Predicting Business Sales Outcome Using Emails and Historical Data

David Assaraf and **Benjamin Liu**

Abstract

In this work, we extract contextual information from emails in order to understand the incentives of customers through email exchanges and historical data. Varying the methods for aggregating email exchanges, we created different representations for this unstructured data. We also used different approaches in combining structured data and unstructured data in order to benefit from the different representations. Starting with a baseline model of 0.72 test AUC, the different incremental changes we made to the modelling phase led to a 0.89 test AUC. We outline here next steps to be conducted in our research in order to enable results interpretability for better adoption from companies.

1 Introduction

The emergence of large scale NLP models such as Transformers (Vaswani et al., 2017) allowed the NLP practitioners to extract information from much larger settings. In this setting, the data used in order to extract information has widened across diverse sources and for different purposes. In this work, we will leverage email data between customer and vendors in order to understand the incentives of customers for buying this product. The objective is, given some email data and some historical data of a customer, to predict whether or not this customer will buy our product or not. This project is a part of a wider project. Indeed, from the probability we predict for the customer to buy or not our product, we can build a model in order to predict the willingness to pay of a customer and use this as a pricing model for the different product and services offered by the company. This allows to standardize the pricing grid of large scale companies, that suffer the large setting of their vendors, and the usual behavior of pricing the items on-the-go. Using Transformer architectures in order to leverage the amount of data we have (400GB of email exchanges), we will efficiently try to extract relevant representations for the text email data to perform downstream classification. Another challenge in our setting is to combine the historical data we got from this customer with the email data and we will try different architectures in order to solve this task.

2 Related Work

Email classification at a large scale have been done by transfer learning. In the work by Liu et al. (2019), it is mentioned the challenge of sales engagement emails classification tasks due to the complexity of content, context, and the lack of standardized large corpora. Their work went for transfer learning implementations, and compared feature embeddings using fine-tuned and not fine-tuned BERT, ELMo, Flair, and GloVe. Their results showed that the fine-tuned BERT model outperforms the other ones for larger sample sizes.

In this project, we deal with both structured and unstructured text data. The task of combining these 2 types of data is usually discussed in the medical field where clinical notes are modeled together with structured electronic health records. Zhang et al. (2020) mentioned in their work that often research studies build models upon either structured or unstructured data, and many neglected the potential value of utilizing both in a fusion model combining structured and unstructured data. This work experimented on a Fusion-CNN and a Fusion-LSTM model combining structured and unstructured data, but improved the AUC no more than 5% in general compared to using either structured or unstructured data. It has been a dif-

ficult task to integrate these two, and there has not been a state-of-the-art method to do so.

3 Data Overview

Our data is from Tata Communications who sells its service/products to various companies, which we will call customers. The data is mainly divided into 2 parts: unstructured email data and structured tabular data. The following sections will give an overview of these 2 types of data as well as the label we use for supervised learning. An important record associated with both types of data, which is also a term that we will use throughout this project report, is what is called an *opportunity*. An opportunity is essentially a sale of a service/product. Assuming a customer is making inquiries on one of Tata Communications' product, there will be emails exchanged between the customer and Tata Communications asking for information or negotiating the price. There are also emails exchanged within Tata Communications about this potential sale. This sale is an *opportunity*, and the emails mentioned are associated with this opportunity. There could also be structured tabular data recorded about this opportunity, such as information about the customer. 1 opportunity can be matched to multiple emails, not necessarily a fixed number. 1 opportunity is only matched to 1 row of the structured tabular dataset. There exist cases where some emails or structured data are not matched to an opportunity, but in this project we will drop those data and assume that every email and structured data correspond to an opportunity. After a sale is closed, the opportunity will be labeled as a binary indicator showing success/failure of this opportunity's outcome. 1 opportunity uniquely has 1 outcome label.

3.1 Unstructured Email Data

We have 14,754,617 raw emails with a total size of 206.54 GB before labeling them into success/loss of a business opportunity. These are our unstructured data meaning that they are purely texts and the length for each email varies. The length distribution of these raw emails is shown as the histograms below in Figure 1. We only show the emails with length smaller than 1000, but in reality there are a small number of emails whose length can be as long as about 5000. Since there are many emails with length below 10, we also split the histogram to length 0-10, and 10-1000 to better show the length distribution. There are currently 6,461,981 emails labeled with success/loss and matched to 53,333 opportunities. These 6M emails are the ones we are interested in using for the classification task. Section 3.2 will further discuss the matching process.

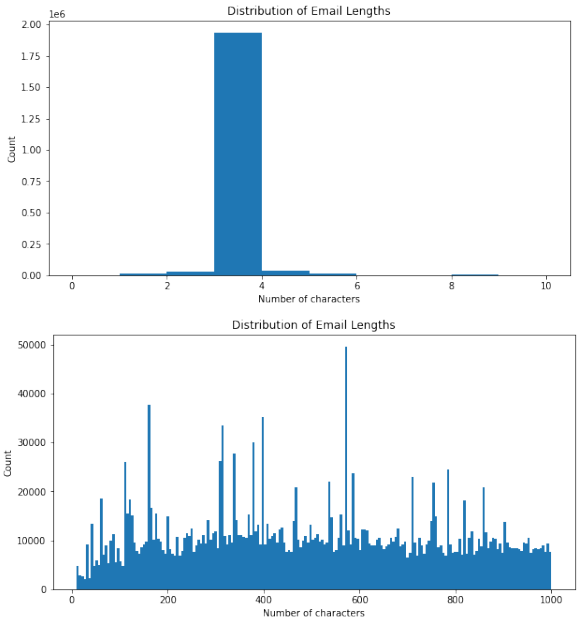

Figure 1: Upper plot: distribution of email length (0 - 10 characters); lower plot: distribution of email length (10 - 1000 characters)

3.2 Data Pre-processing

3.2.1 Preparing the raw emails for modelling

The first step of our pre-processing pipeline was to clean the 14M emails we had in our database. The emails have specific vocabulary that can render the modelling difficult. The first step was to remove overlong emails, that are recognized to contain low information about customer incentives. In a same setting, we remove short emails (that correspond to 'FYI' emails for example). Then, we cleaned the actual content of the emails. It consisted in removing the different formulas for introducing or concluding an email. It also consisted in parsing the different emails in order to remove the weird characters that were coming from the HTML text of the emails (the emails were loaded through Elastic Search which is robust to such HTML text). Last, we also removed the stop words. This entire cleaning pipeline takes around 72 hours to run and we used the Harvard servers in order to load and store the data (about 400GB of email text). We decided not to use lemmatization in order to let the different methods for creat-

ing embeddings take care of that. Then, a mandatory step was to prevent our model from suffering from information leakage. Indeed, at inference time, we want Tata to be able to use our model *during* the negotiation step and not once the negotiation is done. Since we have data at hand for closed opportunities, the emails we had spanned the entire negotiation process, until the very end. Therefore, the last emails contained information about the prices agreed, or negative contents for failed opportunities. Including these emails would not make any sense since a simple Rule Based System on the last emails would be enough to categorize these opportunities. Therefore, we used a time buffer in order to select the emails where the negotiation step was happening. We used some heuristics indicated by a Tata representative, and selected only the emails spanning from in the 30%-70% range of the entire emails for this opportunity.

3.2.2 Creating the opportunities

The data we received consist of 3 data frames. First, we have an email data frame with all the emails, each corresponds to an *email ID*. Second, we have an account data frame, mapping *email IDs* to their corresponding *opportunity ID*, which uniquely identifies a business sales opportunity such as new orders or renewals from clients. Third, we have an opportunity data frame, mapping *opportunity IDs* to its binary *outcome*– success or failure.

Our pre-processing goal is to have a mapping of emails to its corresponding opportunity and then to its binary success/failure *outcome*. To accomplish this, we join the first 2 data frames by *email IDs* and then join the third data frame by *opportunity IDs*.

The 14M emails in total include various types of opportunities such as renewal and new order. This project is mainly interested in finding the predictive power of business emails for *new order* opportunity outcomes. We have a list of 53,333 opportunities that correspond to new orders. We filter the data frame by these *opportunity IDs* and the resulting data frame has 6,461,981 emails. Each email corresponds to an *opportunity ID* and a binary *outcome*. Each *opportunity ID* can correspond to multiple emails, meaning they span multiple rows.

3.3 Structured Data

The structured data has 151,900 rows and 1319 columns. There is 1 observation/row per opportunity. It is in tabular form uniquely identified by *opportunity IDs*. It can be joined to the email data by *opportunity IDs*. The 1319 columns consist of information related to the opportunities, such as date created, date closed, price, various account IDs and types, etc. We then filter the column by the following keywords in Table 1 which are deemed as significant by Tata Communications for this classification task. We will keep the column if the keywords are a subset of the column name. We end up having 265 columns for structured data.

Year
Service_Record
Num_Circuits
Account_Industry
Month
Country

Table 1: Keywords to identify significant structured data columns

3.4 Labels

Our response variable, or labels, are binary, indicating success or failure of an opportunity. 1 opportunity has 1 label. This also means that 1 label is matched to 1 row of the structured data, but can be matched to multiple emails.

In reality, making prediction at email level would not be reasonable, since the outcome labels are at opportunity level. It would not make sense if multiple emails belonging to the same opportunity were predicted to have different outcomes. Thus, we are interested in making the prediction at opportunity level. This means that we need a method to pool emails belonging to the same opportunity together. More on the pooling step will be discussed in Section 4.1.1 and Section 5.

Table 2 shows the distribution of the binary labels in unstructured and structured datasets. We consider the labels to be relatively balanced.

4 Modelling

4.1 With non fine tuned embeddings

4.1.1 Creating the non fine tuned embeddings

Our initial email database was composed of about 53000 opportunities, amounting up to 6M emails. The lengths of the emails varied from 5 tokens up

Data	Label	Percentage
Unstructured Data	0	43%
	1	57%
Structured Data	0	46%
	1	54%

Table 2: Distribution of binary labels. 0 indicates the failure of an opportunity; 1 indicates the success of an opportunity

until 1345 tokens (we are talking here about the bert-base-uncased tokenizer from Hugging Face). The first step of this baseline model was to create embeddings for these different emails. We did so using a pre-trained DistilBERT model (Sanh et al., 2019), because it was more light weight than the BERT model and faster to perform inference using this model. The constraints we had when creating these embeddings is that the maximal number of tokens for one tokenized email should be 512. We did not want to spend too much time creating the embeddings for this baseline model so we removed the emails that had more than 512 tokens (we did not want to over-architecture the baseline). The DistilBERT created one contextual embedding representation for every token in the tokenized email. In order to get an embedding representation for one email, we used the representation of the CLS token, following the approach of (MacAvaney et al., 2019). Therefore, we ended up with an embedding vector of size 768 for every email in the different opportunities. This procedure took about 36 hours to run on a single K80 Tesla GPU.

The last step of data preparation was to create a representation of one opportunity based on the representations from the different emails of this opportunity. We did so using a simple mean pooling scheme: for one opportunity, we computed the average of all its email embeddings. As a result, we got a vector embedding with 768 features for one opportunity. The resulting data was of size (23000, 768). For these opportunities considered, the setting of the classification was 46%/54% so no imbalance issues here.

4.1.2 Unstructured data baseline model

Our baseline model for unstructured data leveraged only the email data in order to perform classification for success of the opportunity. Our architecture was a Feed Forward Neural Network with 3 layers.

In order to get a reliable baseline model, we decided to perform Hyperparameter tuning for the model. We used Bayesian Optimization in order to accelerate the searching process. We optimized over the following hyperparameters: weight decay, batch size, dropout, learning rate, optimizer (Adam, SGD or Adam with scheduler). Please see below for a sketch of the HP tuning procedure. The results we got were

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	36608
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8256
batch_normalization (BatchNo	(None, 64)	256
dense_2 (Dense)	(None, 1)	65

Total params: 45,185
Trainable params: 45,057
Non-trainable params: 128

Figure 2: Architecture of the Feed Forward Neural Network we used for Unstructured model baseline

Val AUC	Val Accuracy	Test AUC
0.73	0.68	0.72

Table 3: Results of the baseline model

4.1.3 Structured data model

The next step of our work was to capture the predictive power contained inside the structured data. As a reminder, for every opportunity, we had 265 structured features coming from customer historical data. After an initial filtering step where we selected the opportunities for which we got embeddings, the resulting data was of size (17000, 286). The setting of our classification was 46%/54%. Since the data was very sparse, we tried different approaches in handling this data: Sparse Auto-Encoders (Ng and others, 2011), Principal Component Analysis (Jolliffe, 1986) but none of them yielded good results. A much simpler useful technique that yielded good results was data normalization (Sun, 2019). We used the same Feed Forward architecture for the Neural Network and performed HP tuning on a large setting (batch size, dropout, learning rate, optimizer and weight decay). The results of the HP tuning procedure are presented here in Figure 3

From these results, we can see that the structured data had quite a lot of predictive power, and that using a single Neural network on structured data allowed to have a very good predictive AUC.

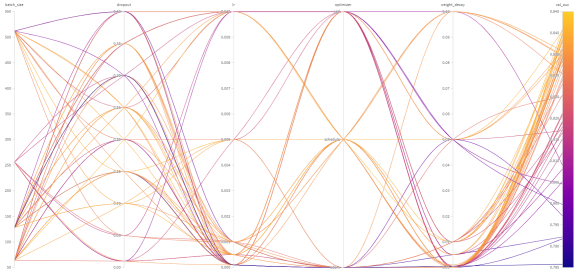

Figure 3: Hyperparameter tuning for the structured data model

Val AUC	Val Accuracy	Test AUC
0.84	0.78	0.84

Table 4: Results of the Structured data model

4.1.4 Combining structured and unstructured data

The last step of the baseline model is to find a way to effectively combine the structured and unstructured data into a big model.

Approach 1: Non Modular approach

The first approach we have used in order to do so is a non-modular approach, in the sense that it imposes some structure on the structured and unstructured data. We gather the embeddings created by DistilBERT for the opportunities and combine them with the structured data for this opportunity. We end up with a feature vector of size 1053 for every opportunity. The dataset size is therefore (17000, 1053) and we use a Feed-Forward Neural Network on this data set. This approach is non modular in the sense that it enforces to use a FFNN for both the structured and unstructured data. Here, we used a FFNN with Batch Normalization (Ioffe and Szegedy, 2015), dropout and specific kernel initialization. A sketch of the architecture is presented in Figure 4.

Once having set up this architecture, we performed a large scale hyperparameter tuning, optimizing over batch size, dropout, learning rate, optimizer (Adam, SGD, or Adam with Learning rate scheduler) and weight decay. The best results we got are presented in Figure 4.

Val AUC	Val Accuracy	Test AUC
0.88	0.80	0.87

Table 5: Results of the ensemble Neural Network after a large scale hyperparameter tuning.

The results here are good in the sense that we successfully combined the strengths of the two

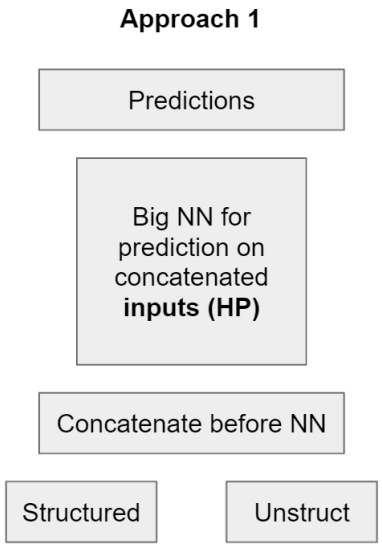

Figure 4: Overview of the Ensemble Learning architecture 1. Here, the structured data is the 285 features for one opportunity and the unstructured data is the 768 features stemming from the mean pooling strategy of the different DistilBERT embeddings of emails inside the same opportunity

types of data in order to create a robust model that will be able to capture patterns whether they come from structured or unstructured data. However, one potential issue with this architecture is that it enforces to use a Feed Forward Neural Network for both the structured and the unstructured data. If we were to use some recurrent architecture for the unstructured data (inherently sequential), we would need a more modular approach.

Approach 2: Modular approach

Therefore, we needed an architecture that would combine two independent Neural networks, one having been trained on Structured data and the other having been trained on the Unstructured data. Our architecture comes from the idea that a we can view a trained Neural network as a feature map. Indeed, when we train a fully connected neural network for binary classification, what we are doing with all the layers except the last one is to create a representation $\phi(x)$ for our input x. Using sigmoid activation and Binary classification loss, we will enforce this representation to map our input space $\mathcal{X}$ to a mapped space $\mathcal{X}^*$ where the two classes are linearly separable. We introduced a toy example here for the sake of presentation.

Here, we can see that the inputs in Figure 5 are clearly non-linearly separable (there are two classes; one red class and one blue class). On top of this, we are going to train a Neural network for binary classification (ie trained with the Binary

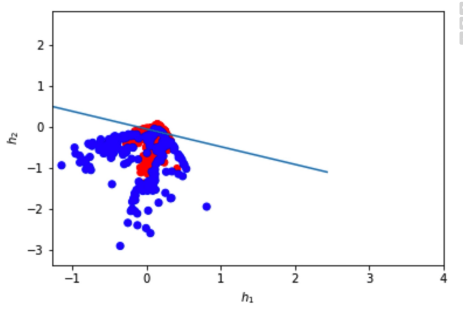

Figure 5: Visualization of the inputs of a binary classification

Cross Entropy loss). What we made specific about this Neural Network is that the penultimate layer has only 2 nodes, in order to be able to visualize the feature map $\phi : R^2 \mapsto R^2$. We visualize here this feature map at the end of the training.

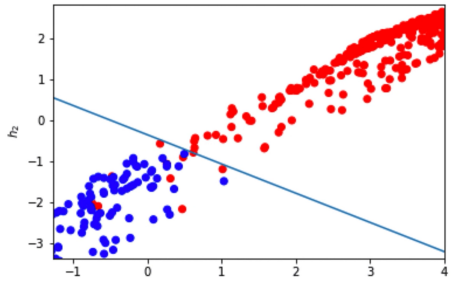

Figure 6: Visualization of the feature map created by a NN trained for binary classification

Therefore, after having witnessed the strengths of Neural networks for representation learning and taking into account the fact that we want to use two different architectures for the structured and unstructured data, we used the following idea: we consider one neural network for the structured data and another for the unstructured data. We train them to perform classification on their own. Then, we chop off the prediction head of both networks in order for both networks to behave as feature maps. Then, the forward pass will consist as follows: go through the structured NN and the unstructured NN and create the features for both the structured and the unstructured NN. Then, concatenate these features and train a last prediction NN on the concatenated features. We visualize the architecture in Figure 7.

Practically speaking, we implemented this in the following way:

- Perform a 70/15/15 train/validation/test split of your data. Make sure that these three ensembles are composed of the same opportunities for both the structured and unstructured data
- Train a NN on the structured data and perform HP for it (please refer to 4.1.3)
- Train a NN on the unstructured data and perform HP for it (please refer to 4.1.2)
- Remove the prediction heads for both NN and freeze all the weights inside those
- Train the last NN (prediction NN in Figure 7) on the same training set and perform HP for it

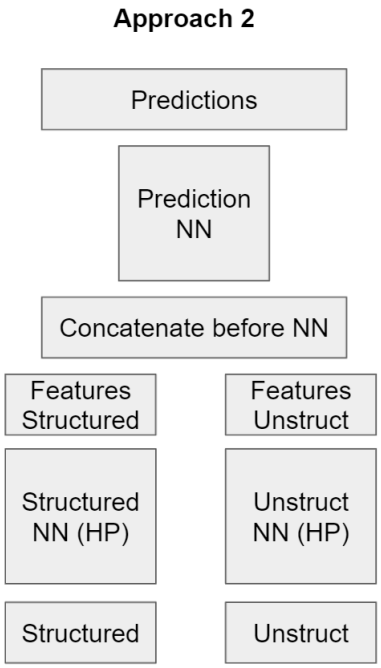

Figure 7: Overview of the Ensemble Learning architecture 1. Here, the structured data is the 285 features for one opportunity and the unstructured data is the 768 features stemming from the mean pooling strategy of the different DistilBERT embeddings of emails inside the same opportunity

The results we got are presented here:

Val AUC	Val Accuracy	Test AUC
0.88	0.81	0.89

Table 6: Results of the ensemble Neural Network after a large scale hyperparameter tuning.

Therefore, this architecture got better results than the other way of performing Ensemble Learning. But this marginal increase is not the most important part of this architecture, it's its modularity. Therefore, in the remaining of our work, we will be able to use a sequential pooling strategy of the

different embeddings of one email inside one opportunity. This will create a sequential representation for the opportunity and therefore we will want to use recurrent architectures to deal with these representations. Doing so, we will be able to use this architecture when it comes down to combining structured and unstructured data.

4.2 Fine-tuning Masked Language Model

To get the representations of the email texts, we use a pre-trained BERT model due to its suitability for context-heavy texts. The BERT model is pre-trained with a large corpus (Devlin et al., 2019), which enables the model to learn the usage and structures of various words. We then fine-tune the BERT masked language model (MLM) so that it can learn our email corpus better. This helps us generate the embeddings of the emails that will be fed into our models for classification or combined with structured data.

The fine-tuning process consists of the following steps, inspired by Allhorn (2020). First we process the whole email corpus with 14M emails into one txt file and structure them into 1 sentence per line and insert 1 line break in between emails. Then we leverage the MLM from Hugging Face (Wolf et al., 2020) to run the MLM on our corpus. We fine-tune the MLM by randomly masking 15% of the tokens and performs a prediction of the masked token and learns from the loss, which we use perplexity. In order to find the best combinations of learning_rate, optimizer, batch size, epochs, and max sequence length, we run a Bayesian hyperparameter optimization using the WandB package. Due to the cost of time and computation, we ran a baseline fine-tuning using a small subset of the corpus.

Based on the hyperparameters suggested by the Bayesian optimization, as well as considering the computational difficulty on our machine, we selected the following hyperparameter values in Table 1 to use while fine-tuning the MLM.

Hyperparameters	Values
batch size	8
epochs	4
learning rate	5e-05
max sequence length	512
optimizer	Adam

Table 7: Hyperparameter used to fine-tune the MLM.

Plotting the perplexity loss across epochs, we observe a well trained MLM with a decreasing loss, meaning the BERT model now is better understanding the email corpus than the pre-trained BERT.

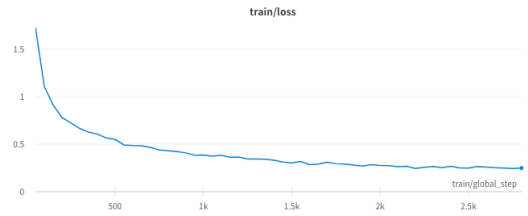

Figure 8: Masked language model fine-tuning loss

4.3 Results with fine-tuned embeddings

Using the fine-tuned embeddings, we went through all the different modelling steps in order to witness the impact of fine-tuning the embeddings specifically for our text data. We show here the results for the model for unstructured data only and the second approach of Ensemble learning (we will no longer be using the first approach for ensemble learning).

Val AUC	Val Accuracy	Test AUC
0.79	0.72	0.79

Table 8: Results of the unstructured data model using fine-tuned embeddings

As expected, fine-tuning the embedding for our specific email dataset did improve a lot the predictive performance of the model relying only on the text data. We got an improvement of 7% test AUC which is significant. This shows how important fine-tuning the BERT embeddings to our specific case is.

Val AUC	Val Accuracy	Test AUC
0.91	0.82	0.89

Table 9: Results of the ensemble model using fine-tuned embeddings

However, when it comes down to the improvements of the EL model, the results are less significant. This is due to the fact that the EL model gives more importance to the features coming from the structured data since they have more predictive power. Including the unstructured data yielded better results but improving the results of the unstructured data was not significant.

5 Next Step

As mentioned in Section 4.1.1, mean pooling scheme is used in both the baseline models and the models using fine-tuned embeddings. While mean pooling the email embeddings gives us relatively well results, as further work, we would like to use dynamic pooling strategies such as Bidirectional LSTM to generate the span representation across emails in the same opportunity. Also, it would be interesting to add attention mechanism (Toshniwal et al., 2020) at email level to understand which emails are weighted more. Inspired by the work of Lee et al. (2017), a concatenation of the outputs of the bidirectional LSTM, further concatenated with the weighted sum of word vectors in the span and a feature vector encoding the size of the span, can give the attended span feature representation.

6 Conclusion

With a large number of email data and structured tabular data, we proposed an ensemble learning method that outperformed the models using either emails or structured data alone, with a test AUC of 0.89. Our ensemble learning is proven to be able to extract more values from both types of data. We also improved the embedding strategy by fine-tuning a masked language model. Using mean pooling to get the opportunity level embeddings, models using fine-tuned embeddings generate better performance, especially for the unstructured data model, improving from the baseline test AUC of 0.72 to 0.79. The test AUC for ensemble model stayed at 0.89 due to the nature of the ensemble model focusing heavily on the structured data.

References

Arndt Allhorn. 2020. *Transfer Learning for Hate Speech Detection*. Ph.D. thesis, MA thesis. Beuth Hochschule für Technik Berlin.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR.

Ian T Jolliffe. 1986. Principal components in regression analysis. In *Principal component analysis*, pages 129–155. Springer.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark, September. Association for Computational Linguistics.

Yong Liu, Pavel Dmitriev, Yifei Huang, Andrew Brooks, and Li Dong. 2019. An evaluation of transfer learning for classifying sales engagement emails at large scale. In *2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 542–548.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. Cedr: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1101–1104.

Andrew Ng et al. 2011. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ruoyu Sun. 2019. Optimization for deep learning: theory and algorithms. *arXiv preprint arXiv:1912.08957*.

Shubham Toshniwal, Haoyue Shi, Bowen Shi, Lingyu Gao, Karen Livescu, and Kevin Gimpel. 2020. A cross-task analysis of text span representations.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October. Association for Computational Linguistics.

Dongdong Zhang, Changchang Yin, Jucheng Zeng, Xiaohui Yuan, and Ping Zhang. 2020. Combining structured and unstructured data for predictive models: a deep learning approach, 08.