INTENT CLASSIFICATION OF DIALOGUES AND DIALOGUE TURNS

Thesis for Master in Data Science
2019 -2020

Ziazopoulos Panteleimon

Academic Supervisor: Ion Androutsopoulos

Company Supervisor : Stauros Lagousis

Offered by Department of Informatics and the Department of Statistics
School of Information Sciences and Technology
Athens University of Economics and Business

Abstract

This work deals with automatic classification of dialogues and dialogue turns. These dialogues came from the the call center of the company KTEO Hellas. The Pobuca company converted these dialogues from speech to text. The later is the format in which the data will be processed in this whole project. The analysis that follows aims to create 2 types of classifiers:

- those that will specify the reason why the customer made the call,
- those that will make predictions about the content of each turn of the dialogue.

The models that were implemented for both types of classifiers are :

1. logistic regression,
2. a classifier that uses a recurrent neural network (RNN),
3. a classifier based on a pre-trained Greek BERT model.

Τίτλος διπλωματικής εργασίας

Ταξινόμηση πρόθεσης διαλόγων και εκφωνημάτων διαλόγων

Περίληψη

Η εργασία αυτή ασχολείται με την ταξινόμηση διαλόγων και εκφωνημάτων (turns) διαλόγων σε κατηγορίες. Αυτοί οι διάλογοι προήλθαν από το τηλεφωνικό κέντρο της εταιρείας KTEO Hellas. Η εταιρεία Pobuca μετέτρεψε τους διαλόγους από προφορικό λόγο σε κείμενο. Η τελευταία μορφή είναι αυτή στην οποία τα δεδομένα θα υποβληθούν σε επεξεργασία στο σύνολο του έργου αυτού. Η ανάλυση που ακολουθεί στοχεύει στη δημιουργία 2 τύπων ταξινομητών:

- εκείνων που θα προσδιορίσουν την πρόθεση που είχε ο πελάτης πραγματοποιώντας την κλήση,
- εκείνων που θα κάνουν προβλέψεις για το περιεχόμενο του κάθε εκφωνήματος (turn) του διαλόγου.

Τα μοντέλα που χρησιμοποιούνται και για τους δύο τύπους ταξινομητών είναι :

1. λογιστική παλινδρόμηση (logistic regression),
2. ένας ταξινομητής που χρησιμοποιεί ένα ανατροφοδοτούμενο νευρωνικό δίκτυο (recurrent neural network, RNN),
3. ένας ταξινομητής βασισμένος σε ένα προ-εκπαιδευμένο ελληνικό μοντέλο BERT.

Contents

1 Introduction **1**
- 1.1 Overview of the Problem . . . 1
- 1.2 Importance of this Project . . . 2
- 1.3 Deeper Understanding of the Dataset . . . 2
  - 1.3.1 Presentation of the Dataset . . . 2
  - 1.3.2 Data Pre-processing . . . 5
- 1.4 Evaluation Metrics . . . 7

2 Related Work **9**

3 Models **10**
- 3.1 Logistic Regression Classifier . . . 10
  - 3.1.1 Logistic Regression Classifier using tf-idf Features . . . 10
  - 3.1.2 Logistic Regression Classifier using Word Embeddings . . . 11
- 3.2 Recurrent Neural Network (RNN) . . . 11
  - 3.2.1 BiRNN . . . 12
  - 3.2.2 Self Attention RNN . . . 12
- 3.3 BERT . . . 13

4 Model Implementations and Results **15**
- 4.1 Dialogue Classification . . . 15
  - 4.1.1 Logistic Regression with tf-idf Features . . . 15
  - 4.1.2 Logistic Regression with Word Embeddings . . . 17
  - 4.1.3 BiRNN . . . 18
  - 4.1.4 Self-Attention RNN . . . 20
  - 4.1.5 BERT Model . . . 21
- 4.2 Dialogue Turn Classification . . . 24
  - 4.2.1 Logistic Regression with tf-idf Features . . . 24
  - 4.2.2 Logistic Regression with Word Embeddings . . . 25
  - 4.2.3 BiRNN . . . 27
  - 4.2.4 Self-Attention RNN . . . 29
  - 4.2.5 BERT Model . . . 30

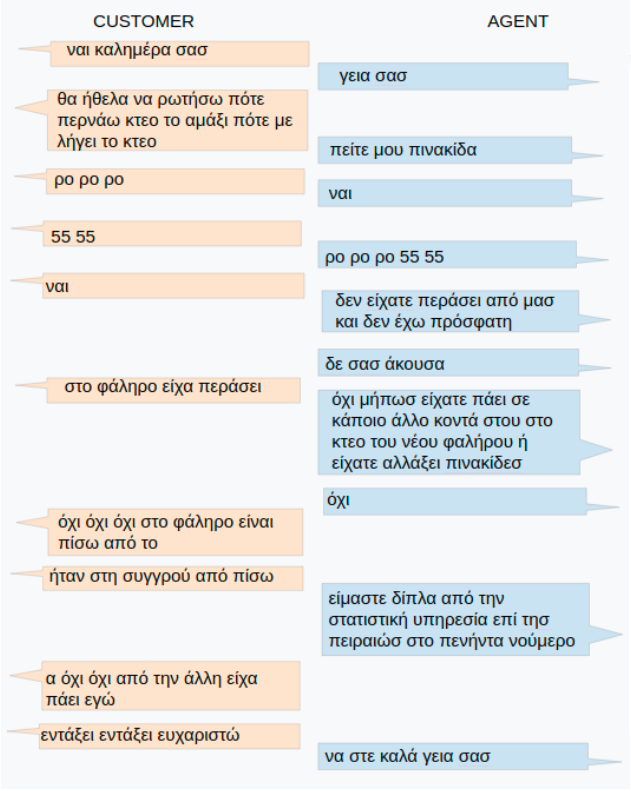
5 Conclusions and Future Work **35**
- 5.1 Dialogue Classification . . . 35
- 5.2 Dialogue Turn Classification . . . 35
- 5.3 Future Work . . . 36

References **37**

1 Introduction

1.1 Overview of the Problem

The following project analyses a dialogue classification problem. The dialogues have come from the call center of the company KTEO Hellas. The dialogues are in the Greek language. The Pobuca company, which provided the data, converted the dialogues from oral to written form. This conversion was done through an application built by Pobuca. This application was mainly a speech recognition system that automatically converted the speech of the dialogues to text. Moreover, through this application the annotator was able to manually annotate the intent of the customer that made the call by considering the text of the entire dialogue. However, a dialogue is a sequence of turns. That is why each turn of the dialogue was also manually labelled. In addition, each dialogue was characterised by 10 additional arithmetic variables, like the duration of the dialogue. The form of the dialogues that are processed in this project is shown in Figure 1.




```
[['ναι', 'καλημέρα', 'σασ'], ['γεια', 'σα
σ'], ['θα', 'θελα', 'να', 'ρωτήσω', 'πότ
ε', 'περνάω', 'κτεο', 'το', 'αμάξι', 'πότ
ε', 'με', 'λήγει', 'το', 'κτεο'], ['πείτ
ε', 'μου', 'πινακίδα'], ['ρο', 'ρο', 'ρ
ο'], ['ναι'], ['55', '55'], ['ρο', 'ρο',
'ρο', '55', '55'], ['ναι'], ['δεν', 'είχα
τε', 'περάσει', 'πάλι', 'από', 'μασ', 'κα
ι', 'δεν', 'έχω', 'πρόσφατη'], ['δε', 'σα
σ', 'άκουσα'], ['στο', 'φάληρο', 'είχα',
'περάσει'], ['όχι', 'μήπωσ', 'είχατε', 'π
άει', 'σε', 'κάποιο', 'άλλο', 'κοντά', 'σ
του', 'στο', 'κτεο', 'του', 'νέου', 'φαλή
ρου', 'ή', 'είχατε', 'αλλάξει', 'πινακίδε
σ'], ['όχι'], ['όχι', 'όχι', 'όχι', 'στ
ο', 'φάληρο', 'είναι', 'πίσω', 'από', 'τ
ο'], ['ήταν', 'στη', 'συγγρού', 'από', 'π
ίσω'], ['είμαστε', 'δίπλα', 'από', 'την',
'στατιστική', 'υπηρεσία', 'επί', 'τησ',
'πειραιώσ', 'στο', 'πενήντα', 'νούμερο'],
['α', 'όχι', 'όχι', 'από', 'την', 'άλλη',
'είχα', 'πάει', 'εγώ'], ['εντάξει', 'εντά
ξει', 'ευχαριστώ'], ['να', 'στε', 'καλά',
'γεια', 'σασ']]
```

Figure 1: An example of a random dialogue. On the left the dialogue is depicted as it is split to its turns. On the right the raw form of the same dialogue is shown.

Since each turn of the dialogue is also annotated, two different perspectives of classification are adopted:

- Build classifiers that predict the intent of whole dialogues.
- Build classifiers that predict the intent of dialogue turns.

The types of classifiers that were tested in this project were:

1. logistic regression,
2. a classifier that uses a recurrent neural network (RNN),
3. a classifier based on a pre-trained Greek BERT model.

1.2 Importance of this Project

The present work aims to provide an automated approach to the classification of calls received by a call center. For a company that receives a variety of calls on a daily basis (such as KTEO Hellas) it is very important to be able to have an objective and instant update of its customers needs. It is very helpful for it to know if its customers have a complaint and if so why. Also, knowing how often and for what reasons its customers request additional information may indicate points that the company has not clarified or advertised to the necessary extent. Furthermore, when the above knowledge is offered through automated processes, the human factor is eliminated, a state that provides the company with a more realistic picture of what actually happens.
All of the above, in combination with the progress of computer science and the ease with which the above processes are automated nowadays, make it necessary for every modern company to integrate in its structure classifiers such as those that will be described in the following chapters.

1.3 Deeper Understanding of the Dataset

1.3.1 Presentation of the Dataset

The data to be processed in this project are pre-processed recorded dialogues from the call center of the KTEO Hellas company. The initial form of the dialogues was conversations between a possible customer and an agent from the company. Due to the company's domain the possible customer's intents to call the call center of such a company are limited. So it can be assumed that the following work concerns domain specific, task oriented dialogues. It is important to be mentioned that in the work below only the text of the dialogues will be used. The possible characteristics that might be deduced from the speech of the speakers, like sentiment, do not affect any of the annotation of the data this project deals with. The manual annotation of the training and test dialogues (ground truth) was also done exclusively according to the text of the dialogues or the dialogue turns by considering suggestions made by Pobuca's app and were always checked from the human annotator.
Each dialogue was classified according to the intent of the customer in one out of four possible categories. The possible intent categories for dialogues are:

- Make a reservation.
- Ask for information.
- Make a complaint.
- Anything else (other).

Aside from the text, each dialogue was also annotated with some other features like the gender of the customer or the store he/she wants to make a reservation for, but those characteristics are

beyond the scope of our task. Furthermore, each dialogue came with additional information. This kind of information is shown in Table 1.

Table 1: Additional characteristics of the dialogues

Variables	Mean	St.D.
Agent duration: total time (in seconds) the agent spoke	49.32	34.88
Customer duration: total time (in seconds) the customer spoke	46.36	32.20
Agent's largest monologue: duration (in seconds) of the agent's largest turn	5.89	3.15
Customer's largest monologue: duration (in seconds) of the agent's largest turn	6.74	3.87
Agent's words per minute	239.07	69.52
Duration: total duration (in seconds) of the dialogue	89.69	59.94
Patience: total time (in seconds) that nobody spoke during the dialogue	0.47	0.95
Interrupts by the agent: number of times the agent interrupted the customer in a dialogue	9.15	6.94
Interrupts by the customer: number of times the customer interrupted the agent in a dialogue	6.10	4.76
Switches: number of times a change of the speaker happened in a dialogue	26.04	17.24

Some statistics about the dialogues are:

- Number of words per dialogues: 196.58 ± 133.57 (Figure 2).
- Length of the words used: 4.51 ± 2.45.
- Number of unique words (vocabulary size): 4,990.
- Total number of words: 123,455.

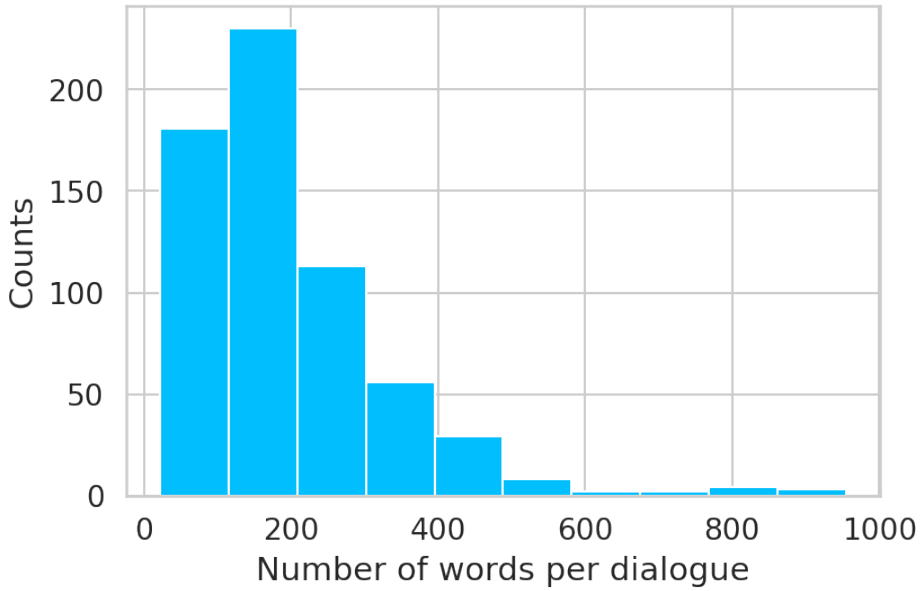


Figure 2: Histogram of the number of words used per dialogue, with minimum 22 words in a dialogue and maximum 955.

The small vocabulary size, compared to the total number of the words used, confirms the assumption of domain specific dialogues. Patience lasting significantly less that the total duration of the dialogue validates that the dialogues are task oriented since neither the customer nor the agent have available time to spend. These characteristics combined with the small duration of the dialogues (frequently less than 2 minutes) show the speed with which information is transferred and the swiftness with which customers are served.

On the other hand, the annotator also labelled each turn of the dialogue according to its content. Each time the speaker changed, his/her utterance (dialogue turn) was classified according to the intent of the speaker as inferred from the text. It is important to mention that sometimes there was not an actual change of the speaker but the company's app split the talk of the current speaker due to noise (actual background noise or extended pause).
The possible intents for a dialogue turn are 22. These intents are mutually exclusive. Also, these intents are different for the two types of speakers (agent, customer). There are 12 intent categories for the turns of the agent, 9 intent categories for the turns of the customer and 1 'None' intent category that includes all the turns that do not belong in any of the intent categories already specified.
Some statistics about the turns of the dataset are:

- 8,253 turns during which the customer was talking.
- 9,111 turns during which the agent was talking.
- Number of words per turn: 7.20 ± 6.97 .

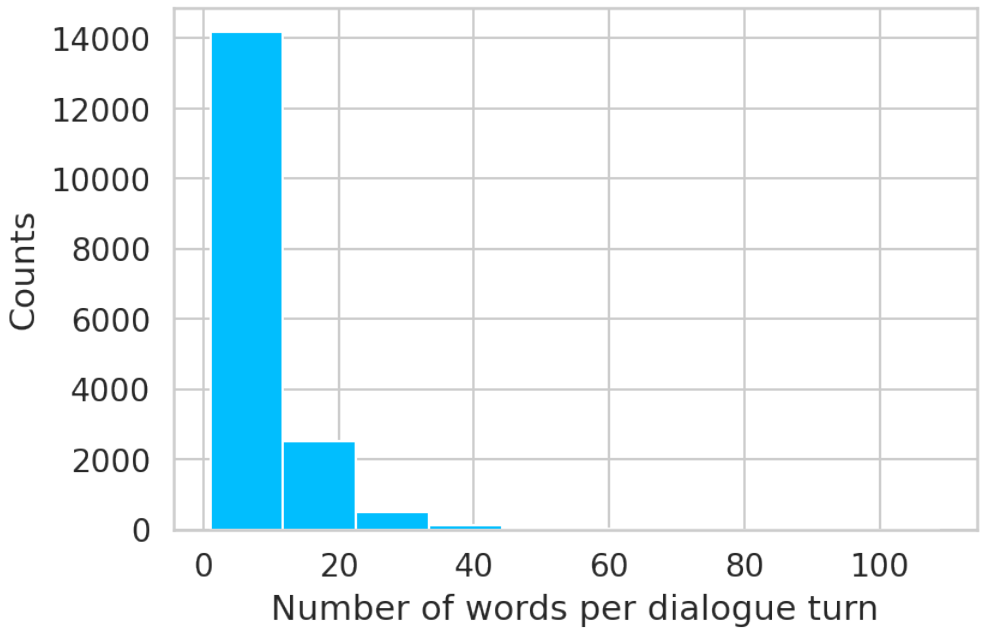


Figure 3: Histogram of the number of words used per turn, with minimum 1 word in a dialogue turn and maximum 109.

Considering that the agent is the person who would always talk first, by greeting the customer, and also the last to speak, by saying goodbye, it is expected for the agent to have, on average, at least one more turn in each dialogue. Taking into account the fact that the number of dialogues is around 600, we see that the dialogues are quite balanced between the 2 speakers. This statement is also supported by the almost equal duration of speech of the customer and the agent.

To sum up, we end up having 2 datasets:

- 635 dialogues classified to 4 intent categories.
- 17,364 dialogue turns classified to 22 intent categories.

1.3.2 Data Pre-processing

One problem in the dataset is the large class imbalance. Regarding the dialogues, there are only 9 examples for the 'complaint' category. With only them it is impossible to conduct adequate training of a classifier and at the same time leave examples for hyperparameter tuning and also test the performance of the classifier. For this reason the category 'complaint' was removed and was not considered in the rest of this work, even though the prediction of the 'complaint' class might be the most meaningful category for the company's needs. However, the removal of this category will not significantly affect the way in which the classification problem is approached as it remains a single-label multi-class classification problem and a not binary one (there are more than 2 classes, but each dialogue or turn is classified to only one class). So it is quite easy that for the methods that were developed in this project to be generalised from a 3-class classification problem to a 4-class one, once the company has collected a sufficient number of examples that fall into the 'complaint' category.
Therefore the final dataset for the classification of the dialogues consisted of:

- 269 examples of 'reservation'.
- 206 examples of 'information'.
- 153 examples of 'other'.

So the first problem of this project is a 3-class dialogue classification with 628 data examples. Since the dialogues are classified as a whole, all the turns of each dialogue were concatenated to form a text. The text format contains one turn after the other (Figure 5).

As for the dialogue turns, there are some categories that have few examples for both training and testing a classifier. For example, the 'CUSTOMER RESERVATION CONFLICT' class consists of 9 examples and the 'AGENT HANDLE SUPERVISOR' class has only 1. Furthermore, some other categories may have sufficient amount of data but consist of greatly fewer examples compared to other classes. For example, the 'None' class has 13,467 when 'CUSTOMER DEFINE PHONE' has 33 and 'CUSTOMER DEFINE NAME' has 35. This is a ratio 408/1. The existence of these 'minorities' acts like noise in the whole classification process because the classifier cannot be trained to learn the characteristics of these small categories, let alone predict them correctly when it encounters an example similar to them. In addition, most examples are marked as 'None' since the content of most sentences cannot be included in one of the above classes. That having been said it is preferable to limit the extent of this particular class.

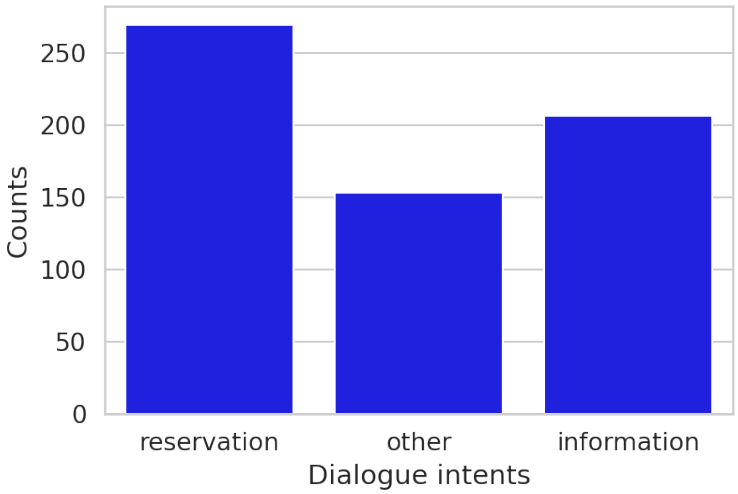


Figure 4: Barplot of the intent classes of dialogues after the 'complaint' category was left out.

```
'ναι καλημέρα σασ γεια σασ θα θελα να ρωτήσω πότε περνάω κτεο το αμάξι πότε με λήγει το κτεο πείτε μου πινακίδα ρο
ρο ρο ναι 55 55 ρο ρο ρο 55 55 ναι δεν είχατε περάσει πάλι από μασ και δεν έχω πρόσφατη δε σασ άκουσα στο φάληρο ε
ίχα περάσει όχι μήπωσ είχατε πάει σε κάποιο άλλο κοντά στου στο κτεο του νέου φαλήρου ή είχατε αλλάξει πινακίδεσ ό
χι όχι όχι όχι στο φάληρο είναι πίσω από το ήταν στη συγγρού από πίσω είμαστε δίπλα από την στατιστική υπηρεσία επ
ί τησ πειραιώσ στο πενήντα νούμερο α όχι όχι από την άλλη είχα πάει εγώ εντάξει εντάξει ευχαριστώ να στε καλά γεια
σασ'
```

Figure 5: Example of a dialogue after all its turns were concatenated to form a text.

All the above contributed to make a cluster with all the examples of 'minorities' classes and form a bigger class. As 'minorities' we considered all classes with fewer than 50 examples. In this new cluster the examples of the 'None' class were also included. Then this cluster was split in two new classes: 'AGENT OTHER' and 'CUSTOMER OTHER'. The separation of this cluster was done according to the speaker of each particular turn. The new classes are two and not one in order to limit the extent of the 'None' class and at the same time maintain the separation of the classes between the two speakers. Now all the labels refer to the person who talks, something that proved very useful in the study that followed. The rest of the classes remained as they were. However, even after merging those categories the dataset remained highly imbalanced (Figure 6).

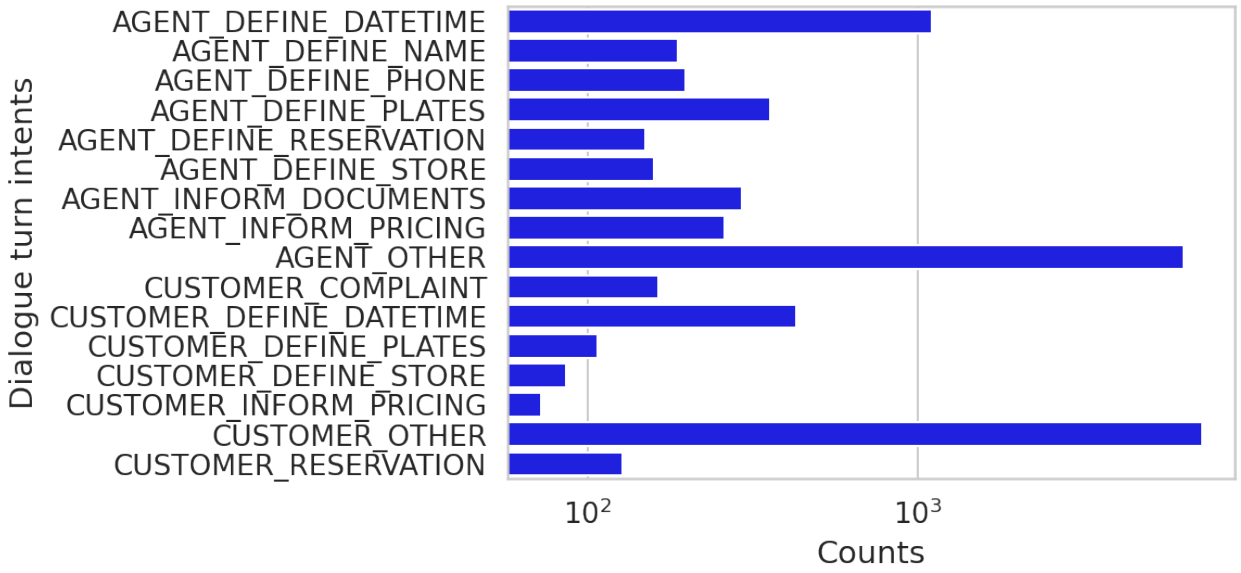


Figure 6: Barplot of the intent classes of dialogue turns, after creating the classes 'AGENT OTHER' and 'CUSTOMER OTHER'.

In numbers 'AGENT OTHER' constitutes 36.90% of the whole dataset while 'CUSTOMER OTHER' accounts for 41.87% of the dataset. Taking into account that 'AGENT DEFINE DATETIME' and 'CUSTOMER DEFINE DATETIME' make up 6.34% and 2.46% respectively we conclude that the rest 12.43% is shared between the 12 remaining categories (roughly 1% for each of the remaining classes). The percentages mentioned above just highlight the huge imbalance of the dataset of this classification problem. A promising fact for the dialogue turn problem is that some of the 'small' categories that remained consisted of unique phrases and with small variations between its different examples. This characteristic may help the classifiers cope with these 'small' categories. So, despite the few examples, it is easier to predict them correctly when a test example of such a class occurs. Such an example is the category 'AGENT INFORM PRICING' where almost all of its examples were the exact phrase 'είναι 55 ο έλεγχος και ενσωματωμένη η κάρτα καυσαερίων' ('it is 55 the control including the exhaust card).

To sum up, the dialogue turn classification problem was converted to a 16-class intent classification problem over a highly imbalanced dataset (Figure 6).

1.4 Evaluation Metrics

The classifiers of this project will be evaluated with the metrics that follow:

1. Accuracy: the fraction of the total successful predictions by the total number of examples.

2. Balanced accuracy: the average of the recall over all classes. For each, recall is defined as follows:

$$recall = \frac{\text{true positive}}{\text{true positives + false negatives}}$$

In other other words, balanced accuracy is macro-averaged recall.

3. F1 score:

$$F1 = \frac{2^* \text{ precision } * \text{ recall}}{\text{precision + recall}}$$

The weighted option of F1 score will be used in order to take into account the imbalance of the classes (especially for the dialogue turn classification problem). That means calculate the F1 score for each class and then find their average weighted mean according to the number of true instances for each label. It is a weighted calculation of macro-F1.

$$\text{Weighted F1} = \frac{\sum_i^N \text{F1}_i \cdot ntrue_i}{\text{total number of examples}}$$

where N is the number of classes, $F1_i$ is the F1 score regarding class i and $ntrue_i$ is the number of true instances of class i.

4. Logarithmic Loss (or just 'log-loss'): the negative log-likelihood for multi-classification problem.

$$\text{Multiclass log-loss} = -\frac{1}{N}\sum_i^N \sum_j^M y_{ij} \cdot ln(p_{ij}))$$

where N is the number of examples, M is the number of classes, y_{ij} is the binary variable with the ground truth (manually provided) labels and p_{ij} is the classification probability output by the classifier for the i-example and the j-class. This metric was not used only for evaluation of the models but also as loss metric when training the models.

2 Related Work

The TOD-BERT (Wu et al., 2002, [1]) is a BERT model built especially for task-oriented dialogue classification tasks. Unlike the BERT models utilised in this project, the TOD-BERT was trained on nine different task-oriented datasets which are English, human-human and multi-turn. That is why it is able to learn better representations than existing pre-trained models for task-oriented downstream tasks. Also, in the TOD-BERT two additional special tokens, [USR] and [SYS], are used in order to capture speaker information and the underlying interaction behavior in dialogue. Similar attempts to split the text of the dialogues between the two speakers were implemented in the logistic regression models of this project. For training the TOD-BERT, all the turns of the same dialogue are concatenated into one flat sequence, similar to the pre-processing that was made for the dialogue classification problem of this project. The TOD-BERT outperforms strong baseline BERT models on intention classification, dialogue state tracking, dialogue act prediction and response selection. Regarding intent recognition on the OOS dataset, the TOD-BERT achieves 1.7% all-intent accuracy improvement compared to BERT.
Herzig et al. (2016) [2] propose a different approach for classifying emotions in customer support dialogues in social media. In the classifiers they analysed they included features like the time elapsed between the timestamp of the last customer/agent turn and the timestamp of the subsequent turn, the day of the week, enumerating the turns of each dialogue. They demonstrated the importance of such features when combined with other textual features, like n-grams, exclamation marks, question marks. Similarly in this project, aside from data extracted from the text of the dialogues, features like the time duration of the dialogue, the number of times a change of the speaker occurred in the dialogue, etc. (Table 1) were included in the dialogue classification experiments.

3 Models

3.1 Logistic Regression Classifier

The logistic regression classifier is a discriminative classifier that learns what features from the input are most useful to discriminate between the different possible classes. Discriminative systems are often more accurate and hence more commonly used in supervised classification, that is why logistic regression was chosen as a first approach for tackling the problem. Moreover, logistic regression is an easy and quickly trained algorithm. This allows making many tests regarding the way the data may be further processed. Two main paths can be distinguished during the application of logistic regression regarding the type in which the data will be converted in order to become proper input for the classifier. The two types to be tested are tf-idf features and word embeddings. The hyper-parameters of logistic regression that were tuned were:

- Regularization strength.
- Type of regularization (e.g. squared L1 or L2 norm of the weights).

For all logistic regression models the splitting of the dataset was 72.25% for training, 12.75% for the development set and 15% for testing. It is important to mention that all the data splits of this project were stratified in order to keep the same proportions of the classes with initial dataset in all sets.

3.1.1 Logistic Regression Classifier using tf-idf Features

Tf-idf stands for term frequency multiplied by inverse document frequency.

$$\text{tf(i)} = \frac{\text{number of times word i appears in the current document}}{\text{total number of words of the current document}}$$

$$\text{idf(i)} = \log\left(\frac{\text{total number of documents}}{\text{number of documents word i appears in}}\right)$$

$$\text{tf-idf(i)} = \text{tf(i)} * \text{idf(i)}$$

The process of calculating the tf-idf scores of the words and arranging these scores to a vector for each dialogue is done by the vectorizer. This function, by using the train set, calculates the tf-idf scores of the top N most frequent words (N is a hyperparameter to be tuned), computing tf scores over the entire dataset. Then, each dialogue is converted to a vector with elements the tf-idf scores of the most frequent words the vectorizer held that appeared in this particular dialogue, now computing the tf scores separately for each dialogue.
The set of hyperparameters the vectorizer introduces to the problem is:

- Max features: the number of most frequent n-grams across the corpus.
- N-gram range: the range of n-grams whose frequency would be checked (like only unigrams, or unigrams and bigrams, or just bigrams, etc).
- Max df: the maximum document frequency of a word. If a word appears more often than this threshold then it is ignored.

- Min df: the minimum document frequency of a word. If a word appears less often than this threshold then it is ignored.
- Sublinear tf: In sublinear term frequency, scaling will be applied to tf-idf calculation ({ tf(i) } term replaced by {1 + log(tf(i))}).

Some additional arguments that are applied to the vectorizer are:

- A vocabulary of stopwords.
- Smoothing of df(i) in order to prevent zero divisions ({ df(i)} $\rightarrow$ { 1 + df(i) }).
- All words converted to lowercase.

3.1.2 Logistic Regression Classifier using Word Embeddings

Word embeddings are vector representations of words. Each word is mapped to a dense vector in such a way that words with similar meaning are 'close' in the vector space of the vector representation. Common metrics that quantify the 'closeness' of 2 vectors is cosine similarity and Euclidean distance. Moreover, words that appear often in the same text may have cosine similarity closer to 1 than words that are completely irrelevant.
Due to the small vocabulary size of the dialogues it is not recommended to build word embeddings based on the data that KTEO Hellas provided. It is more promising to use the Greek FastText pre-trained word vectors (Grave et al., 2018 [3]). These vectors were trained on Common Crawl and Wikipedia using fastText and Continuous Bag of Words Model (CBOW) with position-weights, with 300 dimensions, with character n-grams of length 5 (Joulin et al., 2016 [4]).
But, since each vector represents a single word, a representation for each example (dialogue or dialogue turn) as a whole has to be made. The basic approach is to calculate the centroid of the embeddings of the words of each example (tf-centroid). Another method is to calculate the tf-idf-centroid of each example, that is, to multiply every word vector with its respective idf score and then calculate the centroid of the 'new' vectors for each example (tf-idf-centroid).

3.2 Recurrent Neural Network (RNN)

An RNN is any network that contains a cycle within its network connections. RNNs are networks with loops in them, allowing information to persist. In this project RNNs are used since it is unclear how a traditional neural network could use its reasoning about previous examples to inform later ones. However, even in RNNs it is possible for the gap between the relevant information and the point where it is needed to become very large. That is why the use of the Long Short Term Memory network architecture (LSTM) is adopted.
For all RNN models the split of the dataset was 76.5% for train, 13.5% for development and 10% for test in order to give more examples for training. This was considered necessary since RNNs had many more parameters to be trained. However, this split has the disadvantage that less than 70 examples were left for test. That means that one misclassification leads to around 1.5 drop at the accuracy score. Another disadvantage is that the results of the RNN are not directly comparable to those of the logistic regression classifier, since they are trained on different splits (and with different sizes of the training set).

3.2.1 BiRNN

LSTMs are explicitly designed to avoid the long-term dependencies problems. However, LSTM layers are computational heavy to train. The initial thought was to use GRU layers (Cho et al., 2014 [5]) instead of LSTM since they are faster to train, but the results they gave were lower than the RNNs that used LSTMs. So only two LSTM layers were inserted to the RNNs utilised in this project, one reading each example from left to right and the other one reading from right to left (bidirectional LSTM). This way 2 output vectors will be created for the same text and then these 2 will be concatenated to form a single vector. Afterwards, pairs of dropout layers and dense layers are stacked, and they are fed with the output (concatenated vector) of the bidirectional LSTM. The hyperparameters to be tuned for this architecture are:

- Number of pairs of (Dropout + Dense) layers.
- Number of neurons of LSTM layer.
- Number of neurons for the dense layer.
- Dropout rate before the LSTM layer is applied (right after the embedding layer).
- Dropout rate between danse layers.
- Activation function of dense layers.
- Learning rate of the optimizer.

3.2.2 Self Attention RNN

Even with LSTMs, RNNs often cannot actually cope with long-distance dependencies, i.e., information that has to be remembered from earlier parts of a long sequence of words. Often the LSTM has forgotten the earlier parts of the sequence once it has processed the entire sequence. Applying bidirectional LSTMs only helps the classifier to remember better information about the very start and the end of the text. Some technigues that deal with this problem suggest applying pooling (Max or Mean pooling) to the intermediate states of the LSTM layer. Such technigues were tested in this project but their results were not promising. The attention mechanism (Vaswani et al., 2017 [6]) was born to resolve this problem. Attention in deep learning can be broadly interpreted as a vector of importance weights.
In this project the attention mechanism is implemented as:

$$\widetilde{h}_i = [\vec{h}_i^T; \overleftarrow{h}_i^T]$$
$$\vec{e}_{\mathrm{i}} = tanh(W \cdot \widetilde{h}_i)$$
$$a_{\mathrm{i}} = softmax(\vec{u} \cdot \vec{e}_{\mathrm{i}})$$
$$\vec{c}_{\mathrm{i}} = \sum_{i=1}^{N} a_{\mathrm{i}} \cdot \widetilde{h}_i$$

where:

- $\widetilde{h}_i$ is the concatenation of the forward and backward vectors of state j of the LSTM layer.
- $\vec{e}_{\mathrm{i}}$ is the hidden layer of a self-attention MLP.
- a_{i} reflects the importance of the annotation $\mathrm{h_j}$ with respect to hidden state $\mathrm{h_i}$.

- $\vec{c}_i$ is the output context vector, which represents the entire sequence.
- W matrix and $\vec{u}$ vector are the weights to be learned throughout the training process.

The main architecture for the self attention RNN model was the same as BiRNN model (Chapter 3.2.1) with the only addition of an attention mechanism over the LSTM layer. The hyperparameters to be tuned were also the same.

3.3 BERT

BERT (Bidirectional Encoder Representations from Transformers, Devlin et al., 2019 [7]) is based on the Transformer architecture (Vaswani et al., 2017 [6]). The general Transformer uses an encoder and a decoder network, however BERT uses only the encoder to learn a latent representation of the input text. As input, BERT takes the sum of token embeddings, segment and position embeddings. The positional embeddings represent the position of words in a sequence. Segment embeddings are useful when BERT is trained on sentence pairs for tasks that take sentence pairs as input (e.g. question answering). In this project, since their are no sentence pairs the segment embeddings are of no importance.

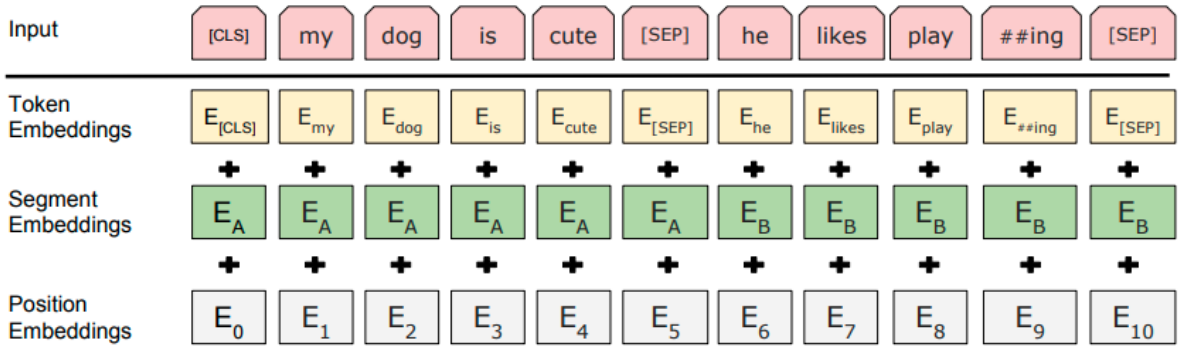


Figure 7: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings (Devlin et al., 2019 [7]).

Fine-tuning is straightforward since the self-attention mechanism in the Transformer allows BERT to model many downstream tasks. Each token embedding represents a sub-word of the input sentence[1]. In order to hold a representation for the meaning of the whole input sentence without changing the token embeddings, an additional token is inserted, the [CLS] token. When using BERT for classification tasks, such as text classification or sentiment analysis, the final-layer [CLS] token is fed into an additional output neural network. The other token embeddings are used for tasks like part of speech tagging, where an output token is needed for each input word. So, in this project only the final [CLS] token of the Greek-BERT (Koutsikakis et al., 2020 [8]) was used, which is fed to additional dense layers that finally classify each example. The [SEP] token represents separation of the input example so that BERT will be able to recognise the sentences. Since no sentence pairs were used in this project, each example had only 1 [SEP] token at its end (Figure 8).

[1] 'Sub-word' term as it is defined by Hugging Face, see
https://huggingface.co/transformers/master/tokenizer_summary.html.

In this project Greek-BERT (Koutsikakis et al., 2020 [8]) was used since all text data are in Greek. Greek-BERT is a transformer-based language model for Greek, a BERT-base pre-trained on Greek corpora, trained on 29 GB of Greek text (Koutsikakis et al., 2020 [8]). At the final layer of this model a dropout layer and a dense layer, as output layer, were attached. Being limited in resources but mainly in time, the fine-tune process that followed included the hyperparameters below:

- Learning rate.
- Dropout rate after output layer of the Greek-BERT.
- The existence or not of an additional hidden dense layer between dropout layer and output layer (the number of its neurons was another hyperparameter).
- Keep frozen or not the weights of the Greek-BERT layers while training.

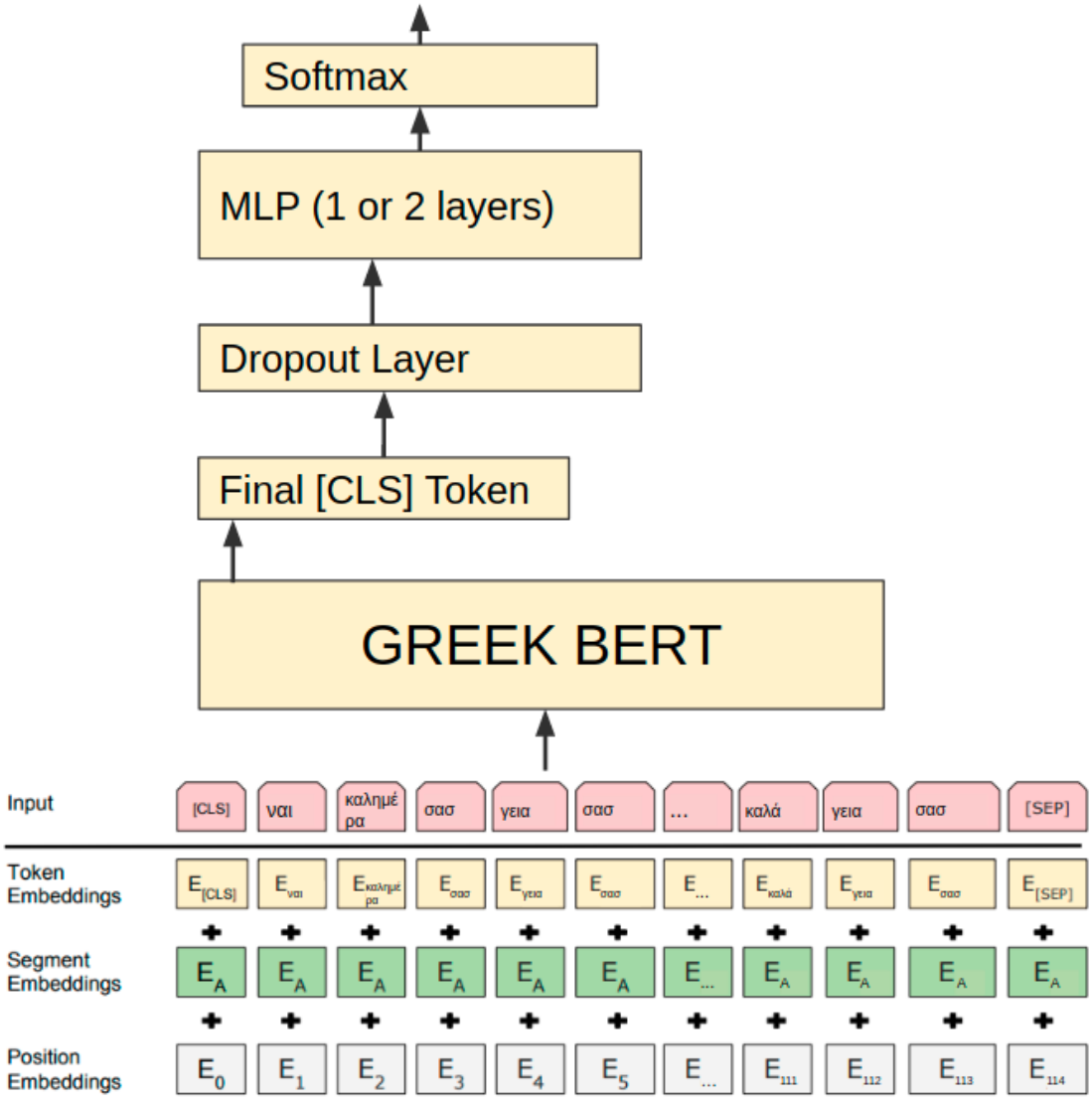


Figure 8: Basic BERT model architecture.

4 Model Implementations and Results

All five types of classifiers described in chapter 3 were implemented for both classification problems. The main tools used to implement these classifiers were extracted from the libraries:

1. Scikit-learn (Pedregosa et al., 2011 [9]) for the implementation of the logistic regression classifiers.
2. Tensorflow (Abadi et al., 2016 [10]) for the implementation of the RNN classifiers .
3. Pytorch (Paszke et al., 2017 [11]) for the implementation of the BERT classifiers.

All the experiments ran in virtual machines of the Google Cloud Platform[2], utilising CPU for the logistic regression classifiers and GPU for the RNNs and the BERT model classifiers.

4.1 Dialogue Classification

4.1.1 Logistic Regression with tf-idf Features

Since the logistic regression classifier is easy to train many experiments were conducted. First of all, an attempt was to make a dialogue classification by using only the values of the variables of Table 1 (Chapter 1.3.1), without any text data. The values of these variables were normalised and then mapped to [0,1] as:

i-th value of variable x: $x_i \rightarrow \widetilde{x}_i = \frac{x_i - \sum_{j=1}^{N} x_j}{\sqrt{\sum_{j=1}^{N} |x_j - \sum_{k=1}^{N} x_k|^2}} \rightarrow \widetilde{\widetilde{x}}_i = \frac{\widetilde{x}_i - min(\widetilde{x})}{max(\widetilde{x}) - min(\widetilde{x})}$

Finally arranged to 10-dimensional vectors ('variable vectors'). The results of this classification are shown in Figure 9. The scores of this classifier were:

- Accuracy = 53.68%
- F1 score = 46.57%
- Log-loss = 0.9334

It is obvious that such a classifier is not a good one since 1 out of 3 classes in not predicted at all. However, even this classifier can find patterns in variables of Table 1 that characterise the 'information' and the 'reservation' categories. This statement is based on the fact that the numbers of right predictions for these 2 classes (Figure 9, upper left and bottom right boxes) are higher than the respective misprediction numbers. This observation proved useful in improving the results of the next classifier.

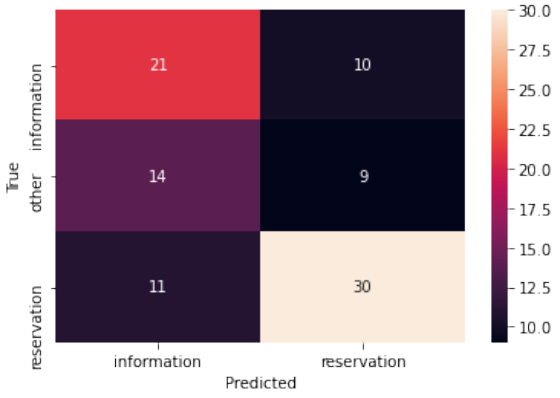


Figure 9: Heatmap of logistic regression predictions regarding dialogue classification, using as input data the 'variable vectors'.

[2]Google Cloud Platform. is a suite of cloud computing services offered by Google

After that, another logistic regression classifier was tested (tuned and evaluated) whose input was only tf-idf features from the text part of the dialogues. Its results were far better than the previous classifier. But, since the variables from Table 1 held information about this classification problem another technique was used.

Before the vectors of the tf-idf vectorizer were inserted to the logistic regression classifier, they were concatenated with the 'variable vectors' (see above) and the concatenated vectors were inserted to logistic regression. According to the development set results the highest F1 scoring model was selected. The hyperparameters of this model were: use only of unigrams, 200 tf-idf features, max tf 0.7, min tf 0.1 no sublinear tf scaling and regularization strength of logistic regression set to 1. Penalty norm was fixed to L2 since L1 norm did not give classifiers that converged.

The final scores were:

- Accuracy = 82.11%
- F1 score = 81.88%
- Log-loss = 0.5279

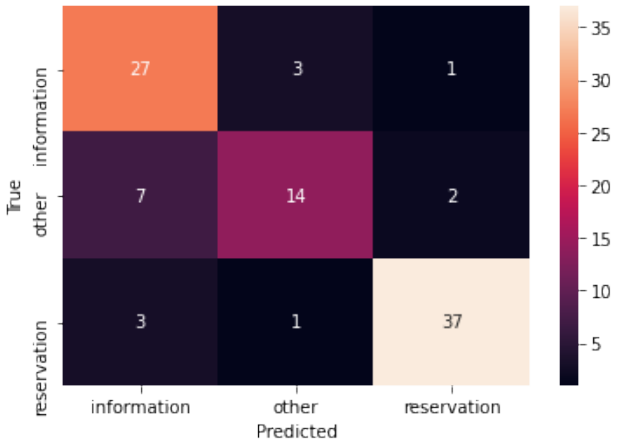


Figure 10: Heatmap of logistic regression predictions regarding dialogue classification, using as input data the concatenated vectors of tf-idf features and 'variable vectors'.

This was the highest scoring logistic regression model (Chapter 4.1.2, Table 2). Note the slightly more intense misclassification between the 'other' and the 'information' classes. This characteristic is one that will be present at most classifiers that followed.

The learning curves of this model are shown in Figure 11. In order to make smaller the margin between train and test curves feature selection was applied by making more strict regularization and taking into account only the words with the highest information gain. However, none of these methods gave scores as high as the classifier described above and at the same time having learning curves with smaller margin between train and test curves.

Other experiments that were tested included differentiation of the words used from the agent and the words used from the customer. That was accomplished by concatenating to every word of the customer the word ′πελατη′ (greek word that means 'customer'). In this way the vectorizer treated as different the words of the agent from the words of the customer. However, the scores of such a model were not as good as the model mentioned before (Chapter 4.1.2, Table 2).

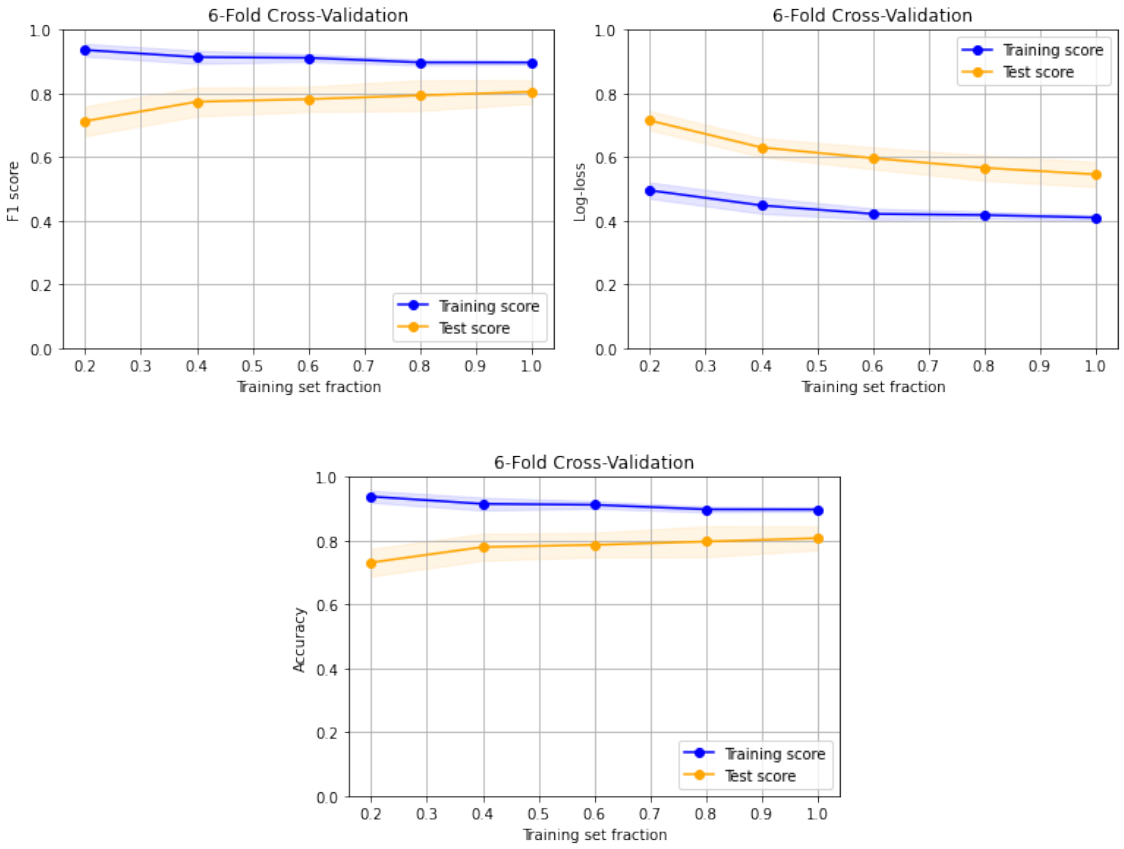


Figure 11: Learning curves of logistic regression regarding dialogue classification, using both tf-idf features and additional data from the variables of Table 1.

4.1.2 Logistic Regression with Word Embeddings

The experiments conducted for this type of classifier were to calculate the tf-centoid and tf-idf centroid of the words embeddings representing the dialogues. From these 2 alternatives, the tf-centroid approch scored higher F1 score (Table 2). For logistic regression, with regularization strength 1 and L2 penalty norm, the scores are:

- Accuracy : 56.84%
- F1 score : 49.28%
- Log-loss : 0.9529

Another approach that was checked was to create 2 texts instead of 1. The first text included only the turns of the agent and the second only the turns of the customer. Then the centroid of each text was calculated and was inserted to the logistic regression classifiers that were trained separately. Afterwards, the mean vector of the two probability vectors of the distinct classifiers was taken. Finally, the evaluation was done according to the mean vectors. Both tf and tf-idf centroids (Table 2) were tested but none of them scored higher at F1 than the tf-centroid classifier mentioned at the previous paragraph.

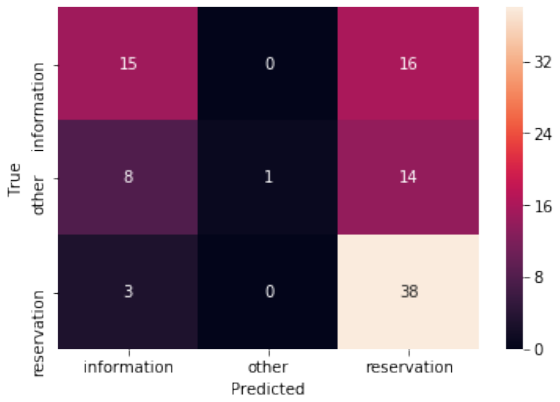


Figure 12: Heatmap of logistic regression predictions regarding dialogue classification, using input data the tf-centroid vectors of word embeddings of a dialogue.

Table 2: Scores of all logistic regression experiments for dialogue classification problem

Logistic Regression	Accuracy	F1	Log-loss
'variable vectors'	53.68%	46.57%	0.9334
tf-idf features	80.00%	80.02%	0.5542
both tf-idf-features and 'variable vectors'	**82.11%**	**81.88%**	**0.5279**
tf-idf-features, 'variable vectors', separation words of the customer	77.89%	77.88%	0.5301
tf centroid	56.84%	49.28%	0.9529
tf-idf centroid	54.74%	46.58%	0.9693
tf centroid while splitting the text for the agent and the customer	55.32%	48.08%	0.9543
tf-idf centroid while splitting the text for the agent and the customer	48.94%	41.02%	0.9882

4.1.3 BiRNN

In practice, RNN classifiers came with some difficulties. The slow speed with which they were trained compared to the logistic regression models limited the range of hyperparameter tuning. In addition, due to several network-related difficulties (non-stable internet connection, non-stable connection to the Google Cloud Platform), the hyperparameter tuning process was divided into parts and in the end the best scoring hyperparameter combination was chosen.
The RNN classifiers were modeled using Tensorflow. That means that the metrics from sci-kit learn library could be applied only after the trainining of the RNN finished and not at the end of each epoch. In addition, it was not a trivial task to model F1-weighted metric in Tensorflow and apply it at the end of each epoch. That is the reason the evaluation metrics that were available to monitor the early stopping process of the networks were only the accuracy and the log-loss metrics. Both metrics were tested. The use of the accuracy metric for monitoring the early stopping process of the networks gave better scores on the development set than the ones resulted when using the log-loss metric. That was the reason the accuracy metric was chosen as the monitoring metric for the early stopping process. Also, the hyperparameters were selected according to the highest accuracy scoring model on the development set.

For dialogue RNN models, the dialogues were tokenized and padded to sequences of length 400. That means that only the first 400 words of each dialogue example were used. The number 400 was selected since most of the dialogue consisted of fewer than 400 words (Figure 2). Only 46 dialogue turns had more than 400 words.
The best scoring model on the development set had a LSTM layer with 75 neurons, 2 pairs of (Dropout + Dense) layers, with dropout rate 0.1 and 75 neurons. The activation function of Dense layers was tanh. No dropout was applied before LSTM layer. The optimizer was Adam with 0.0001 learning rate. The model was left to train for 100 epochs, while applying early stopping with patience 15 epochs.
The test scores of this model were:

- Accuracy : 71.43%
- F1 score : 71.43%
- Log-loss : 0.9480

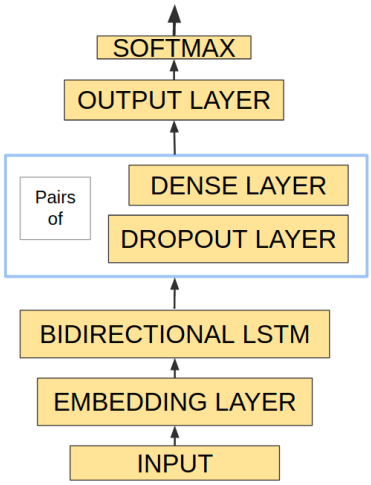


Figure 13: Architecture of the RNNs regarding dialogue classification problem.

A crucial note is that the use of GPU to train the RNNs made them non-reproducible[3]. If the training process was repeated for the same RNN the outputs were not always the same.

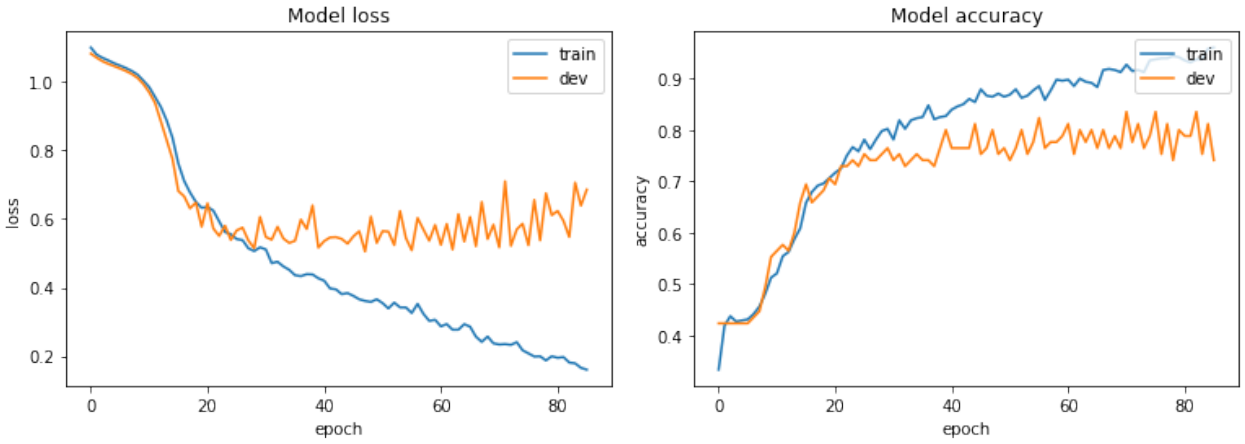


Figure 14: Training curves of Accuracy (right) and Log-loss (left) of the highest scoring BiRNN model for dialogue classification.

[3]See "A Workaround for Non-Determinism in TensorFlow" (https://www.twosigma.com/insights/article/a-workaround-for-non-determinism-in-tensorflow/).

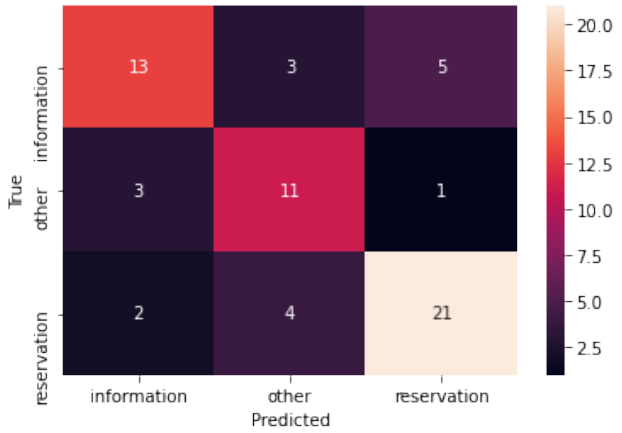


Figure 15: Heatmap of BiRNN model for dialogue classification.

4.1.4 Self-Attention RNN

As in the BiRNN (see Chapter 4.1.3) the hyperparameters of this model were selected according to the highest accuracy scoring model on the development set.
The best scoring model on the development set had a LSTM layer with 75 neurons, 2 pairs of (Dropout + Dense) layers, with dropout rate 0.2 and 75 neurons. The activation function of Dense layers was tanh. No dropout was applied before LSTM layer. The optimizer was Adam with 0.001 learning rate. The model was left to train for 100 epochs, while applying early stopping with patience 15 epochs.
The test scores of this model were:

- Accuracy : 71.43%
- F1 score : 71.71%
- Log-loss : 1.0314

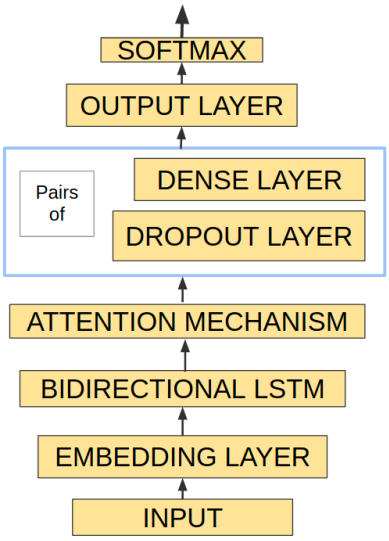


Figure 16: Architecture of the self attention RNN regarding dialogue classification problem.

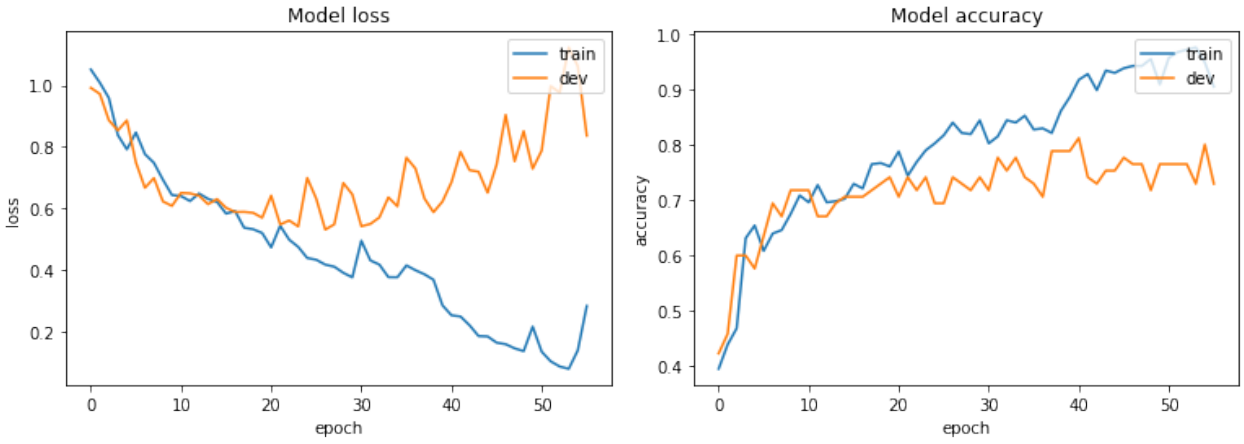


Figure 17: Training curves of Accuracy (right) and Log-loss (left) of the highest scoring self attention RNN model for dialogue classification.

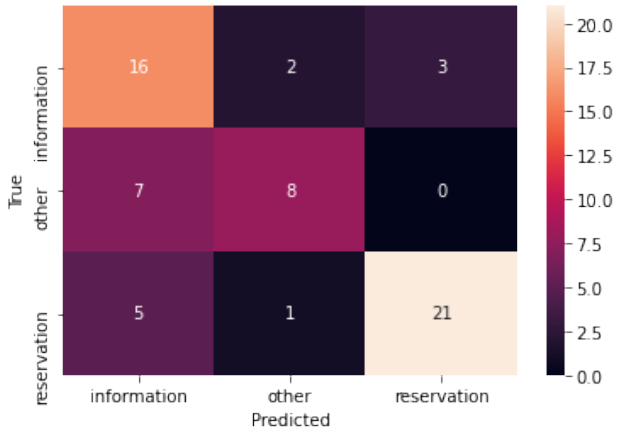


Figure 18: Heatmap of self attention RNN model for dialogue classification.

4.1.5 BERT Model

BERT was modeled in PyTorch[4]. In PyTorch, the difficulties with metrics were more than the ones in Tensorflow. In PyTorch, neither F1 nor balanced accuracy metrics were easy to be implemented while the model was trained. That is why, as with Tensorflow, only the accuracy and the log-loss metrics were used in the training process and the rest of the metrics were calculated after the training process had finished. Another difficulty was that implementing early stopping function in PyTorch was not something trivial. After many tests and experiments in the training process, it was found that when the development log-loss score began to substantially rise, the model has reached its limits and after that epoch the scores, most of the times, worsen. However, it was observed that sometimes a small increase on the development log-loss may be combined with an increase at the

[4]The implementation of the BERT models in Pytorch was made by using parts of Venelin Valkov's publicly available code https://www.curiousily.com/posts/sentiment-analysis-with-bert-and-hugging-face-using-pytorch-and-python/.

accuracy score. So, it was decided that the epoch in which training stopped was the one that the increase of the development log-loss appeared and not the previous one. All the above are summed up to the following monitoring process:

> Let the model train till the epoch where the development log-loss score shows a 'small' increase. If this epoch is met then stop the training.

The measure of how 'small' the increase of the development log-loss score should be, after some manual tuning and some tests, was set to be 1% higher than the development log-loss score of the previous epoch. It should be mentioned that in the whole process mentioned above the test score was not used.

However, the most serious problem with the BERT models was the fact that not only its scores were non-reproducible, they also displayed high variance. Running the exactly same model (all seeds set) 2 consecutive times may give output scores that differ even 6%-7% at accuracy score. For this reason techniques described by Mosbach et al. (2020) [12] were tested. Using such techniques the scores showed less variance but were quite lower. That is why such techniques were not applied at the final BERT model.

The split of the dataset was 75% for train, 15% for development and 10% for test. With this split is was attempted to give as many examples for training as possible and at the same time leave a substantial number of examples to the development set so that the hyperparameter tuning could be as general as possible.

The dialogues were tokenized and padded to 400 sequence vectors.

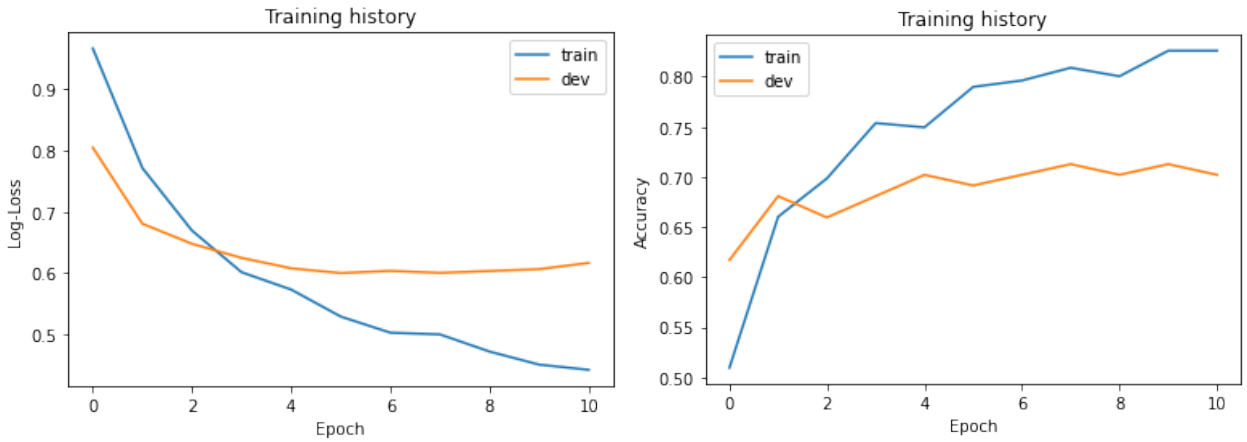


Figure 19: Training curves of Accuracy (right) and Log-loss (left) of the highest scoring BERT model for dialogue classification.

Finally, the model with the highest accuracy score on the development set was selected. This model consisted of the Greek BERT and on top of that a Dropout layer with 0.3 dropout rate. Then the output layer followed, a dense layer with 3 neurons and softmax activation function. The learning rate was tuned to 1e-06. The option where all the weights of the Greek-BERT model were trained was the one that scored higher.

The test scores of this model were:

- Accuracy : 82.54%
- F1 score : 79.46%
- Log-loss : 0.4788

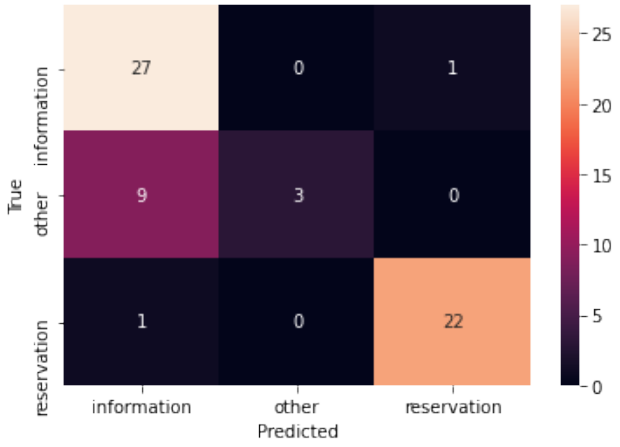


Figure 20: Heatmap of the BERT model for the dialogue classification problem.

The performance of the BERT model may be not as high as it was expected due to the fact that BERT is a language representation model. It learns to understand the language over which it was trained, to recognise the context of the examples it meets. That creates a small problem since the 'other' and the 'information' classes are semantically close to each other. Moreover, the BERT model is a model with many more parameters than the previous classifiers. Fine-tuning on over a small dataset may not be enough in order for its weights to be calibrated to the necessary extent for the BERT model to become able to fully distinguish these 2 classes.

4.2 Dialogue Turn Classification

The classification process of dialogue turns was more complicated. Due to the imbalanced dataset the prediction of all 16 classes at least once was not always possible. Methods that included total removal of the two 'other' classes or splitting the dataset and making the classification separately for the customer and agent classes or making the classification with the data at its raw form, were tested. At the end none of them was considered consistent enough. The existence of the 'other' classes helped to classify any given new turn, even if the initial label was 'None'. Furthermore, two other techniques were implemented in order to enhance the performance of the classifier.
Firstly, due to the vast amount of examples of the two 'other' classes and the 'AGENT DEFINE DATETIME' class, undersampling was applied to these 3 categories. That means, at the training dataset only a fraction of these classes was selected for training so the classifier did not train, and subsequently fit, too much on these categories. The number of examples of the rest of the classes were left as they were. It is important to note that undersampling was not applied to the development and the test sets. The undersampling process introduced 2 additional hyperparameters to the problem, the fraction of the 'other' classes and the fraction of the 'AGENT DEFINE DATETIME' class that will be used to train the classifier.
Secondly, since all classes referred to either the agent or the customer, it was decided that misclassifications between agent and customer categories could be overlooked. That tactic was mainly based on the fact that many of the mispredictions occurred due to the existence of the same intent class for both the customer and the agent (e.g. 'AGENT DEFINE DATETIME' and 'CUSTOMER DEFINE DATETIME'). That means that after the classification process was done, it was possible to check the speaker of each dialogue turn of the test set and classify each dialogue turn to the class the both matched the actual speaker and at the same time held the highest probability as predicted from the classifier. For the sake of brevity, the mapping mentioned above will be referred as 'Speaker Mapping'.

4.2.1 Logistic Regression with tf-idf Features

The hyperparameters for this classifier were chosen according to the highest scoring balanced accuracy model on the development dataset. The undersampling process tuned to 4% of the 'other' classes and 40% of the 'AGENT DEFINE DATETIME' class. The vectorizer of this model looked both unigrams and bigrams, used the 200 most frequent features, words that appeared in less than 30% and more than 1% of the training examples were not included and sublinear tf scaling was applied. For the logistic regression classifier, with L2 penalty function and regularization strength set to 1, scores are shown in Table 3.

Table 3: Logistic regression scores regarding dialogue turn classification, with use of tf-idf features

Metrics	Initial Score	After 'Speaker Mapping'
Balanced Accuracy	64.48%	72.95%
Accuracy	52.02%	72.51%
F1 Score	53.41%	76.06%

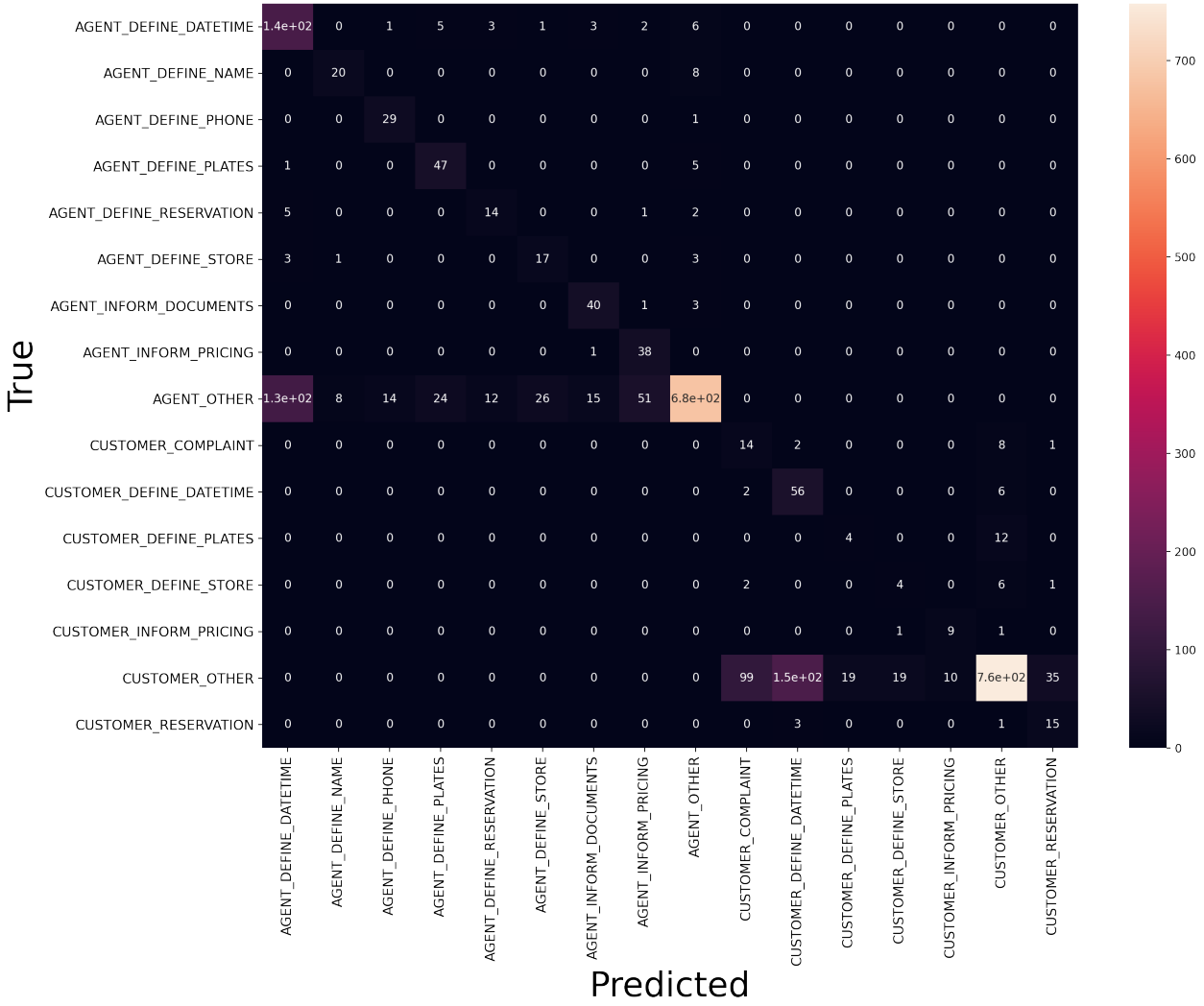


Figure 21: Heatmap of logistic regression predictions regarding dialogue turn classification, using tf-idf features, after 'Speaker Mapping' was applied.

4.2.2 Logistic Regression with Word Embeddings

As in Chapter 4.2.1, the metric according to which the hyperparameters were selected was the balanced accuracy metric. The method that scored higher balanced accuracy was the one using tf-idf centroid vectors. The fractions for the undersampling process were tuned to 4% for the 'other' classes and to 40% for the 'AGENT DEFINE DATETIME' class. For the logistic regression classifier, with L2 penalty function and regularization strength set to 1, scores are shown in Table 4.

Table 4: Logistic regression scores regarding dialogue turn classification, with use of tf-idf centroid approach.

Metrics	Initial Score	After 'Speaker Mapping'
Balanced Accuracy	64.95%	75.86%
Accuracy	46.03%	62.11%
F1 Score	48.54%	67.02%

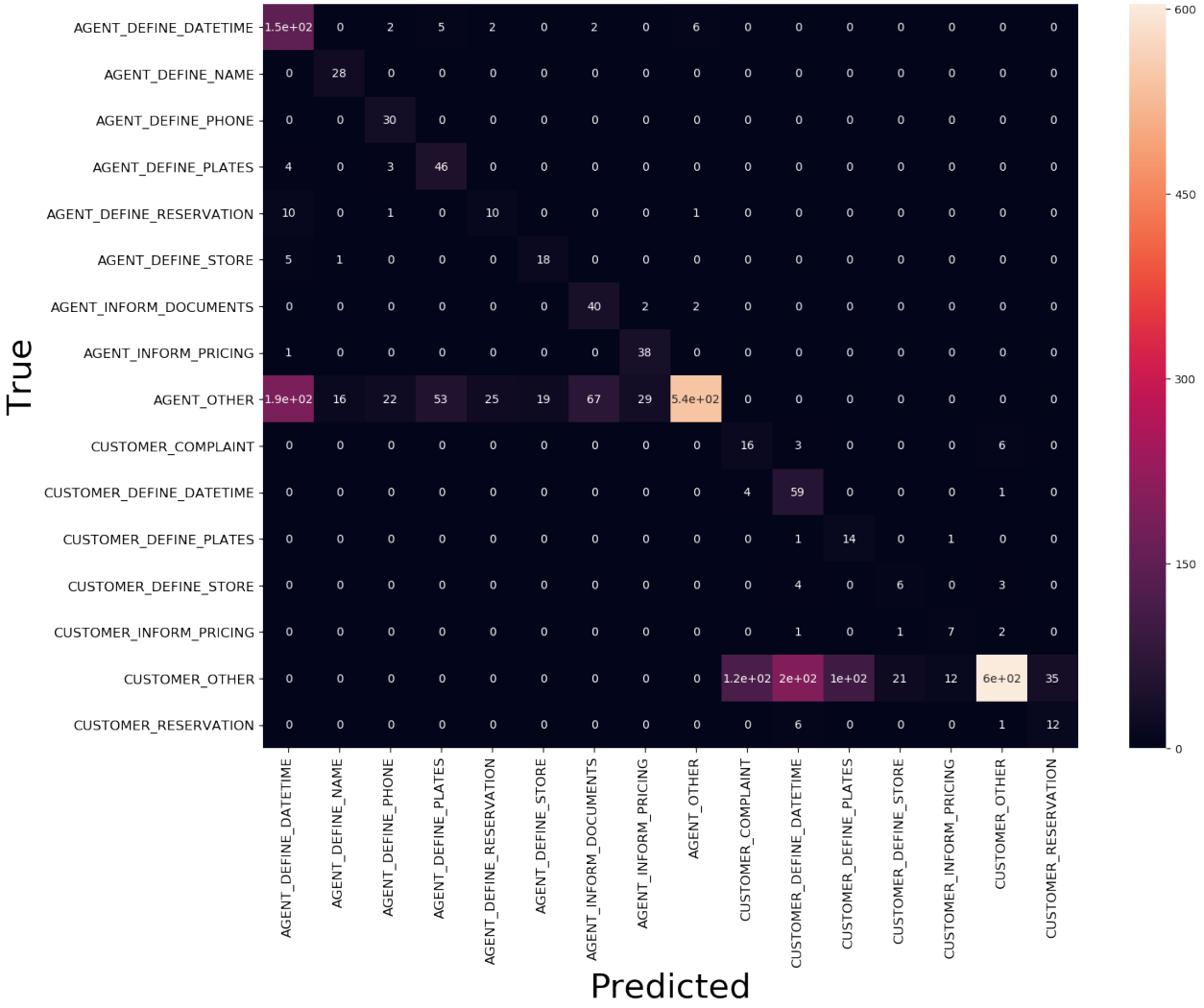


Figure 22: Heatmap of logistic regression predictions regarding dialogue turn classification, using tf-idf centroid vectors of word embeddings, after 'Speaker Mapping' was applied.

Table 5: Scores of all logistic regression experiments regarding dialogue turn classification (after 'Speaker Mapping' applied).

Logistic Regression	Balanced Accuracy	Accuracy	F1
tf-idf features	72.95%	**72.51%**	**70.06%**
tf centroid	68.67%	60.19%	65.42%
tf-idf centroid	**75.86%**	62.11%	67.02%

4.2.3 BiRNN

For dialogue turn RNN models, the turns were tokenized and padded to sequences of length 20. That means that only the first 20 words of each turn were used. The number 20 was selected since most of the dialogue turns consisted of less than 20 words (Figure 3). Only 874 dialogue turns had more than 20 words.
In order to include as many hyperparameters as possible in the hyperparameter tune process the undersampling parameters were fixed to the values that the best logistic regression classifiers had (4% for the 'other' classes and 40% for the class 'AGENT DEFINE DATETIME'). Dealing with the same difficulties as mentioned in 4.1.3, the best scoring model was selected according to the development accuracy score.

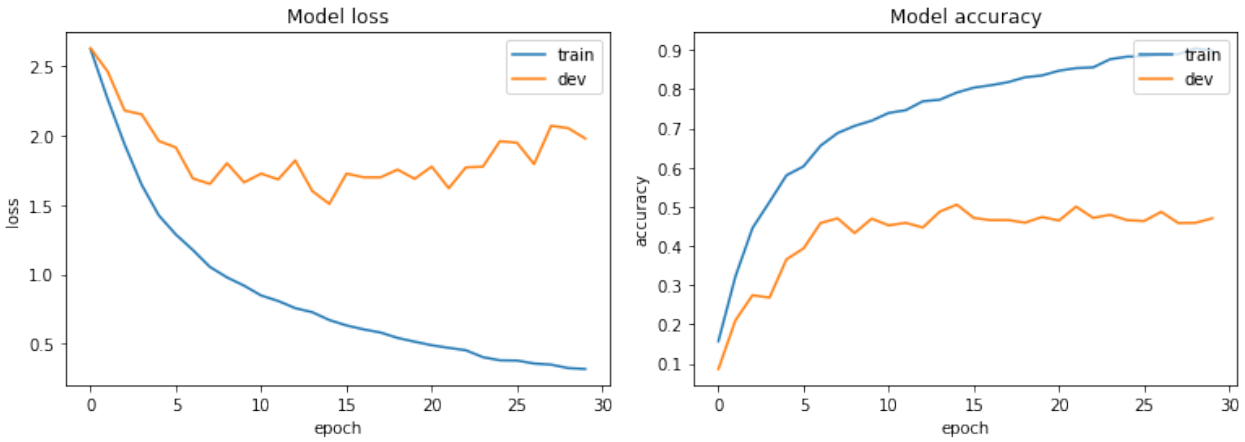


Figure 23: Training curves of Accuracy (right) and Log-loss (left) of the highest scoring BiRNN model for dialogue turn classification.

The best scoring model on the development set had a LSTM layer with 75 neurons, 2 pairs of (Dropout + Dense) layers, with dropout rate 0.1 and 75 neurons. The activation function of Dense layers was tanh. No dropout was applied before the LSTM layer. The optimizer was Adam with 0.001 learning rate. The model was left to train for 100 epochs, while applying early stopping with patience 15 epochs. For the BiRNN model, scores are shown in Table 6.

Table 6: BiRNN model scores regarding dialogue turn classification.

Metrics	Initial Score	After 'Speaker Mapping'
Balanced Accuracy	59.60%	68.30%
Accuracy	50.78%	68.28%
F1 Score	53.55%	73.07%

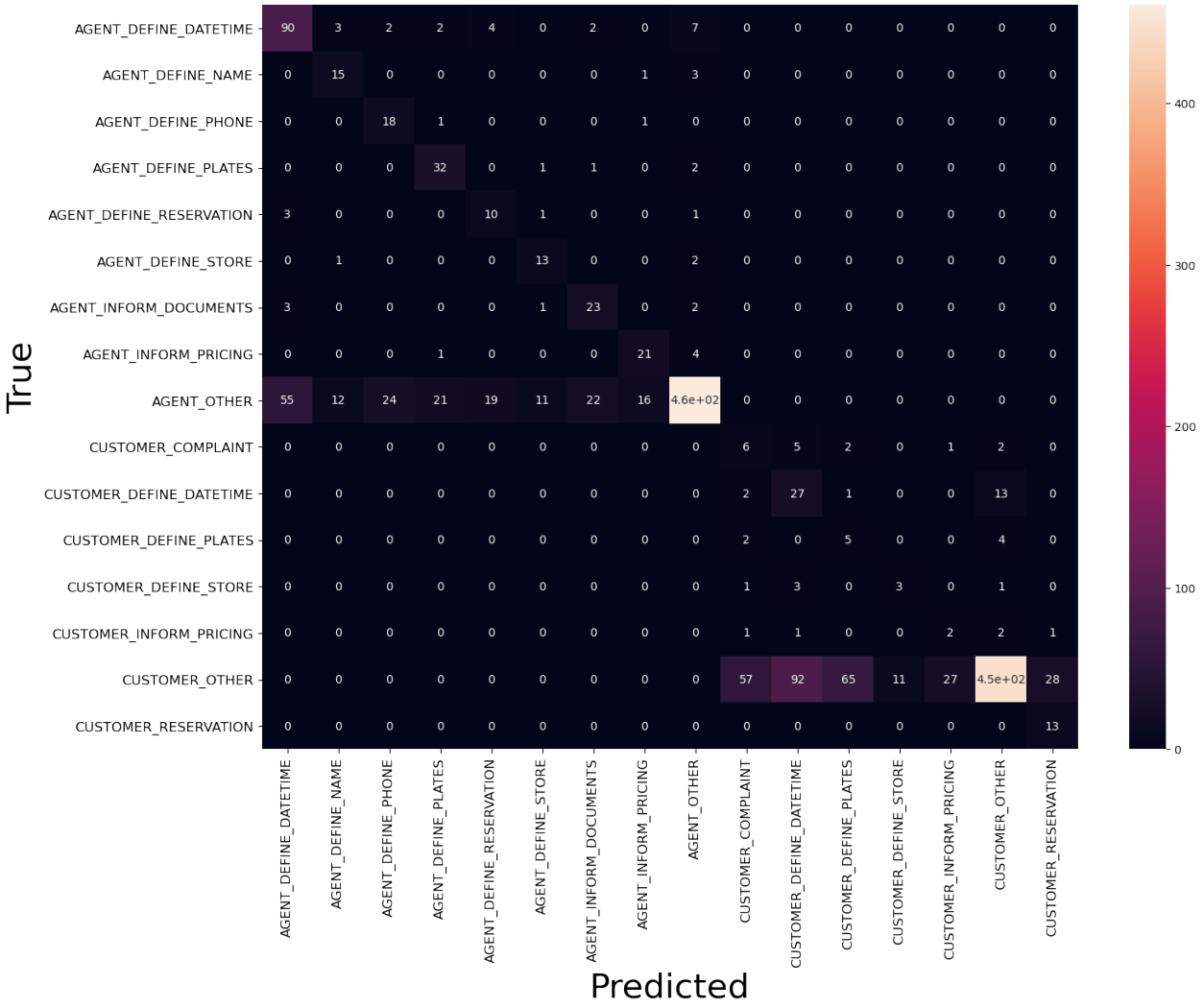


Figure 24: Heatmap of BiRNN model regarding dialogue turn classification, after 'Speaker Mapping' was applied.

4.2.4 Self-Attention RNN

The best scoring self attention RNN model for dialogue turn classification regarding the development accuracy score had a LSTM layer with 75 neurons, 2 pairs of (Dropout + Dense) layers, with dropout rate 0.1 and 75 neurons. The activation function of Dense layers was tanh. No dropout was applied before LSTM layer. The optimizer was Adam with 0.001 learning rate. The model was left to train for 100 epochs, while applying early stopping with patience 15 epochs.

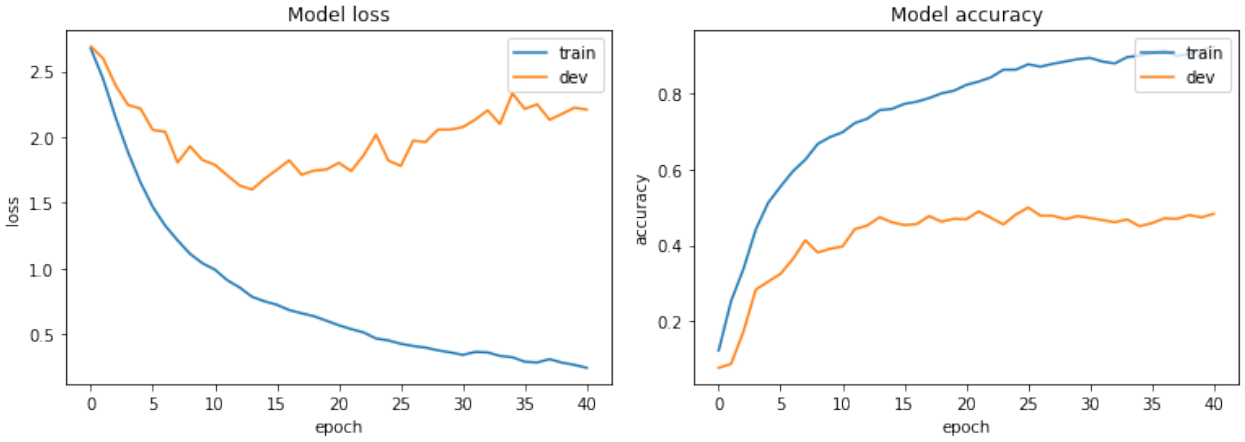


Figure 25: Training curves of Accuracy (right) and Log-loss (left) of the highest scoring self attention RNN model regarding dialogue turn classification.

For the self attention RNN model, scores are shown in Table 7.

Table 7: Self attention RNN model scores regarding dialogue turn classification.

Metrics	Initial Score	After 'Speaker Mapping'
Balanced Accuracy	60.28%	71.06%
Accuracy	48.93%	65.05%
F1 Score	51.15%	70.34%

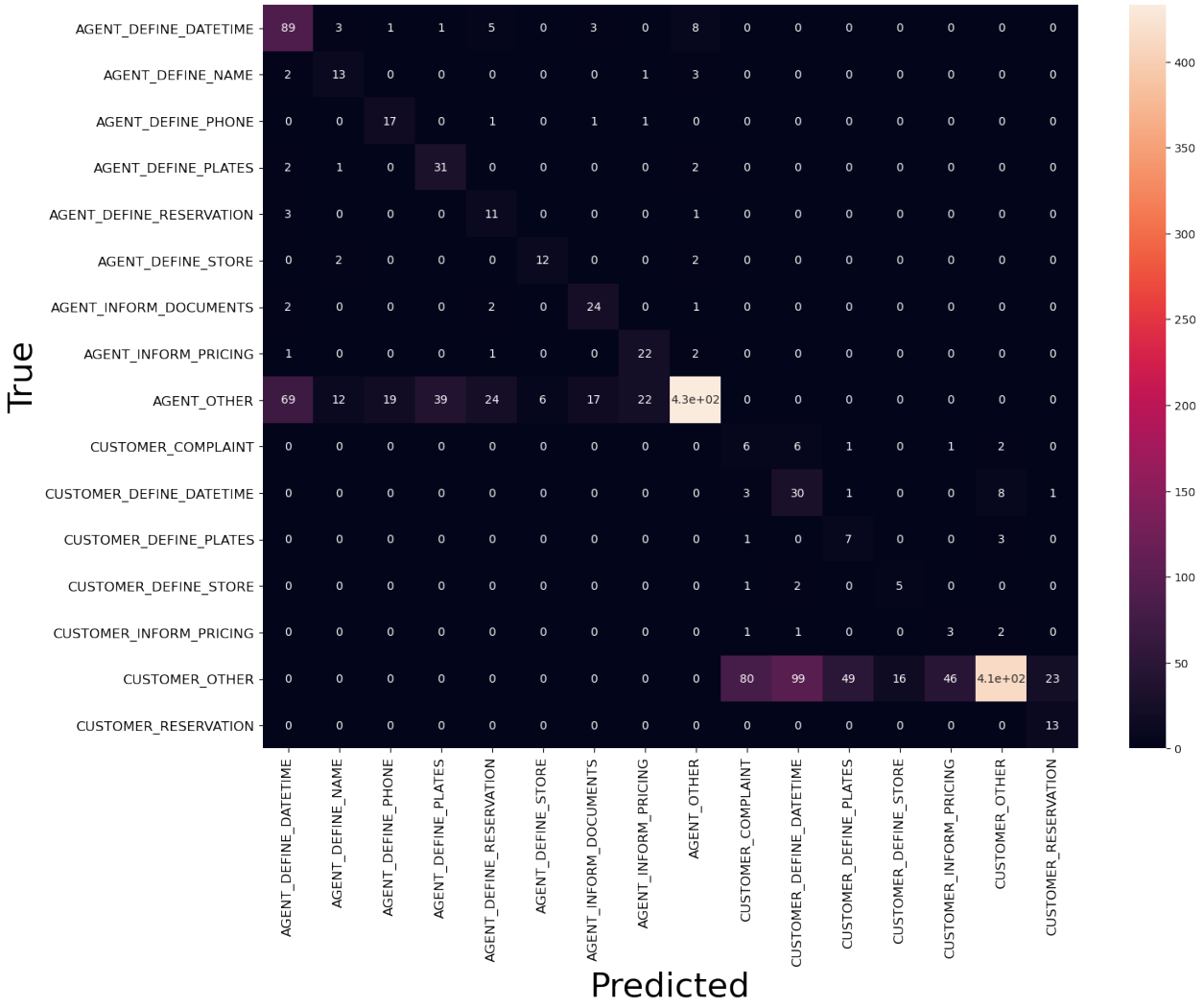


Figure 26: Heatmap of self attention BiRNN model regarding dialogue turn classification, after 'Speaker Mapping' was applied.

4.2.5 BERT Model

The same problems as those for the BERT model in dialogue classification (see Chapter 4.1.5) were met here too. However, in this classification problem the technidues described by Mosbach et al. (2020) [12] gave better results. So, the decay weight was set to 0.01 for the Adam optimizer. Furthermore, warm up for the learning rate was used for the first 7 epochs in order to prevent divergence early in training[5]. Moreover, bias correction was applied. Bias correction has a similar effect to the warm up for the learning rate by reducing the original step size of the optimizer as:

[5]Warm up for the learning rate is the phase in the beginning of the training process of the classifier where learning rate is much smaller than the set value of learning rate and then increases linearly over a few epochs until it reaches the set value of learning rate.

$$\alpha \rightarrow \alpha \cdot \frac{\sqrt{1-\beta_2^t}}{(1-\beta_1^t)}$$

where α is the step size of the optimizer, β_1 and β_2 are the beta parameters of the Adam optimizer and t is the number of iterations that have taken place.
The dataset was split to 75% for train set, 15% for development and 10% for test. The undersampling hyperparameters were tuned to 0.05% for the 'other classes and to 40% for the 'AGENT DEFINE DATETIME' class. Due to the imbalanced dataset the development log-loss showed higher variations than those of the BERT model for dialogue classification. For this problem, in order for the training to stop, the development log-loss of an epoch should rise more than 20% regarding the previous epoch.

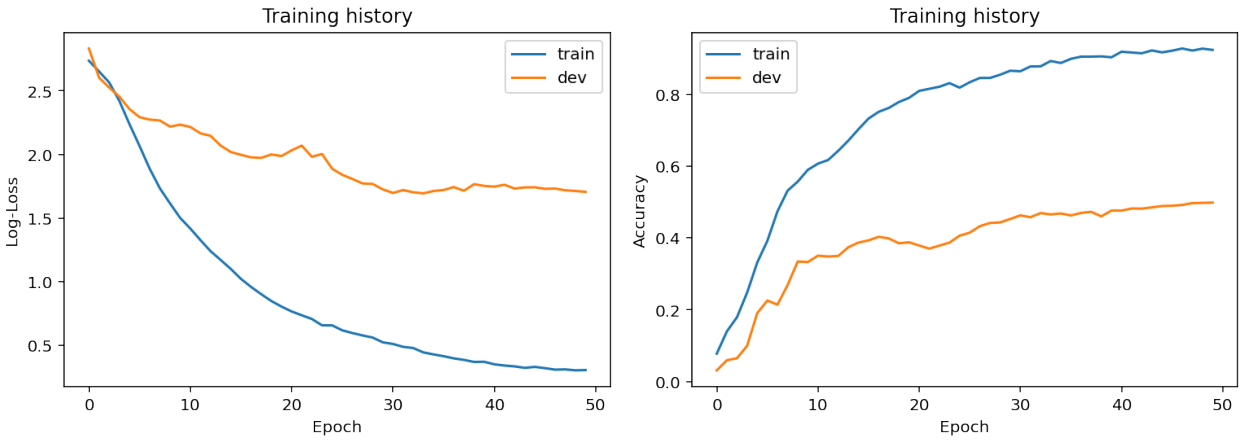


Figure 27: Training curves of Accuracy (right) and Log-loss (left) of the highest scoring BERT model regarding dialogue turn classification.

Finally, the model with the highest accuracy score on the development set was selected. This model consisted of the Greek-BERT and on top of that a Dropout layer with 0.1 dropout rate. Then a Dense layer with 500 neurons followed, right before the output Dense layer (with 16 neurons and softmax activation function). The learning rate was tuned to 1e-05. The option when all the weights of the Greek-BERT model were trained was the one that gave better results on the development set. The model was left to train for 50 epochs. The scores of this model are shown in Table 8.

Table 8: BERT model scores regarding dialogue turn classification.

Metrics	Initial Score	After 'Speaker Mapping'
Balanced Accuracy	77.97%	86.61%
Accuracy	49.68%	66.15%
F1 Score	52.61%	70.97%

The BERT model is the highest scoring classifier for dialogue turn problem regarding the balanced accuracy metric. In Figure 28, the distribution of probability that the BERT model outputs for the turn 'για την καρτούλα έρχεστε χωρίσ ραντεβού έχετε άμεση εξυπηρέτηση και είναι 12 ευρώ το κόστοσ΄ ('for the card you come without an appointment you have immediate service and is 12 euros the cost'). The true label of this turn is 'AGENT DEFINE RESERVATION' and it is obvious that BERT classifies the example right, even without 'Speaker Mapping' process.

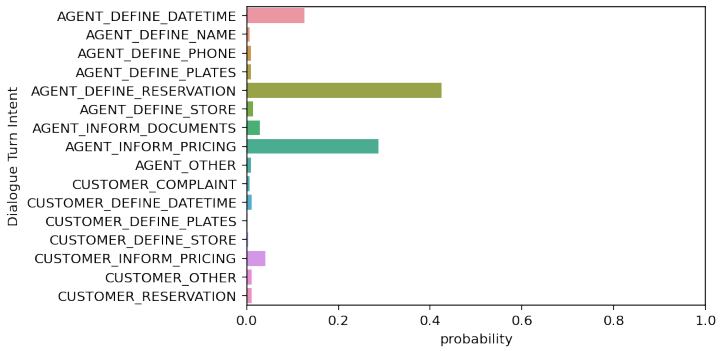


Figure 28: Example of the probability distribution of the BERT model for the dialogue turn classification problem.

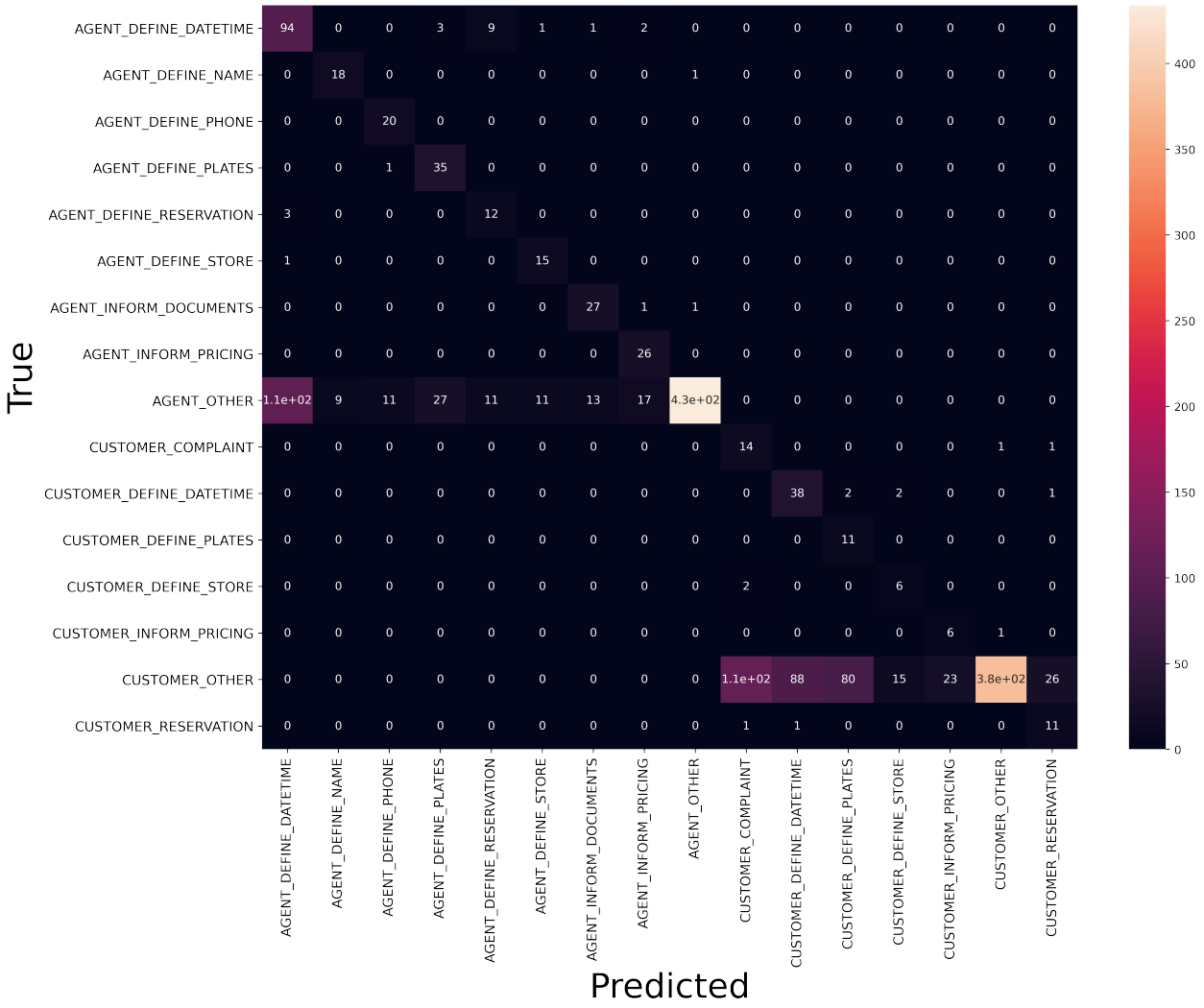


Figure 29: Heatmap of the BERT model regarding dialogue turn classification, after 'Speaker Mapping' was applied.

Observing all heatmaps for dialogue turn classification, especially the one of the BERT model, it was obvious that the majority of misclassifications regarded the two 'other' classes. In order to know the performance of the classifier in relation to the rest of the classes, the following was done: Using the same predictions used for calculating the scores in Table 8, all examples of the test set that regarded 'other' classes were ignored and then 'Speaker Mapping' was applied.
The scores of this approach are shown in Table 9 (after 'Speaker Mapping' was applied).

Table 9: BERT model scores regarding dialogue turn classification, with and without examples of the 'other' classes.

Metrics	Scores (including the 'other' classes examples)	Scores (ignoring other classes examples)
Balanced Accuracy	86.61%	91.23%
Accuracy	66.15%	91.06%
F1 Score	70.97%	92.23%

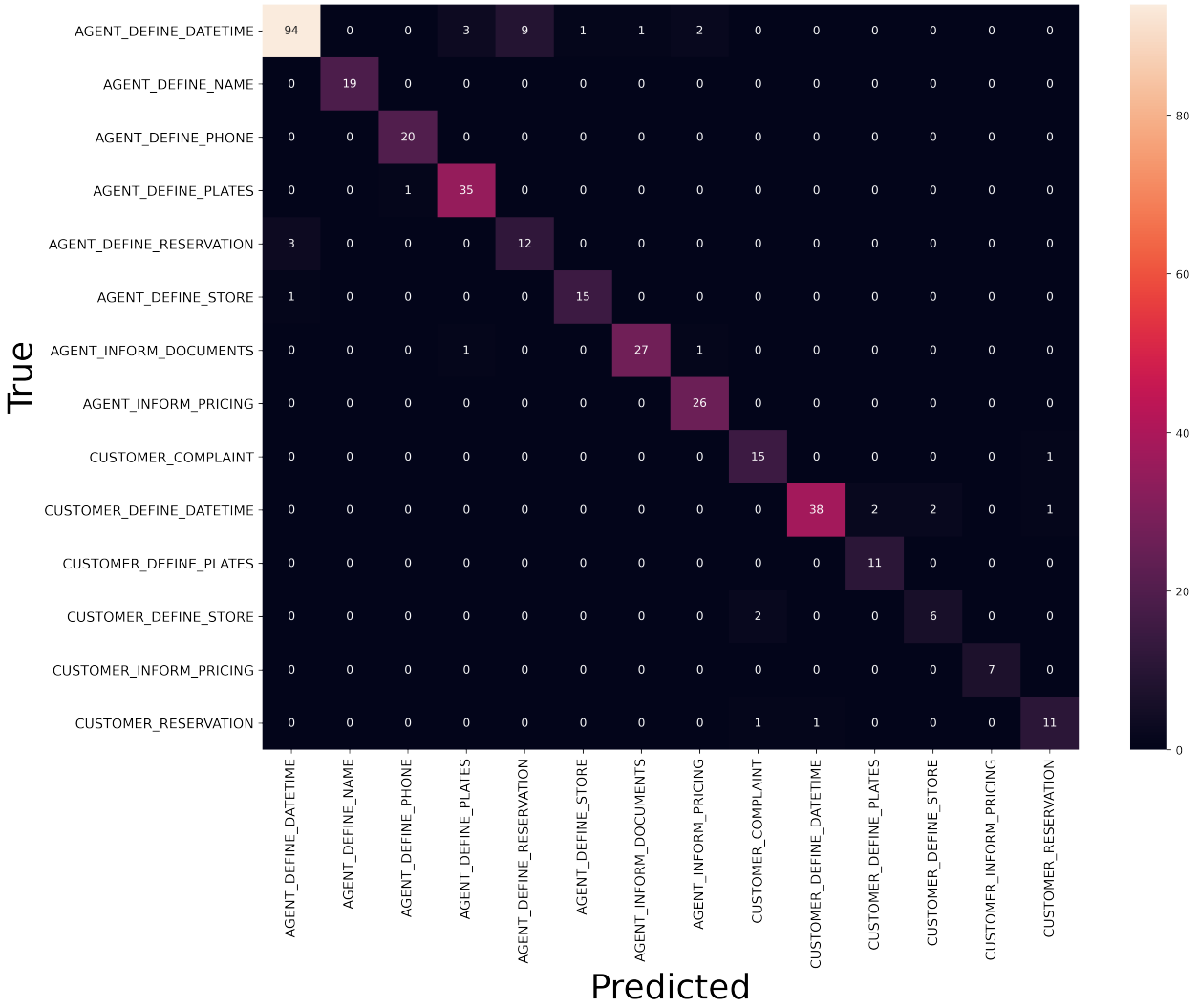


Figure 30: Heatmap of the BERT model regarding dialogue turn classification when the examples of the 'other' classes are ignored and 'Speaker Mapping' was applied.

It is clear that the scores of the BERT model classifier increased substantially. The almost diagonal shape of the heatmap in Figure 30 validates the excellent performance of the BERT model. These high scores are quite promising, yielding the expectation that the 'minorities' classes (the classes that were clustered with the 'None' class to form the two 'other' classes) would also be correctly predicted when more data about them are collected. When this time comes, the 'other' classes would not be a mixture of classes with different intents, a state that may help to further disentangle the 'other' with the rest of the classes.

5 Conclusions and Future Work

5.1 Dialogue Classification

Table 10 summarizes the results of the all classifiers for the dialogue classification problem are summarised. These results are based on the test subset. However, the test set is not the same among the classifiers since different splits of the dataset were applied. So, these results may not be strictly comparable.

Table 10: Results of all classifiers for dialogue classification problem

Classifier	Accuracy	F1 Score	Log-Loss
Logistic regression with 'variable vectors'	53.68%	46.57%	0.9334
Logistic regression with tf-idf features	82.11%	**81.88%**	0.5279
Logistic regression with word embeddings	56.84%	49.28%	0.9529
BiRNN	71.43%	71.43%	0.9480
Self-attention RNN	71.43%	71.71%	1.0314
BERT	**82.54%**	79.46%	**0.4788**

The BERT model is the one that achieved best scores in accuracy and log-loss while the logistic regression classifier with use of tf-idf features achieved the highest F1 score. However, it is important to keep in mind that the BERT model was tuned over a limited set of hyperparameters. On the other hand, the logistic regression classifier was tuned over a wide range of all its main hyperparameters. That means that the logistic regression classifier is almost at its limits when the BERT model has possible margins for improvement.
The main reason why the results are not close to 100% (and probably would never reached such levels) is that the categories 'other' and 'information' are semantically close. Even for the human annotator, the boundaries of these categories were not always clear. This claim is also supported by Figures 10, 18, 20, the heatmaps of the highest scoring classifiers for this problem. In these Figures the confusion between these two classes is clear..

5.2 Dialogue Turn Classification

Table 11 summarizes the results (after applying 'Speaker Mapping') of the all classifiers for the dialogue turn classification problem are summarised. These results are based on the test subset. However, the test set is not the same among the classifiers since different splits of the dataset were applied. So, these results may not be strictly comparable.

Table 11: Results of all classifiers for dialogue turn classification problem (applying 'Speaker Mapping')

Classifier	Balanced Accuracy	Accuracy	F1 Score
Logistic regression with tf-idf features	72.95%	**72.51%**	**76.06%**
Logistic regression with word embeddings	75.86%	62.11%	67.02%
BiRNN	68.30%	68.28%	73.07%
Self-attention RNN	71.06%	65.05%	70.34%
BERT	**86.61%**	66.15%	70.97%

The BERT model is the one that achieved best score regarding balanced accuracy while the logistic regression classifier with use of tf-idf features achieved the highest accuracy and F1 score. As in dialogue classification, the logistic regression classifier is a classifier that was well tuned, making difficult any further improvement at its scores. However, the BERT model showed promising results, when many of its hyperparameters were manually set, like warm up and decay rate of learning rate. Having more time and better resources, would make possible a wider hyperparameter tune process to run and would probably give even better results.

5.3 Future Work

The next steps that might lead to improvement of the scores of this project's classifiers are to:

- Include in the hyparameter tuning of RNN models the recurrent dropout parameter. At this project, this parameter was set to its default value, which was zero. Some trials included that the value of 0.1 recurrent dropout showed promising results. However this approach was not implemented here because even the slightest recurrent dropout rate that was applied slowed significantly the training process. Given better resources this change is expected to lead to better scores for the RNN models. Stacking more than one bidirectional LSTM layers may also improve the results.
- Pre-training Greek-BERT on Greek dialogues.
- For dialogue classification, combine (e.g., by using a linear layer) the output of the Greek-BERT model with the 'variable vectors' (Chapter 4.1.1).

Furthermore, a neural model that might take this whole work one step further is introduced by Chalkidis et al. (2018) [13]. In this paper Chalkidis proposes the structure of a hierarchical recurrent neural network that converts each sentence to a single sentence embedding by using a first-level RNN that reads the word embeddings of each sentence and produces a sentence embedding. Then for classifying multi-sentence text, it uses a second-level RNN that reads the sentence embeddings and classifies each multi-sentence text. In our case, we could classify the entire dialogue on each turn. A similar technique can be applied to combine the 2 problems of this work since a dialogue is not just a series of independent speech acts, but rather a collective act performed by the speaker and the hearer. In our case, firstly classification of each turn of a dialogue can be performed. Then, the predicted labels of the dialogue turns could be used to classify the entire dialogue.

References

[1] C.S. Wu, S. Hoi, R. Socher and C. Xiong, "TOD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogue". arXiv:2004.06871, 2020.

[2] J. Herzig, G. Feigenblat, M. Shmueli-Scheuer, D. Konopnicki, A. Rafaeli, D. Altman and D. Spivak, "Classifying Emotions in Customer Support Dialogues in Social Media". Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Los Angeles, USA, pp. 64-73, 2016.

[3] E. Grave, P. Bojanowski, P. Gupta, A. Joulin and T. Mikolov, "Learning Word Vectors for 157 Languages". Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan, 2018.

[4] A. Joulin, E. Grave, P. Bojanowski and T. Mikolov, "Bag of Tricks for Efficient Text Classification". Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, Valencia, Spain, pp. 427-431, 2016.

[5] K. Cho, B. van Merrienboer, D. Bahdanau and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, pp. 103-111, 2014.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser and I. Polosukhin, "Attention is All you Need". Advances in Neural Information Processing Systems 30, California, United States, pp. 5998-6008, 2017.

[7] J. Devlin, M.W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, Minnesota, pp. 4171-4186, 2019.

[8] J. Koutsikakis, I. Chalkidis, P. Malakasiotis and I. Androutsopoulos, "Greek-BERT: The Greeks Visiting Sesame Street". Proceedings of the 11th Hellenic Conference on Artificial Intelligence (SETN 2020), held on-line, 2020.

[9] F. Pedregosa, et al. "Scikit-learn: Machine Learning in Python". Journal of Machine Learning Research, vol 12, pp. 2825-2830, 2011.

[10] M. Abadi, et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". arXiv:1603.04467, 2016.

[11] A. Paszke, et al. "Automatic Differentiation in PyTorch". NIPS 2017 Workshop on Autodiff, California, USA, 2017.

[12] M. Mosbach, M. Andriushchenko and D. Klakow, "On the Stability of Fine-tuning BERT: Misconceptions, Explanations, and Strong Baselines". arXiv:2006.04884, 2020.

[13] I. Chalkidis, I. Androutsopoulos and A. Michos, "Obligation and Prohibition Extraction Using Hierarchical RNNs". Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Melbourne, Australia, pp. 254-259, 2019. 2018.