Chair of Financial Management and Capital Markets
TUM School of Management
Technical University of Munich

Risk Factors for Municipal Bond Disclosure Documents

Interdisciplinary Project supervised by Lisa Knauer

Kevin Burton ✉

✉ kevin.burton@tum.de

April 4, 2024

Contents

1 Introduction

The United States of America were built for the automobile. With the car becoming ubiquitous after World War II, American city growth changed from focusing on a walkable, small downtown to large urban sprawls, which today are known as American suburbia [7]. The issue however with this scheme of growth is that the tax revenue that is taken in from suburbs is very little compared to the maintenance cost that they create, especially in comparison with city centers. Therefore strong local economic growth is necessary for these cities and towns and can be broken down into three parts, as done by Charles Marohn in [6]:

1. "Transfer payments between governments, where the federal or state government makes a direct investment in growth at the local level, such as funding a water or sewer system expansion."

2. "Transportation spending, where transportation infrastructure is used to improve access to a site that can then be developed."

3. "Public and private-sector debt, where cities, developers, companies, and individuals take on debt as part of the development process, whether during construction or through the assumption of a mortgage."

Especially the third part is important for the project at hand. Cities profit short term from taking on debt to create new development by the increase of tax revenue, but in the long term drown in the maintenance costs. They are therefore forced to take on new debt and build more. [6]
All in all this debt taken on in the form of bonds is vital for American municipalities. Furthermore, is the income received by the holder of these bonds tax exempt. Unfortunately these bonds are only rarely rated by financial rating companies. Therefore this project tries to fill this existing gap by producing software, that is able to accurately rate bonds using only their disclosure documents.

2 Capabilities of the Software

The ultimate goal of the project is to produce a software that is capable of making an accurate risk assessment given a municipal bond disclosure document. This task entails the usage of a natural language processing machine learning algorithm. The risks to be assessed were originally going to be risk of default, ESG risk and cyber-security risk. However, taking into account the limited scope of the project, only risk of default was fully attempted with natural language processing, whereas the ESG risk was done using a much simpler method, focusing only on the environmental part, (see section 4) and the cyber-security risk assessment falling by the wayside completely, primarily because most disclosure documents did not have any description concerning cyber-security. Since disclosure documents can be very large, some being comprised of hundreds of pages, some way of filtering out the relevant information needed to be found. The natural candidate for this was finding a section in the documents dedicated to risks of the bond, which from here on will be called the risk section. These risk sections appear in the disclosure documents most of the time and present a great foundation for learning. However, this approach means that municipal bond disclosure documents without such a section cannot be assessed. For information on how this risk section was located in the documents see section 5.2. Finally, the output of the software when performing a regular risk assessment is a listing of six probabilities where each represents the probability of the bond belonging to a certain risk class, using the Moody's scale from AAA down to B. Furthermore, the ESG risk is expressed by presenting the national risk index (NRI) for the county where the bond's municipality is located.

2.1 Usage of the software

When performing a normal risk assessment the program should be called using the `-a` tag, representing assessment. Following the tag, the system path to the bond disclosure document saved as a PDF is expected. Furthermore, if an ESG assessment should be made, the tags `-c` and `-s` should also be set, indicating the county and the state of the bond's municipality, respectively. For example, a program call to execute a complete risk assessment on the bond with the CUSIP 416415DV6 would look like the following:

```
$ main.py -a /path/to/document/416415DV6.pdf -c Hartford -s Connecticut
```

If desired it is also possible to train a new model, which is subsequently saved. For this purpose the `-t` and `-n` tags are available. The first tag indicates that a new model should be trained and expects the path to a directory containing pre-processed disclosure documents. Such a directory can be obtained by using the second tag with a path to a directory containing the documents in PDF format. Within this directory should be two subdirectories, one containing the training and one containing the testing data set. Furthermore, the documents in the directories have to be seperated into directories by their labels. This structure is visualized in the following.

```
data
├── training
│   ├── AAA
│   ├── AA
│   └── ...
└── testing
    ├── AAA
    ├── AA
    └── ...
```

It should be noted that if a pre-processed data set is already available, `-t` can be used without `-n`, but not vice versa. If a new data set should be created, the path following `-t` indicates where the pre-processed files should be stored. The final way to use the software is to perform an evaluation on the model's performance. This is achieved by using the `-e` tag followed by the path to the directory containing the pre-processed disclosure documents.

```
$ main.py -e path/to/preprocessed/data
```

An evaluation is automatically performed after training, but this option allows for an evaluation to be performed without the time consuming training of a new model.

3 Related Work

In terms of achieving a financial risk assessment via natural language processing, two related projects are presented. The first, which is a cyber-security risk assessment is of conceptual similarity, because large disclosure documents, in this case issued by companies, are analysed before making a judgment. The second project is more closely related since it also regards the financial sector, but the analysis is one of sentiment instead of risk. Furthermore, the latter is of especial interest because of its usage of BERT for encoding (see 5.3.1).

3.1 Cybersecurity Risk

In the article "Cybersecurity Risk" by Florackis et al. a cyber-security risk analysis for technology companies is presented. Here the parts of companies' 10-Ks regarding their cyber-security are compared to those of companies that experienced a security breach, which makes an assessment possible. This is very comparable to the task at hand, since the cyber-security documents are similar to the bond disclosure documents in that they are both authored by the entity being assessed and give an overview of assets, threats and vulnerabilities. Here a binary classification is made, expressed in a probability of a potential security breach. The classification is based on the fact that companies with similar levels of cyber-security risk use similar words to describe their risk exposure. Unfortunately, a similar assumption can not be made for bond disclosure documents. Wording is often similar within states and sectors, which do experience similar risks, but solely based on that an accurate risk assessment does not seem possible. Furthermore, Florackis et al. state that firms that provide a lengthier and more comprehensive disclosure, are less likely to be exposed to cyber-attacks. It is possible that such a correlation (positive or negative) also exists for bond disclosure documents and therefore providing information on length of the relevant text to the model seems effective. The cyber-security risk assessment is ultimately achieved by calculating cosine and Jaccard similarity between the firm to be rated and firms that were subject to a cyber-attack. [3]

3.2 FinBERT: Financial Sentiment Analysis with Pre-Trained Language Models

FinBERT is a pre-trained language model, which is able to perform sentiment analysis on sentences related to the financial sector. Sentences can be labeled in three ways: positive, negative and neutral. This work by Araci is based on BERT (Bidirectional Encoder Representations from Transformers), which is a pre-trained model mainly used to encode text into so-called embeddings (for more information on BERT see section 5.3.1). For

FinBERT explicitly, the model was further pre-trained on words specifically related to finance to convey more precise meaning in the embeddings. This work is of particular interest for the task at hand since using BERT for encoding of the disclosure documents was planned regardless. Having a further pre-trained and more fine tuned encoder would certainly improve results. However, the fine tuned FinBERT version seems to be completely integrated into the sentiment analysis model and no solution was found to extract it. Another possible idea was to use the sentiment analysis on all sentences contained in the risk section of the disclosure document, but a decision was made against this approach, fearing too much loss of information. [1]

4 ESG Assessment

The municipal bond risk assessment from an ESG (Environmental, Social, Governmental) standpoint was achieved by simply outputting the national risk index of the municipality. This index on a scale from 1 to 100 is calculated and maintained by the FEMA (Federal Emergency Management Agency), taking into account environmental hazards like droughts, hurricanes, earthquakes, floods and wildfires. Naturally, using this index ignores both the social and governmental component of ESG. Furthermore, important elements of environmental risk such as air and water quality aren't as prevalent. In detail, a comma seperated values (csv) file containing all NRIs was downloaded [1] to perform a lookup on. This entails that the indexes will most likely be out of date whenever the assessment is performed in the future. All in all, this is only a temporary solution, and if desired, can be expanded on in the future in numerous ways. For example, implementing a natural language processing model to assess risk, although it is unclear which labels to use for supervised learning.

5 Risk Assessment via Classification

The main part of the project was the default risk assessment using natural language processing, which is elaborated on in the following. The goal of this assessment is a successful classification of bonds into one of six categories on the Moody's scale from AAA to B. Supervised learning was used with the labeled data set consisting of rated bonds and their disclosure documents.

5.1 Data Acquisition and Composition

The acquisition of bond disclosure documents used for both training and testing was done using the MUNISRCH application of the Refinitiv database. After filtering for municipal bonds rated by any firm and released after 2010, any further inspection was completely manual. Only if a disclosure document with a proper risk section was available, could it be downloaded and used for training or testing. Because of this very time consuming process, only a total of 129 documents were acquired, of which 117 for training and 12 for testing. Of the 117 training documents, 56 were labeled AA which expressed in percentage is 47.9%, making up the largest portion of labels. 31 or 26.5% were labeled A, 10 or 8.5% AAA, 7 or 6% BAA, further 7 or 6% BA and 6 or 5.1% B. The reason for the uneven spread of labels is that most municipal bonds are rated as AA. As to what effect this skewed data set has on learning is further discussed in section 5.4. The testing data set was evenly spread, with 2 documents for every label.

5.2 Data Preprocessing

To make training the model possible, first the disclosure documents in PDF format had to be converted into a simple text format, which in this case was .txt. Furthermore, as mentioned in section 2, the raw text of the document needed to be condensed to information that was short enough to learn on and still contained enough information to deduce a risk assessment. A risk section, which is often contained in the documents was chosen for this purpose, however locating the section turned out to be not so trivial. This is because different municipalities publish substantially different documents: the risk section might have a completely different name and location across disclosures. Therefore a solution that is likely to find the risk section for documents,

[1] https://hazards.fema.gov/nri/data-resources#csvDownload

with the only similarity between them being that they are municipal bond disclosure documents, had to be found.
The concept that was employed in the end used that the risk sections of the disclosure documents often feature similar words with higher quantity than anywhere else in the document. Therefore, a filter containing these words was conceived and the occurrences of these words in a given frame of the document are counted. The frame with the highest amount of occurrences is likely to be the risk section. Obviously, the choice of which words appear in this filter greatly influences the ability of the program to locate the section. This choice was made after a manual inspection of a couple of documents, where it became evident that apart from obvious phrases like "risk factor" and "investment consideration", the topics of competition, natural disaster and in some cases also cyber-security appeared. To allow for declinations of the words, only the radicals were explicitly incorporated into the filter document. For a similar reason, the entire text of the disclosure document was transformed into lower case. This document resides next to the source code in the project hierarchy.
The issues with this solution is firstly, it does not always find the risk section (around on in ten times), and secondly it is computationally expensive. The reason for the high computational complexity is that the program iterates over the pages of the document, counting the occurrences of the words specified in the filter within a given frame of pages, which in the current version of the software is three. Therefore the program has to go through every word in a three page window as many times as there are words in the filter, as many times as there are pages in the document. It is therefore inadvisable to increase the size of the window, or the size of the filter in the current solution. To improve both of these issues, the iteration over frames stops after a maximum of 50 pages, since no document was found in which the risk section was located later, and sometimes the software would identify a false risk section in some appendix of a disclosure document, containing many occurrences of filter words because of a list or table.
Afterwards the part of the document that the software has identified as the risk section is further condensed into those sentences that contained the words contained in the filter. The reason for this is that in testing, this step had a positive effect on loss on unseen data. As for why this improves the results, a theory is that the unfiltered risk section still contained too many words, and therefore too much noisy information. The clear drawback of this approach is that it increases bias, as the sentences used for learning are indirectly selected by the person writing the filter. Finally the text is appended with the length of the risk section, as a correlation between this length and risk is conceivable, especially when taking into account that such a correlation did exist in the cybersecurity risk project by Florackis et al. (see section 3.1).
The labels of the documents where inferred, meaning that the respective category that a disclosure document belonged to was captured by the directory it was located in. This means that during pre-processing of the documents the directory structure had to be maintained.

5.3 Model Architecture

In the following section the chosen architecture for the machine learning model is explained. It is comprised of a text encoding part, which translates the text into a series of vectors called embeddings, a recurrent neural network with a long short-term memory, which account for the main part of learning, and a final layer which performs classification by condensing the output into a useful format.

5.3.1 Text Encoding with BERT

BERT (Bidirectional Encoder Representations from Transformers) is a masked language model which for this project was used to transform natural text into encodings. The purpose of creating such an encoding is to encapsulate the meaning of words into vectors, called embeddings. Embeddings use the fact that the meaning of words can be inferred from the words in the context. The fact that makes BERT exceptional in comparison with encoders that create static embeddings like word2vec of GloVe, which always encode the same word with the same vector, is that BERT creates so called contextual embeddings. When using these embeddings, every word will be represented by a different vector when it appears in a different context. Another particularity of BERT is the bidirectionality. Instead of relying only on the information that came before the current word to make a computation, the model attends to all inputs, both those that came before the current element of the

input and those that follow. Therefore, potentially useful contextual information that follows the current element is also taken into account. To understand how BERT and other similar language models create an embedding that captures the contextual meaning of a word, how these models are trained has to be examined first. The models in question are trained to solve the problem of correctly predicting a missing or replaced word within a sentence. For this purpose a very large corpus of sentences is used, for example the English Wikipedia. For BERT in particular, 15% of words are replaced. Of these, 80% are replaced with the `[MASK]` token, 10% are replaced with a random word from the vocabulary and the final 10% are left unchanged. The model then outputs a probability distribution over every word in the vocabulary, with the words which the model presumes to be most likely for that position in the sentence given the context, receiving the most amount of probability mass. Figure 1 visualizes the procedure. Written at the very bottom is the original sentence to be trained with. Above that, some words have been replaced, which the model tries to correct. Finally, the softmax function translates the output of the encoder into a distribution over the vocabulary that sums up to one, resulting in a probability for every vocabulary entry. After the model has been pre-trained one way to use it is for text encoding. Every

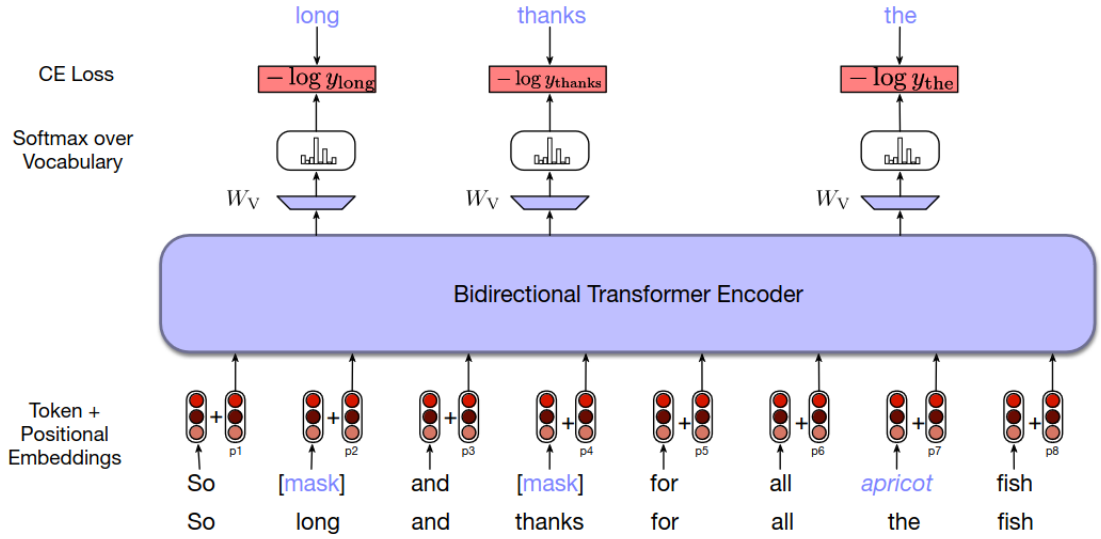

Figure 1 Masked language model training. [4, Figure 11.5]

token is therefore encoded with a vector the size of the vocabulary, representing a probability distribution over that vocabulary. These embeddings are what encapsulate the meaning of the words that were encoded. [4, Chapter 11] [2]

Specifically for the risk assessment BERT with Talking-Heads Attention and Gated GELU position-wise feed-forward networks was employed[2]. This model promises higher accuracy at the cost of taking longer to fine-tune.

5.3.2 Bidirectional RNN

The next layer after the tokens were encoded using BERT is a bidirectional recurrent neural network, or RNN for short. This hidden layer encapsulates the main part of the learning procedure. The main advantage of an RNN over a common neural network is that the size of the input can be completely variable, since it doesn't have to match a certain number of nodes. This is achieved by repeatedly using a single node, where the input is always a token and the state of the node from the previous time step, forming a cycle. The final output of the layer is then the state of the node after every element of the input was passed. The prevalent drawback of RNNs is the so-called vanishing gradient problem. Input elements that where passed through the node at the beginning have less impact on the final state as those that occur later. Therefore it is common to stack multiple

[2] https://tfhub.dev/tensorflow/talkheads_ggelu_bert_en_base/2

RNNs together or use a bidirectional RNN, as was done for this project. Here the input is passed through two RNNs, once from left to right and once from right to left. This way the vanishing gradient problem can be counteracted. This setup is visualized in figure 2. [4, Chapter 9.4]

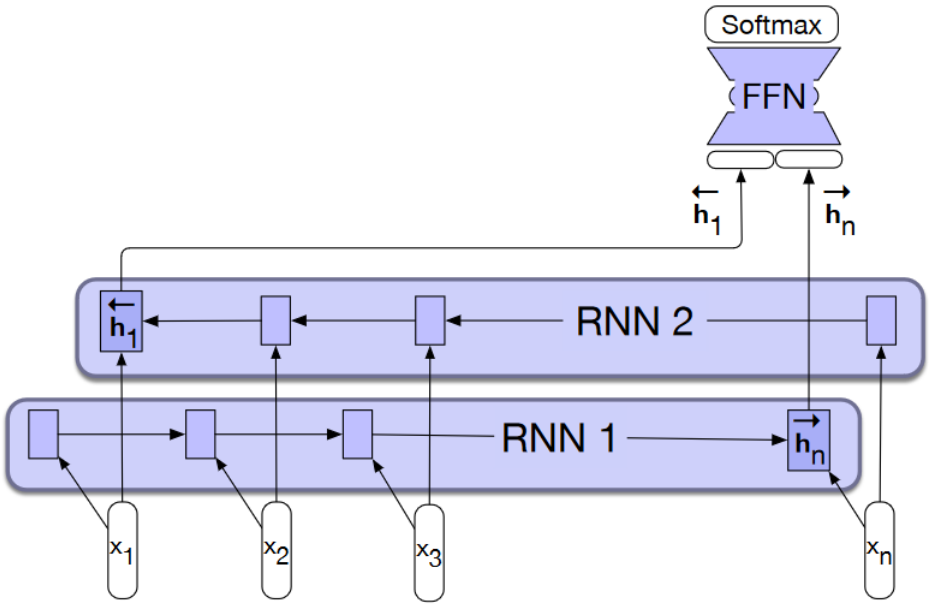

Figure 2 Visualization of a bidirectional RNN, which feeds its output to a feed forward neural network (FFN) [4, Figure 9.12]

5.3.3 LSTM

In similar fashion to the bidirectional approach for the RNN, the long short-term memory (LSTM) serves the purpose of making the model more resilient to the vanishing gradient problem. As Jurafsky and Martin write: "In practice, it is quite difficult to train RNNs for tasks that require a network to make use of information distant from the current point of processing. Despite having ac- cess to the entire preceding sequence, the information encoded in hidden states tends to be fairly local, more relevant to the most recent parts of the input sequence and recent decisions. Yet distant information is critical to many language applications." [4, Chapter 9.5] The LSTM is of especial importance for the task of assessing bond disclosure documents as the text that the risk assessment is being made on is often times very long. Therefore being able to distinguish important distant information to remember is vital. This memory effect is achieved mainly by gates, which are themselves small feed forward neural networks. The input is then passed through the gate and is then, depending on the learned weights of the gate, either passed through unchanged or erased. [4, Chapter 9.5]

5.3.4 Final Classification

Finally the output of the bidirectional RNN, which is of dimensionality two, is passed through a feed forward neural network to upsample the output for final classification. The final layer needs to be of dimensionality six, as the classification discriminates between six different classes. To be interpretable, this final vector has to represent probabilities. This is achieved by the softmax function, which outputs a vector where the entries all sum up to one. Each entry is then a probability of the disclosure document belonging to a specific class.

5.3.5 Training Parameters

The training of the model was done using batches of size 16. This means that for every epoch (round of learning) eight back-propagations to adjust the weights are performed. Usually, larger batch sizes improve performance and accuracy, but increase the amount of memory needed. The batch size of 16 was the maximum that could

reliably be handled by the VRAM of my graphics card. The pre-fetching of the batches was set to autotune, which dynamically tunes the amount of elements to be pre-fetched at run time, improving performance. Furthermore the training data set was shuffled to achieve that every batch has documents of different categories. The seed for this shuffle was randomly set to 42. The categorical labels of the data were inferred as mentioned in section 5.2. Finally 10% of the training data set was reserved for a validation data set, which helps to judge the training process and set meta-parameters as they are not learned on. All of these parameters are set in the code snippet below, which is taken from `classifier.py`, lines 92-105.

```
AUTOTUNE = tf.data.AUTOTUNE
BATCH_SIZE = 16
SEED = 42

raw_train_ds = tf.keras.utils.text_dataset_from_directory(
    training_directory,
    labels='inferred',
    label_mode='categorical',
    batch_size=BATCH_SIZE,
    validation_split=0.1,
    subset='training',
    shuffle=True,
    seed=SEED
)
```

The remaining parameters and how they are applied to the procedure is apparent in the following code snippet, also from `classifier.py`, lines 146-155.

```
epochs = 10
steps_per_epoch = tf.data.experimental.cardinality(train_ds).numpy()
num_train_steps = steps_per_epoch * epochs
num_warmup_steps = int(0.1 * num_train_steps)

init_lr = 3e-5
optimizer = optimization.create_optimizer(init_lr=init_lr,
                                          num_train_steps=num_train_steps,
                                          num_warmup_steps=num_warmup_steps,
                                          optimizer_type='adamw')
```

The number of epochs was set to 10, meaning that the model experiences 10 rounds of learning over the data set to tune its parameters. This figure was determined through testing, where it became apparent that after 10 epochs not much new is learned. The learning schedule was set to the same as BERT uses in its pre-training: a number of warm-up steps with an initial learning rate which decays linearly. The original BERT paper by Devlin et al. recommend a learning rate of either 5e-5, 3e-5 or 2e-5, whichever achieves the best results [2, Appendix 3]. In this case 3e-5 performed the best. Finally an optimizer had to be chosen and this choice fell on AdamW, which is the optimizer that BERT was originally trained with [2, Appendix 2]. AdamW or Adaptive Moments is an optimizer which minimizes prediction loss and achieves regularization via decoupled weight decay [5]. The loss function to be optimized was categorical cross-entropy loss.

5.4 Results and Evaluation

5.4.1 Evaluation of Training

A graph showing both loss and accuracy of both the training and validation data set can be seen in figure 3. The decrease of loss and increase of categorical accuracy in a monotonous fashion indicates that the training of the model with labeled data had a positive effect. After the ten epochs the model could therefore recognize and categorize seen data without making mistakes. However ten epochs is a lot for successful training. Devlin et al. state that for fine-tuning between two and four epochs should already work well [2, Appendix 3]. Furthermore as can be seen with the validation data set, loss and accuracy does not improve over the epochs for data that is not being trained on. This allows the prediction that it is possible that the model will not perform well on new, unseen disclosure documents.

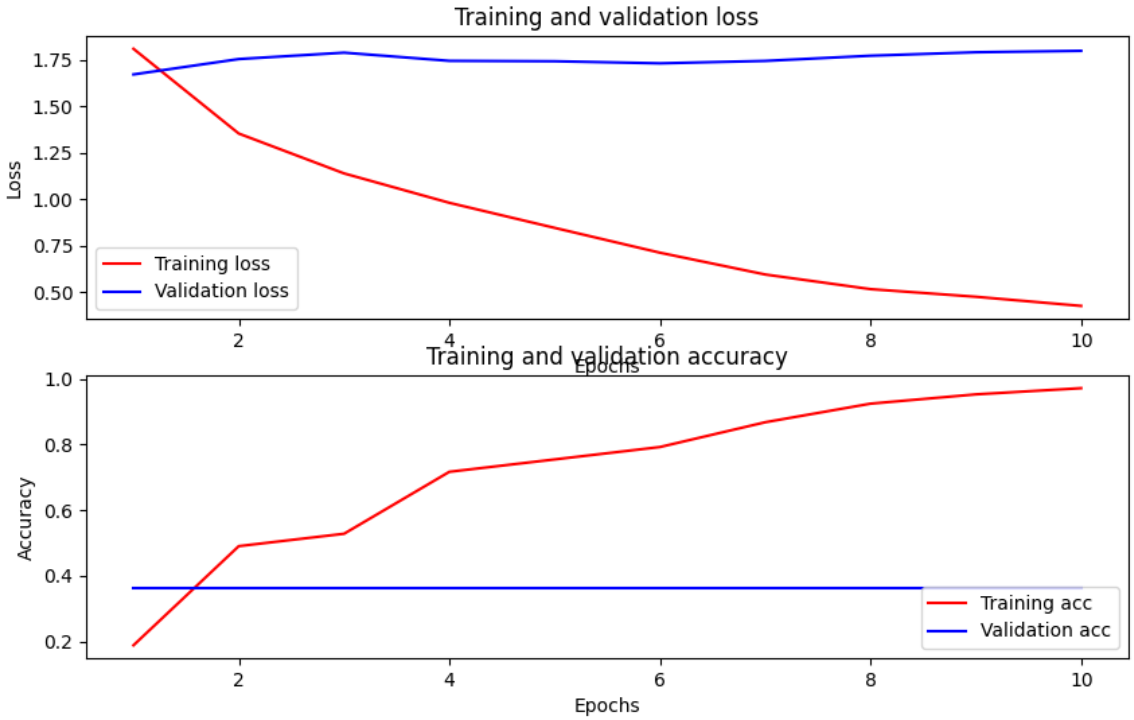

Figure 3 Loss and accuracy of training (red) and validation (blue) data sets over the epochs.

5.4.2 Evaluation of the Model

The evaluation using the evaluation data set, which is of even composition regarding the six categories yielded the following results. Across all documents an accuracy of 25% was achieved, which is exactly one correct classification better than the expectation of guessing randomly, which would yield 16.6%. Furthermore the average categorical cross-entropy loss was 2.17, which when compared to the graph in figure 3, is high. A positive and a negative example of the evaluation shows well where the model performs well and were it does not. First of all the example where the prediction was off:

```
Probability for AAA: 0.03
Probability for AA: 0.62
Probability for A: 0.24
Probability for BAA: 0.04
Probability for BA: 0.02
Probability for B: 0.05
Label: AAA
```

Here a clear bias towards AA can be observed, with almost no positive probability shifted towards the correct category, which was AAA. In this case the model guessed the classes proportionally to their occurrences in the training data set. Most likely, there was very little for the model to infer from the document and in these cases the skewed data composition becomes a problem. Furthermore this missing inference could stem from that not enough training samples were seen to cover the vast differences between the documents, as they can vary a lot between municipalities and states. On the other hand it is also possible that there is no proper correlation between these disclosure documents and the rating, as there is more to the matter of a financial risk classification than just the information given in the disclosure document. Also in the world of finance, such a link is less dependent on single used words, and is much more complicated. For example, in the typical usage of natural language processing to classify movie reviews, where single words carry the entire sentiment, this is much easier.
However the following positive example shows that maybe there is merit to the solution concept:

```
Probability for AAA: 0.04
Probability for AA: 0.17
Probability for A: 0.47
Probability for BAA: 0.04
Probability for BA: 0.09
Probability for B: 0.19
Label: B
```

Unfortunately due to the black box nature of machine learning, only speculation is possible here. The model decided to assign a disproportionally high amount of probability to A, and a disproportionally low amount to AA. Even more interestingly, a for the amount of documents in the category very high probability was assigned to B, which was also the true label. Obviously there were some words or combination of words which led the model to predict a higher risk for this disclosure document.

Another thing to address is the bias that is created by filtering the risk section before using it for training and classification. Because of that, all of the result s presented are dependent on the words in the filter document, and by extension the person who created the filter.

An additional reason for the poor performance is that of label selection. First of all there is the issue of which company's rating to use as a label. As most bonds were rated by Moody's, these were the ratings used. Then the question of what kind of rating should be used had to be answered: just for Moody's there is Short-Term Issuer Rating, Long-Term Issuer Rating, Derived Seniormost Tax Backend Rating and Derived Seniormost Revenue Backend Rating. Most of the bonds were rated via tax backend rating and some via revenue backend rating, with the other two being more rare. If a single bond was rated in more than one category, the label was chosen by overall frequency of the rating method: tax backend over revenue backend over long-term, with short-term being the least frequent. However this process of labeling still introduces selection bias. The final issue with label selection is that the bonds in question are rated multiple times across their lifetime and this rating can change drastically without that being reflected in the disclosure document, as can be seen in figure 4 The obvious choice therefore is to always select the rating as a label that was made closest to the bonds release, as this should be reflected best by the disclosure document. However, what if an investor would like to buy the bond in the middle of its lifetime and before doing so, assesses the risk using the machine learning model. There is likely to be a large discrepancy between what the model predicts and what a rating firm would actually assess. This reinforces the point that there is low correlation between label and document.

6 Future Work

There are a couple of improvements possible to the project. First of all the ESG assessment could be achieved using machine learning as was originally planned. For this purpose however, which labels should be used to perform the prediction is unclear. The most important improvement to the default risk assessment is to automatize the retrieval of disclosure documents. This would entail writing software that crawls the database for labeled documents and downloads those with a suitable document, without creating duplicates. The duplicates are an issue because oftentimes a bond gets issued multiple times with a slightly different CUSIP. Furthermore only documents with proper risk sections would need to be collected. All in all this would diminish the bias that is created by unbalanced training data. However the identification of the risk section remains a problem, especially if the question is deciding if the document has a risk section or not. For this purpose and also to increase the performance of risk section location, I propose a machine learning solution, where the model receives a page of a bond disclosure document and has to solve the binary prediction if the page is part of a risk section or not. Another improvement to the model would be to switch from a categorization task to a regression task, as the labels are not nominal but in fact ordinal. Therefore a simple numerical encoding of the classes is necessary.

RATINGS		
Rating Source	Date	Rating
S&P Long-term Issue Credit Rating	10-Jul-2023	NR
S&P Long-term Underlying Rating	10-Jul-2023	NR
S&P Long-term Underlying Rating	21-Nov-2022	AA-
S&P Long-term Underlying Rating	13-May-2021	A+
S&P Long-term Underlying Rating	13-Apr-2018	A
S&P Long-term Underlying Rating	14-Dec-2017	CCC
S&P Long-term Underlying Rating	26-Sep-2017	CC
S&P Long-term Underlying Rating	14-Sep-2017	B-
S&P Long-term Underlying Rating	11-Jul-2017	BB
S&P Long-term Underlying Rating	15-May-2017	BBB-
S&P Long-term Underlying Rating	22-Sep-2016	BBB
S&P Long-term Underlying Rating	11-Mar-2016	A+
S&P Long-term Underlying Rating	05-Mar-2014	AA-
S&P Long-term Issue Credit Rating	24-May-2013	AA
S&P Long-term Underlying Rating	24-May-2013	A

Figure 4 Rating history of bond 416415DV6. Ratings done by S&P. Picture taken from the Refinitiv database.

7 Conclusion

In conclusion the project yielded mixed results. Positive is definitely that a machine learning model was produced that evidently has the ability to learn and make the correct assessments on seen data. However the good results of the Cybersecurity Risk paper by Florackis et al. [3] could not be reproduced. If the reason for this is lacking correlation between label and data, or the model was not perfectly designed and implemented would have to be examined by introducing the improvements talked about in the future work section. My personal opinion is that an accurate prediction of unseen documents is possible, but if the model were to make investment recommendations, these predictions would have to be nearly perfect. This near perfection I personally do not see as feasible, as long as the disclosure documents remain completely unstandardized and vastly different from each other. Financial risk assessment happens, in my opinion, to be too complicated of an issue to solve with a simple natural language processing model.

References

[1] D. Araci. “Finbert: Financial sentiment analysis with pre-trained language models”. In: *arXiv preprint arXiv:1908.10063* (2019).

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

[3] C. Florackis, C. Louca, R. Michaely, and M. Weber. “Cybersecurity risk”. In: *The Review of Financial Studies* 36.1 (2023), pp. 351–407. DOI: https://doi.org/10.1093/rfs/hhac024.

[4] D. Jurafsky and J. Martin. *Speech and Language Processing*. https://web.stanford.edu/~jurafsky/slp3/. Jan. 2023.

[5] I. Loshchilov and F. Hutter. "Decoupled weight decay regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[6] C. Marohn. *America's Growth Ponzi Scheme*. https://www.strongtowns.org/journal/2020/5/14/americas-growth-ponzi-scheme-md2020. 2018.

[7] S. Towns. *America's Suburban Experiment*. https://www.strongtowns.org/curbside-chat-1/2015/12/14/americas-suburban-experiment. 2015.