Bilingual text classification using transformer models and k-NN

A study of the applicability of transformer models in conjunction with k-Nearest Neighbor models on the classification of Dutch and English messages from the customer services of housing corporations.

Robert Brandemann

Master Thesis
Vrije Universiteit Amsterdam
Business Analytics
June 23, 2023

Master thesis

Business Analytics

Bilingual text classification using transformer models and k-NN

A study of the applicability of transformer models in conjunction with k-Nearest Neighbor models on the classification of Dutch and English messages from the customer services of housing corporations.

Robert Brandemann
2713542

First supervisor:
Dr. V. François-Lavet
Second supervisor:
Dr. E. N. Belitser

Supervisor:
Ing. P. F. Heek
Engineering Lead

Vrije Universiteit Amsterdam
Faculty of Science
De Boelelaan 1111
1081 HV Amsterdam

Zig Websoftware
Botterstraat 51
1271 XL Huizen

Executive Summary

This study has been commissioned by the company Zig. Zig is a provider of software for housing corporations in the Netherlands. These housing corporations tend to get large numbers of questions from their tenants. A significant part of these questions is sent in via contact forms. With these forms, the tenant is required to select a *subject*. This *subject* relates to a *category* which in turn relates to the team of employees that should resolve the question. The assignment of a message to the correct team is therefore, indirectly, the responsibility of the tenant. This is not desirable, as tenants are likely to select a 'miscellaneous' *subject* when confronted with a long list of *subjects* and are likely to make mistakes when choosing the correct *subject*. The intent of this study is to find a method to assign the correct *category* to the messages automatically, based purely on the message text in the contact form. As each customer of Zig used different *categories*, this study is limited to a single housing corporation, namely [redacted]. [redacted]. A significant part of [redacted]'s tenants consists of international s[redacted], hence roughly half of their messages are in English (the other half is Dutch). Furthermore, the *categories* are not set in stone and new *categories* can be added or existing *categories* can be modified. The final algorithm should be flexible enough that it can adapt to these *categories*. Given the recent success of large language models, the focus of this research is mainly on similar pre-trained transformer models. The problem and the challenges can be summarized in the following research question and conditions:

> How can the process of categorizing/routing incoming messages at the customer service of housing corporations be automated with the use of attention-based models, given the bilingualism of the messages and the variability of *categories*?

This primary question should lead to an algorithm that complies with the following conditions:

1. The algorithm should be able to utilize the semantic meaning of the messages to assign the right *category*.
2. The algorithm should be able to process both Dutch and English messages.
3. The algorithm should be able to adapt to new *categories* with few data, without exhaustive retraining of the models.
4. The overarching concept of the algorithm should remain easily explainable to end users with little to no mathematical knowledge.

This research was done in two phases. The first phase concerns itself primarily with the bilingualism of the messages (i.e. the second condition) by combining machine translation with different pre-trained models. In the second phase, the possibility of using k-Nearest Neighbors (k-NN) for resolving the third condition is investigated. The first and fourth conditions are satisfied by the model choice. Large pre-trained transformer models are able to utilize the semantic meaning from text. The transformer models are simple to understand in their intended purpose, as they extract a numerical representation of this meaning. The process of searching for the k most similar examples to obtain a prediction, as is the case for k-NN, is also very intuitive.

For dealing with the second condition in the first phase, four approaches have been investigated. The main question here is whether it is better to use a large multilingual model which can process many more languages than required (e.g. mBERT, which can process 104 languages) or a monolingual model (e.g. BERT for English and BERTje for Dutch) in conjunction with Machine Translation. The first approach feeds the messages into a multilingual BERT (mBERT) model (Model *A1*). The second approach uses Google Translate to translate all messages into English and uses BERT (Model *A2*). The third approach is similar to the second, but the messages are translated into Dutch and BERTje is used (Model *A3*). The final approach translates all Dutch messages into English and vice versa. The Dutch and English version for each message is then concatenated and used as input for mBERT (Model *A4*). It was found that no approach performs significantly better than the others, but Model *A3* performed marginally better on the validation set where the model predicts the right *category* in 71.2% of the cases (i.e. a validation accuracy of 0.712).

In the second phase, the best model from the first phase, Model *A3*, has been combined with a k-Nearest Neighbor algorithm (k-NN). The k-NN is, theoretically, able to adapt to new/modified *categories*. Two simple bag-of-words models have been created which only use k-NN. These use the frequency that each word occurs in a message and the Term Frequency-Inverse Document Frequency respectively as input for Model *B1* and *B2*. Model *B3* uses the document embedding, as obtained from the last transformer block of Model *A3*, as input for the k-NN. Model *B4* combines the predictions of Model *A3* and *B3*. It was found that the validation accuracy is highest when Model *B4* is for 80.5% based on *A3* and 19.5% on Model *B3* with 30 neighbors. This model is able to predict 71.6% of the *categories* in the test set correctly (test accuracy of 0.716). In comparison, a model that would predict a random *category* would lead to a accuracy of 0.183 (18.3%) and a model that only predicts the most common *category*, namely 'Financial matters', would lead to an accuracy of 0.350 (35.0%).

As Model *B4* relies heavily on *A3*, its performance on new/modified *categories* will be poor. Future studies might be able to design a formal experiment to evaluate the performance on new/modified *categories*. These might find a ratio between Model *A3* and *B3* which increases this performance.

In short, the best approach that has been found in this research is to translate all messages into Dutch. Then BERTje should be used to obtain the document embeddings and the predictions for *categories*. A 30-Nearest Neighbors algorithm should be used on document embeddings to obtain a separate set of predictions for the *categories*. A weighted average should be taken between these predictions, where the predictions from BERTje are weighted by 0.805 (80.5%) and the predictions from the k-NN by 0.195 (19.5%), to obtain the final predictions.

If this automated categorization were implemented, the *subject* field in the contact forms would not be the deciding factor for routing the messages to the teams. Hence, the *subject* field could be removed or replaced by a free-form text box. This free-form text box would have the benefit of obtaining a concise summary of the message itself. This summary could then be appended to the message and used as input for the models that are described in this report, without significant changes to the architecture. It would be reasonable to assume that the addition of this summary would improve the performance of the models.

Preface

This thesis has been written in fulfillment of the Master Business Analytics. The research for this thesis was conducted in the 6 month period from February 2023 to July 2023. The research was part of an internship at Zig (a company that creates software solutions for housing corporations).

This research concerns itself with the automatic assignment of *categories* to incoming messages at the customer services of housing corporations. Several methods of combining machine translation with pre-trained models are investigated in an effort to find the best way to work with Dutch and English data at the same time. The automation of the categorization could reduce the amount of manual labor for housing corporations and the investigation into bilingual text data could be very informative for future research at Zig. I hope that this study will be valuable to Zig and that it can be used to improve the satisfaction of Zig's customers and the tenants.

Firstly, I would like to express my gratitude to the company supervisor, Peter Frans Heek, who helped me greatly with the organization and planning of the research during our weekly meetings. Secondly, I want to thank my university supervisor, Dr. Vincent François-Lavet, who was able to provide informative answers to my questions, near immediately after I asked them. His substantive knowledge was of great help during the project. Finally, I want to thank Dr. Eduart Belitser for being the second reader of this thesis on behalf of the Vrije Universiteit.

Robert Brandemann, June 23, 2023

Contents

1 Introduction

This report has been commissioned by the company Zig. This initial section will provide the motivation and business context for this study. Section 1.2 will provide a formalized problem formulation and finally, Section 1.3 will give an overview of the structure of the whole report.

1.1 Business Context

As indicated above, this report has been commissioned by Zig. Zig is an abbreviation and stands for ***Z**ekerheid,* ***I**nnovatie* and ***G**emak* (Security/Assurance, Innovation, and Convenience). Zig creates digital solutions for Dutch housing corporations since 2001. These housing corporations mainly focus on the social rental market (monthly rent $<$ €808.06 in 2023). The objective is to increase the convenience for the employees of the corporations, the house seekers, and the tenants who use Zig's products.

The products can be grouped into two major groups based on the rental process. On one hand, there are the products that help prospective tenants with finding a house and the advertising and distribution of housing for the corporations (Housing Brokerage Platform). On the other hand are the products which deal with 'realized' tenants, these products allow for easy communication between the tenants and the housing corporation (Housing Service Platform). This research focuses on a small part of this second group.

The Housing Service Platform is able to aggregate questions from customers from a plethora of channels. The vast majority of these questions are textual (e.g. e-mails, or contact forms). A certain *category* is assigned to each of the messages. These *categories* differ for each housing corporation and could, theoretically, be changed on a whim. An employee/team of the housing corporation tends to only be able to see a small selection of relevant *categories*. An accountant would, for example, only see the messages assigned to the *category* 'Financial Matters', whereas a service desk employee would be able to see a plethora of *categories* like 'Keys', 'Report Nuisance', and 'Miscellaneous'. It is not uncommon for housing corporations to create a new *category* for specific events that lead to a lot of questions from tenants. For example, several of Zig's customers created a new *category* for COVID-19-related questions in 2021. The acquisition/building of new housing complexes could also justify the creation of a new *category* if enough questions come in. When the flow of questions for such *categories* subsides, the *category* can, and often is, disabled for future use.

When the wrong *category* is assigned to a certain message, it will likely end up at the wrong department. The erroneous assignment will therefore often lead to higher service times as the message would need to be rerouted to the right *category*. The assignment of *categories* is currently handled in three main ways. Firstly, several housing corporations have a plethora of e-mail addresses, each of which results in the assignment of a certain *category* (e.g. messages to *financial-matters@corporation.com* could get assigned automatically to the *category* 'Financial Matters'). In this first approach, the burden of correctly assigning a *category* to the message is on the tenant, who has to select the correct e-mail address. The second approach places this burden on the housing corporation employees. For example, in the case of physical letters, there is no automatic assignment and employees would have to select the correct *category* themselves. The third approach is utilized by contact forms and asks the tenant to select a *subject* from a given list of possible *subjects*. These *subjects* can then be mapped to *categories* (e.g. contact forms with the *subject* 'Service Costs' could be assigned the 'Financial Matters' *category*). In practice, the list of possible *subjects* is often a subset of all possible *categories*. Like with the first approach, this method places the burden of categorization on the tenants.

Placing the categorization burden on employees would require a significant time investment from the housing corporations. It is likely that placing the burden on the tenants instead would lead to a higher utilization of the 'Miscellaneous' *category* and more miscategorized messages in general. This would, in turn, also require a significant, but smaller, time investment from the corporations. To reduce the amount of time that needs to be spent on anything but helping tenants, one would want to automate the categorization. The practical objective of this research is to accomplish this for contact forms specifically. The ideal situation would be to phase out the predefined selection of selectable *subjects* and add the option for free-form *subjects*. This would improve the similitude of

e-mails and contact forms. This research will focus on the free-form text from contact forms. When free-form *subjects* are implemented, future research could focus on the inclusion of the *subject* in the algorithm using all e-mails and contact forms.

Given that [redacted], one of Zig's customers, has specifically requested this research, their data will be used. [redacted][1]. [redacted][2]. A significant part of [redacted]'s tenants are international s[redacted] hence a large part of the messages, to and from their service desk, is in English (as opposed to Dutch). Section 3 will expand further upon the multilinguality of the data.

In 2021 and 2022, the data-team of Zig did research into the application of bag-of-words methods and Recurrent Neural Networks in a near identical context[3]. This proprietary research indicated that large models, which utilize the semantic meaning of messages, could perform well. Given the positive correlation between the model's capacity to capture the meaning of text and the performance, it was proposed that further research should investigate the viability of pre-trained attention-based models. Hence, this research will primarily focus on the application of attention-based models.

1.2 Problem description

The problem of automatic categorization of text messages could be interpreted as a conventional supervised text classification problem with additional challenges. These can best be described in the following research question and conditions:

> How can the process of categorizing/routing incoming messages at the customer service of housing corporations be automated with the use of attention-based models, given the bilingualism of the messages and the variability of *categories*?

This primary question should lead to an algorithm that complies with the following conditions:

1. The algorithm should be able to utilize the semantic meaning of the messages to assign the right *category*.
2. The algorithm should be able to process both Dutch and English messages.
3. The algorithm should be able to adapt to new *categories* with few data, without exhaustive retraining of the models.
4. The overarching concept of the algorithm should remain easily explainable to end users with little to no mathematical knowledge.

The fourth constraint can mostly be considered as a reaction to the implications of the third constraint. Given that the problem should be treated as a supervised classification problem, there should be some correctly labeled messages for each *category*. This would require some manual labeling by the employees of the housing corporation. It seems reasonable to assume that an intuitive overarching understanding of the algorithm would motivate the employees to contribute in this way. It might also motivate the employees to correct algorithmic mistakes in the categorization, instead of resolving the message in the wrong *category*, which would benefit the quality of the data.

This research will explore the viability of applying an attention-based model in conjunction with Machine Translation and k-Nearest Neighbors in an effort to find a solution to the research question while complying with the conditions.

[1] [redacted]
[2] [redacted]
[3] The author was a part of this team and participated in this project

1.3 Report structure

This report will first provide a brief overview of the related works in Section 2. The following section on Exploratory Data Analysis, Section 3, will provide insight into the data that is utilized for this research. Section 4 on the Methodology, will describe the approaches that have been applied to the data and the reasoning behind these, in addition to the evaluation methods. The following section, Section 5, will confer the results of the applied methods, explore the performance of the methods per *category* and several common mistakes for the models will be investigated in an effort to gain more understanding of the workings of large pre-trained models. Finally, the results will be wrapped up in a concise conclusion and discussion (Section 6). This report will conclude with several suggestions for further research and a practical recommendation for implementing this research at Zig.

2 Related works

In the last few years, Large Language Models have entered the realm of public knowledge. This entry can largely be attributed to the introduction of OpenAI's chatGPT. The viability of such large attention-based models was first shown by Vaswani et al. in 2017 (Vaswani et al. 2017) and later, the research by Devlin et al. with their introduction of BERT (Devlin et al. 2018). Since these publications, the field of Natural Language Processing has been dominated by large attention-based models. It has become clear that attention-based models have the potential to surpass the performance of traditional Recurrent Neural Networks on most tasks. Devlin et al. have shown that pre-trained models, like BERT, can be very generalizable. By appending, for example, a simple classifier to the end of a model like BERT and training only this classifier on a downstream task, one is often able to get outstanding results. Fine-tuning the large pre-trained models in such a way allows one to utilize the complex understanding of a text that the model has learned while keeping the computational cost low. A logical reaction to the proven potential is the large number of variations of these models that have been introduced in the previous years (Kalyan et al. 2021). The usage of these pre-trained transformer models has become the norm for most Natural Language Processing applications, like text classification and text generation.

Most research regarding the transfer models assumes that the data is either monolingual or contains a multitude of languages. Models like BERT, XLNet, and GPT-2 are specifically created for the English language, while models like BERTje, CamemBERT, and SloBERTa are created for Dutch, French, and Slovenian respectively. Multilingual models like XLMRoBERta and mBERT are created using 100 and 104 languages respectively. One can expect a lower performance if one were to use a large multilingual model on data with a low number of languages than if one were to use a model that was trained specifically for that smaller set of languages. To utilize the full power of the model, the fine-tune data should match the pre-train data. No research could be found on a bilingual model that uses specifically uses both English and Dutch data.

Previous research has shown the viability of Machine Translation as a pre-processing step for sentiment classification (Araujo et al. 2016; Khare et al. 2018; Poncelas et al. 2020). This research indicates that the performance of the sentiment classification goes down when Machine Translation is used, but it does not make the classification unviable. Furthermore, it highlights the potential benefits of translating the data into a different language which might have a more apt vocabulary, which in turn makes the sentiment classification easier. A different field of study where Machine Translation can be utilized is that of cross-lingual learning. This field concerns itself with training models using models on a language with a large amount of labeled data such that it is applicable for a different language where labeled data is not or less available. Wan introduces a method for sentiment classification on Chinese texts while having only labeled English texts (Wan 2009). For this approach, one combines a Chinese translation of the labeled English texts and an English translation of the unlabeled Chinese texts. These are then used to train a model in a semi-supervised manner.

The problem of bilingualism in the data for this research will be tackled by using a multilingual model (mBERT), by using Machine Translation, and a monolingual model (BERT and BERTje separately) and, partially inspired by Wan (Wan 2009), by using the concatenation of a Dutch and English version of each message as the input for a multilingual model (mBERT).

Little research could be found on adjustments to pre-trained models which would allow them to adjust to new or changing *categories*. The research that has been done primarily focuses on the combination of pre-trained models and k-Nearest Neighbors (Kassner and Schütze 2020; Khandelwal et al. 2020). The existing research utilizes the predictions by a pre-trained model and the predictions from a k-NN algorithm on the 'embedding' (as obtained using the pre-trained model) of text from any relevant corpus. This corpus would generally be the train set. A weighted average is taken between the predictions of the pre-trained model and the k-NN. Note that the prediction of the pre-trained model is fixed to the number of possible *categories* in the train set. The predictions from the k-NN on the other hand are based on the *categories* that are present in the relevant corpus. If one were to include the messages with new or modified *categories* in this corpus, the combined model would be able to predict these.

A major downside of k-NN models in the field of Natural Language Modelling is the required assumption that the data is distributed in such a way that semantically similar points lay together

(i.e. that the input space is separable into neighborhoods of the output *categories*). Kassner and Schütze, and Khandelwal et al. utilize a 'document embedding' in their k-NN (Kassner and Schütze 2020; Khandelwal et al. 2020). This document embedding is extracted from a pre-trained model, but no reason is given to believe the required assumption holds. This research will try to make the assumption more credible by applying Neighborhood Component Analysis on the embeddings before the k-NN algorithm is applied. This Neighborhood Component Analysis is a commonly used supervised method to transform the input space of a k-NN algorithm such that similar points with the same labels are near each other in the transformed space while providing dimensionality reduction (Z. Wu et al. 2018; Yang et al. 2012).

3 Exploratory Data Analysis

As mentioned in Section 1.1, the available dataset comes from the housing corporation [redacted], a customer of Zig [redacted]. The dataset consists of messages from incoming contact forms. These contact forms contain information about the tenant, a *subject*, and *message*. For this project, only the *message* will be taken into account. Also described in Section 1.1 is the fact that the *subject* relates partially to *categories* and that the intent is to utilize this research project to phase out the *subject*. Hence, the *subject* of the contact forms is ignored to ensure that the created algorithms could, at least technically, continue working after the current implementation of the *subject* has been phased out.

While [redacted] has a multitude of *categories*, only nine overarching *categories* were deemed relevant to be algorithmically categorized. Table 1 shows the number of messages per *category*. One might note that two of the *categories*, namely 'General Administration' and 'General Renting', seem very generic. While these *categories* encompass a plethora of topics, it is assumed that these are distinct from the other *categories*. This assumption is validated by the results in Section 4.1.

Table 1: Absolute and relative frequency of the number of messages that appear in the dataset per *category*.

Category	**Frequency**	**Relative Frequency**
Financial matters	[redacted]	[redacted]%
General Administration	[redacted]	[redacted]%
Comment new room	[redacted]	[redacted]%
Appointment with the caretaker	[redacted]	[redacted]%
General Renting	[redacted]	[redacted]%
Keys	[redacted]	[redacted]%
Service costs	[redacted]	[redacted]%
Nuisance	[redacted]	[redacted]%
Acceptance new construction	[redacted]	[redacted]%
Total	**118,445**	

Figure 1 shows the number of messages per *category* and language. It is clear that the distribution of languages is roughly in favor of English, with a notable exception for 'General Administration'. [redacted].

Figure 1: Number of times each *category* and language appears in the dataset.

When looking at the length of the messages per *category* and language, as visualized in Figure 2, it becomes clear that there is no significant difference in the message lengths between Dutch and English. This is indeed what one would expect from two Germanic languages. Somewhat notable is the fact that messages in the *category* 'Nuisance' generally contain more characters than messages in the other *categories*. The vast majority of the messages have less than 1,000 characters. Nine of the messages contain more than 10,000 characters. These nine are valid messages, so there would be no reason to exclude these. The same holds for the very short messages, these all seem valid questions/notifications from tenants, so there is no reason to exclude these.

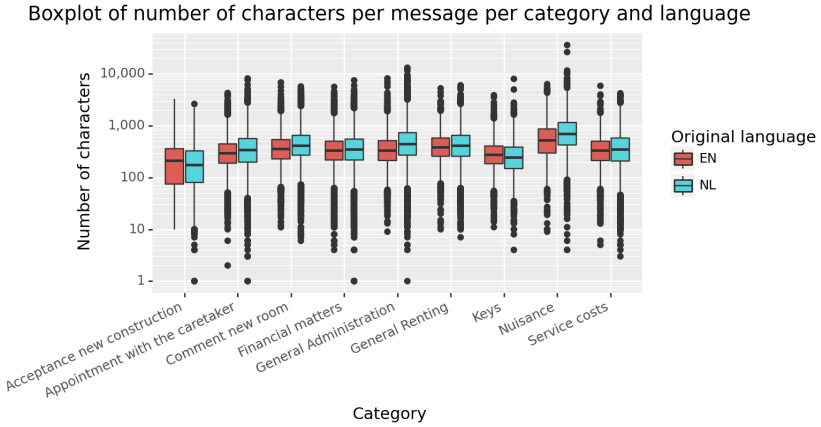
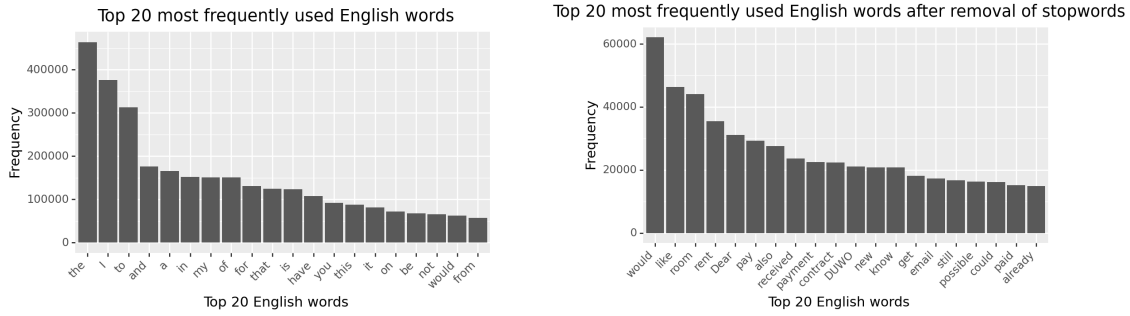


Figure 2: Boxplot of the number of characters per message for each *category* and language, using a logarithmic y-axis.

As indicated in Section 2, each Dutch message has been translated to English using Google Translate and vice versa, such that there is a Dutch and an English version of each message. When investigating the frequency that words have been used in the English versions of the messages, one might note that the distribution, as shown in Figure 3a does not quite follow Zipf's law (Powers 1998). However, it might simply be overzealous to expect the frequency of words to be inversely proportional to the rank by frequency on every text dataset. The removal of pronouns, determiners, articles, and conjunction results in a more meaningful perspective on the content of the messages. Figure 3b shows the usage distribution of the resulting set of words.



(a) Before removal of stopwords.

(b) After removal of stopwords.

Figure 3: Top 20 most frequently used words in the English messages before and after removal of stopwords.

To give an indication of how well the models will be able to adapt to new/modified *categories*, 60 messages from 5 unused *categories* will be used for an evaluation. None of these *categories* fall under the aforementioned 9 most relevant overarching *categories*. The automatic categorization of these messages is not relevant, and these messages will only be used to give an impression of the performance of the model on *categories*, on which the model has not been trained or fine-tuned. The performance is highly dependent on the quality of the *categories*. A vague *category* which overlaps

with other *categories* will be hard to identify, Whereas clearly defined *category* which encompasses a single topic and which does not overlap with other *categories* would be relatively easier. The 5 *categories* are 'Campus control', 'Block heating allowance', 'Internal relocation', 'Lease agreement', and 'Subletting'. These *categories* mostly concern a single topic and are clearly defined and should therefore give an indication of the performance of the model in near-ideal circumstances.

4 Methodology

As described in Section 1.2, the main objective of this research is to create an algorithm that is able to categorize both Dutch and English messages, utilizing a pre-trained attention-based model. This section will describe the different approaches that have been used to accomplish this, in addition to giving information on the evaluation methods used.

This study can be split into two phases. The first phase concerns itself with the first and second conditions, as described in Section 1.2. These state that the algorithm should use the semantic meaning of the messages to assign the right *category* and that the model should be able to process both Dutch and English messages. The second phase is primarily focused on the third condition, which states that the algorithm should be able to adapt to new/modified *categories* without exhaustive retraining.

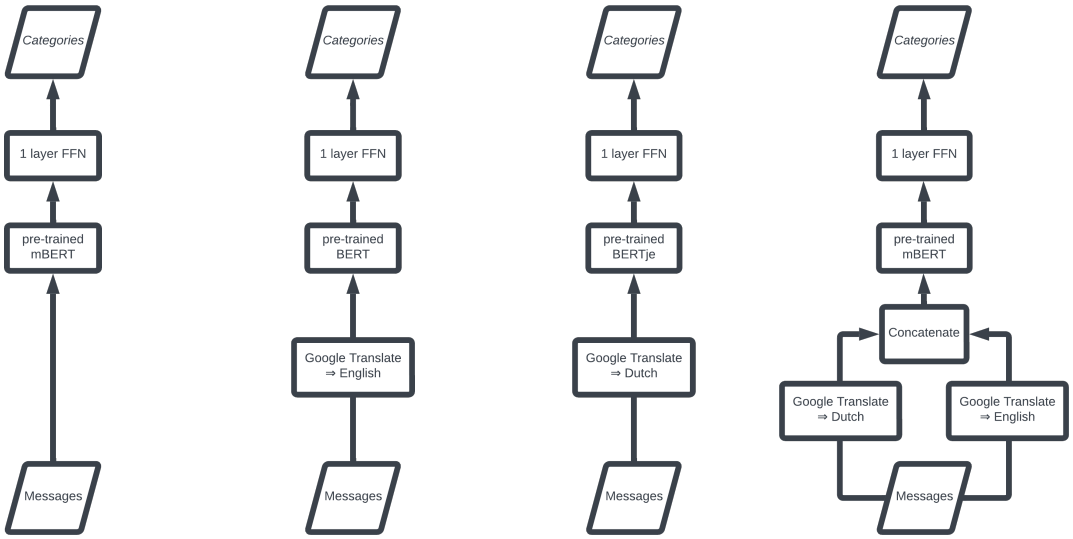
4.1 Phase A

To comply with the first two conditions, as described in Section 1.2, the algorithm should use the semantic meaning of the text to assign the right *category* and it should be able to work with both Dutch and English messages.

To comply with this first condition, large pre-trained transformer models are used. These models are pre-trained on vast quantities of data and allow one to utilize the semantic meaning of texts for a plethora of downstream tasks by fine-tuning the model on this task. These models can, in the conventional case, only be applied to fine-tune data that is similar to the pre-train data. Given that the messages for this research can be either Dutch or English, the pre-train data should, ideally, also contain Dutch and English texts. As indicated in Section 2, no pre-trained model for specifically Dutch and English could be found.

This research will specifically focus on BERT and variants of BERT. Ever since the introduction of BERT (Devlin et al. 2018), which showed the viability and potential of pre-trained attention-based transformer models, a plethora of new language models have been created. As the BERT model could be considered the progenitor of transformer models, a significant part of these models are variants of BERT. These variants generally utilize the same architecture and methodology as BERT but pre-train the model on a different dataset. Given the large amount of architecturally identical variants of BERT, it was deemed most fitting to base this research on BERT and these variants.

The base variant of BERT is pre-trained on English text. This would make it unable to process Dutch messages. Four approaches were designed to allow a variant of BERT to process both Dutch and English messages. The first approach utilized a fine-tuned multilingual BERT model (mBERT). This variant of BERT has been pre-trained on 104 languages (including Dutch and English) and is, therefore, able to categorize both Dutch and English messages. As only 2 of the 104 languages are utilized, one might suspect that this variant of BERT would perform worse than variants which are able to dedicate their entire model to a single language. The second approach uses Machine Translation to translate all messages into English. These translated messages are then classified using a regular BERT model. One could, however, reasonably assume that, given the fact that the messages concern a Dutch housing corporation and houses in Dutch cities, the semantic meaning is best captured in Dutch. Hence the third approach translates all messages into Dutch and uses BERTje, a BERT variant that was pre-trained on Dutch text. The reasoning for the second and third approaches might lead one to suspect a certain semantic value in both the English and the Dutch text, which is unique to that language. To utilize this, a new approach is introduced, inspired by (Wan 2009), where all Dutch messages are translated to English and vice versa, such that a Dutch and English variant of each message is available. The concatenation of these two variants is then used as input for a multilingual BERT model. In short, the following four approaches have been designed to manage the bilinguality of the data. These are shown schematically in Figure 4.



(a) Schematic representation of Model *A1*

(b) Schematic representation of Model *A2*

(c) Schematic representation of Model *A3*

(d) Schematic representation of Model *A4*

Figure 4: Schematical representation for all four models of the first phase.

4.1.1 Transformer Models

As indicated before, the transformer models that will be used for this research are all variants of BERT. These variants, BERT, mBERT, and BERTje, all are built using the same architecture as BERT. The differences between these models can be found in the data that has been used for pre-training and the specific pre-training tasks. The workings of BERT and the differences will be further described in the following subsections.

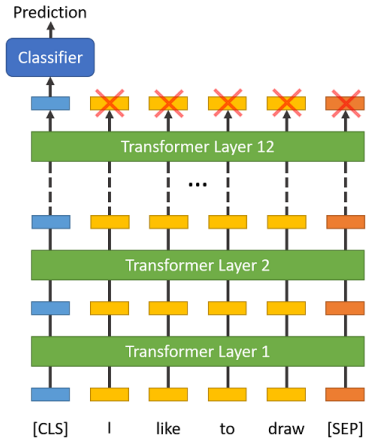


Figure 5: Schematical representation of the architecture of a BERT model with an appended multiclass perceptron.
Source: *https://mccormickml.com/2019/07/22/BERT-fine-tuning/*

4.1.1.1 BERT

Pre-training a large language model like BERT or GPT is computationally very expensive. It is practically unviable for any individual to pre-train such a model. Research has shown that the output of the pre-trained model is such that the semantics of the input texts can be linearly separated (Devlin et al. 2018). The conventional way to learn this linear separation for specific topics (or *categories*) is to append a simple multiclass perceptron to the first output of the pre-trained model (as shown in Figure 5) and fit this perceptron to the required dataset using gradient descent. As Figure 5 shows, in the case of BERT, this first output correlates to the input of a [CLS]

token. This is a special token that is added to the beginning of every message. The corresponding first output is used as the aggregate sequence representation for classification tasks (Devlin et al. 2018). This aggregate sequence representation will from here on out be referred to as the document embedding.

As described above, four approaches are investigated for parsing the semantic information from the bilingual messages. These four approaches use three variants of BERT (more specifically BERT Base). Each of these variants utilizes an identical architecture, but is pre-trained with different datasets and is pre-trained using different self-supervised tasks. The architecture consists of a stack of transformer blocks (or encoders). The BERT Base architecture uses 12 of these blocks. The contents of the transformer blocks are visualized in Figure 6. The primary working parts of the model are the multi-head attention layers. These are able to utilize the interaction between the different input tokens. The major benefits of using attention layers, as opposed to recurrent neural networks, are the fact that there are no recurrent connections and the low amount of non-linearities. This allows for better parallelization of the computations and helps prevent vanishing gradients.

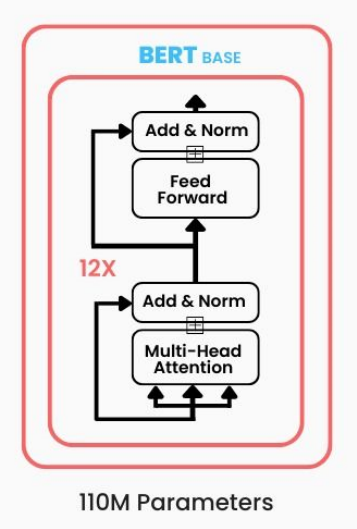


Figure 6: Schematical representation of the architecture of a BERT model with contents of the transformer blocks.
Source: *https://www.turing.com/kb/how-bert-nlp-optimization-model-works*

Multihead Attention
In these models, the input text gets split into separate tokens. In the case of BERT, these tokens are words, or sub-words if the word is not in the vocabulary of the model. These tokens get one-hot encoded and combined with a positional encoding to a single vector for each token. The matrix with these vectors is the input for the initial attention layer and will be denoted as X. Vaswani et al. (2017) describe attention functions as "mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key". In other words, the corresponding output for each input is calculated using a Query and Key-Value pair. These are calculated from the input using simple matrix multiplications:

$$Q = XW^Q$$
$$K = XW^K$$
$$V = XW^V$$

The weight matrices W^Q,W^K, and W^V will be learned using gradient descent. The attention is then calculated as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

Here, d_k stands for the dimensionality of the keys. The actual workings of the attention layer become clearer when a single output is investigated. The output vector for the i-th token will be denoted as y_i. Similarly, the corresponding query, key, and value will be denoted as q_i, k_i, and v_i respectively. The number of tokens in the input will be denoted by n.

$$z_i = \left[\frac{q_i k_1}{\sqrt{d_k}}, \frac{q_i k_2}{\sqrt{d_k}}, ..., \frac{q_i k_n}{\sqrt{d_k}} \right]$$

$$y_i = \sum_{j=1}^{n} v_j \text{softmax}(z_i)_j$$

Note that the query of the i-th token gets multiplied with every key and, after the softmax, multiplied with every value. In other words, $\text{softmax}(z_i)_j$ gives an indication of how much attention should be given to the j-th value when calculating the i-th output. One should also note that no inherent mechanisms to deal with the sequentiality of the input are present. This should therefore be represented in the aforementioned position encoding.

To expand this attention layer to a multi-head attention layer one adds additional learnable weight matrices:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O$$

$$\text{where } head_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Remark that each head now includes 3 additional weight matrices (W_i^Q, W_i^K, W_i^V), and an extra weight matrix, W^O, gets added to combine the output of the different heads optimally. The addition of multiple heads allows the model to use separate heads for multiple purposes on the same 'level'.

As stated before, in the case of BERT, 12 transformer blocks, which are made with 12-head attention layers are stacked on top of each other.

Pre-training
The primary differences between the BERT variants are found in the dataset upon which they are pre-trained and the self-supervised learning (SSL) regiments they followed as pre-training. BERT was trained using two tasks, namely Next Sentence Prediction (NSP) and Masked Language Modelling (MLM). With NSP, the model gets fed two sentences from the dataset and has to predict whether sentence B follows sentence A or not. For this task, 50% of the training examples were cases where the sentences did follow each other and 50% did not. With MLM, 15% of the tokens in a sentence get 'masked' and the model is supposed to predict which words should go in those spots. These SSL schemes were applied to the BooksCorpus (800M words) (Zhu et al. 2015) and English Wikipedia (2,500M words) (Wikimedia Foundation).

4.1.1.2 mBERT

The multilingual version of BERT (mBERT) utilizes an identical pre-training scheme with NSP and MLM. The only major difference is in the training data. For mBERT this is the collection of the top 104 languages from Wikipedia.

4.1.1.3 BERTje

Since BERT has been published, it has been found that the pre-training schemes were less than optimal. Research shows that NSP is ineffective and MLM can be too easy for some sub-words (Lan et al. 2020; Liu et al. 2019). To improve on this, NSP has been replaced by Sentence Order Prediction (SOP) where the objective is to discern the order of two consecutive sentences. Furthermore, the MLM scheme was modified so that entire words get masked instead of single sub-word. For the train data, five Dutch corpora were combined:

- Books: a collection of contemporary and historical fiction novels (4.4GB)
- TwNC (Ordelman et al. 2007): a Multifaceted Dutch News Corpus (2.4GB)
- SoNaR-500 (Oostdijk et al. 2013): a multi-genre reference corpus (2.2GB)

- Web news: all articles of 4 Dutch news websites from January 1, 2015, to October 1, 2019 (1.6GB)
- Wikipedia: the October 2019 dump (1.5GB)

4.1.2 Machine Translation

As indicated at the start of Section 4.1, all messages were translated into Dutch and English such that both a Dutch and an English version of each message is available. This translation has been accomplished using Google Translate[4]. The comparative study by Toral et al. (Toral et al. 2011) indicates that both Google Translate and Bing Translator[5] are very capable at translating Dutch to English. However, it was found that Google Translate performed marginally better in the case of translating Dutch or German to English, but significantly better when translating Italian to English. Later research shows that translating English to Dutch with Google Translate results in a BLEU score (**Bi**lingual **E**valuation **U**nderstudy) of 84 (Aiken 2019). This, roughly, indicates that the resulting Dutch translations match the pre-defined 'correct' translations for 84% word-for-word. With this score, Dutch is among the top 6 languages (out of the 50 languages this paper investigated).

The version of Google Translate that has been utilized for this research uses Google's Neural Machine Translation system (GNMT) (Y. Wu et al. 2016). This system can be considered an improved version of Neural Machine Translation (NMT) (Bahdanau et al. 2016). Major challenges of NMT like the high computational cost for training and usage, and the difficulty of dealing with rare words have been addressed by Google. The resulting model consists of a deep LSTM network with 8 encoder and 8 decoder layers using attention and residual connections. Sub-word units are utilized to improve the performance when rare words are considered and architectural adjustments have been made to improve the parallelism of the calculations, which reduces the computational costs. It is shown that the GNMT system delivers roughly a 60% reduction in translation errors on several popular language pairs when compared to Google's earlier models. It should be noted that none of these 'popular language pairs' include the translation of English to or from another Germanic language. There is, however, little reason to assume a significantly smaller reduction in translation errors can be achieved when the source and target languages are from the same language family.

4.2 Phase B

In the second phase of the research, the condition that the models should be able to adapt to new/modified *categories*, without needing exhaustive retraining is taken into consideration. To overcome this variability of *categories*, while keeping the model explainable (the fourth condition), it was found that the inclusion of a k-Nearest Neighbor algorithm (k-NN) would be most suited. This algorithm searches the input-space for the k messages which are closest and outputs the *category* that is most common in this neighborhood. This search happens the moment a new message needs to be categorized. Hence the data is only evaluated at that moment. Note that, as the *categories* are also only evaluated at that moment, the model is able to adapt to any changes that have been made to the dataset before that moment.

In this second phase, four approaches using k-NN are investigated. These consist of an aggregation method, Neighborhood Component Analysis (NCA), and the k-NN. The aggregation method could be any method which translates the text of the message into a numerical vector. In this study, two bag-of-words methods are used as aggregation methods in addition to the document embeddings, as obtained from the best-performing model in Phase A (See Section 4.1.1.1). First, the reasoning behind the usage of NCA will be described, then the different aggregation methods, and the corresponding models in this research, will be discussed.

The viability of the k-NN algorithm pivots on the assumption that the input-space of the k-NN is separable in such a way that 'similar' points, with the same *category*, are close together, clustered in neighborhoods. To ensure the validity of this assumption, the aforementioned Neighborhood Component Analysis (NCA) is used as a pre-processing step to transform the input-space. NCA is

[4]https://translate.google.com
[5]https://www.bing.com/translator

an algorithm that learns a linear transformation in a supervised fashion to improve the classification accuracy of a stochastic nearest neighbor rule in the transformed space. I.e. NCA can be used to transform the input-space into a space that is separable by *category* using a k-NN algorithm. Hence all models in this second phase, which all include a k-NN, also include the NCA as a pre-processing step.

As indicated above, four approaches using k-NN are investigated. Two of these approaches utilize a bag-of-words method to transform the text data into numerical data and two approaches utilize the document embeddings as obtained from the best-performing model in Phase A instead. The first approach will use the frequency that each word occurs in a message as the input for the NCA, k-NN combination (Model *B1*). This approach can be augmented by incorporating the relative importance of words using the Term Frequency-Inverse Document Frequency statistic (TF-IDF). This statistic increases proportionally to the frequency with which a word is used in a message and is offset by the number of messages that contain that word. This counteracts the influence of inconsequential words which are frequently used, like conjunctions and determiners (Model *B2*). The TF-IDF for term t in document (message) d is calculated as follows:

$$\begin{aligned} \mathrm{TF}(t,d) &:= \text{Frequency of term } t \text{ in document } d \\ \mathrm{df}(t) &:= \text{Number of documents that contain the term } t \\ \mathrm{IDF}(t) &= \log\left(\frac{n}{\mathrm{df}(t)+1}\right) \\ \text{TF-IDF}(t,d) &= \mathrm{TF}(t,d)\cdot\mathrm{IDF}(t) \end{aligned}$$

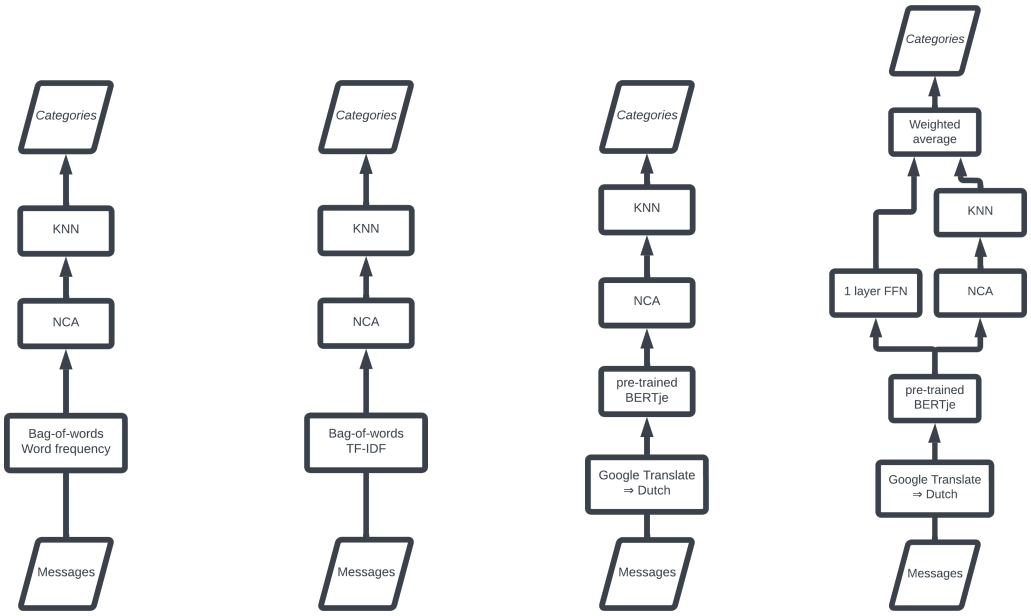
Here, n represents the total number of messages in the dataset. Note that neither of these two methods would be able to learn the semantical meaning of the messages, only the (relative) frequency of each word in a message. Similarly, the sequential nature of text is lost as both approaches are based on bag-of-words methods.

To remedy the lack of semantic understanding and the inability to utilize the sequential nature of text in the messages, the k-NN is combined with the best model from the first phase. The aforementioned document embeddings, which consist of vectors of 768 values for each message, are used as the input-space for the k-NN. Note that, as the models from phase A use a multiclass perceptron to separate these document embeddings into the *categories*, the input-space is assumed to be linearly separable. Such a linearly separable space is not necessarily separable by a k-NN, hence NCA is also utilized for the approaches which combine the pre-trained models and the k-NN.

The third approach will directly apply NCA and the k-NN to the document embeddings (Model *B3*), as obtained from the best model from the first phase. Whereas the fourth method (Model *B4*) combines the predictions from this best model from the first phase (Model *A**) and the predictions from the application of the k-NN to the document embeddings, i.e. Model *B3*, using a weighted average. This approach is inspired by the work of Kassner and Schütze, and Khandelwal et al. (Kassner and Schütze 2020; Khandelwal et al. 2020). With the primary difference being the addition of the NCA. The relative weight for each prediction is managed using an extra parameter p_{k-NN}. Hence the calculation for this fourth approach is as follows $P_{B4}(Category = X) = p_{k-NN}P_{B3}(Category = X) + (1 - p_{k-NN})P_{A^*}(Category = X)$.

Note that only the best model from the first phase is used in conjunction with the k-NN. This choice is based on the assumption that the models from the first phase are independent from those in the second phase. This assumption is factually incorrect, yet the near identical results for the models in the first phase (Section 5.1) suggest that no significant dependence is present. These results indicate that the approaches are interchangeable. Hence only the approach from the first phase that performs best (Model *A**), will be used in conjunction with the last two approaches from the second phase.

In summary, in the second phase, four approaches are investigated, two of which utilize a simple bag-of-words method and the other two use the document embeddings, as obtained from the best model from the first phase. Figure 7 provides a schematical representation for each of these models. Note that Model *B3* and *B4* use BERTje and, into Dutch translated, messages. Section 5.1 will show that this model, Model *A3*, is indeed the best model from the first phase.



(a) Schematic representation of Model *B1*

(b) Schematic representation of Model *B2*

(c) Schematic representation of Model *B3*

(d) Schematic representation of Model *B4*

Figure 7: Schematical representation for all four models of the second phase.

4.2.1 Neighborhood Component Analysis

Neighborhood Component Analysis, or NCA for short, is a supervised algorithm which can learn a linear transformation that can be used to improve the classification accuracy of a k-NN. NCA attempts to transform a space such that points with a similar *category* are are close together in neighborhoods by maximizing the expected accuracy of a stochastic nearest neighbors algorithm. When used as a pre-processing step for a k-NN, the k-NN is expected to perform better, as the input-space is already separated in neighborhoods for the different *categories*.

For calculating the optimal transformation, the k-NN is modified to a stochastic nearest neighbors algorithm. In this case, the prediction of a *category* for a certain point is based on the *category* a randomly chosen point. The probability that point i is classified using point j is calculated using the softmax over the squared Euclidean distances from point i:

$$p_{ij} = \begin{cases} \frac{e^{-||Ax_i - Ax_j||^2}}{\sum_{k \neq i} e^{-||Ax_i - Ax_k||^2}} & \text{if} j \neq i \\ 0 & \text{if} j = i \end{cases}$$

In this calculation, matrix A represents the linear transformation that is to be learned. The probability that a certain point i gets classified correctly is then the sum of all p_{ij} where j has the same *category* (C_i) as point i:

$$p_i = \sum_{j \in C_i} p_{ij}$$

The idea behind NCA, is to maximize this p_i for all points i by modifying the matrix A:

$$A^* = \underset{A}{\text{argmax}} \left\{ \sum_i p_i \right\}$$

Note that the usage of stochastic nearest neighbors leads to a relatively simple objective function which is differentiable. An iterative solver can then be used to approach the optimal value for A. The resulting transformation matrix A can be split into separate components, similar to Principal Component Analysis. These components are ordered by the magnitude of the effect it has on the maximization. The first component is more beneficial than the second when increasing $\sum_i p_i$. One will experience diminishing returns when increasing the number of components one uses.

4.2.2 k-Nearest Neighbors

The k-Nearest Neighbors might very well be the easiest algorithm for a layman to understand. This algorithm takes a message where the *category* is unknown and compares this to the messages where the *category* is known. It finds the k closest matches (generally, by looking at the Euclidean distance between points). The majority *category* of this neighborhood will then be the predicted *category* for the initial message. A major difference between a k-NN and most other classifiers is the fact that no real 'model' is created. For every prediction that is required, the k nearest neighbors need to be found from scratch. This is significantly more inefficient than a simple set of matrix multiplications as for feed-forward networks. While it is possible to determine for each point in the input space which *category* is most common in that neighborhood by determining the decision boundaries, this is not desirable for this research. The fact that the neighbors need to be recalculated is beneficial if the *categories* of labeled data changes. Indeed, this is the desire that is conveyed in the third condition as described in Section 1.2.

4.3 Evaluation

The main metric for evaluating the quality models will be the accuracy. While other metrics like the recall and precision are informative and will be shown, they are of secondary import for the stakeholders in this research. The fine-tuning will, however, be done by optimizing the cross-entropy loss. Optimizing the accuracy could lead to a model where the predicted 'probability' for the correct *category* is barely larger than the 'probability' for the wrong *categories*. Minimizing the cross-entropy loss maximizes the 'probability' for the correct *category*. Here, a confident prediction, with a predicted 'probability' of 1, would be superior to an unsure prediction, where the 'probability' for the correct *category* is barely larger than that of the other *categories*. In this case, one could use the predicted 'probabilities' as a measure of the sureness of the model. When the most likely *category* for a certain message, is still somewhat unlikely, one could elect to have a human evaluate that message.

The final evaluation is to be done on a separate test set, containing approximately 20% of all messages (23,689). Furthermore, an incomprehensive investigation of the differences and similarities in behavior between the models will be provided.

The results of the models will be compared to four baselines, two of which have been described in Section 4.2. These models utilize a bag of words method (either the word frequency or TF-IDF) in conjunction with NCA and k-NN to categorize the messages. In addition to these simplistic models, a majority classifier and a random classifier will be used as baselines.

To evaluate the performance of the models when new *categories* are added or existing *categories* are modified, five extra *categories* are used. For each of these *categories*, 50 labeled messages are added to the train set and then the model performance is investigated using a further 10 messages.

4.4 Experimental Setup

In summary, one can split this research into two 'phases'. In the first phase, four different combinations of Machine Translation and a pre-trained BERT variant will be used to categorize the messages:

A1. Messages → mBERT → *Category*

A2. Messages → Translate to English → BERT → *Category*

A3. Messages → Translate to Dutch → BERTje → *Category*

A4. Messages → (Translate to Dutch + Translate to English) → mBERT → *Category*

The fine-tuning of these models will be done according to the recommendations by the authors of the paper on BERT (Devlin et al. 2018) using a train set, consisting of 80% of the data (94,756 messages). Most hyperparameters will remain the same as during the pre-training of the models. The exceptions are the batch size, the learning rate, and the number of training epochs.

Devlin et al. indicate that, while the optimal hyperparameter settings are task-specific, the following range of possible values work well across all tasks:

Batch size: 16, 32

Learning rate (Adam): 5e-5, 3e-5, 2e-5

Number of epochs: 2, 3, 4

Due to the time and computational constraints, only a batch size of 16 was used. Furthermore, all models were fine-tuned with 3 epochs. The intermediate results of the fine-tuning process did not give sufficient reason to assume the model would improve significantly if it were allowed to train for an extra epoch (See also Section 5.1). To find the best learning rate, an exhaustive search was done and the optimal rate was chosen based on the performance on a validation set of 20% of the train set (18,951 messages). Devlin et al. do indicate that the influence of hyperparameter tuning is expected to be little when the train set is large (e.g. 100k+ training examples).

The second phase of this research focuses on the addition of the k-NN to the model. In this phase, two simplistic models are created and two combinations of the best model from the first phase, Model $A*$, and a k-NN classifier will be tested:

B1. Messages $\rightarrow$ Frequency per word used $\rightarrow$ NCA $\rightarrow$ k-NN $\rightarrow$ *Category*

B2. Messages $\rightarrow$ TF-IDF of each word used $\rightarrow$ NCA $\rightarrow$ k-NN $\rightarrow$ *Category*

B3. Messages $\rightarrow$ 'Document embedding' from Model $A*$ $\rightarrow$ NCA $\rightarrow$ k-NN $\rightarrow$ *Category*

B4. Messages $\rightrightarrows$ $\dfrac{\text{Predictions from Model } A*}{\text{'Document embedding' from Model } A* \rightarrow \text{NCA} \rightarrow \text{k-NN}}$ $\rightrightarrows$ Weighted average $\rightarrow$ *Category*

This second phase introduces three new parameters: The number of components to be used from the NCA (c), the number of neighbors to consider in the k-NN (k) and the weights to be used in the weighted average for Model *B4* (p_{k-NN}). For the first parameter, it is assumed that the first 100 components, as found by NCA, should be more than sufficient to transform the data such that the 9 different *categories* can be identified with a k-NN. Similarly, it is assumed that considering 50 neighbors is most likely more than optimal. Hence the following range for both parameters will be tested exhaustively:

c: 2, 5, 10, 15, ..., 95, 100

k: 1, 2, 3, 4, ..., 49, 50

The assumptions around the maximum values and the step sizes for these ranges are, in part, inspired by Goldberger et al. 2004 and Kariuki et al. 2022. Model *B4* requires an additional hyperparameter $p_{k-NN} = 1 - p_{A*}$ for the weighted average. This parameter distributes the power of both parts. When $p_{k-NN} = 0$, Model *B4* will purely use Model $A*$ and if $p_{k-NN} = 1$, the model will only use the k-NN (i.e. Model *B3*). An exhaustive search will be done to find the optimal value for this hyperparameter. The step sizes for the three hyperparameters are likely smaller than necessary. This fine grain was chosen to ensure the grid search was fine enough while remaining computationally viable. If fewer computational resources were available, one could elect to use a coarser grain and/or use a randomized grid search instead of an exhaustive search.

The fine-tuning and creation of the models, including the generation of all graphs, for this project, will be done using Python. To accomplish this, a variety of third-party modules will be used. The entire list, including the specific versions used, will be provided in Appendix C.

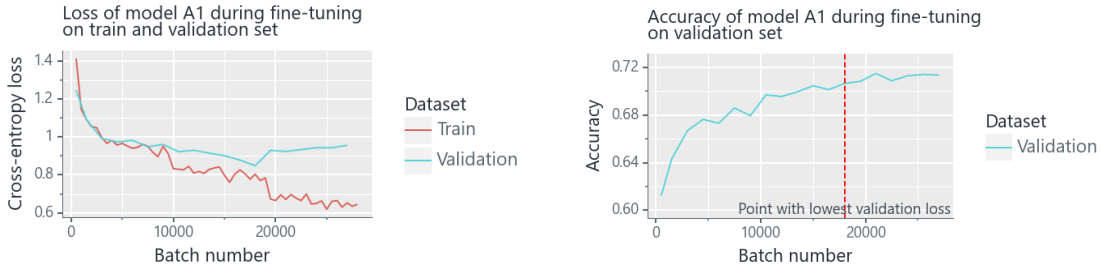
5 Results

In this section, the results from the models, as described in the previous section, will be shown and discussed. The final and intermediary results from the fine-tuning process of the models of Phase A will be provided. This will be followed by a brief investigation into the differences between these models and the common mistakes made by the models. Furthermore, several examples will be given of how the models distribute their attention over each word in a message. In Section 5.2, the results from phase B will be shown. Here, the performance of the models for each hyperparameter setting is shown, in addition to the final results of all models. Also included in this section is the evaluation of the models on new/modified *categories*.

5.1 Phase A

5.1.1 Intermediary results

When considering the results of a fine-tuned model one should first investigate the fine-tuning process itself. The intermediary results are often able to provide useful insight into the specific hyperparameters and the quality of the results. These intermediary results for Model *A1* can be seen in Figure 8a and Figure 8b. These figures show the loss per batch of train data and the loss and accuracy on the validation set every $1,500$ batches. The batches in these graphs encompass the entirety of the three epochs. The figure clearly shows that the training loss steadily decreases over time as expected. One can also see the validation loss decreasing, but increasing after batch $18,000$. The point where the validation loss is least is the point at which the model is presumed to perform optimally. This point is marked by a red line in Figure 8b. In the period after this minimum, the model starts to overfit on the train set. When looking at the validation accuracy, one can see that the line seems to converge. However, at the point where the validation loss is lowest, the accuracy is approximately 0.70.



(a) Train and validation loss during fine-tuning of Model *A1*

(b) Validation accuracy during fine-tuning of Model *A1*

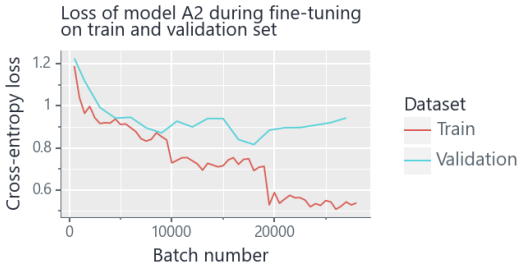
Figure 8: Intermediary results for Model *A1*

Similarly for Model *A2* in Figure 9a and 9b, one can see the training loss decrease during all 3 epochs, while the validation loss starts to increase after batch $18,000$. At this point, the accuracy is approximately 0.72 for the validation set. The remaining graphs of Figure 9c through 9f show the intermediate results for Model *A3* and *A4* respectively. Remarkably, the validation loss is minimal at batch $18,000$ for each of the models. This seems to be coincidental, as the batches are randomly ordered for each model. At this optimum, the validation accuracy is approximately 0.71 for both Model *A3* and *A4*.

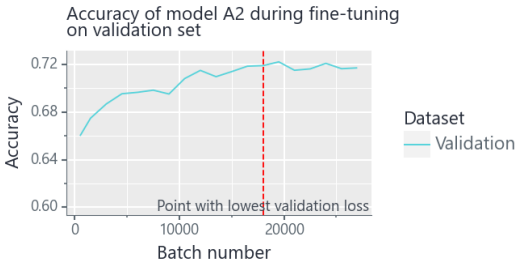
5.1.2 Overall performance

Figure 8, and 9 only show the intermediary results when using a learning rate of 5e-5. While the other graphs show similar results, a learning rate of 5e-5 resulted in marginally better results. The results for these four models with all three different learning rates are summarized in Table 2. In this table, the validation accuracy and macro F1-score at the optimal timestep are shown. A 95% confidence interval is added around the accuracy and F1-score using naive bootstrapping. Note

that there is no clear indication that the underlying distribution of these metrics is heavy-tailed, hence bootstrapping seems like a valid approach (Athreya 1987).

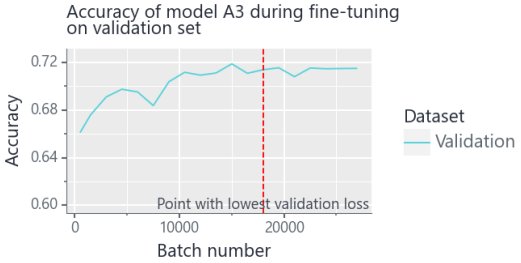


(a) Train and validation loss during the fine-tuning of Model *A2*

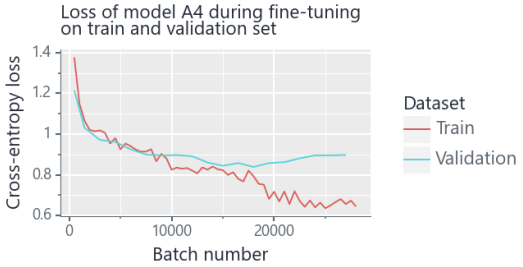


(b) Validation accuracy during the fine-tuning of Model *A2*

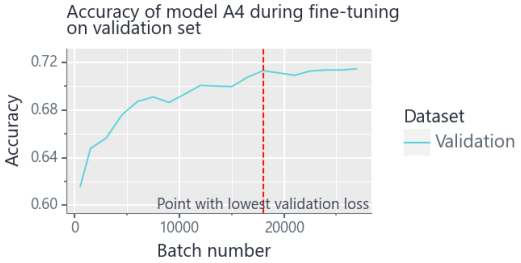
(c) Train and validation loss during the fine-tuning of Model *A3*



(d) Validation accuracy during the fine-tuning of Model *A3*



(e) Train and validation loss during the fine-tuning of Model *A4*



(f) Validation accuracy during the fine-tuning of Model *A4*

Figure 9: Intermediary results for Model *A2*, *A3*, and *A4*

Table 2: Validation accuracy and macro F1-score for each model in the first phase as obtained with the three different learning rates

	Learning rate	Accuracy	Macro F1-score
Model *A1*	5e-5	0.705 ± 0.006	0.659 ± 0.010
	3e-5	0.673 ± 0.007	0.609 ± 0.009
	2e-5	0.655 ± 0.007	0.586 ± 0.008
Model *A2*	5e-5	$\mathbf{0.714 \pm 0.006}$	0.651 ± 0.009
	3e-5	0.673 ± 0.007	0.595 ± 0.009
	2e-5	0.708 ± 0.006	0.644 ± 0.008
Model *A3*	5e-5	0.712 ± 0.006	$\mathbf{0.670 \pm 0.009}$
	3e-5	0.680 ± 0.006	0.625 ± 0.009
	2e-5	0.672 ± 0.007	0.612 ± 0.009
Model *A4*	5e-5	0.712 ± 0.006	0.660 ± 0.008
	3e-5	0.706 ± 0.007	0.653 ± 0.009
	2e-5	0.671 ± 0.007	0.603 ± 0.009
Majority Class	-	0.350 ± 0.009	0.058 ± 0.001
Random Class	-	0.183 ± 0.006	0.108 ± 0.005

As Table 2 shows, all models perform significantly better than the simple baselines yet no model significantly outperforms the others when considering the validation accuracy. A slight difference can be seen when comparing the accuracy of Model *A1* to the others. Model *A1* seems to perform marginally worse than the other models. Larger differences can be seen when investigating the F1-score. Here one can see that Model *A3* performs better than the other models. The runner-up would be Model *A4* which performs 0.01 worse. The confidence intervals are relatively small, which is likely due to the relatively large validation set (18,951 samples), yet it does indicate that the models have a low variance and are quite robust. As stated before, the higher learning rates (3e-5 and 2e-5) result in marginally worse results. From this point, only the models that were obtained using a learning rate of 5e-5 will be considered. Furthermore, given the relatively high performance of Model *A3*, the model that translates the messages to Dutch and uses a fine-tuned BERTje, on both the validation accuracy and the F1-score, this model will be considered the best for this phase.

5.1.3 Performance per *category*

It should be noted that the results are significantly below the hopes and expectations of the author. To elucidate these dissatisfying results, the performance per *category* is investigated. Table 3 shows the validation recall, and precision for each of the *categories*. Note that the recall could, informally, be interpreted as an indication of how easy a *category* is to predict. A low recall could, for example, indicate that the *category* is poorly defined or that the model is not complex enough to understand the *category*. The precision indicates how valid the prediction for the class is. A low precision indicates that other *categories* are often mistaken for this *category*. One might therefore assume that *categories* with a low precision sabotage the performance of the model on the other *categories*.

Table 3: Validation precision and recall per *category* for each of the four models with an average precision and recall for these models combined.

	Model *A1*		Model *A2*		Model *A3*		Model *A4*		Average		
Category	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Freq.
Accept. new construction	0.987	0.459	0.939	0.468	0.754	0.627	0.883	0.621	0.891	0.544	
Appoint. with caretaker	0.742	0.523	0.708	0.481	0.643	0.594	0.741	0.440	0.709	0.510	
General Admin.	0.684	0.639	0.579	0.735	0.713	0.726	0.627	0.672	0.651	0.693	
General Renting	0.704	0.524	0.519	0.607	0.482	0.538	0.506	0.659	0.553	0.582	
Financial matters	0.676	0.957	0.854	0.895	0.926	0.780	0.812	0.909	0.817	0.885	
Comment new room	0.749	0.454	0.685	0.448	0.504	0.771	0.749	0.389	0.672	0.516	
Nuisance	0.690	0.639	0.572	0.694	0.922	0.488	0.606	0.667	0.698	0.622	
Service costs	0.873	0.561	0.785	0.586	0.938	0.523	0.701	0.677	0.824	0.587	
Keys	0.743	0.702	0.684	0.818	0.612	0.924	0.681	0.835	0.680	0.820	

When looking at the *category* 'Acceptation new construction' in Table 3, one can see that precision is high for Model *A1* and *A2*, while the recall is low. This implies that the models predict this *category* not often enough, but when they do, they are often correct. Model *A3* and *A4* have a lower precision, but a significantly higher recall. This indicates that the *category* is more often predicted correctly, but the predictions of this *category* are also more often incorrect. When looking closely at Model *A1* one can see that the precision for each *category* is relatively high, while the recall is low for most *categories*. Given that the recall for 'Financial Matters' and 'Keys' is high, one could conclude that the Model *categorizes* messages too often as these *categories*. This would imply that the model only predicts another *category* if it is very certain (i.e. a high precision for the other *categories*). Remarkably however, the precision for these two exceptional *categories*, is not very low. This is likely due to the fact that the *categories*, and especially 'Financial matters' are rather common. While the recall is low for most of the *categories* for Model *A1*, there is no clear sign that any of the *categories* is truly undesirable. Looking at the overall average gives an indication of how difficult each *category* is to learn. Here it becomes clear that the *category* 'Financial Matters' is relatively easy to predict, while 'General Renting' is hard. Note that the recall for each *category* is far above the expected recall for a randomized model (i.e. the relative frequency of the *category*). This indicates that although 'General Renting' is hard to predict, the models are still able to learn some information about this *category*. Hence, there is no clear reason to conclude that any of the *categories* should be left out to improve the results.

Earlier, Model *A3* was designated as the best model of the first phase. One can see a remarkably high precision for the *categories* 'Financial Matters', 'Nuisance', and 'Service costs' and no tremen-

dously low recalls. It is however clear that the model struggles with identifying 'Nuisances' and the predictions for 'Comment new room' and 'General Renting' are not very trustworthy. Figure 10 shows the confusion matrix as obtained from this model. Here one can see that the diagonal (i.e. the correct predictions) is quite prevalent. Note that 638 of the 'Financial Matters' messages are categorized as 'Comment New Room' and 396 as 'General Renting'. One could reasonably suspect that these mistakes are due to the similar nature of the *categories*. A tenant could, for example, ask about the financial arrangements regarding a new room. It would not be implausible that the tenant or an employee would assign either of these *categories*. It is reasonable to assume that a certain indeterminable part of the messages can not be 'correctly' categorized due to data quality issues and an overlap between the different *categories*. Furthermore, 486 of the 'General Renting' messages have been predicted to belong to 'Comment new room' and 106 to 'Keys'. Once again, these mistakes are somewhat explainable by the nature of the *categories*. Another clear example of explainable mistakes are the 230 'Service costs' messages that have been categorized as 'Financial matters'. One might expect the precision of the generic *categories* ('General Administration' and 'General Renting') to be very low as their definition is rather vague. Table 3 indicated that this was only the case for 'General Renting'. In 325 cases, 'Nuisance' messages are seen as 'General Administration', but this is both understandable and not that significant in comparison to the number of correct predictions. For 'General Renting' on the other hand, 396 'Financial matters' messages are predicted as being 'General Renting'. This is approximately 23% of all 'General Renting' predictions. Similarly, 164 ($\approx 10\%$) 'Service costs' messages are seen as 'General Renting'.

Figure 10: Confusion matrix of model A3 on validation data

To get an informal validation of the supposition that a significant part of the mistakes is explainable by nature of the *categories*, a randomized selection of 10 mistakes from each of the aforementioned types of mistakes have been investigated. The summarized results of this informal investigation can be found in Appendix A. Here it becomes clear that the majority were correctly labeled. Hence, the model does indeed make many mistakes. The messages in question did however often contain topics that could reasonably belong to the predicted *category*. For example, a tenant might notify the housing corporation that they have found a new room and that the contract should be terminated. Such a message would belong clearly to the 'Financial Matters' *category*, yet contains text about a new room, which tends to be associated with the 'Comment new room' *category*. Furthermore, several messages contain multiple questions. One might ask when they can collect their keys and ask about their lease in the same message. In that case, there is no clear correct *category*.

Table 4 provides three examples from this investigation. This table includes an indication of which parts of the sentence are most important in predicting the *category*. This indication is obtained by calculating the integrated gradient from the output to the input tokens (Sundararajan et al. 2017). These gradients show the influence of an input on the predictions. They are compared to the gradients of a baseline sentence consisting of only special padding tokens. The integrated gradients

display how much the input tokens influence the predictions with respect to a zero measurement. A word that has a large influence on the output is shown with a darker background in Table 4, while a word with no influence on the output has a white background.

When looking at the first example in Table 4, it is clear the *category* is difficult to predict. The model would need to understand that, although the tenant specifically says that they want to cancel their contract, the question is about the rent. For the second example, one might conclude that the true label 'General Renting' is not actually correct. Remarkably, the second usage of the word '*sleutel*' ('key') does not seem to have an effect on the prediction, while the first one does. Finally for the third example, one can see that the model tends to focus on words which indicate the recency of an event ('*Onlangs*'-'recently', '*Ondertussen*'-'in the meantime', '*nieuwe ... gevonden*'-'found a new ...', and '*maand vanaf nu*'-'month from now'). The model seems to focus relatively little on the actual request. It is not unlikely that the *category* 'Comment new room' would contain much more references to the currency of the topic at hand than the *category* 'General Renting'.

Table 4: Table containing the Dutch and English texts for three examples of misclassified messages with the relative importance per word shown by the opacity of the background color.

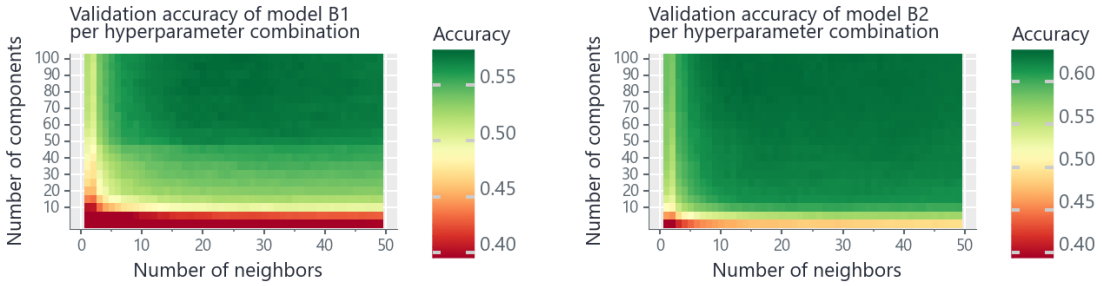
True label	Predicted label	Dutch text	English text
Financial Matters	General Renting	Ik wil graag mijn contract opzeggen en vraag me af of ik de huur voor juli en augustus kan terugkrijgen . Dank je voor je hulp !	I would like to cancel my contract and wonder if I can get the rent back for July and August. Thank you for your help!
General Renting	Keys	Hallo , ik ben vandaag uit mijn kamer verhuisd en heb de sleutel in het deurslot van de conciërge laten vallen . Ik wist niet dat je de sleutel in een envelop moest doen met je kamernummer erop . Ik heb het huisvestingsbureau hierover geïnformeerd en zij zeiden dat het geen probleem was , maar ze wilden dat ik contact met jullie opnam om het jullie te laten weten ! Beste wensen	Hello, I moved out of my room today and dropped the key in the door lock of the concierge. I didn't know you had to put the key in an envelope with your room number on it. I informed the housing office about this and they said no problem, but they wanted me to contact you to let you know! Best wishes
General Renting	Comment new room	Beste , Onlangs heb ik het huurcontract van <adress> opgezegd met de langst mogelijke opzegtermijn . Ondertussen heb ik een nieuwe woning gevonden en zou daarom graag de einddatum van het huurcontract wijzigen naar <date> (precies een maand vanaf nu) . Met vriendelijke groet , <name>	Best, I recently canceled the <address> rental contract with the longest possible notice period. In the meantime, I have found a new home and would therefore like to change the end date of the rental contract to <date> (exactly one month from now). Yours sincerely, <name>

5.2 Phase B

Considering the results from phase A, it is clear that Model *A3*, the model that translates the messages to Dutch and uses a fine-tuned BERTje, performs the best overall, but will still make a significant amount of mistakes. Furthermore, this model is not able to adjust to *categories* that get newly created or get modified. In this section, the results of the four models from the second phase will be discussed. Recall that two of these four models utilize the optimal model from the first phase. These models, which combine a transformer model with a k-NN, will therefore utilize Model *A3*.

5.2.1 Hyperparameter Tuning

Figure 5 shows the validation accuracy for each combination of neighborhood components (for the NCA) and neighbors (for the k-NN). From the figure, the diminishing returns for the number of components and the number of neighbors become clear. Very little improvement is made when increasing either the neighbors or the components from, about 15 neighbors and 50 components. Recall that Model *B1* uses the frequency of each word as the input and *B2* the Term Frequency-Inverse Document Frequency. As expected, one can see that Model *B2* performs better than Model *B1*. This becomes even clearer when inspecting Table 5. This table shows the validation accuracy and F1-score for both models with the optimal combination of the hyperparameters, the baselines as seen earlier in Section 4.1, and the best model from the first phase, Model *A3*. Remarkably, the simplistic k-NN models, which do not use the semantic meaning of the messages and only look at the frequencies that words are used in, perform only slightly worse than Model *A3*. As Model *B2* performs significantly better than Model *B1*, it is clear that utilizing the TF-IDF helps with the performance.



(a) Validation accuracy for Model *B1* for each combination of the hyperparameters

(b) Validation accuracy for Model *B2* for each combination of the hyperparameters

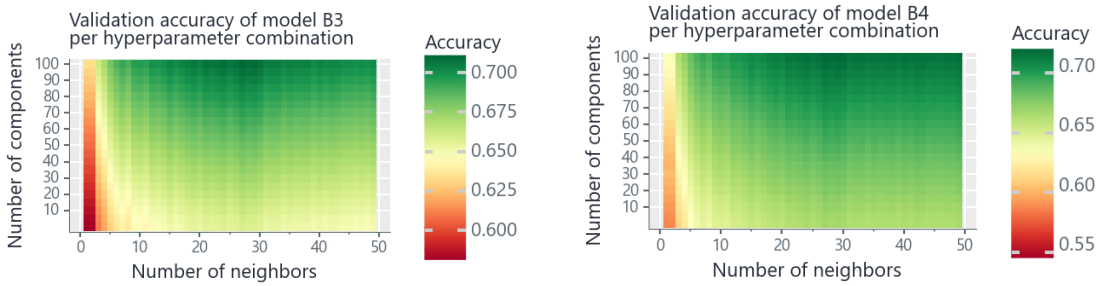
Figure 11: Results for all tested hyperparameters for Model *B1* and *B2*

Table 5: Validation accuracy and macro F1-score for the models from the second phase, when using optimal hyperparameters, and baselines

	Neighbors	Components	Accuracy	Macro F1-score
Model *B1*	19	95	0.575 ± 0.009	0.442 ± 0.013
Model *B2*	23	95	0.631 ± 0.009	0.546 ± 0.012
Model *B3*	31	95	0.658 ± 0.008	0.612 ± 0.011
Model *B4*	30	90	$\mathbf{0.715 \pm 0.009}$	0.663 ± 0.010
Model *A3*	-	-	0.712 ± 0.006	$\mathbf{0.670 \pm 0.009}$
Majority Class	-	-	0.350 ± 0.009	0.058 ± 0.001
Random Class	-	-	0.183 ± 0.006	0.108 ± 0.005

As for the simple models *B1* and *B2*, the optimal hyperparameters were determined for Model *B3* and *B4*. The validation accuracy for each of the combinations is shown in Figure 12. Once again, one can clearly see diminishing returns for increasing the neighbors and components. Here it seems that approximately 30 neighbors and 90 components are optimal for both. Recall that Model *B4* combines the predictions from the transformer model (*A3*) and the k-NN (*B3*) using a weighted

average. Figure 13 shows the validation accuracy for all possible values of p_{k-NN} (in steps of 0.001). Note that when using $p_{k-NN} = 0$, Model *B4* is equivalent to Model *A3* and when using $p_{k-NN} = 1$, Model *B4* is equivalent to Model *B3*. The graph shows that the validation accuracy is highest when the predictions are 19.5% based on the k-NN and 80.5% on the transformer model. There is little difference between the performance of this optimum and Model *A3*. The major difference is that Model *B4* is, theoretically, able to adapt to new/modified *categories*. The actual validation accuracy and F1-score are shown in Table 5. Note that, while Model *B4* performs significantly better than Model *B3*, it does not clearly outperform Model *A3*. Given the fact that Model *B4* has the highest validation accuracy, and is theoretically able to adapt to new/modified *categories*, it is deemed the best model in this study. This single model has been evaluated on the test set, which resulted in a test accuracy of 0.716 ± 0.008.



(a) Validation accuracy for Model *B3* for each combination of the hyperparameters

(b) Validation accuracy for Model *B4* for each combination of the hyperparameters

Figure 12: Results for all tested hyperparameters for Model *B3* and *B4*

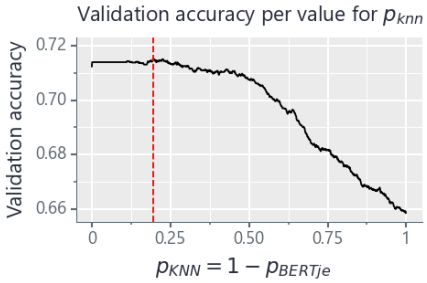


Figure 13: Linegraph of the relation between the hyperparameter p_{k-NN} for Model *B4* and the validation accuracy

5.2.2 Performance per *category*

Table 6 shows the validation precision and recall for Model *A3*, *B3*, and *B4* per *category*. Note that Model *B3* only outperforms Model *A3* when considering the precision for 'General Renting', the recall for 'Financial Matters', and the recall for 'Nuisance'. Despite this, Model *B4* seems to improve on the primary weaknesses of Model *A3* (e.g. the precision for 'General Renting' and the recall for 'Nuisance'). This implies that models *A3* and *B3* have a different understanding of the same data, both of which are utilized in Model *B4*.

Table 6: Validation precision and recall per *category* for Model *A3*, *B3* and *B4*, where the highest precision and recall per *category* is highlighted.

	Model *A3*		Model *B3*		Model *B4*		
Category	Precision	Recall	Precision	Recall	Precision	Recall	Freq.
Acceptance new construction	**0.754**	**0.627**	0.592	0.431	0.723	0.585	
Appointment with the caretaker	0.643	**0.594**	0.546	0.376	**0.674**	0.497	
General Administration	**0.713**	**0.726**	0.557	0.572	0.67	0.679	
General Renting	0.482	**0.538**	0.558	0.289	**0.718**	0.442	
Financial matters	**0.926**	0.78	0.77	0.879	0.806	**0.898**	
Comment new room	0.504	**0.771**	0.476	0.509	**0.572**	0.64	
Nuisance	**0.922**	0.488	0.676	0.5	0.732	**0.602**	
Service costs	**0.938**	0.523	0.631	0.52	0.73	**0.589**	
Keys	0.612	**0.924**	0.547	0.784	**0.625**	0.824	

5.2.3 Adaptability to new/modified *categories*

To evaluate the adaptability of the model to new/modified *categories*, the models from the second phase are applied to previously unseen *categories*. Five new *categories* have been chosen with 60 messages each. As stated in Section 4.3, 50 messages will be added to the 'train set' (i.e. the input for the k-NN) and predictions will be made for the remaining 10. The addition of the messages to the train set represents the manual action from the housing corporation to add existing messages to the new *category*. Table 7 shows the results from this experiment. Recall that these *categories* have been chosen to be easy to predict. The table, therefore, shows an indication of the adaptability of the model in a near-ideal situation. Note that Model *B4*, due to the fact that it relies for 80.5% on the transformer Model *A3*, which can not adapt to new *categories*, performs worst overall. Model *B2* and *B3* however, perform quite well. The performance of these models seems close to the performance on messages with previously known *categories* (Validation accuracy of 0.631 and 0.658 respectively).

Table 7: Number of correctly predicted *categories* for unseen *categories* where only 50 labeled examples are available

Category	Model *B1*	Model *B2*	Model *B3*	Model *B4*
Campus Control	5/10	5/10	**6/10**	2/10
Block heating allowance	6/10	**7/10**	6/10	3/10
Internal relocation	4/10	3/10	**5/10**	1/10
Lease agreement	**5/10**	**5/10**	4/10	2/10
Subletting	5/10	6/10	**7/10**	4/10

6 Conclusion

In this research, the possibilities for the usage of transformer models have been investigated for the automatic classification of incoming messages at the department for customer service of housing corporations. Major challenges here are the fact that these messages are either Dutch or English and that the target *categories* can, occasionally, change or new *categories* can be created. This objective and these challenges have been formulated in the following research question and conditions:

> How can the process of categorizing/routing incoming e-mails at the customer service of housing corporations be automated with the use of attention-based models, given the bilingualism of the e-mails and the variability of *categories*?

This primary question should lead to an algorithm that complies with the following conditions:

1. The algorithm should be able to utilize the semantic meaning of the messages to assign the right *category*.
2. The algorithm should be able to process both English and Dutch messages.
3. The algorithm should be able to adapt to new *categories* with few data, without exhaustive retraining of the models.
4. The overarching concept of the algorithm should remain easily explainable to end users with little to no mathematical knowledge.

The first condition regarding the utilization of the semantic meaning of the messages is satisfied by the usage of attention-based models. For the second condition, regarding the bilinguality of the data, four approaches have been investigated. Firstly a pre-trained multilingual BERT model, mBERT, has been fine-tuned on the messages (Model *A1*). The second approach used Machine Translation to translate the Dutch messages into English and used BERT to fine-tune the messages (Model *A2*). The third approach is similar to the second, but the English messages are translated into Dutch and a Dutch variant of BERT, BERTje, was fine-tuned (Model *A3*). Finally, a new approach was introduced where Machine Translation was used to obtain both a Dutch and an English version of each message. These versions were concatenated and mBERT was fine-tuned using these concatenated messages (Model *A4*). It was found that each of these approaches lead to similar results which were significantly higher than the Majority Class baseline and a random classifier. Although the approaches gave very similar results, the third approach (Model *A3*) was deemed to perform best with a validation accuracy of 0.712.

The third condition addresses the dynamic nature of *categories*. Occasionally new *categories* can be created and existing *categories* can be deleted or modified. To allow the models to adapt to new/modified *categories*, a k-Nearest Neighbor classifier (k-NN) is used in conjunction with the fine-tuned transformer model. Four approaches using k-NN have been investigated. Each of the approaches uses Neighborhood Component Analysis (NCA) to transform the data such that it is separable using a k-NN. Two of the approaches are simple bag-of-words models which do not use the transformer models. The first approach (Model *B1*) uses the frequency that each word is used in a message as the input for the k-NN. The second approach (Model *B2*) is similar but uses the Term Frequency-Inverse Document Frequency (TF-IDF) instead. The third approach (Model *B3*) uses the fine-tuned Model *A3* to obtain a document embedding, which is then used as the input for the k-NN. Finally, the fourth approach combines the predictions from Model *A3* and *B3* using a weighted average. It was found that the fourth approach *B4* lead to the highest validation accuracy (0.715), but the performance on new *categories* was abysmal due to the heavy influence of Model *A3* in the prediction. The second and third approaches seemed to perform relatively well on the new *categories*.

The final condition, regarding the explainability of the models at a conceptual level, is satisfied by the model choice. Each model consists of up to four 'modules': Machine Translation, fine-tuned transformer model, NCA, and/or k-NN. While the Machine Translation is very complex, the concept of translating text using software is known to nearly everyone. It is unlikely that anyone will have never heard of tools like Google Translate or Bing Translator. The exact theory behind the fine-tuned transformer model is also rather complex. The function of the model is

however very simple: To obtain a numerical representation of the semantic meaning of the text. The basic idea that the semantic meaning can be obtained by utilizing the interaction between each word in a message is somewhat more complex, but still sufficiently intuitive in the view of the author. The NCA is best understood when considering its intent. The NCA is used to ensure that the input is separable by a k-NN, hence it ensures that similar data points are near each other. The k-NN is one of the most intuitive models. The overarching concept that one categorizes the messages by looking at the *categories* of k messages which are semantically most similar will not likely lead to confusion.

The final answer to the research question, given the research done in this study, is that it is best to use Machine Translation to translate the messages into Dutch. The Dutch messages should then be fed into a fine-tuned BERTje to obtain document embeddings and predictions. The document embeddings should be transformed using NCA (with 95 components) and fed into a k-NN (with 30 neighbors). The predictions from BERTje and the k-NN should then be combined with an 80.5% reliance on BERTje and 19.5% on k-NN. This model results in the highest validation accuracy and obtains a test accuracy of 0.716. However, the performance on new *categories*, as described in Section 5.2.3, is poor. Future research could set up a more elaborate experiment to test this performance and modify the p_{k-NN} hyperparameter to modify the respective reliance on the k-NN and BERTje, to improve this performance.

6.1 Further research

Given the rise of transformer-based chatbots like OpenAI's ChatGPT and Google's Bard, one could wonder whether these generative models could be utilized for other tasks, such as classification. Recent research into zero-shot learning with ChatGPT has led to promising results. For example, Kuzman et al. show that zero-shot classification with ChatGPT is able to outperform a fine-tuned XLM-RoBERTa model (Kuzman et al. 2023). This might lead one to suppose that learning the meaning of the target *categories*, using a model such as ChatGPT, could be better than learning what sort of messages belong to a certain *category* by fine-tuning a model. It would be interesting to see if such a model would be able to correctly assign *categories* to a message, given a description of each of the *categories*. The downside of these models is that these models can, realistically, only be used via the servers of the respective companies. This would mean that the confidentiality of the messages would be compromised. An experiment such as this should therefore be done on text that does not contain any personal data. Future research could focus on the anonymization of text using, for example, Named Entity Recognition (Mamede et al. 2016).

In this research, it was found that a simple k-NN on the Term Frequency-Inverse Document Frequency performed quite well. It would be reasonable to suppose that a smaller and simpler model would perform similar to the performance of the best models that were found in this research. Future studies could investigate the exact workings of the TF-IDF k-NN to see which words are most important in the determination of the *category*. This might lead to the creation of a simple rule-based model which could give more insight into the *categories* and could, perhaps, be used as an intermediary solution for the problem this study tried to solve, as it would be trivial to implement. Furthermore, one could investigate the use of old messages for the k-NN. One could reasonably suppose that only the most recent year/month or the most recent X messages are relevant for determining the *category* of new messages.

In this study, the models were limited to BERT and variants of BERT, since BERT was released, many larger and better models have been released. One could investigate whether fine-tuning such models leads to significantly better results. These would however require dedicated hardware or access to a supercomputer to fine-tune these gigantic models.

Finally, as stated in Section 6, the model that was deemed to perform best in this study, namely Model *B4*, performed poorly on new *categories* due to a high reliance on the transformer model. In the future, one might investigate how the performance on the new *categories* can be increased while decreasing the performance on existing *categories* minimally. The major challenge here would be to design a statistically sound metric for evaluating the performance on new *categories*.

Alternatively, one could also investigate the data quality. As described in Section 5.1.3, many mistakes that the models make, are explainable. This implies that the *categories* overlap or are

not clearly defined. This would lead to incorrectly labeled data, which is detrimental for the performance of any model that tries to automate the categorization.

6.2 Practical application

When considering the applicability of this research, one should note that the models that have been created use the messages, and more importantly the *categories* of a single housing corporation. This means that the models that have been added as digital addenda are only able to categorize messages into these *categories*. While all the results in this report only relate to the data of this one housing corporation, there is no reason to assume that a similar model would work significantly worse for a different housing corporation. The performance is however highly dependent on the quality of the data. If the corporation uses vague *categories* which encompass a plethora of topics like the *category* 'Customer service' or 'Miscellaneous', the model will likely perform worse, than if the corporations used clearly defined, single topic *categories*.

The problem of vague *categories* and corporation-specific models could be prevented if all corporations were to use the same well-defined *categories*. The standardized *categories* would however lead to a loss of agency for the corporations. To prevent this, one could look further into the possibilities of separating *categories* from teams. Currently, a team is allowed to see several *categories*. One could make the *category* truly descriptive of the message and add the team, which should resolve the message, as an extra attribute to the message. To prevent extra work for the employees, one would need a new model which would assign the correct teams to the messages as one can not expect tenants to assign the right teams.

Currently, the *subject*, as described in section 1.1, functions partially as such a descriptive *category*. A tenant is now able to choose a specific *subject* when filling in a contact form, which is then mapped to a *category*. The responsibility for correctly labeling the message is therefore on the tenant. This predefined *subject* becomes irrelevant when the *category* can be assigned based on the text in the message. The author would however not recommend removing the *subject* altogether. Allowing the tenant instead to write their own *subject*, would provide the tenant with more agency and the model with a summary of the message. The model that accompanies this research does not rely on the predefined *subjects* that are currently in use and would keep working if the implementation of the *subjects* was removed or changed. If the predefined *subjects* were to be changed to free-form text, one could adjust the model to use these by appending the new free-form *subjects* to the messages and fine-tuning the model again.

The actual implementation of the model differs per model. Models *A1*, *A2*, *A3*, and *A4* only need a single message as input to return an output. As the models from the second phase (*B1*, *B2*, *B3*, and *B4*) search for a number of messages which are most similar to the input message, an extra set of messages to compare against is required. Models *B3* and *B4* use document embeddings as input for the k-Nearest Neighbors algorithm. These embeddings are obtained from the pre-trained model and consist of a vector of 768 floating-point numbers per message. It is likely most efficient to calculate and store these embeddings in a database whenever the housing corporation receives a new message. All models from the first phase and Model *B3* and *B4* use a large transformer model. This model is a large collection of matrix multiplications with 110 million parameters. This requires roughly 1.2GB to store. Given the size of the model, one would likely need a specific server or service which is able to handle the requests.

Each model and related code is added as a digital addendum. These addenda consist of the model (in one, or more files), and a file which contains all the code required to use the model (e.g. `model_A1.py`) which includes a minimal example for obtaining a prediction. A further and more inclusive description of these addenda and the list of the required Python modules and their versions are given in Appendix B and C respectively.

References

Aiken, M. (2019). An updated evaluation of google translate accuracy. *Studies in Linguistics and Literature*, *3*, p253. https://doi.org/10.22158/sll.v3n3p253

Araujo, M., Reis, J., Pereira, A., & Benevenuto, F. (2016). An evaluation of machine translation for multilingual sentence-level sentiment analysis. *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, 1140–1145. https://doi.org/10.1145/2851613.2851817

Athreya, K. B. (1987). Bootstrap of the Mean in the Infinite Variance Case. *The Annals of Statistics*, *15*(2), 724–731. https://doi.org/10.1214/aos/1176350371

Bahdanau, D., Cho, K., & Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding.

Goldberger, J., Hinton, G. E., Roweis, S., & Salakhutdinov, R. R. (2004). Neighbourhood components analysis. In L. Saul, Y. Weiss, & L. Bottou (Eds.), *Advances in neural information processing systems*. MIT Press. https://proceedings.neurips.cc/paper_files/paper/2004/file/42fe880812925e520249e808937738d2-Paper.pdf

Kalyan, K. S., Rajasekharan, A., & Sangeetha, S. (2021). Ammus : A survey of transformer-based pretrained models in natural language processing.

Kariuki, H., Mwalili, S., & Waititu, A. (2022). Dimensionality reduction of data with neighbourhood components analysis. *International Journal of Data Science and Analysis*, *8*, 72–81. https://doi.org/https://doi.org/10.11648/j.ijdsa.20220803.11

Kassner, N., & Schütze, H. (2020). BERT-kNN: Adding a kNN search component to pretrained language models for better QA. *Findings of the Association for Computational Linguistics: EMNLP 2020*, 3424–3430. https://doi.org/10.18653/v1/2020.findings-emnlp.307

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., & Lewis, M. (2020). Generalization through memorization: Nearest neighbor language models.

Khare, P., Burel, G., Maynard, D., & Alani, H. (2018). Cross-lingual classification of crisis data.

Kuzman, T., Mozetič, I., & Ljubešić, N. (2023). Chatgpt: Beginning of an end of manual linguistic data annotation? use case of automatic genre identification.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.

Mamede, N., Baptista, J., & Dias, F. (2016). Automated anonymization of text documents. *2016 IEEE Congress on Evolutionary Computation (CEC)*, 1287–1294. https://doi.org/10.1109/CEC.2016.7743936

Oostdijk, N., Reynaert, M., Hoste, V., & Schuurman, I. (2013). The construction of a 500-million-word reference corpus of contemporary written dutch. In P. Spyns & J. Odijk (Eds.), *Essential speech and language technology for dutch: Results by the stevin programme* (pp. 219–247). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-30910-6_13

Ordelman, R., de Jong, F., Hessen, A., & Hondorp, H. (2007). Twnc: A multifaceted dutch news corpus. *Analytica Chimica Acta - ANAL CHIM ACTA*.

Poncelas, A., Lohar, P., Way, A., & Hadley, J. (2020). The impact of indirect machine translation on sentiment classification.

Powers, D. M. W. (1998). Applications and explanations of Zipf's law. *New Methods in Language Processing and Computational Natural Language Learning*. https://www.aclweb.org/anthology/W98-1218

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks.

Toral, A., Gaspari, F., Naskar, S., & Way, A. (2011). Comparative evaluation of research vs. online mt systems, 13–20.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need.

Wan, X. (2009). Co-training for cross-lingual sentiment classification. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 235–243. https://aclanthology.org/P09-1027

Wikimedia Foundation. (n.d.). *Wikimedia downloads*. https://dumps.wikimedia.org

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., ... Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation.

Wu, Z., Efros, A. A., & Yu, S. X. (2018). Improving generalization via scalable neighborhood component analysis.

Yang, W., Wang, K., & Zuo, W. (2012). Fast neighborhood component analysis. *Neurocomputing*, *83*, 31–37. https://doi.org/https://doi.org/10.1016/j.neucom.2011.10.021

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books.

Appendix

A Data quality experiment

As stated in Section 5.1.3, an informal experiment has been done to determine the quality of the data and to gain a better understanding of the reasoning of the model behind the predictions. For this experiment 10 messages have been selected at random from each of 7 common mistakes model *A3* made, such as predicting 'Financial Matters' while the correct *category* is 'Service costs'. These messages have been evaluated by the author. As can be seen in Table 8, in 57 of the 70 cases, the 'true' label was correct. A further 10 messages were assigned a correct *category* by the model, while the 'true' label was wrong. However, more than half of all cases are situations where the model is truly wrong and the 'true' label is correct.

Table 8: Table with results from informal experiment regarding the quality of the data and the predictions of model *A3*.

		True *Category*			
		Correct		Incorrect	
		Prediction		Prediction	
True *Category*	Prediction	Correct	Incorrect	Correct	Incorrect
Financial Matters	Comment new room	2	8		
Financial Matters	General Renting		9	1	
General Renting	Comment new room	1	7		2
General Renting	Keys	3	2	4	1
Service costs	Financial Matters	7		3	
Nuisance	General Administration		9	1	
Service costs	General Renting	5	4	1	
Total		18	39	10	3

B Digital Addenda

This report is accompanied by several digital addenda. The Python code required for each model to run are provided in addition to the files containing the actual models and model-parts. These models/model-parts are the fine-tuned transformer model, the word frequency and TF-IDF vectorization, the NCA transformation, and the k-NN models. Only the Python code is publicly accessible, the model files are exclusively shared with the host company. The model-specific addenda are separated in different folders for each of the eight models that have been explored in this study. Each of these are such that each model can be easily utilized in isolation form the other models.

In addition to these model files, three python-files are included which are used to generate the models. These three files, `model_creation_data_preparation.py`, `model_creation_phaseA.py`, and `model_creation_phaseB.py`, contain the code required to prepare the data, create the models from the first Phase, and the second Phase respectively.

A final script (`rest_api_interface.py`) is added as an example for the usage of the models and to provide an accessible interface for the models in the form of a REST API.

These files are made accessible via GitHub:
https://github.com/RobertBrnn/TransformerKNNTextclassification.

C Product / package versies

As indicated in Section 4.4, Python was used for this project. Table 9 provides the versions of the Operating System, Python, and the Python Modules that were used for this Project. The digital addenda, as described in Appendix B, also utilizes these specific versions.

Table 9: Table with the used products and the versions of these products

Name	Type	Version
Windows	Operating System	Windows 10 Pro 22H2
Python	Programming Language	3.9.12
beautifulsoup4	Python Module	4.11.1
bertviz	Python Module	1.4.0
captum	Python Module	0.6.0
datasets	Python Module	2.10.1
difflib	Python Module	20200713
evaluate	Python Module	0.4.0
flask	Python Module	2.3.2
gensim	Python Module	4.3.1
googletrans	Python Module	3.1.0a0
matplotlib	Python Module	3.5.2
mysql.connector	Python Module	8.0.32
pandas	Python Module	1.4.2
plotnine	Python Module	0.10.1
scikit-learn (sklearn)	Python Module	1.2.2
spacy	Python Module	3.5.1
tensorboard	Python Module	2.11.0
pytorch	Python Module	2.0.0
tqdm	Python Module	4.63.0
transformers	Python Module	4.27.3