	29
content statement	0.17	0.20	0.18	41

Table 29: Classification result LSTM

Class	precision	recall	f1-score	support
Account - no subscription	0.07	0.06	0.06	35
Account details	0.44	0.14	0.21	682
App available	0.00	0.00	0.00	4
App malfunction - refer to website	0.10	0.07	0.09	54
App update	0.05	0.45	0.10	58
Availability countries	0.11	0.36	0.17	11
Browser	0.05	0.08	0.06	39
Cancel subscription	0.87	0.73	0.80	1419
Cancelled subscription - confirmation	0.07	0.05	0.06	78
Cannot log in - username already in use	0.03	0.20	0.06	30
Change payment details	0.57	0.48	0.52	27
Change size of subtitles	0.00	0.00	0.00	6
Chromecast - icon missing	0.06	0.12	0.08	17
Chromecast - issue is being worked on	0.02	0.02	0.02	54
Chromecast - subtitles	0.16	0.27	0.20	33
Continue watching - how to	0.33	0.14	0.20	7
Continue watching - titles do not disappear	0.20	0.20	0.20	5
Devices compatible	0.00	0.00	0.00	4
Devices simultaneously	1.00	0.50	0.67	8
Double subscription on one account	0.02	0.20	0.04	25
Heartland	0.68	0.67	0.68	120
How do I become a member	0.00	0.00	0.00	19
Login details	0.09	0.02	0.03	129
Only trailers	0.19	0.13	0.15	126
Password lost	0.03	0.26	0.06	19
Pause subscription	0.15	0.40	0.22	10
Payment Methods	0.00	0.00	0.00	5
Payment failed	0.03	0.03	0.03	35
Payment of selected subscription	0.00	0.00	0.00	23
Price increase - informing why	0.25	0.10	0.14	10
Refund - email reminder not received	0.00	0.00	0.00	11
Refund after trial period	0.02	0.25	0.03	4
Refund global	0.23	0.11	0.15	89
Resend password reset email	0.03	0.09	0.04	22
Send activation email	0.05	0.27	0.08	11
Smart TV app - issues	0.00	0.00	0.00	5
Subscription costs	0.00	0.00	0.00	20
Technical issues	0.59	0.17	0.26	682
Use google chromecast	0.00	0.00	0.00	20
Use subtitles	0.03	0.06	0.04	18
Voucher code	0.17	0.39	0.24	28
Watch tv	0.35	0.50	0.41	149
What is	0.00	0.00	0.00	4
Why subscription cancelled	0.00	0.00	0.00	4
Working on a solution	0.05	0.10	0.07	29
content statement	0.08	0.02	0.04	41

Table 30: Classification result BiLSTM

Class	precision	recall	f1-score	support
Account - no subscription	0.00	0.00	0.00	35
Account details	0.62	0.08	0.14	682
App available	0.02	0.25	0.04	4
App malfunction - refer to website	0.18	0.19	0.18	54
App update	0.06	0.07	0.06	58
Availability countries	0.11	0.18	0.13	11
Browser	0.17	0.13	0.14	39
Cancel subscription	0.87	0.74	0.80	1419
Cancelled subscription - confirmation	0.02	0.03	0.02	78
Cannot log in - username already in use	0.04	0.37	0.07	30
Change payment details	0.65	0.56	0.60	27
Change size of subtitles	0.67	0.33	0.44	6
Chromecast - icon missing	0.17	0.24	0.20	17
Chromecast - issue is being worked on	0.08	0.07	0.08	54
Chromecast - subtitles	0.21	0.33	0.26	33
Continue watching - how to	0.17	0.14	0.15	7
Continue watching - titles do not disappear	0.40	0.40	0.40	5
Devices compatible	0.00	0.00	0.00	4
Devices simultaneously	0.44	0.50	0.47	8
Double subscription on one account	0.03	0.20	0.05	25
Heartland	0.66	0.69	0.68	120
How do I become a member	0.00	0.00	0.00	19
Login details	0.05	0.02	0.02	129
Only trailers	0.26	0.29	0.28	126
Password lost	0.02	0.05	0.03	19
Pause subscription	0.10	0.30	0.15	10
Payment Methods	0.00	0.20	0.01	5
Payment failed	0.00	0.00	0.00	35
Payment of selected subscription	0.05	0.04	0.05	23
Price increase - informing why	0.36	0.40	0.38	10
Refund - email reminder not received	0.17	0.36	0.24	11
Refund after trial period	0.00	0.00	0.00	4
Refund global	0.70	0.43	0.53	89
Resend password reset email	0.06	0.05	0.05	22
Send activation email	0.29	0.18	0.22	11
Smart TV app - issues	0.00	0.00	0.00	5
Subscription costs	0.24	0.25	0.24	20
Technical issues	0.54	0.51	0.53	682
Use google chromecast	0.03	0.10	0.05	20
Use subtitles	0.33	0.17	0.22	18
Voucher code	0.64	0.32	0.43	28
Watch tv	0.05	0.01	0.02	149
What is	0.25	0.25	0.25	4
Why subscription cancelled	0.00	0.00	0.00	4
Working on a solution	0.00	0.00	0.00	29
content statement	0.09	0.12	0.10	41

Table 31: Classification result BiLSTM + CNN

Class	precision	recall	f1-score	support
Account - no subscription	0.04	0.03	0.03	35
Account details	0.46	0.36	0.40	682
App available	0.00	0.00	0.00	4
App malfunction - refer to website	0.22	0.11	0.15	54
App update	0.08	0.09	0.08	58
Availability countries	0.29	0.18	0.22	11
Browser	0.08	0.21	0.12	39
Cancel subscription	0.91	0.72	0.80	1419
Cancelled subscription - confirmation	0.07	0.05	0.06	78
Cannot log in - username already in use	0.05	0.20	0.08	30
Change payment details	0.63	0.63	0.63	27
Change size of subtitles	0.33	0.17	0.22	6
Chromecast - icon missing	0.10	0.12	0.11	17
Chromecast - issue is being worked on	0.11	0.28	0.16	54
Chromecast - subtitles	0.15	0.15	0.15	33
Continue watching - how to	0.20	0.14	0.17	7
Continue watching - titles do not disappear	0.00	0.00	0.00	5
Devices compatible	0.00	0.00	0.00	4
Devices simultaneously	0.50	0.38	0.43	8
Double subscription on one account	0.02	0.04	0.02	25
Heartland	0.56	0.78	0.66	120
How do I become a member	0.00	0.00	0.00	19
Login details	0.06	0.02	0.02	129
Only trailers	0.24	0.30	0.27	126
Password lost	0.02	0.11	0.04	19
Pause subscription	0.17	0.40	0.24	10
Payment Methods	0.20	0.20	0.20	5
Payment failed	0.17	0.03	0.05	35
Payment of selected subscription	0.06	0.17	0.09	23
Price increase - informing why	0.40	0.20	0.27	10
Refund - email reminder not received	0.05	0.64	0.10	11
Refund after trial period	0.00	0.00	0.00	4
Refund global	0.29	0.02	0.04	89
Resend password reset email	0.07	0.23	0.10	22
Send activation email	0.10	0.27	0.15	11
Smart TV app - issues	0.00	0.00	0.00	5
Subscription costs	0.10	0.20	0.13	20
Technical issues	0.51	0.50	0.51	682
Use google chromecast	0.00	0.00	0.00	20
Use subtitles	0.12	0.39	0.18	18
Voucher code	0.36	0.36	0.36	28
Watch tv	0.42	0.50	0.45	149
What is	0.00	0.00	0.00	4
Why subscription cancelled	0.00	0.00	0.00	4
Working on a solution	0.03	0.03	0.03	29
content statement	0.14	0.02	0.04	41

Table 32: Classification result BiLSTM + Attention

G Distribution tickets period (12-07-2021 until 08-08-2021)

In this appendix in table 33 the distribution of the tickets from 12-07-2021 until 08-08-2021 is given.

General type	size (n)
Account and subscription	1,325
Film/series related	273
Payments	201
Technical	385
Total	**2,188**

Table 33: Distribution incoming tickets period (12-07-2021 until 08-08-2021)