MASTER'S THESIS 2022

Topic Classification for Swedish Podcasts Using Transformers

Olof Bengtsson

ISSN 1650-2884
LU-CS-EX: 2022-44

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2022-44

Topic Classification for Swedish Podcasts Using Transformers

Ämnesklassificering för Svenska Podcasts genom Transformers

Olof Bengtsson

Topic Classification for Swedish Podcasts Using Transformers

Olof Bengtsson
ol862be-s@student.lu.se or olof-bengtsson@outlook.com

August 15, 2022

Master's thesis work carried out at Softhouse Consulting Öresund and GetReachAudio.

Supervisors: Kerstin Johnsson, kerstin.johnsson@softhouse.se
Pierre Nugues, pierre.nugues@cs.lth.se

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Many people listen to podcasts on a weekly basis, and they have grown much in popularity in recent years. However, compared to traditional broadcast mediums such as television and radio, there are very few tools available for analyzing podcasts. In this Master's thesis, I employed different machine learning models to classify the topics covered in podcast episodes, which would facilitate the analysis. The dataset I used consisted of texts collected from Swedish online forums and spanned 65 different topics. After training and evaluating the models on the forum dataset, the best one reached an F1 score of 0.72, which consisted of a pre-trained Swedish BERT model and a logistic regression model head. This model outperformed the TF-IDF baseline, which obtained an F1 score of 0.66. The final evaluation was then done on an excerpt of the podcast transcriptions, where the transformer model obtained an accuracy of at least 0.46, outperforming the TF-IDF baseline by 75%.

Keywords: Topic classification, transformers, podcasts, machine learning, natural language processing

Acknowledgements

Firstly I would like to thank Softhouse Consulting and GetReachAudio for making this interesting project possible and showing great interest as well as providing me with all the necessary tools to be able to carry out this project. Furthermore I would like to give special thanks to my supervisors, Pierre Nugues at LTH, and Kerstin Johnsson at Softhouse Consulting for sharing their expertise in the subject and always being supportive and enthusiastic.

Contents

Chapter 1
Background and motivation

In the recent years, the consumption of podcasts has increased substantially across the world. Podcasts are a large source of entertainment and many people listen to multiple podcast series every week. However, compared to older broadcast mediums, such as television and radio, content creators have access to far less analysis tools to help them know what they are doing right, what they are doing wrong, but also to know what kind of companies they could approach for paid promotion.

GetReachAudio is a company which collects data on consumer behavior for different kinds of streamed media, such as podcasts. They then use this data to help the creators and producers better understand what they should do to reach out to a wider audience. One powerful tool in this would be to analyze the episodes and classify which topics that are covered as well as mapping when they are covered in the episodes. Combining this analysis with consumer data from the distributor, one could see if there are any topics that result in few or many listeners. This information could then be used to create content that attracts more listeners and identify what kind of companies could be interested in promoting themselves on the podcast.

At the moment of writing this thesis, to the best of my knowledge, there is no such algorithm available for creators, and even though it is possible to do the topic classification manually, it is not a feasible task. Instead, machine learning could be used to train a model to recognize which topics are covered.

In this Master's thesis, I used natural language processing and machine learning to predict when a topic is covered in individual podcast episodes. This was done in collaboration with both GetReachAudio and Softhouse Consulting Öresund, which is a software consulting firm located in Malmö. I used the podcast *Dumma Människor* (eng. *Dumb People*) produced by Clara Wallin with Lina Thomsgård and Björn Hedensjö as hosts to test the classification models. Though the ultimate goal is to use the predicted topics to help creators in improving their podcast, this is outside scope of the thesis and I only focused on the topic prediction.

To train the final topic classifiers, I used pre-trained transformers and fine-tuned them on two corpora: The Swedish Reuters corpus and texts from online Swedish forums. As results,

Swedish BERT, trained on Swedish texts, yielded the best performance.

I used this model to categorize the podcasts from *Dumma Människor*, where I first transcribed speech from all available podcast episodes into text using publicly available natural language processing tools from Google Cloud Platform. Finally, I carried out a small-scale evaluation of the models on about one hundred test segments from the podcast transcriptions.

1.1 Contributions

In this Master's thesis, I created a classifier to determine the topics of a series of podcast episodes. I trained the model on data collected from Swedish online forums, spanning 65 different topics.

1. I first created a baseline model by using a TF-IDF representation of the texts and a logistic regression head for classification.

2. The model was then expanded by using transformers to generate contextual embeddings of the texts, which increased the F1 score from 0.66 to 0.71 and 0.72 to 0.75 for an ordered and a shuffled sampling of the same training dataset.

3. Thereafter, I experimented with a small feed-forward neural network instead of logistic regression as a model head. This yielded better results for the training dataset, reaching an F1 score of 0.72 and 0.77. However, this proved to have lower performance on the podcast dataset.

4. I also applied a fine-tuning procedure to the Swedish and the multilingual BERT model by unfreezing one layer. This resulted in the same f1 score for the Swedish BERT and 0.63 F1 score for the multilingual BERT.

Manual evaluation on podcast transcriptions showed that the best model was the one that used Swedish BERT without fine-tuning and a logistic regression model head. This model reached an accuracy of at least 0.46 on the podcast data.

1.2 Related work

The field of natural language processing is constantly advancing and growing. One big contributing factor to this is the vast access to written text data from social media and Wikipedia for example. This makes it easy to collect massive datasets to train powerful natural language processing models. However, as the access to massive texts increases, the need for models to handle texts grows as well. The applications for these kinds of models range from text classification and sentiment analysis to text summarization, translation, and question answering.

In a recent report, Dhar et al. (2021) discuss and surveys the applications of a number of text mining tasks, one of which being topic classification. Other fields covered in the report are sentiment analysis, text summarization, question and answering systems and filtering of documents.

Ikonomakis et al. (2005) discuss in an older report using machine learning for text classification. In this they find that the data used to train the models is very important and can have a large impact in how accurate the model is. A drawback with the models they evaluated was that they could not handle homonyms properly, i.e they did not differentiate between words that were spelled the same but had different meanings.

However, there are currently models which remedy the problem with homonyms. One recent approach proposed by Vaswani et al. (2017) is the transformer architecture pre-trained on very large corpora. These transformers result in large models with up to 340M parameters. They are very powerful, outperforming the state-of-the-art in several of the NLP fields such as translation, question, answering, and categorization. One powerful feature of the models is that they generate text embeddings which consider the context rather than simply looking at every individual word.

One of the most successful transformer-based model is BERT (Devlin et al., 2018). Initially, BERT was trained on English Wikipedia articles. Then BERT was extended to a multilingual version with a training corpus including other languages. The National Library of Sweden (*Kungliga Biblioteket*) has developed a Swedish version of BERT which is publicly available and free to use for a range of natural language processing tasks.

Thompson and Mimno (2020) built a model using BERT for topic modeling. They then compared the transformer-based approach to more conventional methods, such as *Latent Dirichlet Allocation*, or *LDA*. The results showed them that BERT produced topic clusters as good as, or better than conventional methods.

Remmer et al. (2021) describes a topic classifier for Swedish using transformers and is close to our thesis. They focus on the *international classification of diseases* or ICD codes, which is a way of recording the diagnoses of patients. In their paper, Remmer et al. (2021) use Swedish BERT to train a model to classify the ICD code from a written medical journal. They compared this to baseline models, which were $\boldsymbol{K}$-nearest neighbors, support vector machines, and decision trees. They found that Swedish BERT performed better than the baseline when the ICD codes were grouped into 10 categories instead of using all 263, reaching a micro F1 score of 0.80 and a macro F1 score of 0.58. However, Swedish BERT was found to have very poor performance, with a micro and macro F1 score of 0 when considering all 263 codes simultaneously.

There have also been approaches to train cross-lingual classifiers where a model trained on one language is transferred to another language. In a recent report, Schwenk and Li (2018) propose a certain dataset to train and evaluate such models. The dataset is derived from a collection of annotated news articles from the news agency *Reuters* and has balanced class prior probabilities. The original dataset, released by Lewis et al. (2004), is labeled with one or more label per text and can be used for individual languages as well. Among these languages, Swedish is included.

Chapter 2
Dataset

For this thesis, I used three datasets: one for the initial testing, one for training, and the last one for the real world application test. The initial testing dataset consisted of annotated news articles in Swedish, collected from the news agency Reuters. The training dataset consisted of texts collected from Swedish online forums `flashback.org` and `familjeliv.se`, where the texts were annotated with their corresponding thread topic. The last dataset, used for real world testing, consisted of transcriptions of 109 podcast episodes. Originally, this dataset was unlabeled and I used it as test set. I applied the different models to it and experts from GetReachAudio carried out a qualitative evaluation of the automatic annotation.

2.1 Reuters corpus

The Swedish Reuters Text Categorization Collection Data Set is a corpus from *Reuters* news agency. This dataset is part of a large multilingual dataset consisting of over 487,000 Reuters news articles in 13 different languages. The articles were written between the 20th of August 1996 and the 19th of August 1997 and released to the public to be used for natural language processing, machine learning, and information retrieval systems in 2005. Apart from Swedish, the full dataset also contains articles in Danish, Chinese, German and Spanish to name a few languages. In Swedish, there are 15,731 news articles annotated with at least one topic out of 1435 possible topics.

Furthermore, the topics are not evenly distributed: A few ones appear over one thousand times while many only appear once. Figure 2.1 below shows that a lot of topics only appear 100 times or less. Figure 2.2 is the same histogram, but excluding all topics appearing less than 200 times in the dataset.

This corpus is interesting as it is widely available and thus enables performance comparisons. I carried out the initial testing before collecting the final training dataset, enabling me to determine how many samples were required to correctly identify a topic for a base model. This dataset did not need any processing since it was already labeled and divided into

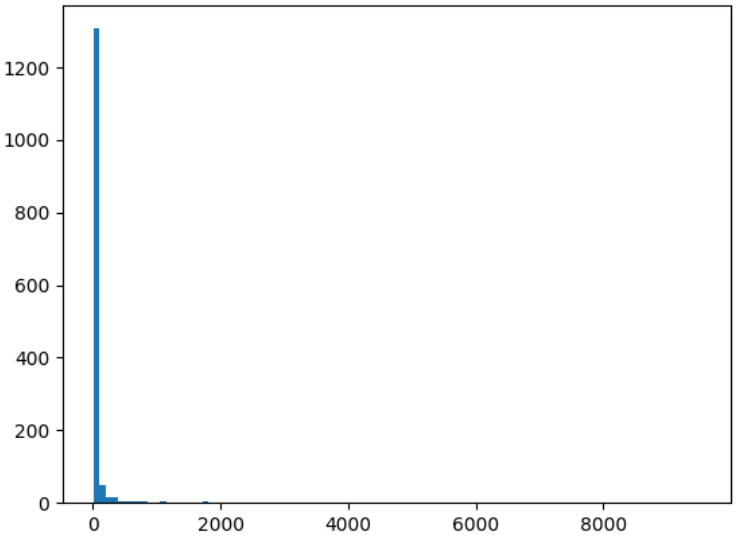

Figure 2.1: Histogram of all topic frequencies in the Swedish Reuters dataset.

separate news articles.

Below is an example text from the dataset which was annotated with six topics, ORSDN, 181502, IINV, C15, C151 and CCAT. The topics correspond to business categories. For example CCAT means *Corporate/Industrial* (Lewis et al., 2004).

> STOCKHOLM, 6 dec (Reuter) - Öresunds substansvärde justerat för utdelning uppgick den 30 november till 190 kr per aktie, innebärande en ökning med 29% sedan årsskiftet. Det skriver Öresund i ett pressmeddelande. Investmentbolagsrabatten var den 30 november 20% på substansen.
> 'STOCKHOLM, 6 Dec (Reuter) - Öresund's net asset value adjusted for dividends on November 30 amounted to SEK 190 per share, implying an increase of 29% since the turn of the year. Öresund writes this in a press release. The investment company discount was 20% on the substance on November 30.'

2.2 Corpus of Forum Posts

`flashback.org` and `familjeliv.se` are two different online forums in Swedish. They are available for download from *Språkbanken*. Since the podcast data was unlabeled and could not be used to train a classifier, I used this as training dataset for the topic classifier. The language in the podcasts is quite informal and these forum texts with labeled data are quite similar to that of *Dumma Människor*.

Since both websites had forums and threads spanning a wide range of topics, I could collect several thousand sentences over 65 different topics. Even though the forums had many more potential topics, I only used the topics which could be interesting for the podcast. See the appendix for the full list of collected categories .

Below is an example text from the *Fysik, matematik och teknologi* (*eng. Physics, mathematics and technology*) category.

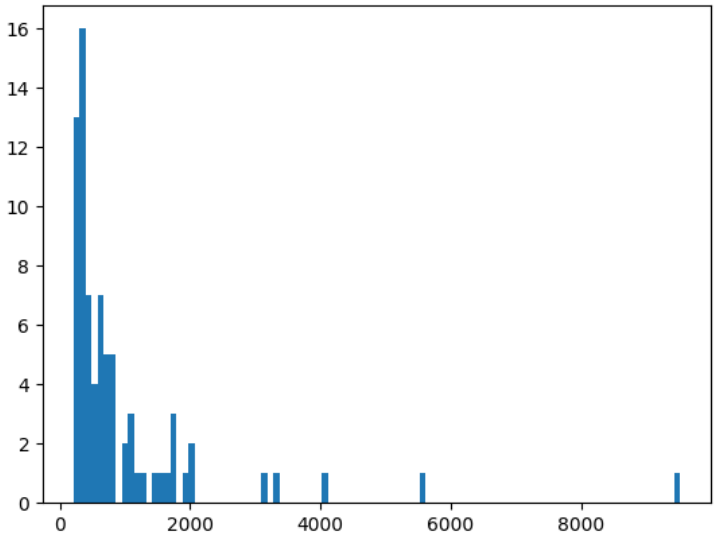

Figure 2.2: Histogram of topic frequencies given the most common topics in the Swedish Reuters dataset.

> det har iaf varit en rolig diskussion vilket då är perfekt, för då får man fart åt den riktningen genom att andas ut i sådana fall. så bara att stå och andas i motsatt riktning dit man skall kommer tillslut göra att man kommer fram, eftersom man " kastar " vatten och kol. det där med att kasta saker är väl rätt lösning men med en stor damm kan det ta sin lilla tid. men när du andas in motverkas inte kraften? befinner sig mitt ute på en isbelagd damm. kan det vara svaret? ja det tycker jag.
> 'it has at least been a fun discussion which is perfect, then you can get speed in that direction by breathing out in that case. just standing and breathing in the opposite direction of where you're going will eventually get you there, since you "throw" water and carbon. I guess the thing with throwing things is the correct solution, but with a large pond this could take quite a while. but when you breathe in doesn't the force counteract? Being located in the middle of an ice covered pond. could that be the answer? Yes I think so.'

Another example text from the *Medicin och Hälsa* (*eng. Medicine and health*) category can be seen below.

> det gör att kroppen får en dunderdos över det normala, alltså mer den normala produktionen. har du ens någon erfarenhet av detta kliniskt? prolaktinnivåerna kan t.ex. ses bli förhöjda. circadin har modifierad frisättning, utsöndring av melatonin sker kontinuerligt under nattens gång. de är inte restriktiva, trodde det var första alternativet för alla som har svårt att somna. har funderat lite på att beställa över nätet men vet inte om jag vågar riktigt. då passar ett melatoninpreparat med
> 'it gives the body a thunderstorm above normal, ie more than the normal production. Do you even have any experience with this clinically? prolactin levels can e.g. be seen to be elevated. circadin has modified release, secretion of melatonin occurs continuously during the night. they are not restrictive, I thought it was the first option for anyone who has difficulty falling asleep. I have thought

a bit about ordering online but do not know if I really dare. then a melatonin preparation fits too'

To create the training dataset, I extracted sentences from each forum. Since an individual post could consist of just a few words, resulting in little meaning, I concluded that a post would not constitute an individual text. Instead, I added posts from the same thread to each other until the text reached the maximum limit of 512 characters, since this was the maximum input length for the transformer models. This would hopefully create more context and meaning to each text used to train the model. Hopefully this is clear in the quotes above. I then labeled each text with the forum in which it was posted. Figure 2.3 shows a histogram of the sequence lengths in the training dataset.

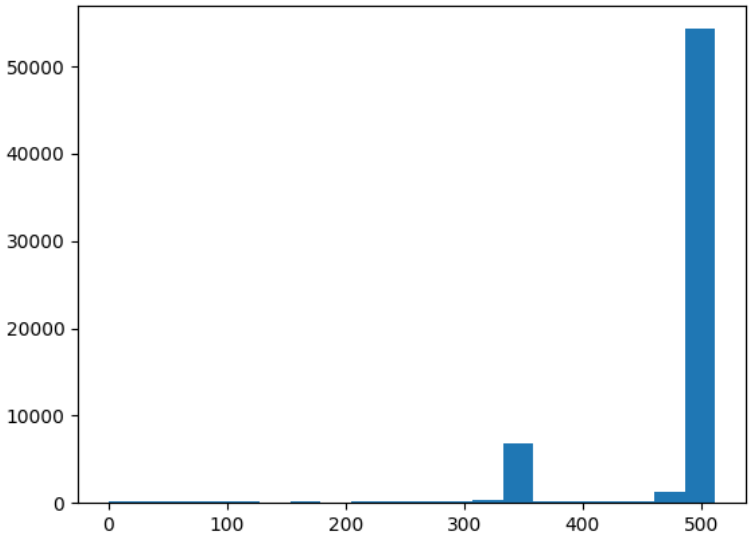

Figure 2.3: Histogram of the sequence lengths in the training dataset.

In the preprocessing step, I lowercased all the texts. The use of the uppercase letters varied across the posts and some contained words written all in capital letters. It would have been possible to use language technology techniques to make all sentences start with a capital letter to restore a "normal" writing style. This would have been done by making all letters after a period followed by a whitespace upper case. However, since the posts are not manually checked for periods and other punctuation signs, I could not take for granted that replacing all such occurrences would not lead to errors for other words. Because of this I kept the texts as they were. In other words, there were possible sources for errors, so I decided that having the text all in lower case was the best option.

The full resulting dataset consisted of about 10,000 text entries from each topic. Because I added texts from the same thread to each other, this meant that the texts from the same thread would be clustered close to each other in the list of texts. Because of the order, I trained two models. One model was trained on the first n texts from each topic and another one was trained on n random texts from each topic. I did this to see whether it made a difference if the model was trained on some data from all threads or all data from some threads. Here $n = 1000$ was used, which is why the histogram in Figure 2.3 only contains 65,000 texts.

2.3 *Dumma Människor* Podcast Corpus

I used the podcast dataset for the real-life application test. It consisted of unlabeled transcriptions from the first 109 full length episodes of the podcast *Dumma Människor*.

The text below is a short excerpt of the 109th episode of the *Dumma Människor* podcast called *Det underbara lidandet* (*eng. The wonderful suffering*) released 2022-02-09.

> Har du någon gång träffat en människa som inte verkar ha några problem som inte verkar ha något lidande i sitt liv du kan sluta vara avundsjuk på den personen i detta nu för att den personen befinner sig garanterat i något som väldigt mycket liknar helvetet
> 'Have you ever met a person who does not seem to have any problems who does not seem to have any suffering in his life you can stop being jealous of that person right now because that person is guaranteed to be in something very similar to hell'

When I started my thesis, these were the only available full length episodes. The podcast however consisted of additional shorter episodes which served as samples to the full length episodes, but these were not included in the dataset.

GetReachAudio provided MP3 files of all the episodes. I used the speech-to-text transcription tool in Google Cloud Platform to generate the transcriptions. The files generated from the Google Cloud Platform service returned the transcriptions in smaller chunks. Within the chunks, every word was given a timestamp. I used these timestamps to assign start and end times for the chunks. This was to make it easier to track the topics in the episodes.

As before, every transcription segment was made sure to not be longer than 512 characters, and if this was the case I truncated the text at the first blank space before the 350th character. The remaining characters then formed a new text segment, and the start and end times were updated. Similarly as for the other datasets, Figure 2.4 shows a histogram of the sequence lengths.

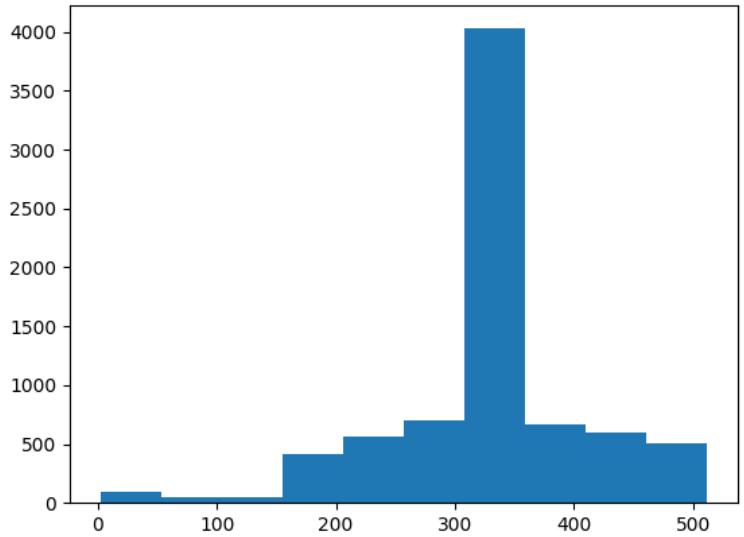

Figure 2.4: Histogram of the sequence lengths in the podcast dataset.

Since there were two people talking in the podcast, often at the same time, the transcription was not perfect and would occasionally miss a few seconds of speech. It was however deemed that the transcriptions captured the essence of what the speakers said.

When listening to the podcast on streaming platforms, one notices that the episodes start with a short segment from sponsors of the podcast. These sponsors might vary, which means that the segments can be different depending on when someone is listening to the episode. This was no problem however, since the opening segment was not included in any MP3 file and therefore not included in the transcriptions.

Chapter 3
Approach

This chapter describes the necessary concepts and components in order to understand all the aspects of this thesis. I start first with techniques to encode the word input and category output into vectors. In the input encoding, I describe two techniques, bag-of-words and subword segmentation, the latter being used by transformers. Then I move on to the description of the learning architectures, logistic regression, neural networks and transformers, and finally the metrics I used to evaluate the performance as well as data stratification.

3.1 Input/Output Formatting

3.1.1 Sentence Tokenization

Bag-of-words. Bag-of-words is a simple tokenization method in which all individual words in a dataset constitute a token. The implementation used in this thesis does not differentiate between higher and lower case, i.e *Sweden* and *sweden* are mapped to the same token. The words constitute the vocabulary, and are mapped to an individual index, where the vocabulary can be very large for large datasets (Nugues, 2014). For each text, the frequency of every word in the text is recorded and used in the vectorization. A short example is shown below.

Text: *Excuse me, could you help me find the house key?*
Tokenization: [excuse:1, me:2, could:1, you:1, help:1, find:1, the:1, house:1, key:1]

The tokens are then mapped to indices corresponding to the word's position in the vocabulary. This tokenization method does not take multiple meanings of the same word into consideration, since a word always is mapped to the same index. Because of this, in a sentence like

Text: *The key to success is communication*

the word *key* would be mapped to the same index as *key* in the first sentence, even though the meaning of the word is different.

WordPiece Another tokenization method is *WordPiece* proposed by Wu et al. (2016). In this method, the words in a sentence are split up into sub-words, or wordpieces, until all sub-words can be found in a defined vocabulary. This tokenization technique is used in BERT models, where the English BERT base model uses a vocabulary with 30,000 tokens. According to Wu et al. (2016) a vocabulary with 8,000 to 32,000 wordpieces achieves both good accuracy and fast decoding. Similarly as for the bag-of-words the tokens are mapped to indices which are used as input to the a model. Below is the tokenization of the sentence

> Hej och välkomna till podcasten Dumma Människor!
> "Hello and welcome to the podcast Dumb People!"

for the tokenizers used in both the multilingual BERT base model and the Swedish BERT base model.

Multilingual BERT:

> He, ##j, och, v, ##äl, ##kom, ##na, till, pod, ##cast, ##en, Dum, ##ma, M, ##än, ##nisk, ##or, !

Swedish BERT:

> Hej, och, välkomna, till, podcast, ##en, Dum, ##ma, Människor, !

After this, the tokens would be mapped to their corresponding index. Here it is clear to see that the tokenizer for Swedish BERT contains more Swedish words compared to the multilingual one, which is reasonable. We can also see that the word *Dumma* is split into *Dum* and *ma*, for both versions. This means that the vocabulary does not have to contain different inflections of the same word, but prefixes and suffixes are stored to account for this.

TF-IDF TF-IDF representation, or term frequency inverted document frequency, is a way to vectorize texts in a collection of documents. This is a more sophisticated version of bag-of-words where the term frequency part constitutes the frequency of a word in a specific document and the inverted document frequency is the total number of documents divided by how many documents the word is found in. In some cases the logarithm of the inverse document frequency is used. The TF-IDF value is then obtained by multiplying the two parts together. This gives a numerical representation of words which can take into account which words are more important in which documents (Nugues, 2014).

In this thesis, a TF-IDF vectorizer was imported from the Python library *scikit learn*, which has slightly different implementation compared to the description above. There were furthermore several settings which could be used when vectorizing a collection of documents, but since the TF-IDF vectorization was to be used in an initial model, it was deemed that the default settings were sufficient. Then the difference was simply that the implementation

added 1 to both the numerator and denominator in the IDF value as well as adding 1 after computing its logarithm, resulting in the formula below

$$\textit{TF-IDF} = \boldsymbol{TF} \cdot [\log(\frac{1+n}{1+df(t)}) + 1], \tag{3.1}$$

where n is the number of documents and $df(t)$ how many documents term t occurs in. Each sentence, which would constitute a row in the matrix representation, was then normalized to have length 1, i.e the $L2$ norm equal to 1. The words were then ordered alphabetically and capital letters were ignored.

Using the following two Swedish documents as example

Document 1: *Jag har alltid velat vara bra på att spela fotboll.*
eng. I have always wanted to be good at playing football.
Document 2: *Vi gick alltid och kollade på bio tillsammans.*
eng. We always went to watch a movie together.

we get a total vocabulary of 16 words. The first document, or sentence, has 10 individual words which gives every word there a term frequency of 1, and 0 for all other words in the vocabulary. Next, using equation 3.1, the inverse document frequency is $\log(3/2)+1 = 1.4$ for *jag*, *har*, *velat*, *vara*, *bra*, *att*, *spela* and *fotboll* since there are 2 document and they only appear in 1 document. For the words that occur in both documents, i.e *alltid* and *på* we instead get $\log(3/3) + 1 = 1$. After normalization this leads to the vector representation

$$\begin{bmatrix} \textit{alltid} & 0.24 \\ \textit{att} & 0.33 \\ \textit{bio} & 0 \\ \textit{bra} & 0.33 \\ \textit{fotboll} & 0.33 \\ \textit{gick} & 0 \\ \textit{har} & 0.33 \\ \textit{jag} & 0.33 \\ \textit{kollade} & 0 \\ \textit{och} & 0 \\ \textit{på} & 0.24 \\ \textit{spela} & 0.33 \\ \textit{tillsammans} & 0 \\ \textit{vara} & 0.33 \\ \textit{velat} & 0.33 \\ \textit{vi} & 0 \end{bmatrix}^{\mathsf{T}}$$

3.1.2 Output encodings

For classification tasks with k different possible targets, the target of a sample should be one hot encoded. This means that a target is mapped to an index and then encoded with a $1 \times k$ vector containing a 1 on the index corresponding to that class and 0 everywhere else. As an

example the three classes *family*, *sport* and *money* would be mapped to 0, 1 and 2 respectively. The one hot encoded vectors for each class would then be

$$\begin{aligned} \text{family} &\rightarrow \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \\ \text{sport} &\rightarrow \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \\ \text{money} &\rightarrow \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{3.2}$$

3.2 Logistic Regression

Logistic regression is a rudimentary machine learning algorithm that fits a linear curve to the logistic function of the odds that a sample belongs to one specific class (Agresti, 2018). See the equation below.

$$\log(\frac{p}{1-p}) = \beta_0 + \beta_1 x_1 + ... + \beta_m x_m = \mathbf{x} \cdot \beta \tag{3.3}$$

In Equation 3.3 above, p is the probability that a sample belongs to a certain class. Since all texts used in this thesis are assigned to one class, the classification is binary and the probabilities used during training are simply either 1 or 0. $\mathbf{x}$ is a row vector with m entries containing the features of the data and the column vector β of the same size holds the trainable weights which constitute the logistic model (Agresti, 2018). One can then expand this to a dataset containing n data points as

$$\log(\frac{\mathbf{p}}{1-\mathbf{p}}) = \mathbf{X}\beta \tag{3.4}$$

where $\mathbf{p}$ instead is an n sized column vector containing the probabilities for each data point and $\mathbf{X}$ is an $n \times m$ feature matrix.

Looking at Equation 3.4 one can see that the left hand side tends to $\pm\infty$ as the probability approaches 1 and 0. The expression can however be transformed as

$$\log(\frac{\mathbf{p}}{1-\mathbf{p}}) = \mathbf{X}\beta \Leftrightarrow \frac{\mathbf{p}}{1-\mathbf{p}} = e^{\mathbf{X}\beta} \Leftrightarrow \mathbf{p} = \frac{1}{1+e^{-\mathbf{X}\beta}} \tag{3.5}$$

where the probability p can take any value between 0 and 1. This equation produces the characteristic s-shaped logistic curve, see figure 3.1. The model is finally fitted by defining a loss function and optimizing the model weights with respect to it.

3.3 Loss functions

A loss, or error function is used to measure the performance of a model, so that an optimizer can update the model weights during training. The loss is defined as a function of the model weights and needs to be differentiable in order for an optimizer to find the optimal weights. To measure the final performance of a model, other techniques are used. See section *Performance metrics*.

For a regression problem the error function could be mean squared error, which computes the average Euclidean distance squared from each data point to the fitted line (Ohlsson and

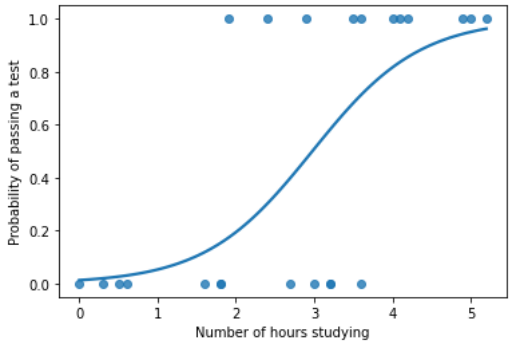

Figure 3.1: Example of logistic curve. Here the graph describes the chance of passing an exam as a function of how many hours of studying.

Edén, 2020). In this thesis however, it is suitable to use crossentropy error, since we are doing classification. This is defined as

$$E(\boldsymbol{\omega}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{c} d_{ni} ln(y_{ni}) \tag{3.6}$$

given a set of weights $\boldsymbol{\omega}$, total number of samples N and number of classes c. y_{ni} is the network output from node i, in this case output i from equation 3.18, given the input from pattern n. The target d_n is a one hot encoded vector, which means that d_{ni} takes the values

$$d_{ni} = \begin{cases} 1 & \text{if input } n \text{ belongs to class } i \\ 0 & \text{if input } n \text{ does not belong to class } i \end{cases}$$

In Equation 3.6 it is clear to see that if $d_{ni} = 1$ and the returned probability for class i, y_{ni}, is close to zero, the loss will be large.

3.4 Optimizing the Weights

With a loss function defined, the optimal model is found by minimizing the loss with respect to the model parameters. There are a several methods available for this kind of problem, but some of the most common ones are, *stochastic gradient descent*, *LBFGS* and *Adam*.

3.4.1 Stochastic Gradient Descent

For every iteration, or epoch, in stochastic gradient descent the training data is stochastically split into minibatches of size P. For every minibatch, the gradient of the error function E_p with respect to the model weight ω is then calculated, whereupon the model weights are updated by taking a step in the opposite direction of the gradient. The reason for this approach is that it reduces the risk of ending up at a local minimum, since the minibatches are different every time. Even though the individual minibatches differ, all samples in the

training data are used before starting a new epoch. Equation 3.7 below shows how the model weight are updated for a learning rate η (Ohlsson and Edén, 2020).

$$\Delta\omega_k = \frac{1}{P}\sum_{p=1}^{P}\Delta\omega_{pk}, \qquad \delta\omega_{pk} = -\eta\frac{\nabla E_p}{\nabla\omega_k} \tag{3.7}$$

3.4.2 LBFGS

LBFGS, where BFGS stands for *Broyden-Fletcher-Goldfarb-Shanno* and the "L" stands for "limited memory", is a quasi Newton optimization method. In a regular Newton optimization method one finds the optimal point $\mathbf{x}_{opt}$ by starting at an initial point $\mathbf{x}_0$ and taking steps as

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}^{-1}\mathbf{g} \tag{3.8}$$

until the solution has converged. In Equation 3.8 above $\mathbf{H}$ is the Hessian and $\mathbf{g}$ is the gradient of the loss function with respect to the model weights ω. It is however cumbersome to compute the inverse of the Hessian which makes Newton methods ineffective, especially if the input data has many dimensions. Quasi Newton methods solve this by approximating the inverse Hessian instead. The approximation $\mathbf{G}_k$, after k iterations is given by the update scheme (Ohlsson and Edén, 2020)

$$\mathbf{G}_{k+1} = (\mathbf{1} - \frac{\mathbf{v}\mathbf{p}^T}{\mathbf{p}^T\mathbf{v}})\mathbf{G}_k(\mathbf{1} - \frac{\mathbf{p}\mathbf{v}^T}{\mathbf{p}^T\mathbf{v}}) + \frac{\mathbf{v}\mathbf{v}^T}{\mathbf{p}^T\mathbf{v}} \tag{3.9}$$

where the variables $\mathbf{p}$ and $\mathbf{v}$ are given by

$$\mathbf{p} = \omega_{k+1} - \omega_k$$
$$\mathbf{v} = \mathbf{g}_{k+1} - \mathbf{g}_k$$

The next point is then calculated as

$$\omega_{k+1} = \omega_k + \alpha_k\mathbf{G}_k\mathbf{g}_k$$

where the parameter α_k is determined by solving the one-dimensional minimization problem (Ohlsson and Edén, 2020)

$$\alpha_k = \min_\alpha \omega_k + \alpha\mathbf{G}_k\mathbf{g}_k \tag{3.10}$$

"Limited memory" in the algorithm refers to that only a certain number of $\mathbf{p}$ and $\mathbf{v}$ vectors are stored at the same time. This naturally results in less memory allocation (Nocedal and Wright, 2006).

This optimizer was used when fitting the logistic regression model.

3.4.3 Adam

The *Adaptive moment estimation* or Adam is another optimization method but works quite differently compared to LBFGS. This is a method which uses a running average of both the gradient and the square of gradients. These running averages are stored in two vectors **m** and **v** respectively, which are given by the following expressions (Ohlsson and Edén, 2020)

$$m_{k+1}^i = \beta_1 m_k^i + (1-\beta_1) g_k^i$$
$$v_{k+1}^i = \beta_2 v_k^i + (1-\beta_2)[g_k^i]^2$$

Using

$$\hat{m}_{k+1}^i = \frac{m_k^i}{1-\beta_1^k} \tag{3.11}$$

$$\hat{v}_{k+1}^i = \frac{v_k^i}{1-\beta_2^k} \tag{3.12}$$

the resulting update algorithm of the weights ω^i is expressed as

$$\omega_{k+1}^i = \omega_k^i - \eta \cdot \frac{\hat{m}_{k+1}^i}{\sqrt{\hat{v}_{k+1}^i} + \epsilon}, \tag{3.13}$$

where i is the index of the weight, η, β_1 and β_2 are the learning parameters of the algorithm and ϵ is a small value in order to avoid division by zero in the first iteration (Ohlsson and Edén, 2020).

Expanding the expression for $\hat{m}_{k+1}^i$ for small k we get

$$\hat{m}_2^i = \frac{m_2^i}{1-\beta_1^1} = \frac{\beta_1 m_1^i + (1-\beta_1) g_1^i}{1-\beta_1^1} = \frac{\beta_1(1-\beta_1) g_0^i + (1-\beta_1) g_1^i}{1-\beta_1^1} = \frac{\beta_1 g_0^i + g_i^i}{1}$$
$$\hat{m}_3^i = \frac{m_3^i}{1-\beta_1^2} = \frac{\beta_1(\beta_1(1-\beta_1) g_0^i + (1-\beta_1) g_1^i) + (1-\beta_1) g_2^i}{1-\beta_1^2} = \frac{\beta_1^2 g_0^i + \beta_1 g_1^i + g_2^i}{1+\beta_1^2}$$

from which we clearly can see that $\hat{m}_{k+1}^i$ can be expressed as

$$\hat{m}_{k+1}^i = \frac{g_k^i + \beta_1 g_{k-1}^i + \beta_1^2 g_{k-2}^i + \ldots}{1 + \beta_1 + \beta_1^2 + \ldots} \tag{3.14}$$

which is a weighted average of the gradients g. Similarly we obtain the following expression for $\hat{v}_{k+1}^i$

$$\hat{v}_{k+1}^i = \frac{[g_k^i]^2 + \beta_2 [g_{k-1}^i]^2 + \beta_2^2 [g_{k-2}^i]^2 + \ldots}{1 + \beta_2 + \beta_2^2 + \ldots} \tag{3.15}$$

which is a weighted average of the gradients squared. Using $\beta_1, \beta_2 < 0$ it is also ensured that the most current gradient has the largest factor, and that the values do not explode (Ohlsson and Edén, 2020). This optimizer was used in all models which used a pre-trained transformer.

3.5 Feed-forward Neural Networks

A feed-forward neural network is a network where the input goes through one or several layers of nodes (weights) and the output from each node in a layer is used as input to the nodes in the next layer. This means that there can be no feedback loops. The smallest feed-forward neural network has only one layer with one node and can be used to map linear dependencies (Ohlsson and Edén, 2020). Looking at the model in Figure 3.2 below, we see that there are two inputs to the output node, the variable **x** and the bias, where the bias is built into the model and does not have to be manually put in. The total output from the model is given by Equation 3.16.

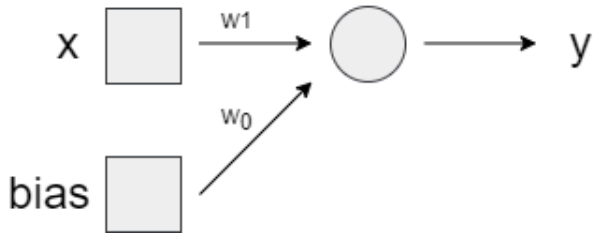

Figure 3.2: Small feed-forward neural with one layer and one node.

$$y = w_0 \cdot \text{bias} + w_1 \cdot x \tag{3.16}$$

Given the values in Table 3.1 and bias set to **1**, the small model would find the optimal values for the parameters w_0 and x_1 according to Table 3.2

x	y
0	2
2	3

Table 3.1: Example values for the small model in Figure 3.2

w_0	w_1
2	0.5

Table 3.2: Optimal fit of parameters w_0 and w_1 in figure 3.2

The neural networks can however be much larger with several layers and nodes to capture more complex patterns. In the illustration above, the model was used for regression, but it could similarly be used for classification. Using the previous model as an example one could say that all samples above the fitted line would belong to a class **1** and all samples below the line would belong to another class **2**. A feed-forward neural network is optimized using the same principle as for logistic regression. In this thesis I used feed-forwards neural networks as model heads after the pre-trained transformer in the model architecture. I addition to this, feed-forward neural networks are also used inside the transformer models. To optimize these models, I used the Adam optimizer.

Layers

There are several different kinds of layers that can be deployed in a neural network, where the application dictates which ones to use.

One of the most widely used layers in a neural network is a fully connected layer. This is a layer where all the nodes are connected to every node in the previous and the next layer. In other words, this means that every output from the previous layer is used as input to every node and the output of a node is used as input to all nodes in the next layer (Ohlsson and Edén, 2020). The architecture of a fully connected layer is shown in Figure 3.3.

In order to reduce over training one can insert a dropout layer in between two layers, which takes the probability of a weight in the next layer not being updated. By not updating some weights, the model is less prone to only recognizing the training data, and hence generalizes better (Ohlsson and Edén, 2020).

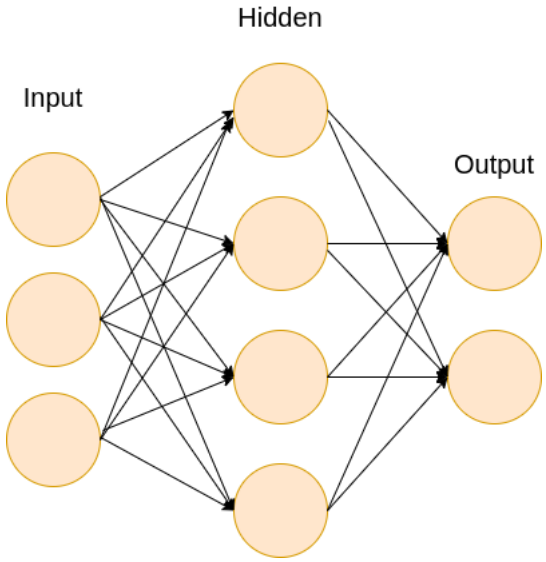

Figure 3.3: Overall architecture of fully connected layers.

Activation functions

In order to model non-linear behaviour the nodes in a layer can have activation functions, which transforms the output using non-linear functions (Ohlsson and Edén, 2020). List 3.5 below includes the most common choices, see Figure 3.4 for graphical representations of the functions.

Rectified linear unit (ReLU): $f(x) = \max(0, x)$

Sigmoid: $f(x) = \dfrac{1}{1 + e^{-x}}$

Hyperbolic tangent: $f(x) = \tanh(x) = \dfrac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$

Using the small network in Figure 3.2 as an example, I apply an activation function f to the output node. The new output y_{act} is then given by Equation 3.17 below.

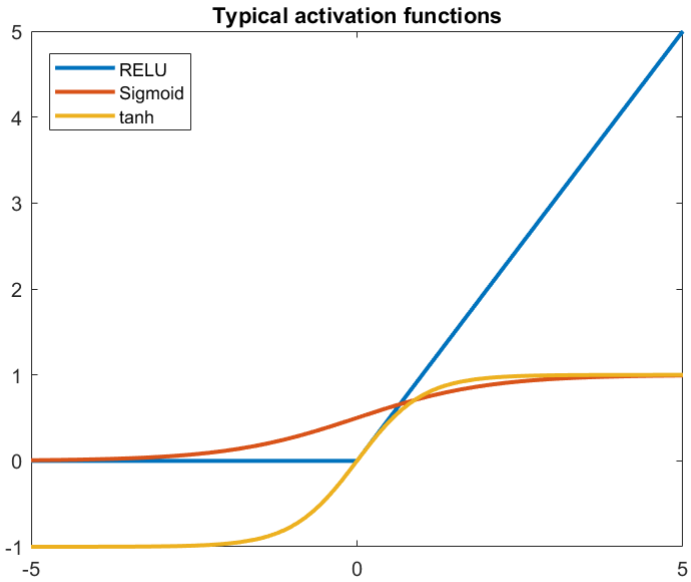

Figure 3.4: Plots of the activation functions listed in List 3.5

$$y_{act} = f(w_0 \cdot \text{bias} + w_1 \cdot x) \tag{3.17}$$

In neural networks used for classification, the softmax activation function is often used in the output layer, see equation 3.18 below. This is a natural choice since the function normalizes the output values to be between 0 and 1 and also add up to 1. The output values for each index then correspond to the probability of a data point belonging to the class of that index (Ohlsson and Edén, 2020). The predicted class is then found by taking the argmax of the returned vector.

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{3.18}$$

In the equation above x_i is the output from node i in the last layer and n the number of nodes in that layer.

3.6 Transformers

The encoder-decoder architecture is a common and powerful setup in machine learning and typically used to train a model to be able to replicate, or generate data similar to the data used to train it (Ohlsson and Edén, 2020). This can for instance be used to generate images, but it has also been widely used in language and text models. Here the encoder takes an input sequence of symbols, $\mathbf{x} : \{x_1, \ldots, x_n\}$ and produces and encoded vector $\mathbf{z} : \{z_1, \ldots, z_n\}$ of it. This is then used as input to the decoder which outputs a sequence of symbols $\{y_1, \ldots, y_m\}$, one symbol at a time. The symbols are then put into the decoder again to generate subsequent symbols. The transformer, proposed by Vaswani et al. (2017), uses such an encoder-decoder architecture and has proven successful in natural language processing models.

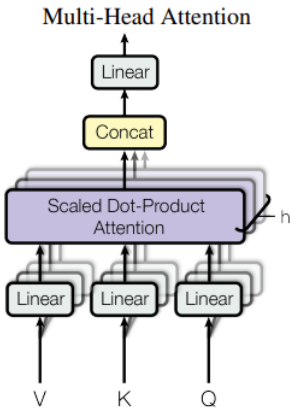

Figure 3.5: Visual representation of multi-head attention. Taken from Vaswani et al. (2017).

An important concept used in transformers is *attention*. This entails taking a query and a set of key-value pairs, where the query is mapped to an output of a weighted sum of the values. Here, the queries, keys and values are all vectors. The weights are obtained by using *scaled dot-product attention*, which takes the query and key vectors of dimension d_k and the value vector of dimension d_v as input. The weights are then determined by calculating the dot-product of the query and every key, dividing by $\sqrt{d_k}$ and applying a softmax layer. See Equation 3.19, which uses matrices Q, K and V containing a set of queries, and the keys and values. The factor $\sqrt{d_k}$ is to ensure that the softmax does not have small gradients for large input matrices (Vaswani et al., 2017).

$$\text{attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{3.19}$$

Instead of using the attention mechanism once where the query, key and value vectors have the same dimension as the model d_m, Vaswani et al. (2017) used *multi-head attention*. This entails the vectors being linearly projected to d_k, d_k and d_v dimensions h times. Meanwhile, the attention function is performed on all projected vectors, outputting vectors with dimension d_v. The vectors are then concatenated and projected again to produce the final output. See Figure 3.5. In their approach Vaswani et al. (2017) used $h = 8$ and $d_k = d_v = d_m/64$ with a model dimension of 512.

The approach used by Vaswani et al. (2017) has an encoder with 6 layers of the same kind and within those layers two sub-layers. The two inner layers are 1: A multi-head attention layer and 2: A fully connected feed-forward neural network, see Figure 3.6. Every sub-layer ends with a normalizing layer which takes the input plus the output of a sub-layer, called residual connections by Vaswani et al. (2017). In order for this to be feasible, the dimension of the input and output of a hidden layer need to be the same, which in this approach was set to 512.

The decoder of the transformer also consists of 6 layers, but instead has 3 sub-layers. The additional one, which is the second sub-layer, is a multi-head attention layer that takes the output from the encoder as input. The first sub-layer is a multi-head attention layer, just as for the encoder, but here it masks subsequent positions. This is because for a sequence of

inputs, the prediction for input at index i should only depend on inputs with indices smaller than i. Lastly, the third and final layer is a fully connected dense layer. The decoder also uses residual connection between every sub-layer, just as the encoder. After the 6 layers the decoder ends with a linear layer with a softmax activation function.

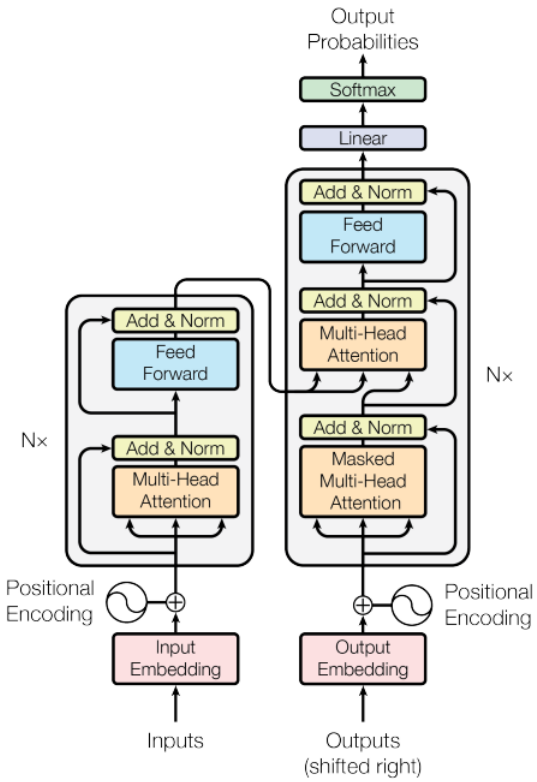

Figure 3.6: The transformer architecture, from Vaswani et al. (2017)

Multi-head attention is used in the transformer in three ways

Encoder-decoder attention, where the queries are the output from the decoder of the previous layer and the key-value pairs are the output from the encoder. This way, the decoder output can have attention to all positions in the input.

Self-attention, where the queries and key-value pairs are the output of the encoder in the previous layer. Here all positions in the current encoder can attend to all positions in the previous encoder.

In the decoder, the self-attention mechanism is designed to block output contributions from future positions. This means that the decoder should only attend to positions lower than or equal to its own position. All values corresponding to those positions are then masked by setting them to $-\infty$ in the softmax layer.

Furthermore, the feed-forward networks Vaswani et al. (2017) used in both the encoder and decoder are built up by two fully connected layers with a relu activation function after the first layer. The input and output of these models have the same dimension as the model, while the hidden dimension shared between the layers is 2048.

Vaswani et al. (2017) trained embeddings to generate vectors of dimension d_m from the input and output tokens. These embeddings are then fed into the linear transformation followed by a softmax layer to generate probabilities for the next possible tokens in the decoder. Furthermore, positional encodings are used to include the position of a token in a sequence. These positional encodings are then added to the input embeddings at the beginning of both the encoder and decoder, meaning that they also have dimension d_m. The equation to calculate the positional embeddings is shown below in Equation 3.20.

$$\begin{aligned} PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_m}) \\ PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_m}) \end{aligned} \tag{3.20}$$

Here pos and i are the token position in the sequence and the token position in the embedding vector.

3.7 BERT

BERT, or *Bidirectional Encoder Representations from Transformers*, proposed by Devlin et al. (2018) at Google, uses the encoder part in the transformer architecture and has been used to reach state-of-the-art performances for many NLP applications. Bidirectional means that in order to predict a token at index i, BERT uses tokens with indices both smaller and larger than i. This differs slightly from the approach by Vaswani et al. (2017) for the transformer, though BERT still uses the same overall encoder architecture. This model produces contextual embeddings of texts, which means that it differentiates between the word *key* from the bag-of-words example. This is achieved by training the model on a large corpora of texts. The pre-training of BERT has been done on a text corpus consisting of the text from English Wikipedia and a corpus called *BooksCorpus* (Zhu et al., 2015). During pre-training, two semi-supervised methods were used: *masked language modeling* and *next sentence prediction*.

In masked language modeling 15% of the tokens in the tokenized input are masked. During masking there is an 80% probability that the token is replaced by `[MASK]`, 10% probability that the token is changed to another token and 10% probability that the token remains unchanged. The model is then trained by trying to predict the masked token and minimize the crossentropy loss (Devlin et al., 2018).

In next sentence prediction, the BERT model learns to understand relationships between sentences, which is an important feature for many NLP applications. Each training sample contains two sentences, `A` and `B`. There it is 50% probability that `B` is the sentence which follows `A`, and 50% probability that `B` is a completely random sentence. During training a binary classifier is then trained to predict whether `B` is the next sentence or not.

Devlin et al. (2018) developed two models, $\text{BERT}_{\text{base}}$ and $\text{BERT}_{\text{large}}$ which have the same overall architecture, but differ in model size. Denoting the number of transformer blocks as L, the hidden size between layers as H and the number of self-attention heads as A the models can be described as

$$\begin{aligned} &\text{BERT}_{\text{base}}(L = 12, H = 768, A = 12), \text{ 110M parameters} \\ &\text{BERT}_{\text{large}}(L = 24, H = 1024, A = 16), \text{ 340M parameters} \end{aligned}$$

In their paper, Devlin et al. (2018) reached higher performance for the large model. However, I chose to only use the base model since my access to powerful computation tools was limited.

BERT has led to several implementations in other languages. The ones I used in this Master's thesis are *Swedish BERT* (Rekathati, 2021), developed at Kungliga Biblioteket and *Multilingual BERT* developed by Devlin and his team.

3.8 Performance Metrics

3.8.1 Accuracy

The most straightforward way to do measure the performance of a model is to measure the accuracy which simply means to count how many times the model classifies a sample correctly and divide that by the total number of samples.

3.8.2 Precision, Recall and F1 score

A more meaningful way to determine how good a model is, is by calculating the *precision* and *recall*, which for multi class classification will assign one metric to each class. Using the following metrics for a class i.

TP – True positives, sample belonging to class i, classified as class i
FP – False positives, sample not belonging to class i, classified as class i
FN – False negatives, sample belonging to class i, not classified as class i

we define precision and recall as.

$$\text{precision} = \frac{TP}{TP + FP}$$
$$\text{recall} = \frac{TP}{TP + FN}$$

Looking at the definition we can see that recall is the number of correctly classified samples divided by the total number of samples from a class, which is a measure of how good the model is at detecting a class. Precision on the other hand is the number of correctly classified samples divided by the number of samples predicted to belong to a class. This gives us a measure of how certain the model is that a predicted class actually is correct.

These two metrics can then be combined by calculating the harmonic mean between the two, resulting in the *F1 score*.

$$\text{F1 score} = \frac{2 \cdot \text{prec} \cdot \text{rec}}{\text{prec} + \text{rec}}$$

This is the metric used to evaluate the models during training in this thesis, since it takes the most factors into account.

3.9 Data stratification

Splitting data into training, validation and test sets is standard procedure when training artificial intelligence models. This is straightforward to do if the data only has one label per example. Then it's simply to determine how many samples are needed from each class and splitting the data from that. When the data has more than one possible class however, there is no simple method.

The stratification used in this thesis is based on a method for multi-label classification proposed by Sechidis et al. (2011). Down below in Algorithm 1, the pseudo code is given.

For the algorithm below D is the full dataset which is going to be split into n subsets $D^1, \ldots D^n$, where every data point consists of one pattern x and a set of classes $\mathbf{Y}$. C is the set of classes that have not yet been assigned to any subset D^i.

Algorithm 1 Data stratification algorithm

while $|C| \geq 0$ **do**
  $c_l \leftarrow$ Class with least samples to fill
  for each$(x, \mathbf{Y}) \in D$ **do**
    if $c_l \in \mathbf{Y}$ **then**
      $D^i \leftarrow$ Subset that asks for most samples of c_l ▷ In case of tie, pick randomly
      $D^i \leftarrow D^i \cup \{(x, \mathbf{Y})\}$
      $C_r \leftarrow$ All classes that have been fully assigned to any subset
      $C \leftarrow C \setminus C_r$
      $D \leftarrow D \setminus \{(x, \mathbf{Y})\}$

This stratification method was used to divide the Reuter corpus into training and test set during the initial testing phase.

Chapter 4

Method

4.1 Testing on the Reuters dataset

I initially trained a classification model on the Reuters dataset, see Section 2.1. The English version of this dataset is widely used and corresponds to a similar problem to the categorization of *Dumma Människor*. In this model, I used a TF-IDF representation to vectorize the news articles, which were then used as input to a logistic regression model. The input was then stratified using the method proposed by Sechidis et al. (2011) to create a training and test set. To fit this model, I used the LFBGS optimizer. Since all articles could be part of one or more topics, I trained one binary model for each topic which classified all articles as either being part of a topic, or not.

Below in Figure 4.1 is the F1 score plotted as a function of topic frequency in the training set.

From the plot, I concluded that a about 1500 samples was needed in order to classify a topic with sufficient accuracy.

4.2 Collecting data and building models

The training dataset for this thesis was collected from the public online forums *flashback.org* and *familjeliv.se*. From there I chose 65 suitable topics, which could be applicable to the podcast data. For each topic I ensured that it had sufficient number of samples in order to correctly classify it. Furthermore it was ensured that all texts had a maximum of 512 characters, since this was the maximum number of characters which could be put in to the transformer models.

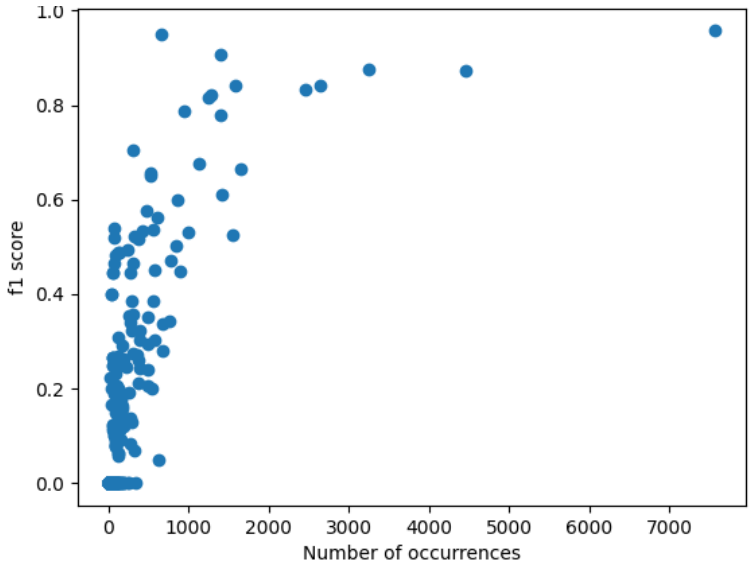

Figure 4.1: The F1 score as a function of the number of occurrences of a topic in the training set.

4.3 Training data

From the full dataset collected from Flashback and Familjeliv, I extracted 1000 texts from each topic to use for training and testing, resulting in 65,000 texts. This proved to be the best compromise for performance, since a larger dataset would take too much time to train on for the more advanced models later on. From this I used 80% for training and validation and 20% for testing.

4.4 Building an initial model

I built the first model by using the same setup as for the Reuters data. However, since the collected dataset no longer had multiple classes for each text, the approach with one binary classifier per topic proved to be flawed, resulting in low performance. Because of this I changed all binary topic models to a multinomial logistic regression model which assigned a text to only one label.

4.5 Using BERT embeddings

Next instead of using a TF-IDF representation of the texts, I used BERT sentence transformers to generate contextual sentence embeddings. For this step, the Swedish cased version of BERT model was downloaded from the python library *sentence_transformers*. This model used a tokenizer which deployed the WordPiece tokenization described in the previous chapter. I used this BERT base to generate embeddings for the dataset, which were then used as input to a logistic regression model. In this step, the layers of the BERT model were frozen and hence not fine-tuned. The reason for using the cased version while the training data was lower case,

was because this model was only available as the cased version.

4.6 Adding Feed-Forward Neural Networks

Using the same BERT embeddings as before, I trained new models by replacing the multinomial logistic regression with a feed-forward neural network. This network, similarly as the logistic regression model, also only predicted one topic per text. The neural network consisted of one layer with 480 nodes with a sigmoid activation function followed by a hidden layer with an equal number of nodes and the same activation function. Lastly, since the model should predict one out of 65 topics, I used an output layer with 65 nodes and softmax output activation function. Before every layer there were also a dropout layer with $p = 0.3$. The batch size for this experiment was 32 and as optimizer, I used Adam with $\eta = 10^{-3}$, $\beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-7}$.

As a stopping criterion during training for this model, I monitored the loss on the validation data, which consisted of 10% of the training data. If the validation loss did not decrease for 5 epochs, the training loop was terminated. In addition to this, I set the maximum number of epochs to 100 epochs.

4.7 Multilingual BERT and Fine-tuning

The next step in trying to improve the model was to unfreeze the last layer in the BERT model and fine-tune the transformer model to the input data. For this I used the PyTorch implementation of BERT, since this was the only implementation that supported freezing and unfreezing layers. Once again WordPiece was used for the sentence tokenization. This model setup consisted of the transformer as base followed by a layer with as many output nodes as there were categories. The last layer once again had a softmax output activation function.

Using this setup I fine-tuned the last layer of the *Swedish BERT base cased* and *multilingual BERT base cased* models. Here, the cased versions were used since they were the recommended versions in PyTorch. For this experiment, I used a batch size of 15 and the Adam optimizer with $\eta = 10^{-4}, \beta_1 = 0.9, \beta_2 = 0.999$ and $\epsilon = 10^{-8}$. Similarly as before the validation loss determined when the model were done training. Since these models were much more time consuming to train, I terminated training if the loss not decrease for 2 epochs instead of 5. Due to the same reason, the training procedure in this part was only done on the shuffled sampling of the training dataset.

4.8 Model evaluation

In the evaluation of the models, the podcast transcriptions were used as test dataset. Just as for the training data, it was ensured that all texts had a maximum length of 512 characters. The models were evaluated by extracting one segment per episode from the podcast data and using the most promising models to predict their topics. For this, I chose all the models which had been trained using a transformer model. Thereafter a qualitative evaluation was made

on the models to determine the best model. This qualitative evaluation was done by sending the topic classification results to GetReachAudio where they ranked the models using three grades *Good*, *Quite good* and *Bad*. After this, I calculated the accuracy for the best one and compared it to the accuracy of the baseline model. Due to time consuming training procedure I evaluated the fine-tuned Swedish BERT model myself, whereupon it was compared to the other five model to determine its grade. The reason for not evaluating all models was to relieve the work load, since the process was quite time consuming.

Chapter 5
Results

All tables and figures below contain results from the test split of the training dataset, extracted from *Flashback* and *Familjeliv*.

5.1 Overall Performance

Table 5.1 below presents the macro F1 score, weighted F1 score and accuracy of the models trained on the shuffled sampling of the training dataset. Table 5.2 shows the corresponding values for the ordered sampling of the dataset. Here we can see that the initial model which deployed several binary classifiers was not suitable for this application. Furthermore, all models which used Swedish BERT outperformed the baseline model on all performance metrics. We can also see that the multilingual model is worse compared to the models which were exclusively pre-trained on a Swedish corpus. Furthermore, the fine-tuned Swedish BERT model and the original Swedish BERT model with a feed-forward neural network (FFNN) model head achieved the same performance on the training dataset.

Method	Macro F1	Weighted F1	Accuracy
Binary TF-IDF+ logistic	0.22	0.22	0.14
TF-IDF + logistic	0.66	0.66	0.66
Swedish BERT + logistic	0.71	0.71	0.71
Swedish BERT + FFNN	**0.72**	**0.72**	**0.72**
Fine-tuned multilingual BERT	0.62	0.62	0.63
Fine-tuned Swedish BERT	**0.72**	**0.72**	**0.72**

Table 5.1: Macro and weighted F1 score as well as accuracy of the models trained on the shuffled dataset from *Flashback* and *Familjeliv*.

Method	Macro F1	Weighted F1	Accuracy
Binary TF-IDF + logistic	0.24	0.24	0.15
TF-IDF + logistic	0.72	0.72	0.71
Swedish BERT + logistic	0.75	0.75	0.75
Swedish BERT + FFNN	**0.77**	**0.77**	**0.77**

Table 5.2: Macro and weighted F1 score as well as accuracy of the models trained on the ordered dataset from *Flashback* and *Familjeliv*.

5.2 Breakdown by Category

Figure 5.1 illustrates the individual F1 scores for the different categories in models trained on the shuffled sampling of the training dataset. Figure 5.2 shows the corresponding graph for the ordered sampling of the dataset. To reduce text in the figure, the F1 scores are plotted against the index of its corresponding category. In these figures, we can see that the initial binary model approach has lower F1 score than all other models for almost every topic. We can also see that the pre-trained Swedish transformer models have quite similar F1 scores as each other for the same topics. For some topics, the baseline model actually has the highest F1 score, but over all topics the transformer models perform better.

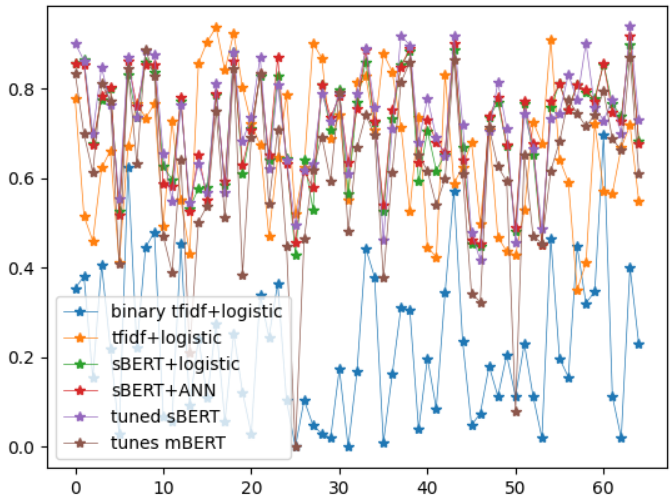

Figure 5.1: F1 scores of all topics in models trained and tested on the shuffled dataset from *Flashback* and *Familjeliv*.

Figure 5.3 shows the precision and recall for the initial model which used 65 individual binary classifiers to predict the topics. Here we can see that the recall is low while the precision is high for both the ordered and shuffled sampling of the training dataset. This suggests that the model assigned very few samples to a topic, but that it usually was correct.

Table 5.3 shows the five topics with the best and worst F1 scores for the model with the highest performance on the forum dataset, i.e the Swedish BERT model with a FFNN head. Here we can see that the topics with the highest performance are usually easily defined, such as *Bilar* (*Cars*) or *Aktier* (*Shares*). These topics do not share much overlap between other topics, see the appendix for the full list of topics. Meanwhile, the topics with low performance

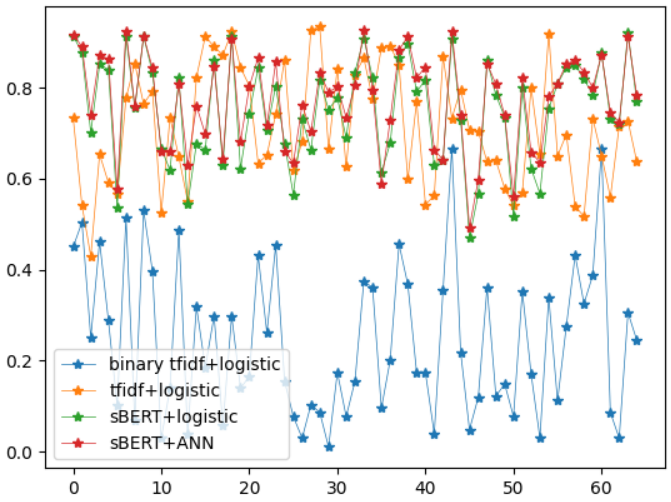

Figure 5.2: F1 scores of all topics in models trained and tested on the ordered dataset from *Flashback* and *Familjeliv*.

can be somewhat harder to define. For example *Relationer och samlevnad* (*Relationships and cohabitation*) and *Barn och familj* (*Childen and family*) can be quite similar and difficult to separate from each other. To illustrate this, Table 5.4 shows the two topics which the lowest performing topics are most frequently misclassified as.

Category	Category (eng.)	F1 score
Bilar	Cars	0.93
Matlagning och metoder	Cooking and methods	0.93
Aktier	Shares	0.92
Politik: utrikes	Politics: foreign	0.92
Musik	Music	0.91
Relationer och samlevnad	Relationships and cohabitation	0.60
Medier och journalistik	Media and journalism	0.59
Barn och familj	Children and family	0.58
Rädsla/skräck	Fear/terror	0.56
Psykologi	Psychology	0.49

Table 5.3: The five topics with the best and worst F1 scores from the model using Swedish BERT and a feed-forward neural network, trained on the ordered sampling of the texts from *Flashback* and *Familjeliv*.

5.3 Qualitative Evaluation

Table 5.5 shows the manual qualitative evaluation of the performance on the podcast transcriptions. This evaluation was done by using all transformer-based models and classifying all text segments in all the podcast episodes. From this, one text segment per episode was

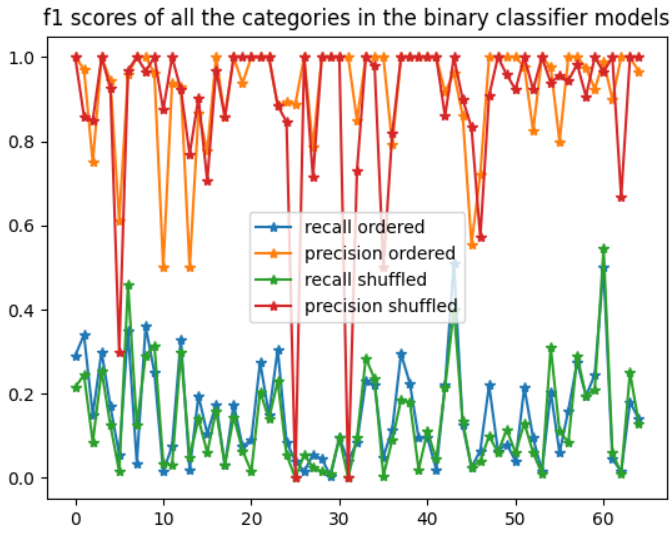

Figure 5.3: F1 scores for all categories in the binary classifier models, tested on the test split of the dataset collected from *Flashback* and *Familjeliv*.

extracted together with its predicted topics for all models. The text segments together with their topic predictions were then manually evaluated by GetReachAudio. The models were numbered as model 1-6 so as to not create bias for any model. In the evaluation, GetReachAudio noted every text which was classified correctly by any model. From this, they then determined which models that were correct the most times, which models that were correct some of the times and which models that were correct the least times. The models were then given the grades *Good*, *Quite good* or *Bad* depending on how well they performed on that scale.

In Table 5.5 we see that the model that achieved the highest F1 score on the training dataset, i.e Swedish BERT + FFNN, did not perform the best on the podcast dataset. Instead, the models which used Swedish BERT and logistic regression were found to be the best.

The accuracy of the model trained on the ordered sampling was found to be at least 0.45 and the accuracy of the model trained on the shuffled sampling at least 0.46. For the baseline model, the accuracy was found to be at least 0.26.

However, these scores should not be taken too literally since the accuracy measurement was not straightforward. In the evaluation, I recorded a successful classification only if I was sure that the topic classification was correct. There were many occurrences where it was either difficult to determine the topic of a text segment, or it was difficult to tell whether the topic classification was correct or not. Furthermore, there were also some occurrences where a topic classification was close, but not good enough to say that it was correct. Because of this the accuracy could be higher, which is why I wrote “at least” before the accuracy. However, the accuracy scores can be compared between the models, which shows that the transformer-based models outperformed the baseline model by 73% and 75%. Again these values might vary, but it is clear that they are better than the baseline model.

Category (eng.)	Most confused topics (eng.)
Children and family	Equality and discrimination
	Feminism
Media och journalism	Censorship and freedom of speech
	Politics: domestic
Psychology	Social anthropology, ethnology and sociology
	Work
Relationships and cohabitation	Eroticism and sexuality
	Children and family
Fear/terror	Health
	Psychology

Table 5.4: The five topics with lowest F1 score and their most confused topics from the model using Swedish BERT and a feed-forward neural network, trained on the ordered sampling of the texts from *Flashback* and *Familjeliv*.

Model	Grade	Accuracy
Swedish BERT + FFNN shuffle	Quite good	–
Swedish BERT + FFNN order	Quite good	–
Swedish BERT + logistic shuffle	Good	> 0.46
Swedish BERT + logistic order	Good	> 0.45
Fine-tuned multilingual BERT shuffle	Bad	–
Fine-tuned Swedish BERT shuffle	Quite good	–

Table 5.5: Scores from the manual evaluation on the podcast data.

Chapter 6
Discussion and Conclusion

6.1 Discussion

The first model, which consisted of one binary classifier for every topic proved to have very low performance on this dataset. Because the dataset just has one label per text, for every individual classifier, only 1/65 of the data is going be part of the topic which the model tries to recognize. The rest is not going to be part of that topic. This means that the dataset is skewed for every topic and the model seems to expect that very few texts are going to be part of a category. We can see this in Figure 5.3, where the recall typically is very low and the precision is very high. This happens because the model in fact assigns very few samples to a topic, but when it does, it is usually correct. A model with this characteristic can be useful, but for this application it is not the desirable behavior.

Changing the binary models into one large multinomial logistic model proved to increase the performance greatly across the board, see Table 5.1 and Table 5.2. This is not surprising since the model now was trained on a dataset where all possible labels were evenly distributed among the texts.

As Tables 5.1 and 5.2 show, the pre-trained transformers outperform the baseline model in all cases apart from the multilingual model. The fact that using contextualized sentence embeddings generated by BERT leads to higher performance, as opposed to using a TF-IDF representation is no surprise. We can also see from the graphs in Figures 5.1 and 5.2 that there is not much separating the F1 scores for individual topics for the Swedish transformer based models. This indicates that using Swedish BERT to represent the texts is the most contributing factor to a high performing model. The results here also suggests that the choice of model head does not make a big difference, which is why I chose to only use a model head with one layer for the fine-tuned BERT models.

Furthermore, the fine-tuned Swedish BERT model achieved the same performance as the BERT models with a FFNN model head which were not fine-tuned. It is somewhat strange that it did not outperform the original models, since fine-tuning BERT intuitively should

produce a better model. One possible reason for this could be that the fine-tuned model only had one final layer after the transformer whereas the other had three layers. This results in that the latter model adapts as well to the training data, resulting in another kind of fine-tuning.

Another reason for these results is that the training procedure for the BERT models with one unfrozen layer was much more time consuming. For the model where all BERT layers were frozen I could experiment a lot with the number of layers, hidden nodes, learning rates, batch sizes, dropout probability and activation functions to find the optimal set of hyperparameters. This was because the only highly time consuming factor was to generate the contextual embeddings, which only had to be done once. However, one epoch for fine-tuning BERT lasted for more than one hour, even when running the computations on a GPU. This meant that I had to wait a long time before I knew whether a setup was good or not, and then possibly make changes to improve the model.

One way to remedy this could be to use less data, but this would also have affected the model performance negatively. We want the model to see as much data as possible to be able to generalize. Recalling the experiment on the Reuters corpus, which showed that a category should have about 1500 samples in the training data. From this the training set should have been even larger, since I only used 1000 texts from each topic. However, using more than 1000 proved to be unfeasible when when it came to time restrictions. With more powerful computation tools it is likely that the models could have been better, at least with respect to the training dataset.

Comparing the two fine-tuned models it is clear that Swedish BERT performs better than its multilingual counterpart. This is not particularly surprising since the Swedish version has been pre-trained only on Swedish texts, whereas multilingual BERT has not. As previously mentioned, the model could likely be improved given more time to find the optimal set of hyperparameters. A multilingual model could however prove useful in order to expand the podcast analysis to more languages than Swedish.

The best final model model however, did not come from the model with the highest F1 score on the training dataset from *Familjeliv* and *Flashback*. Instead, the qualitative analysis found that the models using a logistic regression model head performed the best. This is somewhat surprising, since high performance on similar data should lead to high performance for the test data. The most likely explanation for this is that using a model head with a few fully-connected layers, as opposed to logistic regression, leads to overfitting to the training data. This is due to the model with fully-connected layers being more complex. Because of this, instead of finding general patterns and qualities in the data, it might pick up on features specific to the training data. This naturally lowers the performance on another dataset. The same argument can be used for the fine-tuned Swedish BERT model, which also seems to overfit to the training data. However, the less complex logistic regression model seems to generalize better and performs better on the podcast data. In order to use fine-tuned models to classify the topics in a podcast, it then seems that the training data needs to be more similar to the podcast data.

We can also see that the qualitative evaluation of the podcast dataset did not seem to differentiate much between the ordered or the shuffled dataset, although for all models the F1 score was higher for the ordered dataset, see Tables 5.1 and 5.2. However, since we want the model to generalize as much as possible, a lower F1 score on the training data might not be that bad, which is shown by the qualitative evaluation. Since the shuffled dataset should

contain more versatile texts, as it picks any text from any thread, this should be the best option even though the F1 score is lower in the training set. The reason the ordered models achieve higher F1 score might be due to that the texts are sampled from less threads. Making the model slightly biased towards those threads.

When comparing these results to to those of Remmer et al. (2021), we see that the performance metrics are roughly the same for the case where they divided the dataset into smaller blocks. The F1 score obtained for this setup is a little bit lower compared to Remmer et al. (2021), whereas they used less topics (10 compared to 65). The similarity in results is reasonable since they were trained to perform the same kind of task.

One possible way to improve the performance of the models could be to use that the same topic is often covered for a longer period of time. This is at least the case for this podcast. The hosts tend to talk about one subject for some time and not change subject frequently. This could have been introduced to the model by using the previous two or three topics as input in addition to the text vectorizations. Furthermore, one could use that some topics are probably more likely to follow each other than others. For example it does not seem likely that someone would talk about religion and then directly after talk about cooking and cooking methods. In order to be able to use this however, I would have needed a dataset where these features could be extracted.

For the dataset I used, I could have used the previous topics as input and not shuffle the data in the training loop. However, this would not have been a good idea since all texts with the same label followed each other, apart from in between topics. This would likely not resemble how someone would speak. Instead, the dataset would have had to consist of annotated texts from a blog or another podcast. To my knowledge, no such datasets were available, but it would have been interesting to see what impact these features would have had on the final models.

There is a large difference in how well the models recognizes different topics, see Figure 5.2 and 5.1. The 5 topics with the highest F1 scores and the five topics with the lowest F1 scores for the model with the best performance metric are shown in Table 5.3. Here you can see that the topics with high F1 scores are topics which are quite "narrow", i.e they don't include much texts which could belong to another topic. For example *Aktier* (Shares), *Matlagning och metoder* (cooking and cooking methods) and *Bilar* (cars). These labels are quite niche which makes it easy for a computer to differentiate between them.

On the other hand, there are the categories *Relationer och samlevnad* (Relationships and cohabitation), *Barn och familj* (Children and family) and *Psykologi* (Psychology) which have low F1 scores. This can be linked to that they have much in common with other topics. Particularly the first two, which can be seen in Table 5.4. Furthermore, psychology is a very wide subject and can contain texts which are quite different from each other. Texts from different categories that are similar to each other, or texts from the same category which are different from each other naturally makes it harder for the model to recognize the patterns. One way to remedy this could be to merge topics which are similar.

Furthermore, the training data itself is not perfect. The texts from *Flashback* and *Familjeliv* are completely unprocessed, apart from making the text lower case. Because the data is taken from public forums, the text might contain spelling errors, poor grammar or the threads can go off-topic. All of these factors makes the texts less predictable and can lower the performance. However, this was the most suitable publicly available dataset I could find for the podcast data since the language most resembles spoken language.

Finally, the podcast transcripts adds error to the final evaluation since they are not perfect either. This is to some degree due to unchangeable factors, such as that there are two people speaking in the podcast which makes the transcriptions worse. Furthermore, NLP models in Swedish are usually not as advanced as NLP models in other languages, such as English. Advances in the field could improve the Swedish transcription, which would likely also improve the model performance on the podcast data. This was a factor to consider in the qualitative evaluation as well. If the transcriptions are accurate, it would be easier to evaluate the models, and the accuracy measure would be more trustworthy.

6.2 Conclusion

In this Master's thesis, I explored different approaches to topic classification for podcasts. To do this, I collected a dataset of texts from Swedish online forums, spanning 65 different topics which I used to train the models. The dataset was chosen due to its colloquial language, which should resemble the language in the podcast as much as possible. Furthermore, it was easy to label all the texts with its corresponding forum topic. This means that no time consuming manual annotation had to be made.

Initially I constructed a multi-label classifier, which could assign any, or no, topic to each text segment. This approach turned out to be flawed, since the dataset used for training only contained texts with one label each. The baseline here consisted of a TF-IDF representation of the texts and a logistic regression model head, which only reached an F1 score of 0.22 and 0.24 for a random and ordered sampling of the training dataset respectively. Because of this, I instead proceeded to train a classifier which only assigned one topic per text.

The same baseline model setup was used to produce a much better performance, reaching an F1 score of 0.66 and 0.72 for the shuffled and ordered sampling respectively. This approach was then expanded by using a pre-trained transformer to generate contextual embeddings of the texts, instead of using a TF-IDF representation. The transformer used here was Swedish BERT, which increased the F1 score to 0.71 and 0.75 respectively, as well as the accuracy to 0.71 and 0.75. Changing the logistic regression model head to a small feed-forward neural network increased the F1 score further to 0.72 and 0.77 and the accuracy to 0.72 and 0.77.

As a final experiment I tried to fine-tune a multilingual transformer, multilingual BERT, and the Swedish BERT to the the training data by unfreezing the last layer in the encoder architecture. Here, the Swedish BERT model performed equally well as the best BERT model in the previous step. The multilingual model, however, only reached an F1 score of 0.63.

However, a qualitative evaluation on an excerpt of the podcast transcripts showed that the best model for podcast topic classification was the one which deployed Swedish BERT and a logistic regression model head. This reached an accuracy of at least 0.46 on the podcast dataset, outperforming the TF-IDF baseline model by 75%.

6.2.1 Future Work and Final Words

In order to obtain a higher performance, one has to use better data which is more similar to the podcast transcriptions. Furthermore, since the training data was collected from online forums, it is possible that some automatically assigned topics are not completely correct.

It would also be beneficial to find more data to expand the model with more topics. Furthermore, considering a multilingual approach to the topic classifier could be a good option. Since the company is located in Malmö, it could be useful to be able to classify topics in both Danish and English podcasts.

In order to obtain a more trustworthy evaluation, the podcast transcriptions needs to be a bit more accurate. This would likely also lead to better model performance. Since advances are constantly made in natural language processing tools, this is highly achievable.

To my knowledge, at the time of writing, there are no topic classifiers available, such as the models created in this master´s thesis. Because such tools could be useful, the work done here is highly relevant. Lastly, even if the model does not classify all topics in a podcast correctly it does give a good outline of the topics covered in an episode. This can likely be used in a rudimentary episode analysis, and with some improvements it could prove to be a powerful tool in improving podcast content.

References

Agresti, A. (2018). *An Introduction to Categorical Data Analysis*. John Wiley & Sons, Ltd, 3rd edition.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Dhar, A., Mukherjee, H., Dash, N. S., and Roy, K. (2021). Text categorization: past and present. *Artificial Intelligence Review*, 54:3007–3054.

Ikonomakis, M., Kotsiantis, S., and Tampakas, V. (2005). Text classification using machine learning techniques. *WSEAS transactions on computers*, 4(8):966–974.

Lewis, D., Yang, Y., Russell-Rose, T., and Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397.

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, New York, NY, USA, 2e edition.

Nugues, P. M. (2014). *Language Processing with Perl and Prolog. Theories, Implementation, and Application*. Springer Verlag, Berlin Heidelberg New York, second edition.

Ohlsson, M. and Edén, P. (2020). Introduction to artificial neural networks and deep learning. Technical report, Lund University.

Rekathati, F. (2021). The KBLab Blog: Introducing a Swedish Sentence Transformer.

Remmer, S., Lamproudis, A., and Dalianis, H. (2021). Multi-label diagnosis classification of swedish discharge summaries–icd-10 code assignment using kb-bert. In *RANLP 2021: Recent Advances in Natural Language Processing, 1-3 Sept 2021, Varna, Bulgaria*, pages 1158–1166. Association for Computational Linguistics.

Schwenk, H. and Li, X. (2018). A corpus for multilingual document classification in eight languages. *arXiv preprint arXiv:1805.09821*.

Sechidis, K., Tsoumakas, G., and Vlahavas, I. (2011). On the stratification of multi-label data. In Gunopulos, D., Hofmann, T., Malerba, D., and Vazirgiannis, M., editors, *Machine Learning and Knowledge Discovery in Databases*, pages 145–158, Berlin, Heidelberg. Springer Berlin Heidelberg.

Thompson, L. and Mimno, D. (2020). Topic Modeling with Contextualized Word Representation Clusters.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is All you Need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.

Appendices

Appendix A
Tables

Category	Category
Aktier	Korruption och missförhållanden i offentlig verksamhet
Alkohol och dryck	Medicin och hälsa
Arbete	Medier och journalistik
Resor: Asien	Mode
Astronomi och rymdfart	Motorsport
Barn och familj	Musik
Bilar	Olyckor och katastrofer
Biologi	Organiserad brottslighet
Bitcoin och andra virtuella valutor	Paranormala fenomen, ockultism och ufologi
Bollsport	Politik: inrikes
Bostad, hem och trädgård	Politik: utrikes
Censur och yttrandefrihet	Privatekonomi
EU	Psykologi
Erotik och sexualitet	Relationer och samlevnad
Etik	Religion
Feminism	Restauranger, barer och nattklubbar
Film och filmproduktion	Rollspel och Sällskapsspel
Filosofi	Rädsla/skräck
Fysik, matematik och teknologi	Serier och serietidningar
Glädje	Skönhet
Historia	Socialantropologi, etnologi och sociologi
Hockey	Sorg
Humor	Språk
Husdjur	Stadsplanering, infrastruktur och arkitektur
Hälsa	Tatuering och piercing
Irritation	Telefoni, surfplattor och läsplattor
Jakt, fiske och vildmark	Träning
Jämställdhet och diskriminering	Träning, kost och kosttillskott
Kemi och pyroteknik	Utbildning och studier
Klimat, miljö och geovetenskap	Vapen och militär
Konspirationer och alternativa teorier	Vintersport
Matlagning och metoder	Resor: Övriga världen
Litteratur	

Table A.1: List of all possible topics in the training dataset collected from Swedish online forums in Swedish.

Category (eng.)	Category (eng.)
Shares	Corruption and Misconduct in Public Business
Alcohol and Beverage	Medicine and Health
Work	Media and Journalism
Travel: Asia	Fashion
Astronomy and Space	Motorsport
Children and Family	Music
Cars	Accidents and Disasters
Biology	Organized Crime
Bitcoin and other virtual currencies	Paranormal phenomena, occultism and ufology
Ball sports	Politics: Domsetic
Housing, Home and Garden	Politics: Foreign
Censorship and Freedom of Speech	Personal Finance
EU	Psychology
Erotica and Sexuality	Relationships and Cohabitation
Ethics	Religion
Feminism	Restaurants, Bars and Nightclubs
Film and Film Production	Role Play and Board Games
Philosophy	Fear/Horror
Physics, Mathematics and Technology	Comics and Comic books
Joy	Beauty
History	Social Anthropology, Ethnology and Sociology
Hockey	Care
Humor	Language
Pets	Urban Planning, Infrastructure and Architecture
Health	Tattoo and Piercing
Irritation	Telephony and Tablets
Hunting, Fishing and Wilderness	Training
Gender Equality and Discrimination	Exercise, Diet and Supplements
Chemistry and Pyrotechnics	Education and Studies
Climate, Environment and Earth Sciences	Weapons and Military
Conspiracies and Alternative Theories	Winter Sports
Cooking and Methods	Travel: Rest of the World
Literature	

Table A.2: List of all possible topics in the training dataset collected from Swedish online forums in English.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2022-06-03

EXAMENSARBETE Topic Classifcation for Swedish Podcasts Using Transformers
STUDENT Olof Bengtsson
HANDLEDARE Pierre Nugues (LTH), Kerstin Johnsson (Softhouse Consulting)
EXAMINATOR Jacek Malec (LTH)

Klassificering av ämnen i svenska podcaster med hjälp av förtränade maskininlärningsmodeller

POPULÄRVETENSKAPLIG SAMMANFATTNING **Olof Bengtsson**

Trots den stora populäriteten hos podcaster finns det få verktyg för kreatörer att utvärdera och analysera sina avsnitt. I detta arbete har jag tränat olika modeller för att klassificera ämnen i podcasts, som ska kunna användas i ett sådant verktyg.

Väldigt många personer lyssnar på en eller flera podcaster varje vecka, och antalet lyssnare ökar ständigt. Trots detta finns det väldigt få verktyg tillgängliga för kreatörer att utvärdera innehållet och ta deras podcastserie till nästa nivå. Vidare är det också värdefullt för kreatörer att veta vilka företag de kan vända sig till för betalda samarbeten. GetReachAudio är ett företag som samlar användardata och statistik för podcaster och hjälper kreatörer ta beslut om hur de ska utveckla sin podcast. Ett värdefullt verktyg i ett sådant beslut är att kunna analysera vilka ämnen som berörs i ett avsnitt, samt när i avsnittet de tas upp. Detta kan visa huruvida vissa ämnen resulterar i större lyssnarintresse, samt ge en indikation om vilka sorts företag som skulle vilja marknadsföra sig själva i podcasten.

I mitt arbete har jag tränat olika modeller för att klassificera ämnena i olika podcastavsnitt. Podcasten som användes för att testa modellerna var *Dumma Människor*.

För att träna modellerna skapade jag ett dataset genom att hämta en stor mängd texter från onlineforumen *Flashback* och *Familjeliv*. Dessa användes för att språket skulle påminna om språket i podcasten, det vill säga vara så nära talspråk som möjligt. Från dessa forum samlades texter från 65 olika kategorier, där 1000 texter från varje kategori användes i träningsprocessen. Här bestämdes en texts kategori av vilken rubrik den tillhörde i forumet.

För detta tränades en rad modeller som använde sig av för-tränade modeller på en stor textcorpus. Se figur 1 för hela arkitekturen. Dessa modeller jämfördes sedan med en enklare basmodell för att se hur stor förbättring de gav.

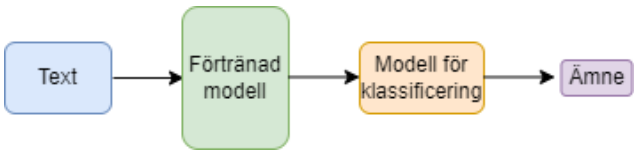

Figure 1: Den övergripande modellarkitekturen som användes.

En kvalitativ utvärdering av modellerna på podcastdatan visade att den bästa modellen var den som använde sig av en modell för-tränad på svenska texter. Resultaten visade även att de modeller som finjusterats för träningsdatan presterade sämre för podcastdatan. Den slutgiltiga modellen klassificerade rätt kategori i podcasten ungefär hälften av gångerna och presterade 75% bättre än basmodellen.