FRANKFURT UNIVERSITY OF APPLIED SCIENCES

Natural Language Processing for ESG Goal Extraction

Module Name - Project
Semester - WS 2022-23
M.Eng. Information Technology

Contributors:
Shubham Pramod Suryawanshi - 1384291

Guidance:
Prof. Dr. Gabriela Alves Werb

Contents

1. Introduction

A company or organization generally publishes an annual ESG report or Sustainability report to inform an audience about its environmental, social and governance (ESG) impacts. This type of reporting focuses on three core areas: environmental impact (carbon footprint and other ecological effects), social (local contributions to the community), and governance (transparency). It enables the company to reflect on its impacts at the same time it also helps the shareholders and other audiences to assess company policies. It could potentially be helpful to the investors as well to assess the financial health of the organization or the company.

Usually, these reports are hundreds of pages long, thus an evaluation of such document to gather future goals and aspirations of the company or organization is a time-consuming process. The objective of the project is to develop a python application that capitalizes on Natural Language Processing(NLP) techniques to extract goals from a PDF document like the ESG report. The application will be fed an ESG report as a document and a text file containing extracted goals will be generated at the end of the inference process. This modular method allows multiple files to be read by the application and generate respective text files for each document.

As part of this project, I have developed a 4 stage NLP pipeline and a python application to support it. I have ensured that the application is compact, easy to use and can be easily recreated for a given environment. The pipeline consist of 4 stages and the tokenized text is passed through 3 transformer based models. The transformer models used for first and second stages are trained and maintained by spaCy[1].

The final stage comprises of a fine-tuned BERT[2] model. As there is no readily available dataset with annotated goals, for the fine-tuning process a dataset was curated by me. It had 4500 samples with 20 % of the samples being goals. A separate validation set was also created, it comprised of 208 samples with 50 % of the samples being goals. In Deep Learning, early stopping is a form of regularization commonly used to avoid overfitting when training a learner with an iterative method, such as gradient descent. Such optimization methods update the model so as to make it better fit the training data with each iteration. Up to certain iteration, the training improves the model's performance on data outside of the training set. However, past that iteration, improving the model’s fit to the training data comes at the cost of increased validation error. Early stopping helped me to avoid this scenario.

2. Project Plan

To start off the project, publicly available ESG reports from companies and organizations from a variety of industries were collected. It was important to maintain a variance in the information to avoid bias regarding a single company or industry. The DAX[3] index was primary focus group for selecting companies during the information gathering phase. The DAX is a stock market index consisting of the 40 major German blue chip companies trading on the Frankfurt Stock Exchange.

After the information-gathering process, I started exploring previously implemented and widely used NLP techniques as well as other avenues to extract goals from a given corpus of text. After reviewing a few techniques, it occurred to me a custom NLP pipeline is required for this specific task. To embody and support this NLP pipeline appropriately, a python application is required, it should be compact, portable and versatile.

As part of this project, I developed a packaged versatile python application that could ingest data in multiple ways. This application can be used in interactive mode, file mode, or directory mode. In interactive mode the application accepts a string given by the user as a terminal argument, this string is processed to verify if it is a goal or not. In file mode, the application process a single PDF file on the given path, and in directory mode, and entire directory is scanned for PDF files ignoring other files. In directory mode, for each PDF file process, there is a respective report created with extracted goals. This flexibility makes the application very easy to use and portable.

Finally I designed an NLP pipeline, wherein each stage would assist in certain way to narrow down the token sequences to extract goals from entire text. For example, the coreference resolution stage is essential for the second stage (Named Entity Recognition) to operate successfully. Thus, each stage complements the following stage in certain way, implementing and placing these stages in a strategic fashion significantly assisted in optimizing the entire process.

The final stage of the NLP pipeline comprises of a fine-tuned BERT base model. After training this base model on a custom dataset, it was validated on multiple test datasets. To ensure it performed consistently across wide range of data, cross-validation techniques were also used as part of this project.

3. NLP Pipeline

In this project the fundamental part of the python application is a 4 stage NLP pipeline. The pipeline is designed to process raw text from a given ESG report and extract goals, then the python application populates a text file with the extracted goals.

When the text is extracted from a given sample ESG report, A large proportion of these sentences or token sequences are not of interest to us as they are not goals. Therefore, it is instrumental to narrow down the number of sentences which could potentially be goals. This would not only reduce the processing time for a given ESG report but significantly increase the computational efficiency of the inference cycle for the transformers models.

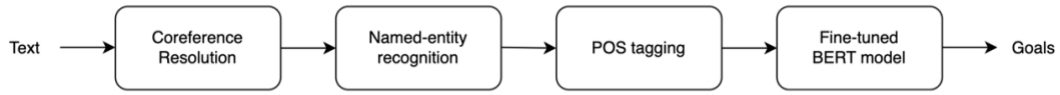

Fig 1: Overview of the NLP pipeline stages

As described in the figure, there are four prime stages that process the raw input text extracted from the ESG report pdf file. Initially after converting the text a python string object it is tokenized and processed in the first stage by a spaCy coreference resolver transformer model. After replacing the recurring occurrences for the entities with the respective entity names, the string is untokenized at the end of first stage. This untokenization process is essential for second stage to process successfully. In the second stage of the pipeline the locations mentioning the organization or company withing the text are highlighted. The third stage tags all the tokens into predefined tagging categories (nouns, verbs, adjectives, etc.). The second and third stages allow us to filter token sequences where the organization is drive an action. At the end of three stages we have all the actions statements enabling us to perform and efficient classification in the fourth stage. The classification is performed by a fine-tuned BERT model. After all the stages the extracted goals along with other processing statistics are stored in an output text file.

4. Coreference Resolution

Coreference resolution[4] is the process of determining which elements in a given piece of text refer to the same entity (Object, Personality or Organization). It is a type of natural language processing (NLP) procedure that is used in many applications, including machine translation, question answering, and automated summarization. Coreference resolution is also a way to increase the accuracy of text understanding systems by helping to identify and disambiguate references to entities in text. I have used this stage in the NLP pipeline to enhance the ability of further stages to located statements where the company or organization is directly involved.

The coreference resolution process begins with the identification of referents in a given piece of text. This is done by parsing the text and applying rules to determine which elements in the text refer to the same entity. Different types of rules can be used for this purpose, such as grammatical rules, semantic rules, and statistical rules. For this stage a Robustly Optimized BERT (RoBERTa)[5] based transformer model maintained by spaCy (en_coreference_web_trf[6]) is used. After the referents have been identified, the coreference resolution process can then be used to determine which of the identified referents refer to the same entity.

The primary motivation to add this stage in the NLP pipeline for this project was to improve the performance of the further stages. I experimented with an NLP pipeline by skipping this stage and a significant drop in the quality of extraction for action statements was observed. Along with better accuracy, this stage also saves a remarkable amount of computational overload and processing time.

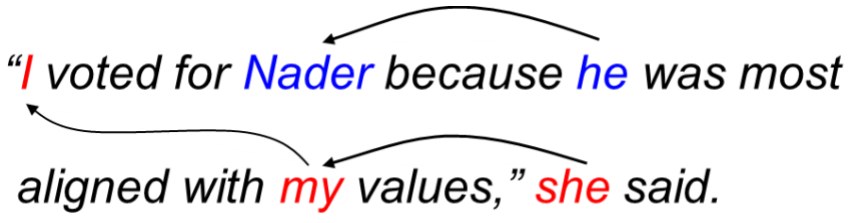

Fig 2: Example of Coreference Resolution process[7]

5. Named Entity Recognition

Named-entity recognition (NER)[8] is a type of information extraction procedure that seeks to identify and categorize named entities mentioned in an unstructured piece of text into pre-defined categories such as personalities, organizations, geographical locations, time, quantities like monetary values, proportions, etc. In the context of this project, this stage would enable us to locate sentences which include the organization or the company of interest, as these sentences are more likely to be goals as compared to other general sentences.

Fig 3: Example of Named Entity Recognition process

For this stage of the NLP pipeline, the core English pipeline (en_core_web_trf[9]) by spaCy is employed. The prime component of this pipeline is a RoBERTa based transformers model for performing a variety of tasks, one of these tasks is NER. This stage would highlight the company tokens that belong to 'Organization' category, the process is demonstrated in the above schematic. This component has been utilized to identify statements which have at least one mention of the organization or the company, as these statements are of interest to us as they have a higher chance of being a goal compared to other generic statements.

6. Part-of-speech (POS) Tagging

The process known as part-of-speech tagging (POS tagging)[10] involves classifying words into their parts-of-speech classes or lexical categories and labeling them accordingly. The predefined collection of tags is known as a tagset. The primary interest of the project is to extract goal statements, and it is observed that generally almost all of the goal statements have an action token preceded by an organization token. In this project, the process of POS tagging has been used to identify such sequences where the Organization is the primary driver of the action.

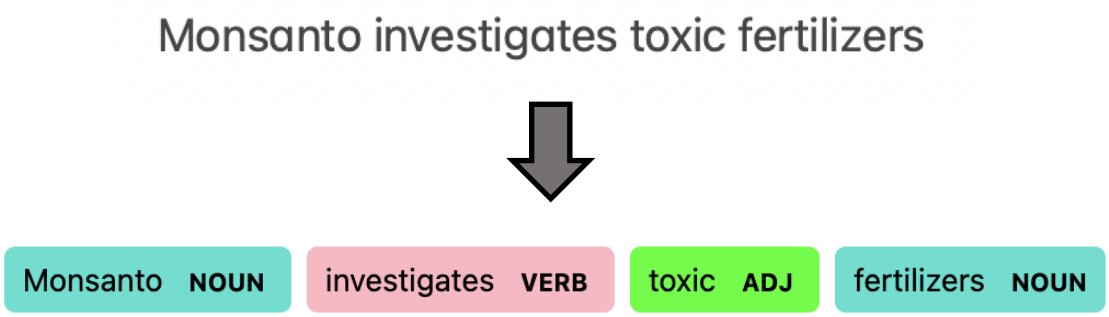

Fig 4: Example of Part-of-speech Tagging process

The process known as part-of-speech tagging(POS tagging) involves classifying words into their parts-of-speech classes or lexical categories and labeling them accordingly. The predefined collection of tags is known as a tagset. Capitalizing on the versatility of the spaCy pipeline used in the previous stage, it has been reused to perform POS tagging in this stage. This not only reduces the footprint of the application but also reduces disk IO as no new model has to be reloaded in the heap memory.

The primary interest of the project is to extract goal statements, and it is observed that generally almost all of the goal statements have an action token preceded by an organization token. In this project, the process of POS tagging has been used to identify such sequences where the Organization is the primary driver of the action.

7. BERT base model

As part of the final stage of the NLP pipeline, I have used a fine-tuned BERT model as a classifier for separating goals from action statements extracted by previous processes. As a precursor to this stage, I have used a BERT base model (uncased), which is a pre-trained model on the English language using a masked language modeling (MLM) objective. It has over 110 million trainable parameters. BERT base mode was originally released with multiple variations, for various languages and case structures in the input text. Along with cases of tokens, the uncased models also ignore the accent markers in a given text. One can use the raw base BERT model for masked language modeling or next sentence prediction as explained below, but as part of the project, it was fine-tuned for a classification downstream task. It is fine-tuned to use the whole sentence(masked or unmasked) to perform tasks like sequence classification. In this project, the fine-tuned model consumes the entire token sequence, to decide if the sequence structure resembles that of a goal statement.

The BERT base model is a transformers model pre-trained on a very large corpus of English text in a self-supervised technique. This means it was trained on the raw texts only, with no human intervention in the labeling process. In other words, it was trained with an automatic process that generated inputs and labels from those texts. More precisely, it was pre-trained with two objectives:

1. Masked language modeling (MLM): In this method, the model takes in a sentence as input, then 15% of the words/tokens in the input will be randomly masked. Then the entire masked sentence is run through the model. Now the task for the model is to predict the masked words. This method allows the model to learn a bi-directional representation of the sentence. It is different from traditional recurrent neural networks (RNNs) which usually consume the tokens in a sequential pattern. In the case of autoregressive models like GPT masking process for future tokens takes place internally.
2. Next sentence prediction (NSP): The model concatenates two sentences as inputs during pretraining phase. Sometimes the text(as a result of the concatenation of two sentences) being fed as input has consecutive sentences. The task for the model is to predict whether the provided sentences were following each other or not.

8. Fine-tuning process

I have used a BERT base model (uncased) and fine-tuned it further using a custom dataset for goal extraction. As this is a downstream classification task (classifying token sequences: goal or not goal), I fine-tuned the base model to act as a classifier. There are significant benefits to using a pretrained model. It reduces computation costs, avoids the hassle to maintain dataset, and allows you to use state-of-the-art models without having to create one from scratch.

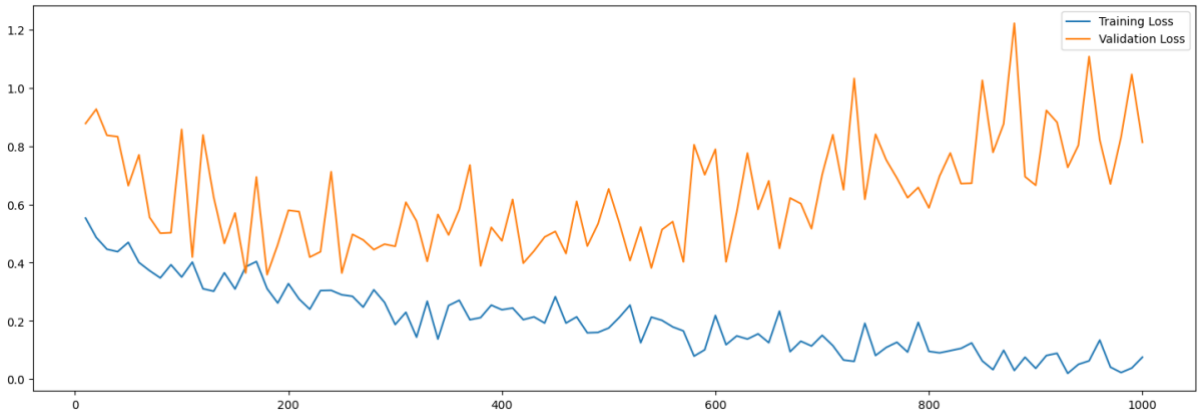

Fig 5: Training steps of BERT base model

Hugging Face[11] is a platform to share datasets and interesting models with other people. Along with being a platform to share interesting artifacts, it also has an open-source python module called Transformers API. This modular API is specially built and optimized to perform NLP tasks. This API also has a Trainer class[12], which is an easy-to-use wrapper over PyTorch models and datasets. This Trainer class was used for the fine-tuning process over the curated datasets containing annotated goals.

After some experimentation, it was observed the BERT base model started to overfit after 160-220 iteration steps, it can be seen in above figure. Due to the problem of over fitting, early stopping procedure was used to find the best model after the fine-tuning process.

9. K-Fold Cross Validation and Testing

In the field of Statistics and Machine Learning, the term generalization usually refers to the ability of a ML classifier to be effective across a wide variety of inputs. This implies that the ML classifier does not encounter performance degradation when provided unseen inputs from the same distribution of the training data.

Cross-Validation is one of the techniques which can be used to verify generalization in a classifier. It is easy to comprehend and implement and it also tends to have a lower bias than other validation methods. These features make cross-validation technique a powerful tool to choose the best preforming ML model. There are a variety of different techniques available that can be used to perform cross-validation process. Still, all of them have a similar algorithm, below steps describe general cross-validation process:

1. Divide the dataset into two parts: one for training, other for testing.
2. Train the model on the training dataset.
3. Validate the model on the testing dataset.
4. Repeat 1-3 steps such that each data point is part of both the training and testing datasets at least once.

As part of this project, to validate the performance of the trained classifier we have used a variation of cross-validation process called K-Fold cross validation. In this process the data is divided in k equal size data points (or folds), all the folds except the one fold is used for training and the remaining fold is used as validation set. At the end of this process and average of performance describes the overall performance of the classifier.

K-Fold cross validation technique has a drawback, increasing the number of folds or the k value results in too many folds or partitions. Thus, the repetitive process of training and validation is time-consuming and resource intensive as well. This problem is highlighted when heavy transformer models containing tens of millions of parameters is used for classification purposes. Therefore, the number of folds have to be chosen after considering factors like time-consumed for a single training and validation session and other factors.

10. Results

A number of test runs and inference runs were conducted after finalizing the pipeline and choosing the optimal BERT classifier after cross-validation process. A readily available and curated dataset for goal extraction was not available, reusing certain portions of the manually curated dataset were used to validate and test the NLP pipeline. Following figure and metrics describe how effective the NLP pipeline performs on a 208-sample dataset. The result is indicative of applications performance during inference runs.

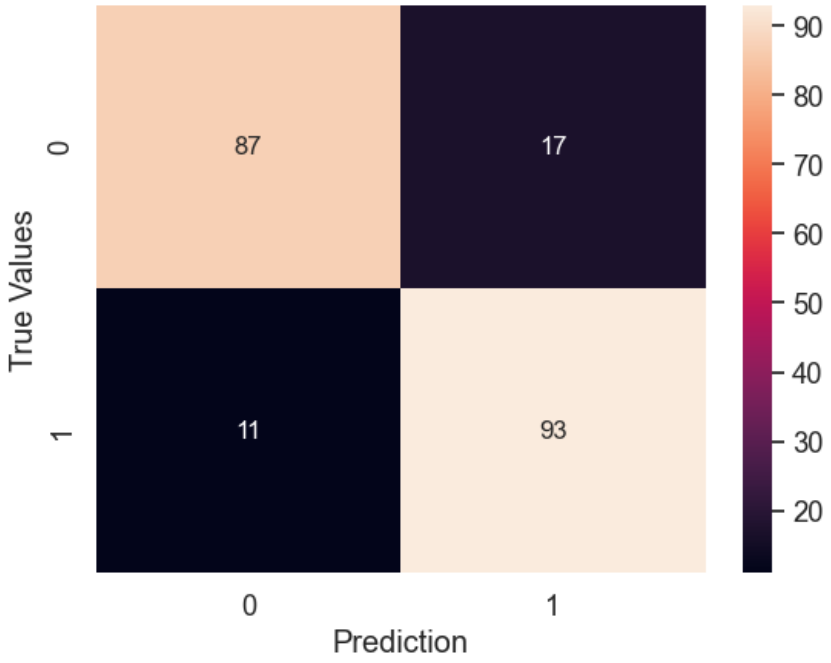

Fig 6: Confusion matrix after an inference on a balanced test dataset

Metric	Score
F1 Score	0.87
Accuracy	0.87
Precision	0.85
Recall	0.89

Fig 7: Classification metrics achieved on a balanced test dataset

11. Conclusion

Extracting goals from a fairly large piece of text like an ESG report is a complex task. As part of this project, I have implemented an NLP pipeline which can perform this task and achieve impressive classification scores. It was observed that along with the fine-tuned BERT classifier model, the initial stages of the NLP pipeline contribute significantly to the performance and efficiency of the entire process. A suitable python application was designed around the NLP pipeline to support it in subprocesses like disk I/O and interacting with user if necessary. The python application is compact and portable, so that it could be easily recreated on a given environment.

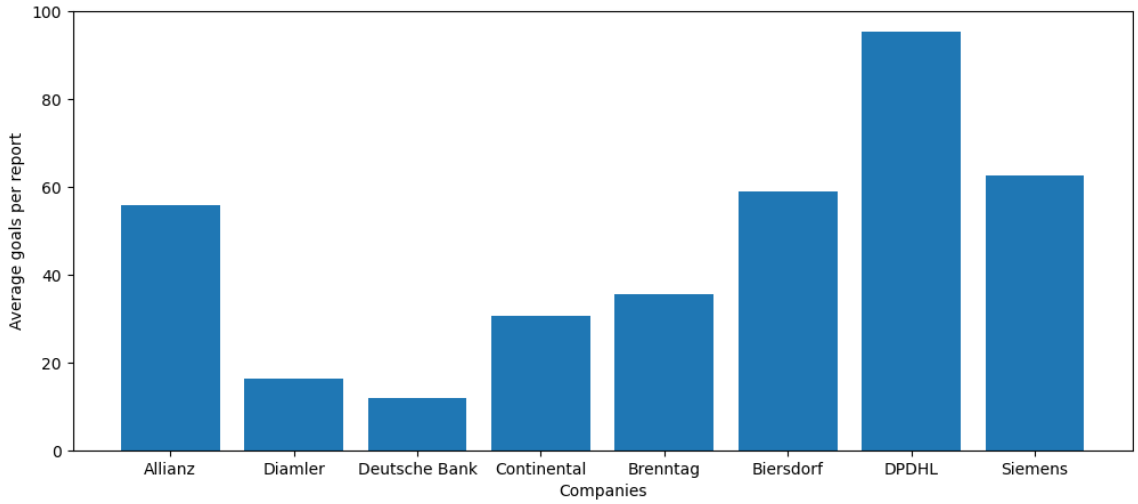

Fig 8: Average goals in a report for each organization

The figure no. 8 describes the organizational behaviour towards goals and commitments in their respective ESG reports. Such interesting analysis can be swiftly done using the NLP pipeline designed as part of this project. A tool can further be built embodying this technique for ever more versatility and other use cases as well.

It was observed during the fine-tuning process of BERT base transformer model that the model is prone to overfitting, the cause could be limited number of annotated samples or lack of variance in the token sequences. A larger dataset would have allowed for desirable amount of bias in the dataset used for fine-tuning.

12. References

Sr. No.	Reference Description
1	Open-source contributors (no date). spaCy [Online]. Available at: https://spacy.io (Accessed: 03 April 2023)
2	Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Online]. Available at: https://arxiv.org/abs/1810.04805 (Accessed: 03 April 2023)
3	Open-source contributors (no date). DAX [Online]. Available at: https://en.wikipedia.org/wiki/DAX (Accessed: 03 April 2023)
4	Sobha Lalitha Devi, António Branco, Ruslan Mitkov (2009). Anaphora Processing and Applications. A Deeper Look into Features for Coreference Resolution. Publisher: Springer Berlin, Heidelberg.
5	Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach [Online]. Available at: https://arxiv.org/abs/1907.11692 (Accessed: 03 April 2023)
6	Open-source contributors (no date). Coreference Resolution in spaCy [Online]. Available at: https://explosion.ai/blog/coref (Accessed: 03 April 2023)
7	Kevin Clark, Dan Jurafsky, Christopher Manning, Christopher Potts (no date). The Stanford Natural Language Process Group [Online]. Available at: https://nlp.stanford.edu/projects/coref.shtml (Accessed: 03 April 2023)
8	Satoshi Sekine, Elisabete Ranchhod (2009). Named Entities: Recognition, Classification, and Use. Publisher: John Benjamins Publishing Company
9	Open-source contributors (no date). spacy/en_core_web_trf [Online]. Available at: https://huggingface.co/spacy/en_core_web_trf (Accessed: 03 April 2023)
10	Claude Sammut, Geoffrey I. Webb (2010). Encyclopedia of Machine Learning. POS Tagging. Publisher: Springer New York, NY
11	Open-source contributors (no date). HuggingFace [Online]. Available at: https://huggingface.co (Accessed: 03 April 2023)
12	Open-source contributors (no date). Transformers [Online]. Available at: https://huggingface.co/docs/transformers/main_classes/trainer (Accessed: 03 April 2023)

13. Appendix