Analyzing News Articles Using NLP Techniques

Dhruv Jain, Edward Montoya, Nghi Nguyen, Tam Huynh

Department of Applied Data Science, San Jose State University

DATA 298A: MSDA Project I

Professor Simon Shim

2023, October 3

Contents

Abstract

Several researchers have been applying a diversity of Natural Language Processing (NLP) techniques to classify news articles. Understanding the importance of handling an enormous amount of text from various resources, this project is studied to help readers and researchers save time gathering useful information from English articles. The study of this paper differs from others by creating a combined system that is capable of classifying news topics, recognizing named entities, and summarizing the article content. Leveraging the power of multiple pre-trained models including BERT, RoBERTa, ERNIE, ELECTRA, T5, and BART, the research encompasses the mentioned tasks. The niche of this research is the implementation of Progressive Ensemble Fine-Tuning (PEFT) and Query-Level Optimization with Reinforcement learning Algorithm (QLoRA) has further enhanced the abstractive summarization task. After rigorous evaluation processes, this research has found that ERNIE is the best model for Text Classification task with 85.34% accuracy, ELECTRA is the best performance of 98.65% accuracy for Named Entity Recognition task, and Flan-T5-Small has reached 0.3438 in Rouge-1 and 0.3280 in Rouge-L for Abstractive Summarization task. This endeavor not only showcases the versatility of state-of-the-art NLP models, but also culminates in the delivery of an advanced news analyzing application.

1. Introduction

1.1 Project Background and Executive Summary

Natural Language Processing (NLP), a field of Computer Science that focuses and enables the interaction between human languages and computers, has been around since the 1950s (Hutchins, 2003) and has been developed ever since. Text classification, a task to assign predefined categories to documents, has been an important NLP task since the 1990s. One of the most significant and earliest work in this task is tackled using Support Vector Machines (SVMs) by Joachims (1998). During the same time, entity recognition, which is another NLP task that can identify and extract specific types of named entities from text, was also being researched. Grishman & Sundheim (1996), put the earliest works on this task to introduce a system for entity recognition in newswire text. Unlike text classification and entity recognition, text summarization techniques were touched on in the 1960s. Text summarization, which also belongs to the NLP field, is a task that returns a brief yet contains enough important information from an input document. Edmundson (1969), has one of the earliest works on the text summarization system for news articles. However, it was not until the 1990s that this task was actively studied.

Transfer learning is a machine learning technique that was first introduced from the word2vec model by Mikolov et al. (2013). This technique used unsupervised learning for words in the form of vector representations, which were later applied in other NLP tasks, including text classification, entity recognition, and text summarization. In transfer learning, knowledge is gained by training a model in one task and then applied to a related task. However, not until the introduction of transformer architecture was first introduced by Vaswani et al. in 2017 that transfer learning achieved state-of-the-art (SOTA) performance on a variety of NLP tasks.

Transformer architecture, which is based on the attention mechanism, allows the model to focus on different parts of the input sequence at different times during the processing step. Transformer has facilitated the development of pre-trained models, which are models trained on huge amounts of data, then are applied in domain specific tasks with smaller sets of data, also known as fine-tuning. Transfer learning with pre-trained models and fine-tuning on domain specific datasets on particular desired NLP tasks is the main approach for this project.

Nowadays in the fast-paced living world, there is more and more news shared online over thousands of resources. There are hundreds of categories for this news to fall under. With the fast-paced news spreading these days, people would need a tool to help them gain information from interested news categories and interested entities. Moreover, a long article would take a lot of time and energy to go over, not to mention the ability to remember and summarize important content. These days summarization is done manually by using freelance writers, which takes up a lot of time and money to get it done. But automating such a process would save both time and money, and also be able to increase the amount of articles that can be summarized in parallel. An application that has models with the ability of news classification (NC), named entity recognition (NER), and abstractive text summarization (ATS) to provide researchers and users important information would be significant. Apart from personal use, an effective classification and summary tool can help organizations, especially consulting companies, filter unrelated information, focus on the most important news of the day, and save time to read entire papers. As a result, the organizations can perform better in terms of customer experiences, time saved, and profits.

This project's ideal application includes creating a multi-label text classification to classify categories based on the five categories including World, Science-Technology, Business,

Entertainment, and Sports and evaluate the best performing pre-trained model among BERT, RoBERTa and ERNIE. These five classes are chosen due to their popularity and huge amount of available data. Moreover, the project aims to recognize mentioned entities from papers using the best pre-trained model among BERT, ERNIE, and ELECTRA. The application can detect a person, a location, an organization, and miscellaneous in a text input. In addition to NER, this project will provide users with ATS using the best suited model between BART and T5. Once a user puts in an input article, the app returns a summary of the text. In each of the tasks, there are appropriate metrics used to evaluate the performance of each model. In specific, Accuracy, precision, recall, and F-1 Score with a confusion matrix will be applied in text classification and entity recognition. Meanwhile, Recall-Oriented Understudy for Gisting Evaluation (ROUGE) and BiLingual Evaluation Understudy (BLEU) will be used for the evaluation of model performance in text summarization. In short, the goal of this project is to study and apply SOTA approaches in building an end-to-end analyzing news system to help researchers and readers gain the most accurate and useful insight from an enormous amount of news articles on a daily basis.

1.2 Project Requirements

The datasets that the group utilizes for this project are publicly available. Hence, there are no privacy concerns while using them to implement this project. The tools to be used in the project includes Python programming language on environments such as Jupyter notebooks and Google Colab, and AWS tools such as S3 and EC2. Flask is used during deployment. GitHub will be used for collaboration, utilizing pretrained models and to make the codes available for anyone interested in the project.

The project consists of three major NLP tasks. The first task will be focusing on classifying news articles into five possible multi-label categories; World, Sci/Tech, Business,

Entertainment, and Sports. The following models will be used for this task.

The first approach to the multi-label classification task will be handled with the use of a Bidirectional Encoder Representation from Transformer (BERT), which is a bidirectional transformer that can be used for various NLP tasks. Based on the research by Devlin et al. (2018), BERT was pre-trained on unlabeled text from the BooksCorpus, with 850 million words, and the English Wikipedia corpus, with 2,500 million words. BERT works by using masked language modeling (MLM), which means that the model will randomly mask input text for the purpose of allowing the model to predict words around those masked. BERT also makes use of next sentence prediction (NSP), which means that the model jointly pre-trains text-pairs and allows for generalization for natural language inference (NLI). Finally, BERT offers the ability to fine-tune for a particular task with just a single output layer. Meaning that BERT can excel in the news article multi-label classification with only modifications in fine-tuning, which makes it a very appropriate model for the first task.

The next model that will be implemented for the multi-label classification task will be another transformer called RoBERTa. The research by Z. Liu et al. (2021) on RoBERTa built upon the famous BERT model and modified it for the purpose of performing even better than BERT on NLP tasks. RoBERTa works by removing the next sentence prediction that BERT utilized and modifying hyperparameters and training data size. The researchers found that the original BERT model was undertrained, and by training their RoBERTa model for longer with more data and bigger batches, they could produce better results. This better performance justifies why it is best suited for the multi-label classification task and has the potential to be arguably the best performing model for the task once fine-tuning is applied.

Finally, our last model for the multi-label text classification task will be in the form of a SOTA approach. Researchers at Google have produced a model called the Pathways Language Model (PaLM), which is capable of performing NLP tasks without the need for a significant amount of task specific examples, which reduces the challenge of building and fine-tuning a very capable model. The PaLM model was trained using a new ML system called Pathways, which allows for highly efficient training and scaling of thousands of Tensor Processing Units (TPU) (Chowdhery et at. 2022). PaLM has demonstrated SOTA few-shot performance for many NLP tasks and should be very capable for the purpose of the multi-label text classification task.

The second task will be a form of token classification in which NER and chunking will be performed for the purpose of being able to identify which entity is speaking, their respective titles, and the tokens that correspond to the entity. This task will consist of BERT, ERNIE, and ELECTRA.

The first model for the token classification task will be BERT. BERT will act more like a baseline for this task, because of its general capability at performing well on NLP tasks. BERT will be fine-tuned differently than the prior BERT model to ensure that it performs well on this NLP token classification task, but it is unlikely that it will be the best performing model for this particular task.

The second model is ERNIE, which is shown to outperform BERT and achieve SOTA on various NLP tasks. ERNIE is popular for NER tasks on Chinese languages but it is not the same case on English datasets, it is worth exploring the model to see if it can achieve outstanding performance on English language NER.

The third model for this token classification task is ELECTRA. Clark et al. (2020) designed ELECTRA to be more efficient than BERT, which was achieved by the way that it

conducts its pre-training, leaving the rest of the model virtually the same as BERT. ELECTRA is essentially two transformer models, where one transformer acts as the generator role, where it replaces tokens in a similar fashion to BERT with a masked language approach, and the other transformer model acts as the discriminator. The discriminator is the model that is able to identify which tokens were masked by the generator. This pairing of transformers makes for a very efficient and capable model that works especially well for token classification type tasks.

The last NLP task that will be performed is text summarization. This task will consist of two separate transformer models. The first model for the text summarization task is BART. The pre-training scheme that BART utilizes is like BERT in that it uses a similar encoder, but differs by using a decoder that is similar to GPT (Lewis et al. 2020). During the pre-training phase BART shuffles the sentences randomly into new orders, and makes use of a novel in-filling scheme. This equates to a model that is very capable on text summarization tasks where BART has achieved up to 6 ROGUE scores.

Finally, the last model that will be implemented for the summarization task will be T5 (Raffel et al. 2020). Researchers from Google produced T5 as a pre-trained model that works by implementing multi-task pre-training. This means that T5 is pre-trained on unsupervised and supervised tasks. The researchers stated that multi-task pre-training is an optimal approach to training, and when fine-tuning is conducted afterward, the results can rival other SOTA models. On benchmarks like GLUE, SuperGLUE, and SQUAD, the T5 model is one of the best performing models. For summarization tasks, T5 excels and performs exceptionally well on the CNN Dailymail dataset, which is applicable to this project because of the domain. It is expected that T5 will perform very well on the summarization task.

Our project mainly focuses on transformers, which are pre-trained models. These models do not need other sources of data for training but require domain-related data to fine-tune the models to improve their capabilities. Hence, we utilize a corpus that consists of more than 1 million news articles accumulated from over 2000 sources in more than 1 year (Gulli, n.d.). The dataset consists of 1,281,103 rows and 9 columns and is in CSV format. Each instance in this corpus is a news article. This dataset is used for the classification task. Along with this corpus, we work on creating a corpus of our own by web-scraping major news providers. This corpus will focus on collecting data related to all our tasks and subtasks within the project, which help us demonstrate the system. The ConNLL 2003 dataset is used for the NER and chunking tasks. This dataset is a subset of Reuters corpus and consists of 1400 articles for the NER task specifically. Apart from these datasets, we also utilize the CNN Daily Mail dataset. This consists of 300,000 news articles from the CNN and Daily Mail newspapers. This dataset was created for the summarization task and is used for the same in this project

1.3 Deliverables

The project deliverables include several reports and presentations, model prototypes for each function, a committed code, a prototype of the application, a web-based application that is available to the public.

The reports consist of two progression reports, two final reports, and final presentation slides. The progression reports record details on all assignments for each section of the project and capture the issues, resolutions, and achievements during project conduction time. The first final report covers the following sections: Introduction, Data and Project Management Plan, Data Engineering, and a part of Model Development. The second final report includes all sections from the first report plus the remaining parts of Model Development, Data Analytics and

Intelligent System, System Evaluation and Visualization, and Conclusion. Last but not least, the final presentation slides contain the summary of each step in each period.

The model prototypes illustrate the back-end work of the project and are the best performing models after going through different project phases such as data cleaning, data processing, model training, and tuning. The committed code is the back-end code of the application and is pushed to a GitHub master branch. The code is the final version and ready for deployment.

The application prototype is a working beta version that will be tested and adjusted. The testing data are English news and articles from all around the world. The final is a web-based portal that allows interactions with users. The web-based portal has three main functions: articles classification, entity classification, and articles summary. First of all, the application allows users to input a news or article, and it will classify which category the article falls into. Next, the website lets users load an article and it will categorize each word or word group in the article; for example, the portal might detect if a word or word group in the text is a “person”, “location”, “organization”, or “other.” Finally, the application has the ability to summarize input news.

1.4 Technology and Solution Survey

Devlin et al. (2018) were researchers from Google AI who introduced Bidirectional Encoder Representation from Transformer (BERT) in 2018. According to the authors, BERT can improve NLP tasks’ performances such as General Language Understanding Evaluation (GLUE), MultiNLI accuracy, Stanford Question Answering Dataset (SQuAD v1.1 Test F1), and Stanford Question Answering Dataset 2.0 (SQuAD v2.0 Test F1). The paper mentions the three most popular pre-train approach groups for NLP tasks are unsupervised feature-based, unsupervised fine-tuning, and transfer learning from supervised data. The project’s framework

has two stages: pre-training in which the model is trained using unlabeled data and fine-tuning in which all parameters are tuned. The authors point out that in the pre-training stage, the two main tasks are Masked Language Modeling (Masked LM) and Next Sentence Prediction. For Masked LM, the paper explains that a percentage of words are masked, and the model will try to predict these masked words based on the context around the words. Next Sentence Prediction task is more advanced: it trains the model to predict if two sentences follow each other. In the fine-tuned stage, a task-specific output layer is added, and the model is trained on a small number of downstream labeled data. BERT is experimented with eleven different NLP tasks, and the results show that BERT is efficient and may achieve human-level performances on all these tasks.

Another research paper proposes a transformer-based methodology which combines robustly optimized BERT (RoBERTa) model and a recurrent neural network to identify figurative language (FL) forms including sarcasm and irony (Potamias et al., 2020). The authors refer that RoBERTa is a modification model of BERT, which uses ten times more data, 8 times larger batch sizes, is trained with many more epochs than BERT and so forth. In the paper, the proposed method, recurrent CNN RoBERTA (RCNN-RoBERTa), is a combination where RoBERTa maps words onto a rich embedding space and RNN captures the dependencies within RoBERTa's pretrained embeddings. The authors point out the drawbacks of previous projects such as handcraft engineered features or lexicon dictionaries or preprocess time consuming issues. In conclusion, RCNN-RoBERTa overcomes all these disadvantages and outperforms other traditional deep learning methodologies and machine learning methods such as RNNs, SVM, k-nearest neighbor, etc.

Clark et al. (2020) mention Masked Language modeling (MLM) and propose an alternative approach called, Efficiently Learning an Encoder that Classifies Token Replacements

Accurately (ELECTRA). The core idea of ELECTRA is shown in the paper as follows: the generator network creates some words which replaces the tokens in an appropriable way; then, the model predicts if each token of the input was replaced by the alternative word or not; after that, the discriminative model is transferred to the downstream tasks. In the paper, the model is evaluated on the General Language Understanding Evaluation benchmark and Stanford Question Answering dataset. The experiment shows that ELECTRA is more efficient than other techniques such as ELMo, GPT, BERT-Small, and BERT-Base with the same model size, data, and compute; for example, ELECTRA-Base has the highest GLUE score among all the mentioned models.

In another paper, Lewis et al. (2020) present BART as a denoising autoencoder whose pretrained model integrates Bidirectional and Auto-Regressive Transformers. The authors indicate that BART technique consists of two main steps: using an arbitrary noising function to corrupt the text and training a model to recreate the original text. In the paper, BART is compared with BERT on several NLP tasks such as SQuAD, MNLI, ELI5, XSum, ConvAIR 2, CNN/DM, etc. The result shows that BART is not only significantly effective at text generation but also good for comprehension tasks, especially when pre-train is scaled to large batch sizes. Also in the experiment, BART is as good as RoBERTa for discriminative tasks. Figure 1 below shows the comparison between BERT-Base and BART-Base pre-training models on different NLP tasks. Based on the results, out of six benchmarks, BART Base outperformed BERT Base in four benchmarks, that are SQuAD 1.1, XSum, ConvAI2, and CNN/DM. With BART being a better embedding model in CNN/DM, we are proceeding our project with BART on the summarization task rather than BERT.

Figure 1

Results from pre-training BERT and BART model.

Model	SQuAD 1.1 F1	MNLI Acc	ELI5 PPL	XSum PPL	ConvAI2 PPL	CNN/DM PPL
BERT Base (Devlin et al., 2019)	88.5	**84.3**	-	-	-	-
Masked Language Model	90.0	83.5	24.77	7.87	12.59	7.06
Masked Seq2seq	87.0	82.1	23.40	6.80	11.43	6.19
Language Model	76.7	80.1	**21.40**	7.00	11.51	6.56
Permuted Language Model	89.1	83.7	24.03	7.69	12.23	6.96
Multitask Masked Language Model	89.2	82.4	23.73	7.50	12.39	6.74
BART Base						
w/ Token Masking	90.4	84.1	25.05	7.08	11.73	6.10
w/ Token Deletion	90.4	84.1	24.61	6.90	11.46	5.87
w/ Text Infilling	**90.8**	84.0	24.26	**6.61**	**11.05**	5.83
w/ Document Rotation	77.2	75.3	53.69	17.14	19.87	10.59
w/ Sentence Shuffling	85.4	81.5	41.87	10.93	16.67	7.89
w/ Text Infilling + Sentence Shuffling	**90.8**	83.8	24.17	6.62	11.12	**5.41**

Note. Performance of BART on various datasets compared to BERT with F1, accuracy, and perplextiy. Adapted from "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," by Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L., 2020, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Copyright 2020 by Association for Computational Linguistics.

Few-shot learning is a term mentioned by Chowdhery et al. (2022) which significantly reduces the number of examples to train the models. To better understand the effect of scale on

few-shot learning, the authors use 540 billion parameters to train a Transformer language model called Pathways Language Model (PaLM). The model is trained by 780 billion tokens of high-quality text to scale up the modeling capacity of large language models. The paper refers that PaLM has two innovations: it reduces the number of computations by a hierarchical variant of softmax and introduces a parameter-efficient attention mechanism which allows the model to participate in a token subset in the input sequence. The experiments draw three important conclusions: first, the model has a capacity to improve with sufficient scale; second, the technique is beneficial for not only language generation tasks but also categorical prediction or regression; third, PaLM is a potential foundation to develop large scale systems for future generations of models.

Raffel et al. (2020) proposes a large-scale study of transfer learning in Natural Language Processing (NLP) tasks using a new model called Text-to-Text Transfer Transformer (T5). T5 model processes with an encoder-decoder structure that returns a sequence of text as output from a text input sequence. In addition to the encoder-decoder structure, T5's remarkable feature is the use of shared vocabulary, which maps both input and output text sequences to a common token space. This feature facilitates the seamless knowledge transfer across multiple NLP tasks. Based on special features of T5, the authors conduct a wide range of experiments on various NLP tasks such as text classification, question answering, and summarization to evaluate the performance of T5 and compare it to other state-of-the-art models including BERT, RoBERTa, and XLNet. After setting up multiple experiments and understanding the results, the authors conclude that the T5 model achieves strong performance across a variety of NLP problems, and the model's performance improves with more training data.

In the Figure 2 below, the authors achieved SOTA performance on 18 out of 24 tasks experimented. The largest model, T5–11B (with 11 billion parameters), performed the best among its variants of T5-Small, T5-Base, T5-Large, and T5-3B. In the figure, T5-11B reached SOTA performance on CNN/DM. It proves the ability for T5 to deal with text summarization task, hence we are proceeding with T5 in this specific task.

Figure 2

T5 Variants Performance Comparison with the Best Existing Approaches (as of October 24th, 2019)

Model	GLUE Average	CoLA Matthew's	SST-2 Accuracy	MRPC F1	MRPC Accuracy	STS-B Pearson	STS-B Spearman
Previous best	89.4[a]	69.2[b]	97.1[a]	**93.6**[b]	**91.5**[b]	92.7[b]	92.3[b]
T5-Small	77.4	41.0	91.8	89.7	86.6	85.6	85.0
T5-Base	82.7	51.1	95.2	90.7	87.5	89.4	88.6
T5-Large	86.4	61.2	96.3	92.4	89.9	89.9	89.2
T5-3B	88.5	67.1	97.4	92.5	90.0	90.6	89.8
T5-11B	**90.3**	**71.6**	**97.5**	92.8	90.4	**93.1**	**92.8**

Model	QQP F1	QQP Accuracy	MNLI-m Accuracy	MNLI-mm Accuracy	QNLI Accuracy	RTE Accuracy	WNLI Accuracy
Previous best	74.8[c]	**90.7**[b]	91.3[a]	91.0[a]	**99.2**[a]	89.2[a]	91.8[a]
T5-Small	70.0	88.0	82.4	82.3	90.3	69.9	69.2
T5-Base	72.6	89.4	87.1	86.2	93.7	80.1	78.8
T5-Large	73.9	89.9	89.9	89.6	94.8	87.2	85.6
T5-3B	74.4	89.7	91.4	91.2	96.3	91.1	89.7
T5-11B	**75.1**	90.6	**92.2**	**91.9**	96.9	**92.8**	**94.5**

Model	SQuAD EM	SQuAD F1	SuperGLUE Average	BoolQ Accuracy	CB F1	CB Accuracy	COPA Accuracy
Previous best	90.1[a]	95.5[a]	84.6[d]	87.1[d]	90.5[d]	95.2[d]	90.6[d]
T5-Small	79.10	87.24	63.3	76.4	56.9	81.6	46.0
T5-Base	85.44	92.08	76.2	81.4	86.2	94.0	71.2
T5-Large	86.66	93.79	82.3	85.4	91.6	94.8	83.4
T5-3B	88.53	94.95	86.4	89.9	90.3	94.4	92.0
T5-11B	**91.26**	**96.22**	**88.9**	**91.2**	**93.9**	**96.8**	**94.8**

Model	MultiRC F1a	MultiRC EM	ReCoRD F1	ReCoRD Accuracy	RTE Accuracy	WiC Accuracy	WSC Accuracy
Previous best	84.4[d]	52.5[d]	90.6[d]	90.0[d]	88.2[d]	69.9[d]	89.0[d]
T5-Small	69.3	26.3	56.3	55.4	73.3	66.9	70.5
T5-Base	79.7	43.1	75.0	74.2	81.5	68.3	80.8
T5-Large	83.3	50.7	86.8	85.9	87.8	69.3	86.3
T5-3B	86.8	58.3	91.2	90.4	90.7	72.1	90.4
T5-11B	**88.1**	**63.3**	**94.1**	**93.4**	**92.5**	**76.9**	**93.8**

Model	WMT EnDe BLEU	WMT EnFr BLEU	WMT EnRo BLEU	CNN/DM ROUGE-1	CNN/DM ROUGE-2	CNN/DM ROUGE-L
Previous best	**33.8**[e]	**43.8**[e]	**38.5**[f]	43.47[g]	20.30[g]	40.63[g]
T5-Small	26.7	36.0	26.8	41.12	19.56	38.35
T5-Base	30.9	41.2	28.0	42.05	20.34	39.40
T5-Large	32.0	41.5	28.1	42.50	20.68	39.75
T5-3B	31.8	42.6	28.2	42.72	21.02	39.94
T5-11B	32.1	43.4	28.1	**43.52**	**21.55**	**40.69**

Note. The performance of T5 variants on various datasets with BLEU and ROGUE metrics. Adapted from " Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," by Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J., 2019, arXiv.1910.10683. Copyright 2019 by arXiv.

Yamada et al. (2019) proposed a new pre-trained model called LUKE (Language Understanding with Knowledge-based Embeddings) for contextualized representation of words and entities. This approach has obtained state-of-the-art performance on five popular datasets for

NLP tasks including relation classification, entity typing, question answering, and named entity recognition. LUKE's architecture is based on transformer architecture like BERT; however, there are some added features that authors have proposed. In the paper, LUKE incorporates an entity-aware self-attention mechanism to effectively capture the meaning and context of the tokens (words or entities) when calculating attention scores during training. LUKE is pre-trained using conventional Masked Language Model and a new pre-training task to learn entity recognition from a large entity-annotated corpus from Wikipedia. The model is then fine-tuned on various downstream tasks based on domain-specific datasets such as Open Entity, TACRED, CoNLL-2003, ReCoRD, and SQuAD 1.1.

Figure 3, 4, 5, and 6 below shows the performance of LUKE on various NLP tasks compared to other models. LUKE outperformed other models in NER task on CoNLL-2003 dataset, classification task on TACRED dataset, Cloze-style QA on ReCORD dataset, and extractive QA on SQuAD 1.1 dataset.

Figure 3

Comparison Results on NER on CoNLL-2003 dataset

Name	F1
LSTM-CRF (Lample et al., 2016)	91.0
ELMo (Peters et al., 2018)	92.2
BERT (Devlin et al., 2019)	92.8
Akbik et al. (2018)	93.1
Baevski et al. (2019)	93.5
RoBERTa	92.4
LUKE	**94.3**

Figure 4

Comparison Results on Relation Classification on TACRED dataset

Name	Prec.	Rec.	F1
BERT (Zhang et al., 2019)	67.2	64.8	66.0
C-GCN (Zhang et al., 2018b)	69.9	63.3	66.4
ERNIE (Zhang et al., 2019)	70.0	66.1	68.0
SpanBERT (Joshi et al., 2020)	70.8	70.9	70.8
MTB (Baldini Soares et al., 2019)	-	-	71.5
KnowBERT (Peters et al., 2019)	**71.6**	71.4	71.5
KEPLER (Wang et al., 2019b)	70.4	73.0	71.7
K-Adapter (Wang et al., 2020)	68.9	**75.4**	72.0
RoBERTa (Wang et al., 2020)	70.2	72.4	71.3
LUKE	70.4	75.1	**72.7**

Figure 5

Comparison Results on Cloze-style QA on ReCORD dataset

Name	Dev EM	Dev F1	Test EM	Test F1
DocQA+ELMo (Zhang et al., 2018a)	44.1	45.4	45.4	46.7
BERT (Wang et al., 2019a)	-	-	71.3	72.0
XLNet+Verifier (Li et al., 2019)	80.6	82.1	81.5	82.7
RoBERTa (Liu et al., 2020)	89.0	89.5	-	-
RoBERTa (ensemble) (Liu et al., 2020)	-	-	90.0	90.6
LUKE	**90.8**	**91.4**	**90.6**	**91.2**

Figure 6

Comparison Results on Extractive QA on SQuAD 1.1 dataset

Name	Dev EM	Dev F1	Test EM	Test F1
BERT (Devlin et al., 2019)	84.2	91.1	85.1	91.8
SpanBERT (Joshi et al., 2020)	-	-	88.8	94.6
XLNet (Yang et al., 2019)	89.0	94.5	89.9	95.1
ALBERT (Lan et al., 2020)	89.3	94.8	-	-
RoBERTa (Liu et al., 2020)	88.9	94.6	-	-
LUKE	**89.8**	**95.0**	**90.2**	**95.4**

Note. The performance of LUKE on NER tasks and question answering compared to popular known LLM . Adapted from “Luke: Deep Contextualized Entity Representations with Entity-aware Self-attention,” by Yamada, I., Asai, A., Shindo, H., Takeda, H., Matsumoto, Y.,

2020, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Copyright 2020 by Association for Computational Linguistics.

Yang et al. (2019) have introduced a generalized autoregressive (AR) pre-training method called XLNet. Thanks to this autoregressive pre-training method, XLNet was proven to overcome the BERT model in having a better learning of long-range dependencies. The model is trained to predict any word in the original sequence conditioned on all other words, regardless of the order. This is achieved by random permutations of the factorization order to capture bidirectional context. The authors incorporate the state-of-the-art AR model, Transformer-XL features into the XLNet pre-training framework including relative positional encoding scheme and segment recurrence mechanism. The model is thus able to capture relationships between words and their contexts over all factorization orders of the last segment. Similar to BERT, XLNet was pre-trained using the BooksCorpus and English Wikipedia. On top of that, the authors also include Giga5, ClueWeb 2012-B, and Common Crawl for the process. The model is then fine-tuned on multiple datasets to compare and evaluate the performance with existing approaches.

The authors concluded that XLNet outperforms BERT on 20 NLP tasks including question answering, natural language inference, sentiment analysis, and document ranking. Results are shown from Figure 7 to Figure 9. Figure 7 shows the error metric for text classification task on seven datasets, and XLNet achieves the lowest error score. For reading comprehension and document ranking tasks, XLNet outperforms GPT, BERT, and RoBERTa by achieving the highest accuracy score.

Figure 7

Comparison Results for Text Classification Tasks

Model	IMDB	Yelp-2	Yelp-5	DBpedia	AG	Amazon-2	Amazon-5
CNN [15]	-	2.90	32.39	0.84	6.57	3.79	36.24
DPCNN [15]	-	2.64	30.58	0.88	6.87	3.32	34.81
Mixed VAT [31, 23]	4.32	-	-	0.70	4.95	-	-
ULMFiT [14]	4.6	2.16	29.98	0.80	5.01	-	-
BERT [35]	4.51	1.89	29.32	0.64	-	2.63	34.17
XLNet	**3.20**	**1.37**	**27.05**	**0.60**	**4.45**	**2.11**	**31.67**

Figure 8

Comparison Results for Reading Comprehension and Document Ranking

RACE	Accuracy	Middle	High	Model	NDCG@20	ERR@20
GPT [28]	59.0	62.9	57.4	DRMM [13]	24.3	13.8
BERT [25]	72.0	76.6	70.1	KNRM [8]	26.9	14.9
BERT+DCMN* [38]	74.1	79.5	71.8	Conv [8]	28.7	18.1
RoBERTa [21]	83.2	86.5	81.8	BERT†	30.53	18.67
XLNet	**85.4**	**88.6**	**84.0**	XLNet	**31.10**	**20.28**

Figure 9

Comparison Results for Question Answering

SQuAD2.0	EM	F1	SQuAD1.1	EM	F1
Dev set results (single model)					
BERT [10]	78.98	81.77	BERT† [10]	84.1	90.9
RoBERTa [21]	86.5	89.4	RoBERTa [21]	88.9	94.6
XLNet	**87.9**	**90.6**	XLNet	**89.7**	**95.1**
Test set results on leaderboard (single model, as of Dec 14, 2019)					
BERT [10]	80.005	83.061	BERT [10]	85.083	91.835
RoBERTa [21]	86.820	89.795	BERT* [10]	87.433	93.294
XLNet	**87.926**	**90.689**	XLNet	**89.898‡**	**95.080‡**

Note. Performance of XLNet compared to other popular LLM on text classification task, reading comprehension, and question answering. Adapted from "XLNet: Generalized Autoregressive Pretraining for Language Understanding," by Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V., 2019, arXiv.1906.08237. Copyright 2019 by arXiv.

Sanh et al. (2020) first introduces DistilBERT model as a compressed version of BERT model. DistilBERT's architecture is also transformer-based, but with fewer layers and attention heads. This model is trained using a combination of knowledge distillation and parameter tuning techniques to compress the model. After comparing the performance of DistilBERT with BERT-base model and ELMo on the dev sets of the GLUE benchmark, the authors find that DistilBERT retains 97% (shown in Figure 10) of BERT performance while being 60% faster and 40% smaller (shown in Figure 11).

Figure 10

Scoring Results Comparison among ELMo, BERT-base, and DistilBERT model

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Figure 11

Model Parameter Size and Inference Time Comparison Among ELMo, BERT-base, and DistilBERT Model

Model	# param. (Millions)	Inf. time (seconds)
ELMo	180	895
BERT-base	110	668
DistilBERT	66	410

Note. The F1-score performance of DistilBERT on various datasets, and the parameter size of DistilBERT. Adapted from " DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," by Sanh, V., Debut, L., Chaumond, J., Wolf, T., 2020, arXiv:2011.06993. Copyright 2020 by arXiv.

While DistilBERT is proved to be a faster, smaller, more efficient model compared to BERT-base model, authors suggest that future work can be done to understand the performance of this model in conducting domain-specific tasks, and the generalizability of DistilBERT to other languages.

Another paper presents a new technique to build sentence encoders called T5 (Ni et al., 2022). The project conducts on three model architectures: encoder-only first, encoder-only mean, and encoder-decoder first. The authors mention that a two-stage contrastive learning method is applied, and the technique improves the performance of the downstream tasks. The models are compared with each other and with BERT models; as a result, the encoder-only models outperform Sentence BERT and SimCSE in some transfer tasks and the encoder-decoder method has an outstanding performance on the semantic textual similarity (STS) task. Additionally, T5 models can be scaled up and obtain the state of art in several NLP tasks at the time of the project.

PEGASUS is introduced in a research paper by Zhang et al. (2020) as a self-supervised transformer that shows outstanding performance on NLP tasks, including summarization. The principle of PEGASUS is explained to be removing or masking sentences from the input; then, the pre-trained model produces a sequence from the remaining sentences as an output. A variety of datasets are used to train the model, including XSum, CNN/DailyMail, NEWSROOM, Multi-News, Gigaword, arXiv, PubMed, BIGPATENT, Wikihow, Reddit TIFU, AESLC, and BillSum. PEGASUS model is compared with multiple transformers methods, such as BERT, BARB, T5, MASS, UniLM. The experiment shows that PEGASUS achieves state-of-the-art results in terms of ROUGE score in 12 NLP summarization tasks.

Table 1 below shows the comparisons of all the methodologies mentioned in section 1.4. The table includes model names, their performances in the research papers, and the pros and

cons of each model. Model performance in the table are partial results from each paper. For full performance results, please refer to the original research papers.

Table 1

Comparisons of Deep Learning Methods

Model	Model Performance	Pros	Cons
BERT	- 80.5 GLUE score - SQuAD v1.1 Test 1 score 93.2 - SQuAD v2.0 Test F1 score 83.1	- Can be fine-tuned for a specific task by adding an additional layer without substantial modifications	- Recent years transformers have developed from BERT and have better performance in some specific tasks
RCNN-RoBERTa	- 0.82 accuracy score, 0.81 precision score, 0.80 recall score, 0.80 F1 score, and AUC 0.89 on semantic analysis tasks	- Outperform a variety of models on various semantic analysis tasks, including RoBERTa itself	- RoBERTa is built based on BERT but use 10 times more data than BERT - Expensive computation
ELECTRA	- ELECTRA-base: score 85.0 on GLUE task - ELECTRA-1.75M: SQuAD 1.1 dev F1 score 94.9, SQuAD 2.0 dev F1 score 90.6, SQuAD 2.0 test F1 score 91.4.	- Outperform RoBERTa and XLNET - Adding a sentence-level contrastive objective slightly decreased scores	- Work well with even small amounts of compute

BART	- SQuAD 1.1 F1 score 94.6 - SQuAD 2.0 F1 score 89.2 - Score 6 ROUGE on summarization tasks	- Matches the performance of RoBERTa on GLUE and SQuAD tasks	- Computationally expensive - Training time is slow - May not perform well on specialized domain tasks
PaLM	- Few-shot PaLM 540B: SQuADv2 F1 score 83.3 - Few-shot PaLM 540B: Drop F1 score 70.8 - Few-shot PaLM 540B: CoQA F1 score 81.5	- Achieve the state-of-the-art performance in many NLP tasks	- Brand-new methodology that may take time to fit a specific task.
T5	- T5-11B: CNN/DM ROUGE-1 score 43.52 on summarization. - T5-11B: GLUE average score 90.3 text classification - T5-11B: SQuAD F1 score 96.22 on QA	- Flexible, not limited to be applied to any specific NLP tasks - Achieved SOTA performance on many benchmarks	- T5 input length is limited to 512 tokens, making it challenging to deal with long documents.
LUKE	- CoNLL-2003 F1 score 94.3 on NER - SQuAD 1.1 Dev F1 score 95.0 on extractive QA	- Incorporate entity-aware self-attention on top of original architecture	- Limited to entities related tasks
DistilBERT	- 66 millions parameters while ELMo: 180 and BERT-base: 110 - On average of 9 tasks, accuracy score is 77.0 while	- Faster and smaller comparing to BERT - Easier to implement on low-resource devices	- Not performing as well as BERT in accuracy - Limited to ability to capture complex and nuanced relationships

	ELMo is 68.7 and BERT-base id 77.0		
PEGASUS	- Achieve state-of-the-art results on 12 abstractive summarization tasks	- Work well with summarization tasks	- Limited to summarization tasks
XLNet	- AG corpus Error rates 4.45 on text classification - SQuAD 2.0 F1 score 90.689 on comprehension	- Capture long-range dependencies - Capture relationship betweens word on relative positions	- Limited transferability to other domains

1.5 Literature Survey of Existing Research

In the paper by Wu et al. (2021), it discusses how pre-trained models have been a revolutionary way to approach NLP problems, especially for the domain of news applications. The paper presents the issue of how pre-trained models currently are “huge in size with hundreds of millions of parameters” (Wu et al., 2021). The paper offers the solution to this problem in the form of a teacher-student learning model by way of distillation, which they named NewsBERT. The basis of the model works from a teacher and student pairing that allows for the student to learn from its parent. The methodology that the paper implemented was by allowing for the paired models to jointly learn together, while also obtaining useful information from the teacher model. This was implemented by using transformer layers for both models, and having both models share a dense layer, which provided the final prediction for the paired models. The results that Wu et al. (2021) achieved was that they found that UniLM-based models were better in performance compared to BERT models at being able to handle and classify news texts. The

paper found that their NewsBERT was capable of outperforming much larger models, while also being 12 times faster. The paper concluded that the future could benefit from both teacher and student being able to learn from the experiences of both models, rather than just the student being able to learn. The paper also proposed the idea of implementing a momentum distillation method, which uses momentum to combine gradients from both models for the purpose of increasing the learning of the student.

Chen et al. (2022) proposed the problem of how Chinese news is mostly in long text, and how this can be an issue for NLP text classification due to its complexity and structure. The solution that the group offered was a "BERT-based local feature convolutional network (LFCN) model including four novel modules" (Chen et al., 2022). This approach was used to handle the issue that BERT models suffer from, which is the length of text that can be used for the input sequence. The group overcame this problem by implementing their own method called Dynamic LEAD-n (DLN), which they used to obtain short text from the original text based on their DLN algorithm. Finally, they used a CNN module to collect the in-text features that were key phrases. After the local features were captured, they fused the key phrases and this created the final output for the text. The final layer was used for the purposes of classification, in which they implemented a single feedforward layer to classify the output text. The group was able to successfully solve the issue that plagues large BERT models with long text while also improving the performance. Their proposed model was able to outperform any other traditional transformer that they tested against based on the metric accuracy for both datasets.

Figure 12

Accuracy Scores of Transformer Models on THUCNews and MCNews Datasets

	THUNews		MCNews
Model	Dev	Test	Dev
CNN	94.9	96.5	91.0
ERNIE	98.2	97.7	-
BERT	97.7	97.8	93.4
BERT-wwm	98.0	97.8	94.4
BERT-wwm-ext	97.7	97.7	93.4
RoBERTa-wwm-ext	98.3	97.8	94.0
MacBERT	98.2	97.7	-
ChineseBERT	98.1	97.9	-
LFCN (ours)	**98.8**	**99.0**	**96.2**

Note. BERT-based local feature convolutional network (LFCN) accuracy performance on THUNews and MCNews. Adapted from "A Long-Text Classification Method of Chinese News Based on BERT and CNN," by Chen, X., Peimin, C., & Lv, S. (2022). *IEEE Access, Volume 10,* p. 34046-34057. Copyright 2022 by IEEE.

In conclusion the paper was able to solve the issue of trying to classify long text with a transformer model by using a novel LFCN model along with their own DLn algorithm to obtain smaller texts.

In the paper by Liu et al. (2022), the paper discusses how trying to perform abstractive summarization with models that are using maximum likelihood estimation can have issues with their accuracy of text for summarization tasks. The paper states that this approach can lead to problems with inferring from the text. The solution that is presented is a novel approach that uses "... a non-deterministic distribution so that different candidate summaries are assigned probability mass according to their quality" (Liu et al., 2022). The actual methodology that the paper uses is by implementing Contrastive Learning for Coordination, which can create different summaries with varying output quality and allows the model to assign higher probabilities to better candidates. This is achieved by the abstractive model having dual roles that outputs the

summaries and allows for evaluation for the estimation of probability over candidate outputs. The paper uses the ROGUE score as a measure for evaluating the candidate performance. Contrastive loss is then used to fine-tune the model based on the better performing candidates. The paper built two variant models BRIO-ctr and BRIO-mul. The BRIO-ctr variant is tuned with constructive loss, and the BRIO-mul variant is tuned with multi-task loss. The latter model ended up being the best performing model on all datasets based on the ROGUE-1, ROGUE-2, and ROGUE-L metrics, therefore demonstrating SOTA performance.

Figure 13

Results of Transformer Performance on CNNDM Dataset with ROGUE-1, ROGUE-2, and ROGUE-L scores

System	R-1	R-2	R-L
	CNNDM		
BART*	44.16	21.28	40.90
PEGASUS*	44.17	21.47	41.11
GSum*	45.94	22.32	42.48
ConSum*	44.53	21.54	41.57
SeqCo*	45.02	21.80	41.75
GOLD-*p**	45.40	22.01	42.25
GOLD-*s**	44.82	22.09	41.81
SimCLS*	46.67	22.15	43.54
BART‡	44.29	21.17	41.09
BRIO-Ctr	47.28†	22.93†	44.15†
BRIO-Mul	**47.78**†	**23.55**†	**44.57**†

Figure 14

Results of Transformer Performance on XSUM Dataset with ROGUE-1, ROGUE-2, and ROGUE-L scores

	XSum		
BART*	45.14	22.27	37.25
PEGASUS*	47.21	24.56	39.25
GSum*	45.40	21.89	36.67
ConSum*	47.34	24.67	39.40
SeqCo*	45.65	22.41	37.04
GOLD-*p**	45.75	22.26	37.30
GOLD-*s**	45.85	22.58	37.65
SimCLS*	47.61	24.57	39.44
PEGASUS‡	47.46	24.69	39.53
BRIO-Ctr	48.13†	25.13†	39.84†
BRIO-Mul	**49.07**†	**25.59**†	**40.40**†

Note. BRIO performance on CNNDM and XSUM dataset with ROGUE metric. Adapted from "BRIO: Bringing Order to Abstractive Summarization.," by Liu, Y., Liu, P., Radev, D., & Neubig, G., 2022. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, *Volume 1: Long Papers*. Copyright 2022 by Association for Computational Linguistics.

In the paper by Zhang (2023), they discussed the issue of understanding how automatic summarization of news articles are successful. The solution that the paper presented was to compare and contrast ten pre-trained models against how humans perform on the task. The methodology that Zhang (2023) implemented was having humans evaluate a hundred samples from the CNN and Daily Mail dataset. Then, they selected ten pre-trained models from different pre-training approaches to perform summaries on the datasets.

Figure 15

List of Large Models Evaluated

Model	Model Creator	# Parameters	Instruction Tuning	Reference
GPT-3 davinci v1 GPT-3 curie v1 GPT-3 ada v1	OpenAI	175B 6.7B 350M	✗	Brown et al. (2020)
InstructGPT davinci v2 InstructGPT curie v1 InstructGPT ada v1	OpenAI	175B 6.7B 350M	✓	Ouyang et al. (2022)
OPT 175B	Meta	175B	✗	Zhang et al. (2022)
GLM	Tsinghua University	130B	✗	Du et al. (2021)
Cohere xlarge v20220609	Cohere	52.4B	✗	Cohere (2022)
Anthropic-LM v4-s3	Anthropic	52B	✓	Bai et al. (2022)

Note. Performance of various large language models comparing parameters and model creators. Adapted from "Benchmarking Large Language Models for News Summarization," by Zhang, T., 2023. Copyright 2023 by arXiv.org.

Zhang (2023) found that to achieve zero-shot summarization the focus of the model should be on instruction tuning, and not the size of the model. Also, the importance of having high quality references for the models is very important for performance. The paper found that the best performing models were Instruct Curie and Davinci, which were focused on instruction tuning and can be seen in the figure below. Finally, they found that the size of the model was not the best approach. For example, the largest model that they used had 175 billion parameters, but it would often ignore the instruction and generate incorrect outputs.

Figure 16

Performance of Large Models on CNNDM and XSUM Datasets

Setting	Models	CNN/Daily Mail			XSUM		
		Faithfulness	Coherence	Relevance	Faithfulness	Coherence	Relevance
Zero-shot language models	GPT-3 (350M)	0.29	1.92	1.84	0.26	2.03	1.90
	GPT-3 (6.7B)	0.29	1.77	1.93	0.77	3.16	3.39
	GPT-3 (175B)	0.76	2.65	3.50	0.80	2.78	3.52
	Ada Instruct v1 (350M*)	0.88	4.02	4.26	0.81	3.90	3.87
	Curie Instruct v1 (6.7B*)	0.97	**4.24**	**4.59**	**0.96**	4.27	**4.34**
	Davinci Instruct v2 (175B*)	**0.99**	4.15	**4.60**	**0.97**	4.41	**4.28**
Five-shot language models	Anthropic-LM (52B)	0.94	3.88	4.33	0.70	**4.77**	4.14
	Cohere XL (52.4B)	**0.99**	3.42	4.48	0.63	**4.79**	4.00
	GLM (130B)	0.94	3.69	4.24	0.74	4.72	4.12
	OPT (175B)	0.96	3.64	4.33	0.67	**4.80**	4.01
	GPT-3 (350M)	0.86	3.73	3.85	-	-	-
	GPT-3 (6.7B)	0.97	3.87	4.17	0.75	4.19	3.36
	GPT-3 (175B)	**0.99**	3.95	4.34	0.69	4.69	4.03
	Ada Instruct v1 (350M*)	0.84	3.84	4.07	0.63	3.54	3.07
	Curie Instruct v1 (6.7B*)	0.96	**4.30**	4.43	0.85	4.28	3.80
	Davinci Instruct v2 (175B*)	**0.98**	4.13	4.49	0.77	**4.83**	**4.33**
Fine-tuned language models	Brio	0.94	3.94	4.40	0.58	4.68	3.89
	Pegasus	0.97	3.93	4.38	0.57	4.73	3.85
Existing references	-	0.84	3.20	3.94	0.37	4.13	3.00

Note. Performance of various large language models with novel metrics: Faithfulness, Coherence, and Relevance on CNN Dailymail and XSUM dataset. Adapted from "Benchmarking Large Language Models for News Summarization," by Zhang, T., 2023. Copyright 2023 by arXiv.org.

In conclusion Zhang (2023) found that most pre-trained models are being tuned on poor quality references, and that with better quality references the models could perform better for summarization tasks. The paper also noted the difficulty of trying to evaluate the performance of humans on summarization tasks, and that it is subjective. The paper states that their testing concluded that current state-of-the-art summarization models that focus on instruction tuning are on-par with humans that work as freelance writers.

Yan et al. (2019) proposes a transformer encoder that handles the named entity recognition (NER) task which is also known as TENER. The paper explains the named entity recognition task as identifying the range of an entity in a specific sentence and giving a category to the entity. According to the authors, TENER's architecture has three layers: Embedding,

Encoding, and Condition Random Field (CRF). In the paper, the Embedding layer uses character encoder and pre-trained word embeddings to extract the characters; then, the final word of the first layer is the combination of those character features. For the Encoding layer, the authors mention two improvements of the transformer: Direction- and Distance-Aware Attention and Unscaled dot-product attention. Last but not least, the CRF layer's purpose is to optimize the probability of valid label sequence. The transformer is tested on six different NER tasks including two English and four Chinese and the result shows that TENER outperforms other models such as BiLSTM-CRF, CNN-BiLSTM-CRF, BiLSTM-BiLSTM-CRF, LM-LSTM-CRF, and so forth.

To conduct the named entity recognition (NER) task effectively, Ushio and Camacho-Collados (2021) introduce a Python library called Transformer-based Named Entity Recognition (T-NER) that can finetune NER language model (LM). The authors design the framework of the library in a way that any level of users can experiment on any given NER dataset. The three experiments on nine different datasets and there are three types of results: In-domain, Cross-domain, and Cross-lingual. In-domain evaluation is explained as using standard dataset, Cross-domain evaluation focuses on the cross-domain performance of T-NER models, and the Cross-lingual shows the zero-shot cross-lingual performance of T-NER models on only the WikiAnn dataset. The results in all three cases are shown in Figures 17, 18 and 19.

Figure 17

In-domain type-aware F1 score

Dataset	*BASE*	*LARGE*	SoTA
OntoNotes5	89.0	89.1	92.1
CoNLL 2003	90.8	92.9	94.3
WNUT 2017	52.8	58.5	50.3
FIN	81.3	76.4	82.7
BioNLP 2004	73.4	74.3	77.4
BioCreative V	88.0	88.6	89.9
MIT Restaurant	79.4	79.6	-
MIT Movie	69.9	71.2	-
WikiAnn/en	82.7	84.0	84.8
WikiAnn/ja	83.8	86.5	73.3
WikiAnn/ru	88.6	90.0	91.4
WikiAnn/es	90.9	92.1	-
WikiAnn/ko	87.5	89.6	-
WikiAnn/ar	88.9	90.3	-

Figure 18

Type-ignored F1 score in cross-domain on non-lower-cased English datasets

train\test	ontonotes	conll	wnut	wiki	bionlp	bc5cdr	fin	avg
ontonotes	**91.6**	65.4	53.6	47.5	0.0	0.0	18.3	40.8
conll	62.2	**96.0**	69.1	61.7	0.0	0.0	22.7	35.1
wnut	41.8	85.7	**68.3**	54.5	0.0	0.0	20.0	31.7
wiki	32.8	73.3	53.6	**93.4**	0.0	0.0	12.2	29.6
bionlp	0.0	0.0	0.0	0.0	**79.0**	0.0	0.0	8.7
bc5cdr	0.0	0.0	0.0	0.0	0.0	**88.8**	0.0	9.8
fin	48.2	73.2	60.9	58.9	0.0	0.0	**82.0**	38.1
all	90.9	93.8	60.9	91.3	78.3	84.6	75.5	81.7

Figure 19

Cross-lingual type-aware F1 results on WikiAnn dataset

	test					
train	**en**	**ja**	**ru**	**ko**	**es**	**ar**
en	**84.0**	46.3	73.1	58.1	71.4	53.2
ja	53.0	**86.5**	45.7	57.1	74.5	55.4
ru	60.4	53.3	**90.0**	68.1	76.8	54.9
ko	57.8	62.0	68.6	**89.6**	66.2	57.2
es	70.5	50.6	75.8	61.8	**92.1**	62.1
ar	60.1	55.7	55.7	70.7	79.7	**90.3**

Note. Performance of T-NER on various datasets and tasks, including cross-lingual. “ T-NER:

An all-round python library for Transformer-based named entity recognition," by Ushio, A., & Camacho-Collados, J., 2021, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations, p. 53-62*. Copyright 2021 by Association for Computational Linguistics.

In a paper by Deping et al. (2021), the authors attempted to compare the performance of a variation of a Recurrent Neural Network (RNN) called a Long Short-Term Memory network (LSTM) and a Transformer model known as Bidirectional Encoder Representations from Transformer (BERT) to classify new articles. One of the main reasons for the authors to draw this comparison was that the traditional NLP methodologies such as LSTM, had a very high training time since parallel computing of the translation tasks was not easy to achieve. Hence BERT was introduced to reduce the training times by pre-training the model using a very large collection of data that has no connection with the NLP tasks that it was chosen to do. The outcome of the study was that the BERT model performed better than the LSTM model on the same corpus and similar parameters. The evaluation metrics chosen for this study were Accuracy and Loss. The authors then spoke about the shortcomings of this study. They mentioned the problem of overfitting after a certain number of epochs and suggested working on reducing the model complexity through regularization and performing curbing the overfitting by implementing cross-validation.

One of the important tasks in the realm of NLP is Named Entity Recognition (NER). NER is the process of extracting information related to a septic topic from a large unstructured text. Vychegzhanin and Kotelnikov (2019), state that this process can help automatically classify information related to people, organizations, locations and time indicators. These classes are highly relevant to news article classification. In their work, the authors compared 7 tools related

to NER for news articles. The tools are Stanford NER, spaCY, Natural Language Toolkit (NLTK), General Architecture for Text Engineering (GATE), Flair and DeepPavlov. These tools have been written in programming languages such as python and java. All the tools are based on different aspects of machine learning and deep learning. This helps in understanding the differences in performance and training times. They implemented the algorithms in languages such as English and Russian, with 2 corpora in each language. Precision, Recall and F1-score are used to evaluate the performance of the tools. Figure 20 shows the results of the experiments conducted by the authors for the English language. The authors conclude that Flair and DeepPavlov perform well for both exact and partial matching, which makes them highly attractive for our project. They also state that the tools find it easier to classify names of people and locations than the other types.

Figure 20

Results of Named Entity Recognition for the English Language

Tool	Model	F1-score					
		Exact matching			Partial matching		
		Kaggle-2016		CoNLL-2003	Kaggle-2016		CoNLL-2003
		3 types	4 types	3 types	3 types	4 types	3 types
Stanford NER	english_conll_4class	0.554	–	0.860	0.663	–	0.886
	english_muc_7class	0.511	**0.486**	0.611	0.650	0.613	0.683
spaCy	en_core_web_sm	0.483	0.452	0.521	0.649	0.619	0.608
	en_core_web_md	0.503	0.468	0.570	0.665	0.631	0.659
	en_core_web_lg	0.496	0.463	0.597	0.660	0.629	0.697
	xx_ent_wiki_sm	0.501	–	0.597	0.637	–	0.695
NLTK	–	0.476	–	0.467	0.616	–	0.555
Polyglot	–	0.476	–	0.467	0.650	–	0.595
Flair	–	**0.584**	–	**0.887**	**0.691**	–	**0.904**
GATE	–	0.460	0.448	0.528	0.575	0.554	0.598
DeepPavlov	ner_conll2003_bert	0.576	–	0.860	**0.691**	–	0.901
	ner_ontonotes_bert_mult	0.523	0.475	0.687	0.685	**0.637**	0.741

Note. Performance of NER tools for NER task with F1-score. Adapted from “Comparison of Named Entity Recognition Tools Applied to News Articles,” by Vychegzhanin, S., & Kotelnikov, E., 2019, Ivannikov Ispras Open Conference (ISPRAS), pp. 72-77. Copyright 2019 by IEEE.

Generally, the named entity recognition systems are made to recognize entities in each sentence individually within a document. This can lead to confusion when there needs to be more information within a single sentence to figure out the entity type. Hence, Hu et al. (2020), suggested utilizing document-level NER. In such a system the context information is retrieved from the whole document by figuring out the relationship between the sentences within the document. This is particularly useful in NER of news articles where there is a possibility of an entity type to be classified into different entity types in different sentences which can cause inconsistencies. In their paper, the authors focused on multi-token entities. They developed a Multi-token Entity Informed Document-level (MEID) model for NER based on documents. This model is created by combining attention-based NER with a Multi-token Entity Classification task. This helps in tagging multi-token entities. The authors suggest either using a flair or BERT-base word embeddings to improve performance. They compare their model with other sentence-level and document-level models on 2 corpora.

The next paper proposes a new pretraining method called Cloze-driven Pretraining of Self-attention Networks, that improves the efficiency and effectiveness of natural language processing models in tasks such as NER (Baevskiet al., 2019). The method uses a cloze-style language modeling objective, where tokens are randomly masked and the model must predict the masked tokens based on the surrounding context. This is a bi-directional transformer model. Unlike previous pretraining methods, this approach directly optimizes the self-attention mechanism, allowing for better utilization of context information. They show that their pretraining method outperforms existing methods on benchmark datasets, including GLUE. They also demonstrate that their method achieves better sample efficiency and faster convergence than previous methods. To further improve the effectiveness of their pretraining method, the authors

introduce a new variant called reversible tokenization, which allows for reversible mapping between subwords and words, enabling the use of word-level supervision during pretraining. The model can better capture word associations in terms of semantic and syntactic structure thanks to this method.. The authors also analyze the effect of different hyperparameters on the performance of their pretraining method and provide insights into the trade-offs between sample efficiency and convergence speed. The proposed cloze-driven pretraining method with reversible tokenization shows promising results for improving the efficiency and effectiveness of self-attention networks in natural language processing tasks.

Jiang et al. (2019), present an improved differentiable architecture search (DARTS) method for language modeling and named entity recognition tasks. DARTS is a gradient-based method that uses continuous relaxation and gradient descent to search for optimal neural network architectures. The proposed improvements aim to address the challenges of instability and inefficiency in previous DARTS methods. To improve stability, the authors introduce a new regularization term to the loss function, which penalizes the change in the network weights during the search process. This encourages the search to prioritize more stable architectures that do not require significant changes in the weights. To improve efficiency, the authors propose a two-stage search process that first searches for a smaller, simpler network architecture and then uses that architecture as a starting point to search for a larger, more complex architecture. This reduces the search space and allows for faster convergence to an optimal architecture. The authors evaluate the proposed method on several benchmark datasets for language modeling and named entity recognition. The findings demonstrate that the proposed technique, I-DARTS, outperforms earlier DARTS methods as well as other search methods like random search and reinforcement learning-based search, achieving state-of-the-art performance on these tasks.

Additionally, the authors examine the learnt designs and offer insights on the key elements of each job. They discover, for instance, that convolutional layers and highway connections are advantageous for named entity identification whereas attention mechanisms and skip connections are advantageous for the language modeling task. Overall, the suggested enhancements to DARTS for named entity identification and language modeling yield encouraging results and shed light on key architectural elements for both tasks.

Joshi et al. (2019) researched to understand how transformer models such as BERT benefit coreference resolution. They implement a fine-tuned BERT for coreference resolution on two variants namely the independent variant and the overlap variant. BERT-base and BERT-large are compared against the c2f-coref model using F1-score. The corpora selected are based on paragraph-level and document-level text data to test the models. Document-level data contains a collection of articles from news, magazines and web data, which makes it ideal for our research. In the document-level dataset, the BERT-base model performs 0.9% better than the c2f-coef model, whereas the BERT-large model shows a 3.9% improvement. The authors state that the major weakness is the inability of the models to perform the task on documents with 512 tokens and beyond.

Kong et al. (2022), suggested a novel approach for document classification tasks to deal with long documents by applying a hierarchical BERT with an adaptive fine-tuning, also called as HAdaBERT, which includes both local and global encoders. The hierarchical structure works in a way that allows this model to capture information at a hierarchical level. The global encoders take care of sentence levels information, while local encoders take control of the word levels within those sentences. This helps reduce computational expense compared to approaches by splitting long documents into segments of maximum tokens into BERT models. Furthermore,

using two encoders in a hierarchical structure helps the model capturing both short-term and long-term dependencies and thus not losing information from the input. With a solid hierarchical BERT architecture, an adaptive fine-tuning is also applied to the model to improve the performance during training and testing. After conducting experiments on four corpora, including AAPD, Reuters, IMDB, and Yelp-2013, the proposed model was shown to outperform some other existing models.

Figure 21

Results Comparison between HAdaBERT and existing models

Models		**IMDB**		**Yelp-2013**		**AAPD**		**Reuters**	
		Dev.Acc	**Test.Acc**	**Dev.Acc**	**Test.Acc**	**Dev.F1**	**Test.F1**	**Dev.F1**	**Test.F1**
Conventional Neural Networks	CNN	42.9	42.7	56.3	57.7	54.5	51.4	83.5	80.8
	LSTM	48.4	48.0	53.1	53.9	69.8	68.8	84.8	82.7
	BiLSTM	49.3	48.9	57.6	58.4	70.3	69.1	86.9	84.7
	ATT-LSTM	49.7	49.3	60.8	62.4	71.0	69.5	87.4	85.1
	SGM	–	–	–	–	–	71.0	82.5	78.8
Hierarchical Neural Networks	NSC	50.9	50.6	61.4	62.7	69.5	67.3	86.7	84.8
	HAN	51.8	51.2	62.7	63.1	70.2	68.0	87.6	85.2
Pretrained Neural Networks	DocBERT	54.4	54.2	65.1	66.3	75.3	73.4	90.5	89.0
	RoBERTa	54.9	54.5	66.2	67.2	75.1	73.2	91.3	90.7
	ALBERT	52.8	52.6	65.3	66.5	74.7	72.8	90.8	90.4
	HAdaBERT	**57.7**	**57.3**	**66.4**	**67.5**	**75.8**	**75.1**	**91.6**	**90.9**

Figure 22

Results of ablation study of HAdaBERT

Models	IMDB		Yelp-2013		AAPD		Reuters	
	Dev.Acc	Test. Acc	Dev.Acc	Test.Acc	Dev.F1	Test.F1	Dev.F1	Test.F1
DocBERT	54.4	54.2	65.1	66.3	75.3	73.4	90.5	89.0
HAdaBERT	**57.7**	**57.3**	**66.4**	**67.5**	**75.8**	**75.1**	**91.6**	**90.9**
HAdaBERT w/o global encoder	56.6	56.1	65.5	66.4	75.2	74.1	90.8	89.3
HAdaBERT w/o adaptive fine-tuning	57.2	56.4	66.1	66.8	75.5	74.4	91.4	90.4
HAdaBERT w/o local encoder	48.7	47.0	64.2	65.8	75.0	73.3	90.8	89.5

Note. The performance of HAdaBERT on various datasets for document classification task with accuracy metric. Adapted from " Hierarchical BERT with an adaptive fine-tuning strategy for document classification," by Kong, J., Wang, J., Zhang, X., 2022, Knowledge-Based Systems, Volume 238, p. 107872. Copyright 2022 by Science Direct.

Named entity recognition (NER) tasks have been normally trained in a way to deal with sentence-level information. Schweter & Akbik (2021) have applied a transformer-based model for NER problems to deal with document-level features instead of sentence-level. The authors evaluate document-level features in fine-tuning NER-focused transformer models with one added layer for local word predictions. For a non fine-tuning approach, the authors leverage transformers to provide features to LSTM-CRF sequence labeling architecture. The datasets of CoNLL for NER on multiple languages including English, German, Dutch and Spanish are used in the experiments. After evaluating and comparing the two proposed approaches, feature-based and fine-tuning with best published models, the results are shown in Figure 23 below. Approaches with fine-tuning document-features clearly outperform feature-based without document-level features. The authors also prove that enforcing document boundaries does improve F1-score in nearly all experimented languages.

Figure 23

Comparative evaluation among feature-based, fine-tuning and best published approaches

Approach	*Doc. features?*	EN	DE	DE_{06}	NL	ES
Feature-based						
LSTM-CRF (all layer mean)	no	91.83 ± 0.06	82.88 ± 0.28	87.35 ± 0.17	89.87 ± 0.45	88.78 ± 0.08
LSTM-CRF (all layer mean)	yes	93.12 ± 0.14	84.86 ± 0.11	89.88 ± 0.26	91.73 ± 0.21	88.98 ± 0.11
Fine-tuning						
Transformer-Linear	no	92.79 ± 0.10	86.60 ± 0.43	90.04 ± 0.37	93.50 ± 0.15	89.94 ± 0.24
Transformer-Linear	yes	93.64 ± 0.05	86.99 ± 0.24	91.55 ± 0.07	94.87 ± 0.20	90.14 ± 0.14
Fine-tuning (Ablations)						
Transformer-Linear	yes (*+enforce*)	93.75 ± 0.16	87.35 ± 0.15	91.33 ± 0.18	**95.21 ± 0.08**	–
Transformer-Linear (+DEV)	yes (*+enforce*)	94.09 ± 0.07	**88.34 ± 0.36**	**92.23 ± 0.21**	95.19 ± 0.32	–
Best published						
Akbik et al. (2019b)	pooling	93.18 ± 0.09	–	88.27 ± 0.30	90.44 ± 0.20	–
Yu et al. (2020)	yes	93.5	86.4	90.3	93.7	**90.3**
Straková et al. (2019)	yes	93.38	85.10	–	92.69	88.81
Yamada et al. (2020)	yes	**94.3**	–	–	–	–

Note. The performance of FLERT for NER task with F1 score metric. Adapted from "FLERT: Document-Level Features for Named Entity Recognition,' by Schweter, S., Akbik, A., 2021, arXiv:2011.06993. Copyright 2021 by arXiv.

In the paper by Y. Liu and Liu (2021) it discusses the problems that exist with doing abstractive summarization, and how most approaches that use sequence-to-sequence are flawed. The paper states that these approaches make use of a dated teacher-forcing algorithm, which

creates a gap called an exposure bias in the training and test data. The solution offered to this problem is the implementation of contrastive learning. The paper discusses how they implemented two stages to successfully solve the problem. First, they used candidate generation, which uses a sequence-to-sequence approach. Then, the second stage uses reference-free evaluation by way of contrastive learning, which provides a higher score to better candidates. The contrastive learning allows for the candidate to be ranked with the use of a loss function. The results that the paper achieved are SOTA for NLP summarization tasks. When the paper was first published their model was the best performing model for summarization tasks based on the ROGUE metric on the CNN/DailyMail dataset. The model rivals the best models known for abstractive summarization tasks, and currently holds second place in abstract summarization tasks on the CNN/DailyMail dataset (Y. Liu & Liu, 2021).

Figure 24

Performance of SimCLS on CNN/DailyMail with ROGUE Metrics

System	R-1	R-2	R-L	BS	MS
BART*	44.16	21.28	40.90	-	-
Pegasus*	44.17	21.47	41.11	-	-
Prophet*	44.20	21.17	41.30	-	-
GSum*	45.94	**22.32**	42.48	-	-
Origin	44.39	21.21	41.28	64.67	58.67
Min	33.17	11.67	30.77	58.09	55.75
Max	54.36	28.73	50.77	70.77	61.67
Random	43.98	20.06	40.94	64.65	58.60
SimCLS	**46.67**†	22.15†	**43.54**†	**66.14**†	**59.31**†

Note. SimCLS performance with ROGUE metric on CNN Daily mail dataset. Adapted from "SimCLS: A Simple Framework for Contrastive Learning of Abstractive Summarization," by Liu, Y., & Liu, P., 2021, Meeting of the Association for Computational Linguistics. Copyright 2021 by Association for Computational Linguistics.

The paper by Dou et al. (2021) discusses how abstractive summarization models can be flexible and capable of producing coherent summaries, but there exists a problem that these types of models can also be unfaithful with errors at times and difficult to control which content will be abstracted upon. The paper proposes a framework called guided summarization (GSum) to solve the issues of the model being unfaithful, while also providing more control for the user. The approach used implemented a constraint on the summary which forced the summary to deviate less compared to the original document. Plus, the introduction of provisions allowed for the user to have more control over the model. BERT and BART were both used to instantiate the GSum framework. Finally, the guidance that was provided relied on four different approaches, which were highlighting sentences, keywords, salient triples, and summaries. The paper demonstrated that the guided model is generally better at extractive summaries, but can still perform well on abstract summaries. Overall, the model can generate more faithful summaries and show how different types of guidance generate qualitatively different and improved summaries. The paper also demonstrates how this approach was able to provide a degree of controllability to the learned models, and results show that the GSum model is capable of SOTA performance according to ROUGE on four popular summarization datasets (Dou et al., 2021).

Yang et al. (2019), propose a language model-driven approach for topic clustering and summarization of news articles known as the Language Model-based Topic Model (LMTM). This model is combined with a BERT and a Topic Summarization Model (TSM) and is compared. The method involves first pre-processing the articles to remove irrelevant information and then using a language model to generate sentence embeddings. The embeddings are clustered using an agglomerative hierarchical clustering algorithm to group similar sentences

together. The representative sentences from each cluster are then used to generate a summary of the article. The proposed method is evaluated using on Jaccard Coefficient (JC), Fowlkes and Mallows Index (FMI), F1-score, recall and precision on a dataset of news articles and compared to other state-of-the-art methods. The results show that the LMTM-BERT outperforms other methods in terms of both clustering accuracy and summary quality. The authors suggest that their method could be applied to other types of text data, such as social media and scientific papers, to provide more accurate and informative summaries.

Qi et al. (2020) discuss how autoregressive language modeling has an issue of focusing on recent tokens instead of capturing prior and long term tokens. This can be a problem for this style of a teacher forcing algorithm and can generate a bias of overfitting for large language modeling. ProphetNet is a solution to this problem. ProphetNet functions by implementing a self-supervised sequence-to-sequence approach that makes use of future n-gram predictions and a novel self-attention mechanism called n-stream. Traditional language models have typically used sequence-to-sequence next sentence prediction, but PropheNet uses an n-step ahead prediction method that allows for the model to predict the following n tokens based on prior tokens. This new method of using n-gram predictions and n-stream self-attention equate to a model that excels at abstractive summarization. ProphetNet performs with SOTA performance compared to other large language models on the CNN/Dailymail dataset for the abstractive summarization task.

Gehrmann et al. (2019), explore the use of fine-tuned language models for the abstractive summarization of text. The authors employ a pre-trained language model such as transformers and fine-tune it on a specific summarization task. The fine-tuned model generates abstractive summaries of input text by predicting the most important information from the source document.

The paper also investigates the effect of various training objectives and data augmentation techniques on the performance of the model. The authors evaluate their proposed method on multiple summarization datasets and compare it to other state-of-the-art models. The results show that the fine-tuned language model outperforms other models on several metrics, including ROUGE and BLEU scores. Overall, the paper presents a promising approach to abstractive summarization using fine-tuned language models. The study provides insights into the impact of training objectives and data augmentation on the model's performance and highlights the potential of this approach for summarization tasks in various domains. They also state that the state-of-the-art models for the task of summarization may not capture the true semantics of the data.

Figure 25

ProphetNet Rouge Performance on CNN/Dailymail Dataset

Dataset	Method	Corpus	R-1	R-2	R-L
CNN/DailyMail	T5 (Raffel et al., 2019)	750GB	43.52	**21.55**	40.69
	PEGASUSLARGE (C4) (Zhang et al., 2019)	750GB	43.90	21.20	40.76
	PEGASUSLARGE (HugeNews) (Zhang et al., 2019)	3800GB	44.17	21.47	41.11
	BART (Lewis et al., 2019)	160GB	44.16	21.28	40.90
	ProphetNet	160GB	**44.20**	21.17	**41.30**
Gigaword	PEGASUSLARGE (C4) (Zhang et al., 2019)	750GB	38.75	19.96	36.14
	PEGASUSLARGE (HugeNews) (Zhang et al., 2019)	3800GB	39.12	19.86	36.24
	ProphetNet	160GB	**39.51**	**20.42**	**36.69**

Note. ProphetNet ROGUE performance on CNN Daily Mail dataset and Gigaword. Adapted from “Generating Abstractive Summaries with Finetuned Language Models,” by Gehrmann, S., Ziegler, Z. M., & Rush, A. M., 2019, *International Conference on Natural Language Generation*. Copyright 2019 by Association for Computational Linguistics.

Performing NER can be challenging, but the paper by Wang et al. (2021) produced an approach that is SOTA for performing this task. The paper states that concatenating different embeddings can produce improved word representations, but selecting which embedding to

create the concatenated representation can be very difficult. This difficulty can often lead to poor performance. The solution that the team proposed was their own novel method called Automated Concatenation of Embeddings (ACE), which allows for the process of selecting concatenations of embeddings to be automated. The methodology that the team uses to implement ACE is done by using a controller to sample the concatenation of embeddings. The controller then updates the embedding based on a reward system for the specific task using accuracy as a metric. Meaning, ACE is using a form of reinforcement learning to improve its ability to select the best concatenated embeddings for a specific task. ACE was able to achieve SOTA results for NER, POS, and chunking tasks on the Conll2003 dataset.

Figure 26

ACE Performance on Conll2003 Dataset for NER and POS Tasks

	NER						POS		
	de	de_{06}	en	es	nl		Ritter	ARK	TB-v2
Baevski et al. (2019)	-	-	93.5	-	-	Owoputi et al. (2013)	90.4	93.2	94.6
Straková et al. (2019)	85.1	-	93.4	88.8	92.7	Gui et al. (2017)	90.9	-	92.8
Yu et al. (2020)	86.4	90.3	93.5	90.3	93.7	Gui et al. (2018)	91.2	92.4	-
Yamada et al. (2020)	-	-	94.3	-	-	Nguyen et al. (2020)	90.1	94.1	95.2
XLM-R+Fine-tune	87.7	91.4	94.1	89.3	95.3	XLM-R+Fine-tune	92.3	93.7	95.4
ACE+Fine-tune	**88.3**	**91.7**	**94.6**	**95.9**	**95.7**	ACE+Fine-tune	**93.4**	**94.4**	**95.8**

Note. The performance of ACE on the Conll2003 dataset with F1 score for NER and POS. Adapted from "Automated Concatenation of Embeddings for Structured Prediction," by Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., & Tu, K., 2021, *Meeting of the Association for Computational Linguistics*. Copyright 2021 by Association for Computational Linguistics.

Recent research has shown that document-level features have improved in the model performance of NER tasks; however, those features are not always available in input documents.

Wang et al. (2022) have suggested a new approach to find external features by retrieving and selecting related text with input sequences. This approach has achieved new state-of-the-part performance on 8 NER data sets across 5 domains. The architecture of this paper's framework includes input sentences being fed into an embedded search engine to get k number of related texts. These texts are re-ranked by applying BERTScore to get the most semantically related texts by F1 scores and concatenate l relevant texts as external contexts. Either original sentence input or the external context will be fed into a transformer-based embedding of choice (depending on the data set) together with the input sentence to get token representations. In the paper, Bio-BERT is used for biomedical domain datasets and XLM-RoBERTa for the rest. The outputs are fed into a Conditional Random Fields (CRF) layer to get conditional probability to calculate the negative likelihood loss L(NLL) and L(NLL-EXT) with the Cooperative Learning (CL) Loss. CL's purpose is to leverage retrieval-based input view to help improve the accuracy of the model without external contexts. The authors suggested that the model with CL is more consistent in the internal representations or the output distribution between two input views. Figure 27 below shows an experiment result of the model, with proposed models with CL outperform others SOTA approaches on most of the experiment datasets.

Figure 27

Comparison Among the CL Approach Models and Other Recent SOTA Models

	Social Media		News		Biomedical		E-commerce
	WNUT-16	WNUT-17	CoNLL-03	CoNLL++	BC5CDR	NCBI	
Zhou et al. (2019)	55.43	42.83	-	-	-	-	-
Nguyen et al. (2020)	52.10	56.50	-	-	-	-	-
Nie et al. (2020)	55.01	50.36	-	-	-	-	-
Baevski et al. (2019)	-	-	93.50	-	-	-	-
Wang et al. (2019)	-	-	93.43	94.28	-	-	-
Li et al. (2020)	-	-	93.33	-	-	-	-
Nooralahzadeh et al. (2019)	-	-	-	-	89.93	-	-
Bio-Flair (2019)	-	-	-	-	89.42	88.85	-
Bio-BERT (2020)	-	-	-	-	-	87.70	-
Evaluation: **W/O CONTEXT**							
LUKE (2020)	54.04	55.22	92.42	93.99	89.18	87.62	77.64
W/O CONTEXT	56.04	57.86	93.03	94.20	90.52	88.65	81.47
CL-L_2	57.35†	58.68†	93.08	94.38†	90.70†	89.20†	82.43†
CL-KL	58.14†	59.33†	93.21†	94.55†	90.73†	**89.24**†	82.31†
Evaluation: **W/ CONTEXT**							
W/ CONTEXT	57.43†	60.20†	93.27†	94.56†	90.76†	89.01†	83.15†
CL-L_2	58.61†	60.26†	93.47†	94.62†	**90.99**†	89.22†	83.87†
CL-KL	**58.98**†	**60.45**†	**93.56**†	**94.81**†	90.93†	88.96†	**83.99**†

Note. Performance of CL models for NER task. Adapted from "Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning," by Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., Tu, K., 2022, *arXiv:2105.03654.* Copyright 2022 by arXiv.

Based on the research papers, some known issues exist in trying to perform each NLP task. Each research paper has approached the problem differently.

There are a few issues with doing text classification, which can be costly for computing and overcoming the issues with longer text. Some papers have handled this issue with a few different approaches.

In the paper by Wu et al. (2021), they approached the task of news text classification by implementing their own model called NewsBERT. This approach was different from others because it implemented paired models that were jointly learning together for the purpose of classifying. The pairing of the models allowed NewsBERT to be able to handle longer text, which resulted in better performance while also being faster than many other large language models.

The team of Chen et al. (2022) proposed the use of a convolutional network that focuses on local features for a BERT-based model, which they termed LFCN. This approach allowed the model to overcome the problem of inputting a larger length of text. The research paper states that they developed their own algorithm called DLN, which allowed for the obtaining of shorter text from the original document.

Finally, in the paper by Kong et al. (2022), they developed a novel way to handle the classification task. To handle the long text, the team utilized a hierarchical BERT with adaptive fine-tuning, which they named HAdaBERT. HAdaBERT is effective at classifying because it uses local and global encoders to capture information at a hierarchical level. The paper demonstrated that by splitting the long text into segments of maximum tokens into BERT, and using the two encoders in a hierarchical manner, the model would excel at short-term and long-term dependencies. Therefore, allowing a more capable model that would suffer from losing information from the input.

NER can be a challenging task for models because it requires the model to extract the necessary information to produce an entity while having large unstructured text surrounding it. Some models have approached this problem in various ways.

In the paper by Hu et al. (2020), they approach this problem by using a document-level extraction. Using this method allowed the team to figure out relationships between the sentences within the document. This method is especially useful when trying to perform NER within news articles because it helps prevent the model from creating inconsistencies in its ability to classify for the NER task.

Schweter & Akbik (2021) handled the NER issue by applying a transformer-based model that focused on the document-level features instead of just sentence-level. Then, the researchers

evaluated the features with the use of fine-tuning their models with an emphasis on performing the NER task.

The researchers of Wang et al. (2021) took a slightly different approach to handle NER. They implemented a novel method called ACE. ACE allowed for the embeddings to be concatenated in an automated way. This was done by using a controller to sample the concatenated embeddings and then update the embeddings based on a reward system for a designated task. This approach equated to a SOTA F1-score performance for the NER task on the CoNLL2003 dataset.

Wang et al. (2022) developed a different approach to the prior NER research papers. The team placed an emphasis on finding external features by retrieving and selecting related text from the input sequences. This novel approach implemented a framework that inputs the sentence into an embedded search engine to provide a k number of related texts. Finally, the texts are ranked by applying a BERTScore to get the most related text based on F1-score. Similar to ACE, this model provides SOTA performance for NER tasks while using the CoNLL2003 dataset.

Text accuracy, otherwise known as faithfulness, can be an issue when trying to perform abstract summarization. This issue often occurs with models using a sequence-to-sequence approach with a maximum likelihood algorithm. To overcome this shortcoming, some papers have approached this issue with varying approaches.

Liu et al. (2022) implemented a novel approach to handle abstractive summarization called BRIO. The team used a non-deterministic distribution, which allowed for different candidate summaries to be assigned a probability based on their quality. This was achieved by using Contrastive Learning for Coordination, which provided the abstractive model with dual

roles that not only output the summaries but also evaluate for the purpose of a probability for ranking. BRIO is a SOTA model for performing abstractive summaries, and is one of the best-performing models for the CNN Dailymail dataset based on the Rogue metrics.

Y. Liu and Liu (2021) handle the issue of performing abstractive summaries by implementing a contrastive learning method. This is a two-step process, which consists of using a candidate generation based upon a sequence-to-sequence type approach and then using a reference-free evaluation with contrastive learning. Contrastive learning allows the model to rank the candidates with a loss function, resulting in SOTA performance for abstractive summarization.

The research of Dou et al. (2021) used a framework called GSum to tackle issues of text inaccuracies. The team implemented a constraint on the summary, resulting in the output deviating less from the original document. Four different approaches were used: highlighting sentences, keywords, salient triples, and summaries. GSum resulted in a model that performed better for text accuracy with respectable Rogue scores on the CNN Dailymail dataset.

Lastly, Qi et al. (2020) created a model called ProphetNet to overcome sequence-to-sequence issues by developing their own approach, which uses future n-gram predictions and an improved self-attention mechanism called n-stream. ProphetNet predicts the following n tokens based on their prior token, which allows for improved text accuracy.

2. Data and Project Management Plan

2.1 Data Management Plan

The project utilizes four different datasets namely AG's news dataset, ConLL2003 dataset, CNN - Daily Mail dataset, and our own corpus. The raw datasets are in the csv format and are used for different tasks in this project. The AG news dataset is downloaded from Antonio

Gulli's website. Antonio Gulli is the collector of the AG news dataset, and we obtained his permission to use the data for this project. ConLL2003 and CNN - Daily Mail are public datasets that are available in many sources. Our plan is to load these two datasets from Hugging Face, an AI community supporting open source contributions. At but not least, we scrape our own corpus from some famous news websites such as New York Times, Yahoo News, Inshorts, and so forth.

After the datasets are obtained, we create user accounts and set up authentication credentials in AWS, then upload the datasets to different S3 buckets. These datasets could be easily accessed from these buckets using Google Colab by some pre-setup steps. First we need to Install the latest Boto3 using the pip command. Next, we set up Boto credentials and load the dataset by coding pandas.read_csv().

There were separate buckets for raw, processed and modeling datasets. This made accessing relevant datasets faster. The group then performed data understanding, exploratory data analysis and data preprocessing on these datasets individually.

2.2 Project Development Methodology

Cross Industry Standard Process for Data Mining (CRISP - DM) is one of the most popular methodologies used in the industry due to its characteristics. In the CRISP - DM approach, each phase requires the finishing of the previous phase; however, there are steps where implementation can be done recurrently to make sure the processes, collected data and model support the desired goals. There are six individual phases in the methodology, including Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation and Deployment. This project is designed to follow CRISP - DM steps to implement and develop all the targets on time and efficiently.

In the Business Understanding phase, project goals are first designed then investigated

further. All questions, doubts are brought up about how the project works, how data can be collected, which models can be applied including problems and potential solutions. To efficiently address the questions, conducting technical and literature surveys is required to understand how pre-trained language models (PLMs) can deal with each of the tasks and which models are best suited to conduct experiments and evaluate. After this step, a list of PLMs and potential data sources are picked to implement news classification (NC), named entity recognition (NER) and abstractive text summarization task (ATS).

The second step is Data Understanding. A deep dive into pre-listed data sources from the previous step is required to further understand the data in terms of structure, quality, quantity and possible explorations. Three collected corpora are used for fine-tuning the PLMs, including AG's news corpus for NC, CoNLL-2003 dataset for NER, and CNN Dailymail dataset for ATS task. On top of these, there is a need for news articles scraping and cleaning for the application deployment step.

Data Preparation is a very crucial step in the methodology. Even though PLMs have the ability to handle text data well, the data needs to be converted into the same format that the PLMs are pre-trained on. A step to filter pre-made corpora to the scope that the project focuses on is taken at first. Then cleaning data, tokenization, handling sequences, and padding are applied to each of the corpora before the text data is input into the model. Most models can have the ability to handle a fixed number of tokens, usually either 512 or 1024. Therefore, for long sequence inputs, there is a need to consider truncating the sequences into length that are supported by the desired models.

After data is being prepared and pre-processed, it will be going to the Modeling phase. Each dataset is then splitted into training, testing and validation sets for the purpose of training,

evaluating and finalizing the model performance before going to deployment phase. For each task, fine-tuning is done on each PLMs and a hyperparameter optimization needs to be achieved for evaluating task performance in the next phase. Understanding which hyperparameters to be optimized for each task is also a challenge in this modeling step. Model comparisons and performance evaluation through metrics such as F-1 score, accuracy, precision, recall, and ROUGE are investigated for the project to come up with the best performing model.

Evaluation is the next-to-last phase in the CRISP - DM approach. After selecting the best performing model in the previous step, the project needs to be reviewed to see whether or not the model meets the desired targets in the Business Understanding step. If not, a revision is necessary to consider reselecting a better suited model for a reasonable trading off between computational cost and performance before deploying the models.

The last step to bring the project to the users is through Deployment. A plan on how to build a user interface, host the application, monitor and maintain the model for the use of researchers and readers is essential. Upon the completion of the product, a final report is produced to include the retrospective summary of the project on strengths, weaknesses, findings and potential future scopes.

2.3 Project Organization Plan

Work breakdown structure (WBS) is a tool utilized in the project management process, which allows for the aid of presenting hierarchical structure of work required to achieve the objective of the project set by the team. Each phase of the WBS is designed to provide a detailed description of tasks for the project, which provides the team with a roadmap to work efficiently.

In the first phase of the WBS, the team determined the business understanding by performing extensive research into literature related to NLP tasks that involve the use of news

articles. This knowledge allowed for the team to develop a work structure on how to best approach each NLP task that is required for the project to be successful.

The next phase consisted of the team demonstrating data understanding by scraping, gathering, and labeling data from news websites for the purpose of building a news corpus. The news corpus allowed the team to potentially fine-tune models, and also act as a means to demonstrate the live deployment of the project. Additionally, the AG News, CoNLL-2003, and CNN_Dailymail datasets were utilized to fine-tune models. The team spent time gathering, understanding, and exploring these datasets for the purpose of being able to properly utilize them for fine-tuning.

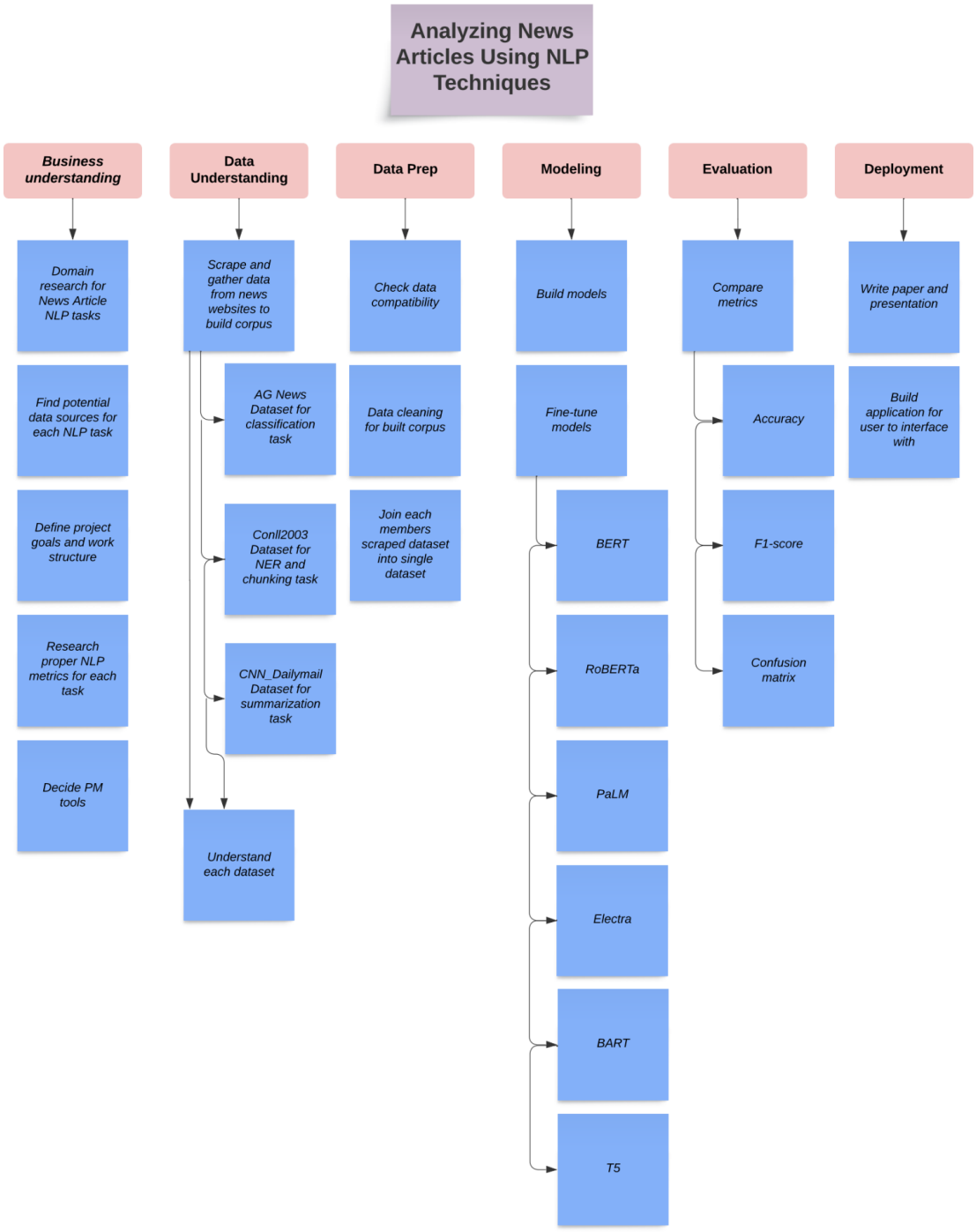
The data preparation phase consisted of the team checking their respective scraped dataset with the intention of being compatible and able to merge with other team members' dataset. During this time necessary cleaning was performed to ensure that compatibility would be possible. Finally, the scraped data sets were merged into a single corpus.

For the modeling phase seven models were built and fine-tuned with task related datasets. For example, the text classification task utilized the AG News dataset, token classification utilized CoNLL-2003, and summarization task utilized CNN_Dailymail dataset. Once the models were fine-tuned they were evaluated with proper corresponding metrics for their specific task. Metrics used were F1-score, accuracy, confusion matrix, and Rogue.

Finally, the best performing models for their respective tasks were implemented for deployment. The deployment consisted of a web application that is capable of running the models and demonstrating the use of the three tasks for text-classification, NER/chunking, and text summarization of news articles.

Figure 28

Work Breakdown Structure for the Project



2.4. Project Resource Requirements and Plan

The technologies must be able to accomplish the following tasks: data planning, data storage, data processing, model development, and hosting a web-based application to deploy the models. Below are the hardwares and softwares being used for the project.

Microsoft Word and Excel works for documenting the project's progress and reports. SJSU offers the license for Microsoft Office at no cost.

Trello is for planning management like build Gantt chart or PERT chart. The cost for five users for Trello is $24 per month.

Draw.io is used to build Work Breakdown Structure (WBS) diagrams. The project only uses the free resource from Draw.io.

Zoom is to conduct meetings among all team members.

Amazon S3 is used for data storage as it is capable of storing unlimited amounts of data with low cost. Amazon S3 is free for the first 12 months of using with 5GB in the standard storage. After that, the quoting price is $0.023 per GB for the next 50 TB per month. The resource can potentially cost nothing as the project is conducted in 1 year and the amount of data is expected to be around 5 GB.

Additionally, AWS Identity and Access Management accounts are created for each team member to retrieve the data from Amazon S3. The service is offered at no charge.

Google Drive is a cloud storage and file sharing platform. The tool is for team members to share documents and codes. Google Drive is a free platform.

Python is the main script for all coding tasks including Data Engineering and Model Development. A variety of free Python platforms are applied.

- Google-Colab - Python: is used for several sections, including Data Engineering and Model Development. This is a useful tool to share codes to all team members and is convenient for team management and discussion. Google-Colab has the ability to connect directly to Amazon S3 for code implementation.
- Jupyter Notebook - Python: is a platform for individual use when team members develop sample codes.
- Spyder - Python: some scraping tasks are not capable in Jupyter Notebook and Spyder is a backup flatform for those cases.

Tableau is applied in the data exploration phase and the visualizations are used for reporting purposes.

Web Scraper is used to scrape data from various news websites. It is a free to use extension plugin and can be used on any browser. It provides a no code interface to scrape data, which makes it easy to use and yet quite powerful.

For Web Hosting, use Amazon EC2 to host a web-based application that includes three functions: classifying input articles, classifying the quoted people, and summarizing input articles. AWS offers 750 hours free per month in the first 12 months. Therefore, the service is potentially free for the project.

Table 2

Resources Summary

Function	**Resource Type**	**Resource**	**Cost**
Project Management	Software	- JIRA - Draw.io Chart - Trello - Google Drive - Zoom Meeting	Free
Data Scraping	Software	Web Scraper	Free
Data Storage	Cloud Software	Amazon Simple Storage Service (S3)	Free first 5GB, $0.023 per GB for the next 50 TB
Dataset Exploration	Software	- Microsoft Word - Microsoft Excel	Microsoft Office is free (Licensed through SJSU)
Dataset Exploration	Software	- Tableau	$70 per month per license
Dataset Exploration &	Software	- Google Colab - Python - Jupyter Notebook - Python	Free

Data Engineering & Model Testing		- Spyder - Python	
Creating REST API	Web framework	- Flask	Free
Web Hosting	Software	- AWS EC2	750 hours free per month in the first 12 months

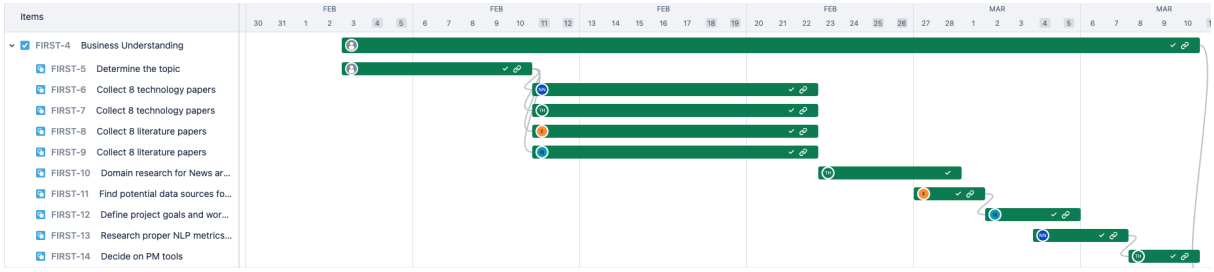
---

2.5 Project Schedule

Gantt chart is used to present a project in detail with tasks and subtasks, timeline, assigned members and the status of deliverables. Figure 29 below shows the timeline for the Business Understanding step.

Figure 29

Timeline for Business Understanding

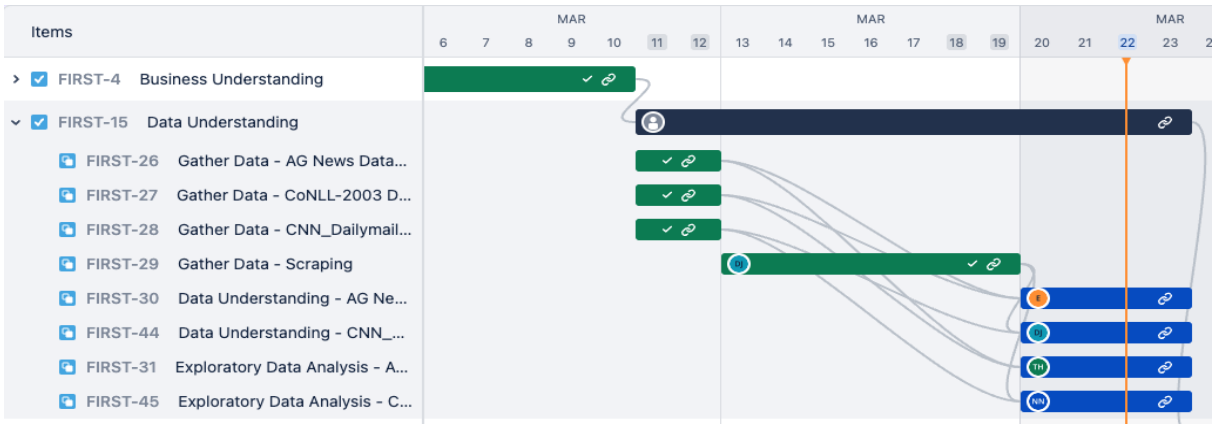


The Business Understanding phase started on Feb 3rd, when the project was being investigated starting with proposing a project topic. It was a shared task, therefore no name was assigned. Then, each team member did research papers on technology and literature to understand how the desired tasks can be done and how the technologies work. A dive into useful

potential useful datasets was conducted with a decision on the project management tool. The phrase was done on March 10th.

Figure 30

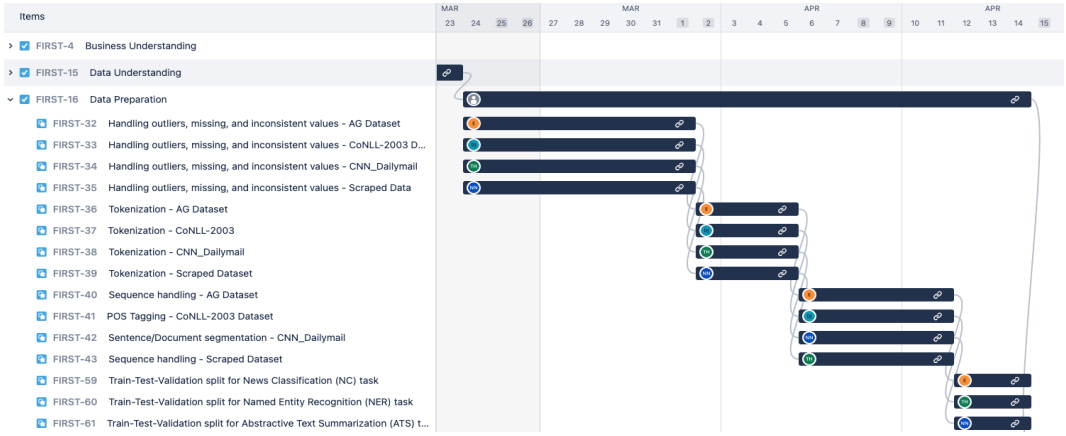
Timeline for Data Understanding



The data understanding phase started on March 11th and ended on March 23rd. This phase consisted of two major tasks. One is to understand the selected dataset and other is to perform exploratory data analysis on those datasets to get insights and patterns. The tasks were divided amongst the group equally. Figure 30 shows the timeline for the data understanding phase.

Figure 31

Timeline for Data Preparation



One of the major phases of the project was data preparation. This phase started on March 24th and ended on April 14th. Tasks were divided amongst the group members equally. This phase consisted of tasks such as handling missing values, outlier and inconsistent values. Preprocessing steps included tokenization, sequence handling, POS tagging, segmentation and splitting of datasets as per the modeling tasks. Figure 31 depicts the tasks for Data preparation.

Modeling tentatively starts on April 15th, right after Data Preparation is done, and ends on July 15th. Each of the members will take care of the development of each task; however, NER is shared by two members. Only after developing the models for tasks, the team can do model evaluation and performance comparison among decided models for NC, NER, and ATS tasks. Figure 32 shows the details with the assigned member and timeline, tasks dependencies, and status of the tasks.

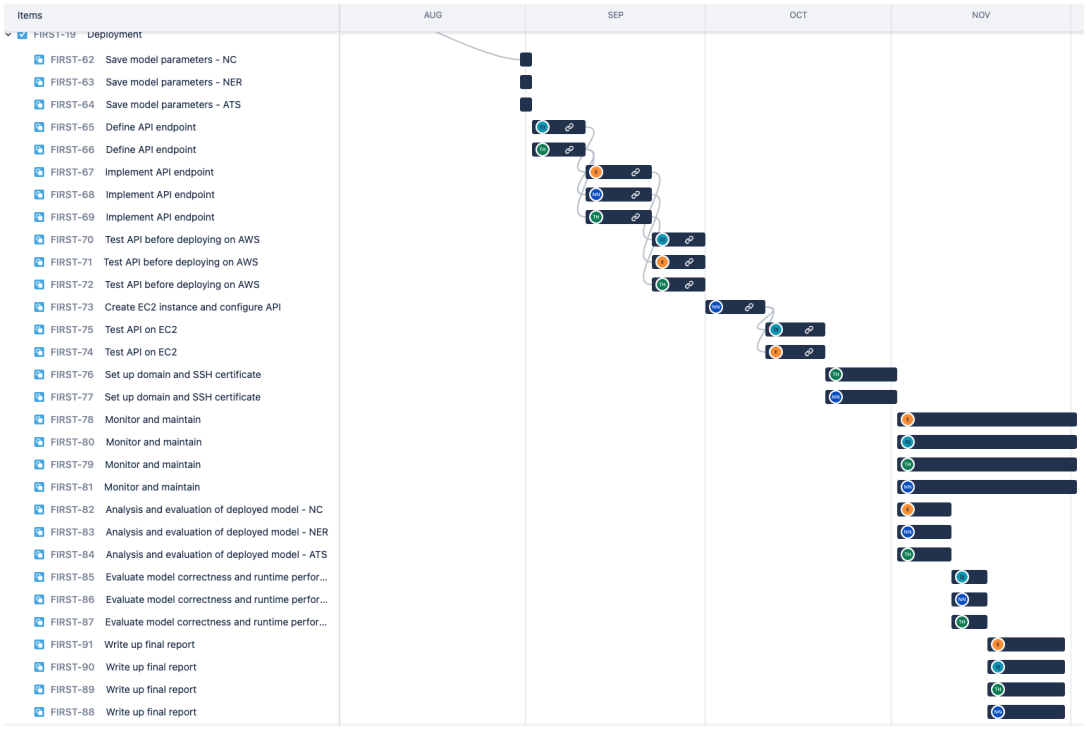
Figure 32

Timeline for Modeling

The last phase of the project was Deployment. It started on August 30th and ended on November 1st. This phase included tasks such as saving selected models, selecting up domain, creating an API between data warehouse and frontend and testing the system. This is when we finished documenting and writing the reports for the project. Figure 33 shows the timeline for the deployment phase.

Figure 33

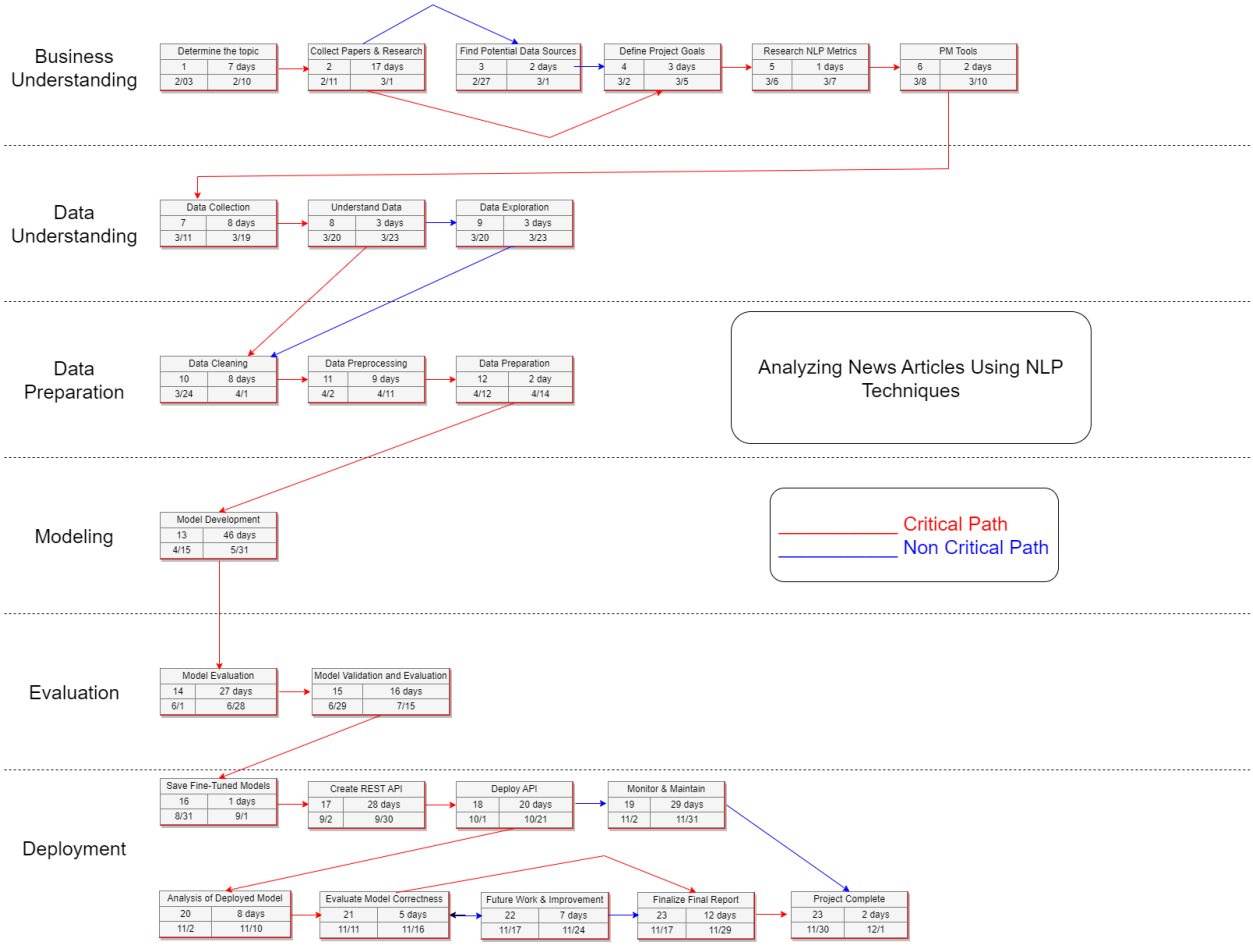
Timeline for Deployment



PERT is a technique used to analyze the various tasks involved in completing a project and to determine the minimum time required to complete the entire project, while accounting for uncertainties in task durations. It is event-oriented and more focused on time than cost, and is typically used in large-scale, complex, and non-routine projects such as infrastructure and research and development. PERT uses arrow and node diagrams to represent activities and events, respectively, and is often used alongside CPM, which employs one time and cost estimate for each activity, whereas PERT may use three time estimates. Despite their differences, PERT is increasingly being used as a critical path scheduling tool (Maribeth, 1968).

Figure 34

PERT Project Schedule



3. Data Engineering

3.1 Data Process

The project requires three separate datasets to support three tasks and one unique corpus as a demo. AG's news dataset is used for news classification, CoNLL-2003 is used for named entity recognition, CNN-DailyMail is used for abstractive summarization training models, and our built corpus is used to perform the three tasks with completed models. Each team member collects and investigates one dataset and performs data preprocessing, data transformation, data preparation, and data statistics.

The data collection plan is first discussed as a group and is done by each individual. Each dataset has its own plan. Except for our unique corpus, the three datasets are collected by

downloading the data from public resources. After the data collection step, it is expected to have a good understanding of each attribute, the size of data, sample of the dataset to map out the next steps in the preprocessing. For our unique corpus, it is collected by scraping news from multiple resources. It is a continuing process; therefore, the final data size for the corpus is not reflected properly in this current section.

Each team member will perform data preprocessing to clean their dataset. The process includes understanding important features to drop unnecessary ones, handling missing and duplicate values. Missing values can be handled by simply dropping the rows or filling the empty values with appropriate algorithms. Duplicate values are redundant and will create bias in the dataset, so it will be handled by dropping. Last but not least, it is important to understand the target feature, whether or not it is imbalanced, and how to handle imbalanced data.

Data transformation is a very crucial part of the whole data process because it is the step where data is transformed into desired formats for each model to understand. For each dataset, tokenization is required, as well as transforming the input data into numbers. For the classification task, it is also needed to convert the target labels into integers. Section 3.4 below will describe in detail each dataset's transformation. For our corpus, it is an optional process now, since we have not evaluated the best model for each task. However, a general pipeline for the transformation process is created for this corpus and ready to be used once the models are completed.

Data preparation requires splitting the data into training, validation, and testing sets. There is no fixed ratio in splitting the data. AG's news has a 70-15-15 splitting ratio in training, validation, and testing sets respectively. ConLL2003 has the training-testing-validating portion of 80-10-10. CNN-DailyMail dataset is pre-splitted by the HuggingFace with the proportion of

92-4-4 for training, validation, and testing. Training set has the largest size, and is used to train the model. The training set contains input data and target values. With known ground truth, it is possible to calculate the model's loss and error to see how well the model is performing, and to update the model's parameters. Validation set is used to evaluate the performance of the model on different data than the training data. The purpose of validation is to see how well the training process has been done and to tune hyperparameters, such as learning rate, regularization strength, number of epochs, batch size, and number of hidden layers. The process of using the validation set is to help better evaluate the model to avoid overfitting problems by adjusting the hyperparameters to optimize loss and improve model's performance. Testing set can be viewed as a demo of a model when it is ready to be used after the training and validating process is completed. Data in the testing set is completely new and unseen in the above processes. By using the testing set, we can achieve an unbiased final model performance evaluation through defined metrics. It is important that the testing set should only be used once to keep the result unbiased.

Data statistics shows the results of the changes of each dataset during each processing step. A summary table is required to show changes in the sizes and shapes of the dataset. Besides, appropriate plots will be shown for each dataset to better visualize the changes in the target and features distribution.

3.2 Data Collection

3.2.1 AG News Dataset

Originally, AG's News dataset is a huge collection of news articles with more than one million rows from more than 2000 sources by ComeToMyHead in the duration of more than one year. The dataset is created for research purposes in multiple areas such as classification, data streaming, information retrieval, etc. Zhang et al. (2015) constructed the topic classification

from AG's news dataset and used this as a text classification benchmark in their research paper.

Dataset takes the size of 621.6 MB on disk, has 1281103 rows and nine attributes. The attributes are:

- Source: source where articles are taken from
- Url: link to article
- Title: title of article
- Image: image in the description, none if there are no images
- Category: news article category/genre
- Description: part of the article, context to classify the article category
- Rank: n/a
- Pubdate: publication date if applicable else 0000-00-00 00:00:00
- Video: if video is in the description else 'N'

Figure 35.1, 35.2 , 35.3 show the data collection plan on AG's News corpus. The plan helps understand every attribute and their data types, how the dataset is collected, the purposes of collecting data, and how to process next, which will be helpful for the following sections.

Figure 35.1

Data Collection Plan for AG's News Dataset

Data Collection Plan

Project number: | Project title: Analyzing News Articles Using NLP Techniques | Project leader: | Date: 4/9/2023

Description of the data collection	
Why are we collecting the dataset?	AG's news is a huge news article dataset with premade label and has been use as benchmark for text classification tasks in multiple research papers.
How are we coolecting the dataset?	By download the xml version of the dataset (thanks to Paolo Ferragina) and load it as a dataframe using Python
What will be done with the data sonce it has been collected?	Data inspecting will be done to understand about data structures/attributes, further perform data cleaning, exploring analysis, transforming, and preprocessing to be ready for the modeling.

Figure 35.2

Data Collection Plan for AG's News Dataset

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)						
		1	2	3	4	5
What?	**Variable title**	source	url	title	image	category
	Input (X) or output (Y) variable?	X	X	X	X	Y
	Unit of measurement					
	Data type	String	String	String	String	String
	Collection method	Existing Data Collection				
	If manual					
	Gauge/instrument	No				
Historical data	Historical data exist?	Yes				
	Source of historical data	http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html				
Who?	**Data collector**	ComeToMyHead academic news search organization				
	Operational definition exist?	No				
	Resources available for data collector?	Yes				
When?	**Start date**	N/A				
	Due date	N/A				
	Duration (in days)	over 365 days				

Figure 35.3

Data Collection Plan for AG's News Dataset

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)					
		6	7	8	9
What?	**Variable title**	description	rank	pubdate	video
	Input (X) or output (Y) variable?	X	X	X	X
	Unit of measurement				
	Data type	String	Integer	String	String
	Collection method	Existing Data Collection			
	If manual				
	Gauge/instrument	No			
Historical data	Historical data exist?	Yes			
	Source of historical data	http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html			
Who?	**Data collector**	ComeToMyHead academic news search organization			
	Operational definition exist?	No			
	Resources available for data collector?	Yes			
When?	**Start date**	N/A			
	Due date	N/A			
	Duration (in days)	over 365 days			

Figure 36 shows a random sample of the dataset. All nine attributes are presented with the variety of article sources, categories, and the url, title for those articles with descriptions, images, and publication dates.

Figure 36

AG's News data sample

source	url	title	image	category	description	rank	pubdate	video
Yahoo Sports	http://us.rd.yahoo.com/dailynews/rss/sports/*h...	BALCO leaker's plea deal rejected \n (AP)\n	none	Sports	<p><a href="http://us.rd.yahoo.com/dailynews/r...	5	2007-06-15 03:41:38	N
BBC science	http://news.bbc.co.uk/go/rss/-/2/hi/science/na...	Rocks reveals Mars' watery past	http___news.bbc.co.uk_nol_shared_spl_hi_pop_up...	Sci/Tech	Exquisite colour images of the Martian surface...	5	2007-02-15 22:01:08	N
WebReference	http://www.webreference.com/programming/javasc...	Using Multiple JavaScript Onload Functions	none	Software and Developement	When scripts are written they're used to accom...	5	2008-01-21 14:40:19	N
Yahoo U.S.	http://us.rd.yahoo.com/dailynews/rss/us/*http:...	Navy Commissions Fast-Attack Submarine (AP)	none	U.S.	AP - With bells ringing and horns blaring, the...	5	0000-00-00 00:00:00	N
BBC - Front Page	http://news.bbc.co.uk/go/rss/-/1/hi/uk_politic...	'More inquiries' in honours probe	http___newsimg.bbc.co.uk_media_images_42827000...	Top News	The Crown Prosecution Service asks cash-for-ho...	5	2007-06-04 13:22:31	N

3.2.2 CoNLL2003 Dataset

The dataset was originally published by Tjong Kim Sang and De Meulder (2003) and contains labeled data for English language sentences from news articles in the Reuters corpus. The dataset was specifically designed for named entity recognition (NER) tasks in English. According to the authors, it concentrates on four different name entities, including persons, locations, organizations, and others that do not belong to the previous three categories. The columns include Chunks, Ner, Pos, and tokens which are explained in more detail as follows.

- Tokens: the individual token or word in the sentence.
- Pos (part-of-speech tag): indicates the role of the token. For example: verb, noun, objective, etc.
- Chunks (syntactic chunk tag): contains the chunk tags of each word and are based on pos tags to group adjacent words into meaningful phrases.
- Ner (named identity tag): classifies the token into four mentioned categories, which are persons, locations, organizations, or others.

The figure below is an example of how the original dataset looks like. The first column presents the tokens, the second column is the Pos, the third column is the Chunks, and the last column is the Ner. Taking the first row as an example, U.N. is the token word, NNP in the Pos column means singular proper noun, I-NP in the Chunk column with I mean inside and NP

stands for noun phrase, and I-ORG in the Ner column indicates inside an organization named entity mention.

Figure 37

CoNLL2003 Raw Dataset

```
U.N.          NNP  I-NP  I-ORG
official      NN   I-NP  O
Ekeus         NNP  I-NP  I-PER
heads         VBZ  I-VP  O
for           IN   I-PP  O
Baghdad       NNP  I-NP  I-LOC
```

For this project, the dataset CoNLL2003 is loaded from the Hugging Face website. There are 20747 rows and four columns. The next figures show the data collection plan for CoNLL2003. It mentions that the data collector is Conference on Computational Natural Language Learning, and Hugging Face captures the dataset and carries on some data processing, such as tokenizing the text and converting the categories to numbers. The data type we use are lists of strings for tokens and lists of integers for pos_tags, chunk_tags, and ner_tags.

Figure 38

Data Collection Plan Description for CoNLL2003 Dataset

Data Collection Plan

Project number:		Project title:	Analyzing News Articles Using NLP Techniques	Project leader:		Date:	4/8/2023

Description of the data collection	
Why are we collecting the dataset	The ConLL2003 dataset contains information about four different name entities, including persons, locations, organizations, and others that do not belong to the first three categories.
How are we collecting the dataset	The dataset can be loaded from HuggingFace using the library datasets and package load_dataset
What will be done with the data once it has been collected?	After collecting the data, we analyze the dataset and prepare it by checking the compatibility, cleaning data, performing EDA, transforming, and make the dataset be ready to apply the transformers.

Figure 39

Data Collection Plan for CoNLL2003 Dataset

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)					
		1	2	3	4
What?	**Variable title**	tokens	pos_tags	chunk_tags	ner_tags
	Input (X) or output (Y) variable?	X	X	X	Y
	Unit of measurement				
	Data type	List of string	List of integers	List of integers	List of integers
	Collection method	Existing Data Collection			
	If manual				
Who?	**Data collector**	Conference on Computational Natural Language Learning (CoNLL)			
	Operational definition exist?	Yes			
	Data collector trained?	N/A			
	Resources available for data collector?	Yes			
When?	**Start date**	3/11/2023			
	Due date	3/12/2023			
	Duration (in days)	2 days			

Additional Comments - e.g. resolution needed, sampling method, R&R results, storing of data, handling outliers, using filters, etc.
The dataset we use for this project is from Hugging Face. Hugging Face already conducts some data processing on the raw dataset; for example, converting letters to numbers.
Original collector's website: https://www.clips.uantwerpen.be/conll2003/ner/ Hugging Face website: https://huggingface.co/datasets/conll2003

Figure 40 depicts the dataset loaded from Hugging Face. Observe that pos_tag, chunk_tag, and ner_tag columns are converted to numbers. Additionally, the Tokens column now contains the tokens of a sentence instead of just a single word like the original dataset.

Figure 40

Hugging Face CoNLL2003 Dataset

	id	tokens	pos_tags	chunk_tags	ner_tags
0	11995	[Two, people, were, killed, and, six, were, in...	[11, 24, 38, 40, 10, 11, 38, 40, 15, 12, 21, 4...	[11, 12, 21, 22, 0, 11, 21, 22, 13, 11, 12, 21...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
1	1830	[Seller, interest, light, .]	[17, 21, 21, 7]	[11, 12, 12, 0]	[0, 0, 0, 0]
2	4978	[Southend, 2, 0, 1, 1, 1, 6, 1]	[22, 11, 11, 11, 11, 11, 11, 11]	[11, 12, 12, 12, 12, 12, 12, 12]	[3, 0, 0, 0, 0, 0, 0, 0, 0]
3	3157	[Earlier, ,, police, declined, to, comment, on...	[31, 6, 24, 38, 35, 37, 15, 15, 28, 38, 12, 21...	[3, 0, 11, 21, 22, 22, 13, 17, 11, 21, 11, 12,...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ...
4	1329	[Estonian, Tallinna, Pank, 11-mo, net, 46.6, m...	[16, 22, 22, 11, 16, 11, 16, 24, 7]	[11, 12, 12, 12, 12, 12, 12, 12, 0]	[7, 3, 4, 0, 0, 0, 0, 0, 0]

The conversion rules of Pos, Chunks, and Ner columns are reflected in figure 41. For example, for the Pos column, “ is assigned as 0, WP is assigned as 45. Observe that the ner_tags column, also our target, has nine categories.

Figure 41

Pos_tag Conversion Rules

- pos_tags: a list of classification labels (int). Full tagset with indices:

```
{'"': 0, "''": 1, '#': 2, '$': 3, '(': 4, ')': 5, ',': 6, '.': 7, ':': 8, '``': 9, 'CC': 10, 'CD': 11, 'DT': 12,
 'EX': 13, 'FW': 14, 'IN': 15, 'JJ': 16, 'JJR': 17, 'JJS': 18, 'LS': 19, 'MD': 20, 'NN': 21, 'NNP': 22, 'NNPS': 23,
 'NNS': 24, 'NN|SYM': 25, 'PDT': 26, 'POS': 27, 'PRP': 28, 'PRP$': 29, 'RB': 30, 'RBR': 31, 'RBS': 32, 'RP': 33,
 'SYM': 34, 'TO': 35, 'UH': 36, 'VB': 37, 'VBD': 38, 'VBG': 39, 'VBN': 40, 'VBP': 41, 'VBZ': 42, 'WDT': 43,
 'WP': 44, 'WP$': 45, 'WRB': 46}
```

Figure 42

Chunk_tag Conversion Rules

- chunk_tags: a list of classification labels (int). Full tagset with indices:

```
{'O': 0, 'B-ADJP': 1, 'I-ADJP': 2, 'B-ADVP': 3, 'I-ADVP': 4, 'B-CONJP': 5, 'I-CONJP': 6, 'B-INTJ': 7, 'I-INTJ': 8,
 'B-LST': 9, 'I-LST': 10, 'B-NP': 11, 'I-NP': 12, 'B-PP': 13, 'I-PP': 14, 'B-PRT': 15, 'I-PRT': 16, 'B-SBAR': 17,
 'I-SBAR': 18, 'B-UCP': 19, 'I-UCP': 20, 'B-VP': 21, 'I-VP': 22}
```

Figure 43

Ner_tag Conversion Rules

- ner_tags: a list of classification labels (int). Full tagset with indices:

```
{'O': 0, 'B-PER': 1, 'I-PER': 2, 'B-ORG': 3, 'I-ORG': 4, 'B-LOC': 5, 'I-LOC': 6, 'B-MISC': 7, 'I-MISC': 8}
```

3.2.3 CNN-Daily Mail Dataset

The CNN-DailyMail dataset consists of more than 311,971 news articles from journalists from CNN and the Daily Mail in English. The dataset consists of three object-type features: article, highlights, and id. Initially, the dataset was amassed by a team of Google DeepMind members, including Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom (Hermann et al., 2015). The first release, Version 1.0.0, aimed to assist supervised neural approaches for machine reading and question answering by leveraging significant amounts of actual natural language training data. Version 1.0.0 consisted of CNN articles that were collected over eight years, from April 2007 to April 2015, while the Daily Mail articles were collected over almost five years, from June 2010

to April 2015. The creators obtained the articles by downloading them from archives of www.cnn.com and www.dailymail.co.uk via the Wayback Machine. This release comprised approximately 312k distinct articles and close to 1M questions in Cloze style.

Later, the dataset was converted back to a summary format by Nallapati et al. (2016). They created both anonymized and non-anonymized versions of Version 1.0.0 by combining the question answers into a summary, Version 2.0.0 and 3.0.0, respectively. Version 3.0.0 allows for both extractive and abstractive summarization with its non-anonymized data. This project uses Version 3.0.0. The project obtained the dataset from Hugging Face's datasets library, which Hugging Face derived from the publicly accessible source for Version 3.0.0 by Abigail See of Stanford University, Peter J. Liu of Google Brain, and Christopher D. Manning of Stanford University.

Figure 44 shows the dataset size and columns, and Figure 45 shows the dataset's raw format in a dictionary form. The 'article' feature contains articles from CNN and Daily Mail. The 'highlights' feature is the summary of the particular article. Finally, the 'id' feature identifies the article provided by the original corpus creators.

Figure 44

CNN-Daily Mail Dataset Size and Columns

```
RangeIndex: 311971 entries, 0 to 311970
Data columns (total 3 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   article     311971 non-null  object
 1   highlights  311971 non-null  object
 2   id          311971 non-null  object
dtypes: object(3)
memory usage: 7.1+ MB
```

Figure 45

CNN-Daily Mail Dataset in Raw Dictionary Form


```
{'article': 'LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he
turns 18 on Monday, but he insists the money won\'t cast a spell on him. Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the
Phoenix" To the disappointment of gossip columnists around the world, the young actor says he has no plans to fritter his cash away on fast cars,
drink and celebrity parties. "I don\'t plan to be one of those people who, as soon as they turn 18, suddenly buy themselves a massive sports car
collection or something similar," he told an Australian interviewer earlier this month. "I don\'t think I\'ll be particularly extravagant. "The things
I like buying are things that cost about 10 pounds -- books and CDs and DVDs." At 18, Radcliffe will be able to gamble in a casino, buy a drink in a
pub or see the horror film "Hostel: Part II," currently six places below his number one movie on the UK box office chart. Details of how he\'ll mark
his landmark birthday are under wraps. His agent and publicist had no comment on his plans. "I\'ll definitely have some sort of party," he said in an
interview. "Hopefully none of you will be reading about it." Radcliffe\'s earnings from the first five Potter films have been held in a trust fund
which he has not been able to touch. Despite his growing fame and riches, the actor says he is keeping his feet firmly on the ground. "People are
always looking to say \'kid star goes off the rails,\'" he told reporters last month. "But I try very hard not to go that way because it would be too
easy for them." His latest outing as the boy wizard in "Harry Potter and the Order of the Phoenix" is breaking records on both sides of the Atlantic
and he will reprise the role in the last two films.  Watch I-Reporter give her review of Potter\'s latest » . There is life beyond Potter, however.
The Londoner has filmed a TV movie called "My Boy Jack," about author Rudyard Kipling and his son, due for release later this year. He will also
appear in "December Boys," an Australian film about four boys who escape an orphanage. Earlier this year, he made his stage debut playing a tortured
teenager in Peter Shaffer\'s "Equus." Meanwhile, he is braced for even closer media scrutiny now that he\'s legally an adult: "I just think I\'m going
to be more sort of fair game," he told Reuters. E-mail to a friend . Copyright 2007 Reuters. All rights reserved.This material may not be published,
broadcast, rewritten, or redistributed.',
 'highlights': "Harry Potter star Daniel Radcliffe gets £20M fortune as he turns 18 Monday .\nYoung actor says he has no plans to fritter his cash
away .\nRadcliffe's earnings from first five Potter films have been held in trust fund .",
 'id': '42c027e4ff9730fbb3de84c1af0d2c506e41c3e4'}
```


For this project, a data collection plan was developed to describe the dataset adequately.

Figure 46, 47 show the data collection plan for the CNN Dailymail dataset.

Figure 46

Data Collection Plan Description for CNN-Daily Mail Dataset

Data Collection Plan

Project number: | Project title: Analyzing News Articles Using NLP Techniques | Project leader: | Date: 4/8/2023

Description of the data collection	
Why are we collecting the dataset	The CNN Dailymail dataset contains three differenct features used for the text summarization task; ID, article, and highlight
How are we collecting the dataset	The dataset can be loaded from Hugging Face by calling the library datasets and package load_dataset
What will be done with the data once it is collected?	The dataset will be pre-processed, transformed, and then used for the NLP task of text summarization

Figure 47

Data Collection Plan for CNN-Daily Mail Dataset

Key Variables - A summary of the chosen input variables (Y's) and/or output variables (X's)				
		1	**2**	**3**
What?	**Variable title**	ID	Article	Highlights
	Input (X) or output (Y) variable?	X	X	Y
	Unit of measurement	N/A	N/A	N/A
	Data type	Integer	String	String
	Collection method	CNN and Dailymail websites by using WayBack Machine		
	If manual	N/A	N/A	N/A
Who?	**Data collector**	Team at Google DeepMind		
	Operational definition exist?	Yes		
	Data collector trained?	N/A		
	Resources available for data collector?	Yes		
When?		CNN	Dailymail	
	Start date	4/1/2007	6/1/2010	
	Due date	4/1/2015	4/1/2015	
	Duration (in days)	2922	1765	

3.2.4 Corpus

The group created this dataset to demonstrate the system once it is built. It is a collection of news articles from sources such as Inshorts and Yahoo News. This dataset will be used to demonstrate the tasks such as the classification of the news article, summarization of the news article, and the named entity recognition from the news article. The dataset is scraped in batches from 12th April 2023. It consists of the following attributes:

- link: the link to the news article
- title: the heading of the news article
- date: the date when the article was published
- content: the content within the news article
- label: the category of the news article
- source: the name of the source of the news article

Figure 48 and Figure 49 show the data collection plan for this corpus.

Figure 48

Data Description for the Corpus

Project number:		Project title:	Analyzing News Articles Using NLP Techniques	Project leader:		Date:	04/12/2023

Description of the data collection	
Why are we collecting the dataset?	The data is being collected to aid during the demonstration of the end product.
How are collecting the dataset?	The data is being collected using webscraping techniques from various news sources.
What will be done with the data once being collected?	The data will undergo a series of preprocessing and transformations steps before feeding it to the models to do their specfic tasks.

Figure 49

Key Variables for the Corpus

Key Variables - A summary of the chosen							
		1	2	3	4	5	6
	Variable title	link	title	date	content	label	source
	Input (X) or output	X	X	X	X	Y	X
	Unit of						
	Data type	String	String	Date/Time	String	String	String
	Collection method	Automated	Automated	Automated	Automated	Manual	Manual
	Historical data exist?	No					
	Operational	No					
	Data collector	Group					
	Start date	12-Apr	12-Apr	12-Apr	12-Apr	12-Apr	12-Apr
	Due date	4-Nov	4-Nov	4-Nov	4-Nov	4-Nov	4-Nov
	Duration (in days)	206	206	206	206	206	206

3.3 Data Pre-Processing

3.3.1 AG News Dataset

3.3.1.1 Drop Unnecessary Features and Filter Target

After inspecting the sample data and understanding the purpose of the task, there is a need to keep only useful attributes, including *source, title,* and *category*. Besides that, after understanding the distribution of the target *category* (as shown in Figure 50), we decide to proceed with the top five most popular and useful categories, which are: World, Sci/Tech, Business/ Entertainment, and Sports. Figure 50 shows the new shape of the dataset.

Figure 50

AG's News Dataset with Categories and their Value Counts

```
World                          186896
Sci/Tech                       154869
Business                       146651
Entertainment                  137437
Italia                         133428
Top News                       126514
Sports                         118131
Europe                          90573
Top Stories                     61579
U.S.                            47707
Health                          42629
Software and Developement       19041
Toons                            8016
Music Feeds                      7632
```

Figure 51

New dataset shape

```
After dropping columns and filter categories dataset shape
 (743984, 3)
```

Figure 52

New dataset sample

title	description	category
Light relay 'should be dropped'	A leading astronomy group in the US voices opp...	Sci/Tech
PeopleSoft Ousts CEO in Oracle Battle	In Oct 1 story headlined "PeopleSoft Oust...	Business
N.B. hospital workers plan to take strike to p...	Canadian Press - FREDERICTON (CP) - New Brunsw...	World
Incoming N.J. Governor a Veteran Lawmaker (AP)	AP - New Jersey's incoming governor is an old ...	World
Devoted Moviegoers Show Up Early	Frequent moviegoers tend to arrive at the thea...	Entertainment
Chair Umpire Who Blew Call Out at Open	The chair umpire who mistakenly overruled a ca...	Sports
Trial Lawyers Group Names New Chief	The Association of Trial Lawyers of America c...	Business

3.3.1.2 Handle missing values

Missing values are handled by first finding out if there are any values missing. Figure 53 shows missing values statistics in terms of counts and percentage in each attribute. With this significantly low value, we decided to drop the missing values as a solution.

Figure 53

Missing Values Statistics

```
               count  percentage
title              5    0.000672
description       15    0.002016
category           0    0.000000
```

Figure 54 shows the shape of the dataset after dropping. As compared to Figure 51, the number of records dropped by 20.

Figure 54

New data shape after handling missing values

```
(743964, 3)
```

3.3.1.3 Handle duplicates

The desired dataset does not need any redundant records; therefore, it is necessary to check if there are any duplicated values in the dataset. Figure 55 shows the sample duplicated data from the original dataset.

Figure 55

Duplicates sample

	title	description	category
34053	Athens closes 2004 Olympic Summer Games with p...	ATHENS (CP) - With pride and relief, Athens bi...	Sports
34074	Athens closes 2004 Olympic Summer Games with p...	ATHENS (CP) - With pride and relief, Athens bi...	Sports
34087	Athens closes 2004 Olympic Summer Games with p...	ATHENS (CP) - With pride and relief, Athens bi...	Entertainment
34111	Athens closes 2004 Olympic Summer Games with p...	ATHENS (CP) - With pride and relief, Athens bi...	Entertainment
34124	Athens closes 2004 Olympic Summer Games with p...	ATHENS (CP) - With pride and relief, Athens bi...	Sports

Figure 56 compares the dataset size before and after handling duplicates. There are 80,438 values duplicated being dropped in the dataset.

Figure 56

Dataset size before and after handling duplicates

```
Data set size before handling duplicates: 743964
Data set size after handling duplicates : 663526
```

3.3.1.4 Remove special characters

Even though picked transformers can handle on-breaking spaces and special characters, we decided to clean the data as a good practice as well as making sure of the input quality for data transformation step. Figure 57 shows unnecessary HTML tags and line break character '\n' existing from the dataset. Figure 58 shows the cleaned data.

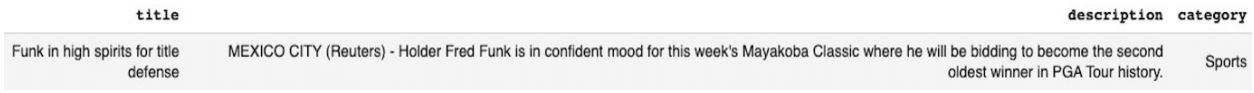
Figure 57

Data sample with HTML tags and special characters

title	description	category
Funk in high spirits for title defense	MEXICO CITY (Reuters) - Holder Fred Funk is in confident mood for this week's Mayakoba Classic where he will be bidding to become the second oldest winner in PGA Tour history. <img src="http://feeds.reuters.com/~a/reuters/sportsNews?i=rrZco0" border="0"></img> <img src="http://feeds.reuters.com/~f/reuters/sportsNews?i=zKDBQBE" border="0"></img> <img src="http://feeds.reuters.com/~f/reuters/sportsNews?i=ne65fLe" border="0"></img> <img src="http://feeds.reuters.com/~f/reuters/sportsNews?i=PjipVNe" border="0"></img> 	Sports

Figure 58

Cleaned data sample



title	description	category
Funk in high spirits for title defense	MEXICO CITY (Reuters) - Holder Fred Funk is in confident mood for this week's Mayakoba Classic where he will be bidding to become the second oldest winner in PGA Tour history.	Sports

3.3.1.5 Handling target imbalance

Our dataset is imbalanced in the category target variable. In order to make our target balanced, we randomly drop records corresponding to over sample values. Figure 59 shows the target variable distribution. Looking at the statistics, we want to make every category the same number of values as Sports. Figure 60 shows the target variable distribution after handling the imbalance.

Figure 59

Target variable distribution before handling imbalance

```
World            168864
Sci/Tech         136485
Business         132117
Entertainment    116703
Sports           109357
Name: category, dtype: int64
```

Figure 60

Target variable distribution after handling imbalance

```
Business          109357
Sci/Tech          109357
Entertainment     109357
Sports            109357
World             109357
Name: category, dtype: int64
```

3.3.1.6 Combine content features

This is an additional process, but after investigating AG's news dataset, it is better to combine the feature *title* with the feature *description* to create a content feature as input for the transformation and modeling process.

Figure 61

Dataset information after combining features

```
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype
---  ------    --------------   -----
 0   category  546785 non-null  object
 1   content   546785 non-null  object
dtypes: object(2)
memory usage: 12.5+ MB
```

3.3.1.7 Convert Target Feature

The target variable is initially labeled as text data. For the computer to understand the target feature, it is necessary to map each label to a distinct integer. As a result, the *Business* category is mapped to a value 0, *Sci/Tech* is 1, *Entertainment* is 2, *Sports* is 3, and *World* has a value of 4.

3.3.2 CoNLL2003 Dataset

3.3.2.1 Handle Missing Values

Figure 62 indicates that there are no null values (no empty value in any column) in the dataset. The id column is dropped as it can not be used for the analysis.

Figure 62

Information of All Variables in the CoNLL2003 Dataset from Hugging Face.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20744 entries, 0 to 20743
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   tokens      20744 non-null  object
 1   pos_tags    20744 non-null  object
 2   chunk_tags  20744 non-null  object
 3   ner_tags    20744 non-null  object
dtypes: object(4)
memory usage: 648.4+ KB
```

Next, we check if for any row, the lengths of the lists in any column are equal, meaning that there are no missing values within any string or integer list. Accordingly, we don't need to handle missing values issues with the CoNLL2003 dataset from Hugging Face.

3.3.2.2 Handle Duplicates

Pos_tags, chunk_tags, and ner_tags can have duplicate values, but the tokens column should not have any duplicates. The code below returns 2013 duplicates in the tokens column.

As a result, we eliminate all the duplicates from the dataset, which remains 18731 rows. Then, we re-index the dataframe.

Figure 63

Duplicates in the tokens column

```
df[df['tokens'].duplicated()].count()
```

```
tokens        2013
pos_tags      2013
chunk_tags    2013
ner_tags      2013
dtype: int64
```

3.3.3 CNN-Daily Mail Dataset

3.3.3.1 Drop Unnecessary Features

The CNN-Daily Mail dataset is hosted on Hugging Face, and when the dataset is loaded, it consists of three features; article, highlights, and id. The purpose of this dataset is to fine-tune models for performing abstractive summarization, which means that the 'id' feature can be removed due to providing no task benefit. Figure 64 shows the dataset size and columns after dropping the id feature, and Figure 65 depicts the dataset.

Figure 64

CNN-Daily Mail Dataset Size and Columns After Dropping 'id'

```
RangeIndex: 311971 entries, 0 to 311970
Data columns (total 2 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   article     311971 non-null  object
 1   highlights  311971 non-null  object
dtypes: object(2)
memory usage: 4.8+ MB
```

Figure 65

CNN-Daily Mail Dataset After Removing 'id' Feature

{'article': 'LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on Monday, but he insists the money won\'t cast a spell on him. Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the Phoenix" To the disappointment of gossip columnists around the world, the young actor says he has no plans to fritter his cash away on fast cars, drink and celebrity parties. "I don\'t plan to be one of those people who, as soon as they turn 18, suddenly buy themselves a massive sports car collection or something similar," he told an Australian interviewer earlier this month. "I don\'t think I\'ll be particularly extravagant. "The things I like buying are things that cost about 10 pounds -- books and CDs and DVDs." At 18, Radcliffe will be able to gamble in a casino, buy a drink in a pub or see the horror film "Hostel: Part II," currently six places below his number one movie on the UK box office chart. Details of how he\'ll mark his landmark birthday are under wraps. His agent and publicist had no comment on his plans. "I\'ll definitely have some sort of party," he said in an interview. "Hopefully none of you will be reading about it." Radcliffe\'s earnings from the first five Potter films have been held in a trust fund which he has not been able to touch. Despite his growing fame and riches, the actor says he is keeping his feet firmly on the ground. "People are always looking to say \'kid star goes off the rails,\'" he told reporters last month. "But I try very hard not to go that way because it would be too easy for them." His latest outing as the boy wizard in "Harry Potter and the Order of the Phoenix" is breaking records on both sides of the Atlantic and he will reprise the role in the last two films. Watch I-Reporter give her review of Potter\'s latest » . There is life beyond Potter, however. The Londoner has filmed a TV movie called "My Boy Jack," about author Rudyard Kipling and his son, due for release later this year. He will also appear in "December Boys," an Australian film about four boys who escape an orphanage. Earlier this year, he made his stage debut playing a tortured teenager in Peter Shaffer\'s "Equus." Meanwhile, he is braced for even closer media scrutiny now that he\'s legally an adult: "I just think I\'m going to be more sort of fair game," he told Reuters. E-mail to a friend . Copyright 2007 Reuters. All rights reserved.This material may not be published, broadcast, rewritten, or redistributed.',
 'highlights': "Harry Potter star Daniel Radcliffe gets £20M fortune as he turns 18 Monday .\nYoung actor says he has no plans to fritter his cash away .\nRadcliffe's earnings from first five Potter films have been held in trust fund ."}

3.3.3.2 Handle Missing Values

The CNN-Daily Mail dataset only requires a little pre-processing based on inspection. The dataset was checked for missing values, but no missing values were found. Figure 66 shows that there are no missing values for the two features.

Figure 66

CNN-Daily Mail Dataset Missing Values

```
article       0.0
highlights    0.0
```

3.3.3.3 Handle duplicates

Performing abstract summarization does not benefit from having duplicate values in the CNN-Daily Mail dataset. Upon inspection, it was found that 3101 duplicate values existed. The duplicate values were dropped, which resulted in 308870 rows of useful data to fine-tune the models. Figure 67 shows the dataset size and columns after dropping the duplicate values.

Figure 67

CNN-Daily Mail Dataset Size and Columns After Dropping Duplicates

```
RangeIndex: 308870 entries, 0 to 308869
Data columns (total 2 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   article     308870 non-null  object
 1   highlights  308870 non-null  object
dtypes: object(2)
memory usage: 4.7+ MB
```

3.3.4 Corpus

3.3.4.1 Dropping unnecessary columns

The corpus consists of certain features that add no value to the NLP tasks that are to be demonstrated. These features can be removed and only required features can be kept. Features such as link, date, source and title are dropped. Figure 68 shows the dataset before dropping the columns and Figure 69 shows the dataset after dropping the columns.

Figure 68

Sample Data Before Dropping Columns

	link	title	date	content	label	source
0	https://www.inshorts.com/en/news/heavy-gym-wor...	Heavy gym work, less net bowling reason behind...	12 Apr 2023,Wednesday	Former India captain Sunil Gavaskar, while tal...	Sports	Inshorts
1	https://www.inshorts.com/en/news/was-my-childh...	Was my childhood dream to bat alongside you: T...	12 Apr 2023,Wednesday	While interacting with Rohit Sharma following ...	Sports	Inshorts
2	https://www.inshorts.com/en/news/mis-tilak-win...	MI's Tilak wins 3rd straight dressing room Pla...	12 Apr 2023,Wednesday	Mumbai Indians (MI) awarded Tilak Varma with t...	Sports	Inshorts
3	https://www.inshorts.com/en/news/kkr-share-vid...	KKR share video of Litton Das watching Rinku's...	12 Apr 2023,Wednesday	KKR have shared a video in which Litton Das ca...	Sports	Inshorts
4	https://www.inshorts.com/en/news/mi-share-vide...	MI share video of Chawla's son in stands, call...	12 Apr 2023,Wednesday	Mumbai Indians shared a video of Piyush Chawla...	Sports	Inshorts

Figure 69

Sample Data After Dropping Columns

	content	label
0	Former India captain Sunil Gavaskar, while tal...	Sports
1	While interacting with Rohit Sharma following ...	Sports
2	Mumbai Indians (MI) awarded Tilak Varma with t...	Sports
3	KKR have shared a video in which Litton Das ca...	Sports
4	Mumbai Indians shared a video of Piyush Chawla...	Sports

3.3.4.2 Removing JSON Tags

During the scraping process the context feature for the articles from Yahoo News was collected in the JSON format. The feature has to be cleaned and combined as a single paragraph for each article to help in the steps further and display clean text without JSON tags in the front end. Figure 70 shows an instance of the context feature before cleaning.

Figure 70

Instance of Context Feature Before Cleaning


```
'[{"context":"BEIJING (AP) — Chinese health officials defended their search for the source of the COVID-19 virus and lashed out
Saturday at the World Health Organization after its leader said Beijing should have shared genetic information earlier."},{"con
text":"The WHO comments were “offensive and disrespectful,\\" said the director of the China Center for Disease Control and Pre
vention, Shen Hongbing. He accused the WHO of “attempting to smear China” and said it should avoid helping others \\"politicize
COVID-19.”"},{"context":"The global health body\'s director-general, Tedros Adhanom Ghebreyesus, said March 17 that newly discl
osed genetic material gathered in Wuhan in central China, where the first cases were detected in late 2019, “should have been s
hared three years ago.”"},{"context":"“As a responsible country and as scientists, we have always actively shared research resu
lts with scientists from around the world,” Shen said at a news conference."},{"context":"The origins of COVID-19 are still deb
ated and the focus of bitter political dispute."},{"context":"Many scientists believe it jumped from animals to humans at a mar
ket in Wuhan, but the city also is home to laboratories including China’s top facility for collecting viruses. That prompted su
ggestions COVID-19 might have leaked from one."},{"context":"The ruling Communist Party has tried to deflect criticism of its h
andling of the outbreak by spreading uncertainty about its origins."},{"context":"Officials have repeated anti-U.S. conspiracy
theories that the virus was created by Washington and smuggled into China. The government also says the virus might have entere
d China on mail or food shipments, though scientists abroad see no evidence to support that."},{"context":"Chinese officials su
ppressed information about the Wuhan outbreak in 2019 and punished a doctor who warned others about the new disease. The ruling
party reversed course in early 2020 and shut down access to major cities and most international travel to contain the diseas
e."},{"context":"The genetic material cited by the WHO’s Tedros was uploaded recently to a global database but collected in 202
0 at a Wuhan market where wildlife was sold."},{"context":"The samples show DNA from raccoon dogs mingled with the virus, scien
tists say. They say that adds evidence to the hypothesis COVID-19 came from animals, not a lab, but doesn’t resolve the questio
n of where it started. They say the virus also might have spread to raccoon dogs from humans."},{"context":"The information was
removed by Chinese officials from the database after foreign scientists asked the CDC about it, but it had been copied by a Fre
nch expert and shared with researchers outside China."},{"context":"A CDC researcher, Zhou Lei, who worked in Wuhan, said Chine
se scientists “shared all the data we had” and “adhered to principles of openness, objectivity and transparency.”"},{"contex
t":"Shen said scientists investigated the possibility of a laboratory leak and “fully shared our research and data without any
concealment or reservation.”"},{"context":"Shen said the source of COVID-19 had yet to be found, but he noted it took years to
identify the AIDS virus and its origin still is unclear."},{"context":"“Some forces and figures who instigate and participate i
n politicizing the traceability issue and attempting to smear China should not assume that the vision of the scientific communi
ty around the world will be blinded by their clumsy manipulation,” Shen said."}]'
```


After cleaning it, the instances look more readable and ready to be displayed in the front end. Figure 71 shows how the instance looks after cleaning.

Figure 71

Instance of Context Feature After Cleaning

```
'BEIJING (AP) – Chinese health officials defended their search for the source of the COVID-19 virus and lashed out Saturday at
the World Health Organization after its leader said Beijing should have shared genetic information earlier.The WHO comments wer
e “offensive and disrespectful," said the director of the China Center for Disease Control and Prevention, Shen Hongbing. He ac
cused the WHO of “attempting to smear China” and said it should avoid helping others "politicize COVID-19.”The global health bo
dy\'s director-general, Tedros Adhanom Ghebreyesus, said March 17 that newly disclosed genetic material gathered in Wuhan in ce
ntral China, where the first cases were detected in late 2019, “should have been shared three years ago.”“As a responsible coun
try and as scientists, we have always actively shared research results with scientists from around the world,” Shen said at a n
ews conference.The origins of COVID-19 are still debated and the focus of bitter political dispute.Many scientists believe it j
umped from animals to humans at a market in Wuhan, but the city also is home to laboratories including China’s top facility for
collecting viruses. That prompted suggestions COVID-19 might have leaked from one.The ruling Communist Party has tried to defle
ct criticism of its handling of the outbreak by spreading uncertainty about its origins.Officials have repeated anti-U.S. consp
iracy theories that the virus was created by Washington and smuggled into China. The government also says the virus might have
entered China on mail or food shipments, though scientists abroad see no evidence to support that.Chinese officials suppressed
information about the Wuhan outbreak in 2019 and punished a doctor who warned others about the new disease. The ruling party re
versed course in early 2020 and shut down access to major cities and most international travel to contain the disease.The genet
ic material cited by the WHO’s Tedros was uploaded recently to a global database but collected in 2020 at a Wuhan market where
wildlife was sold.The samples show DNA from raccoon dogs mingled with the virus, scientists say. They say that adds evidence to
the hypothesis COVID-19 came from animals, not a lab, but doesn’t resolve the question of where it started. They say the virus
also might have spread to raccoon dogs from humans.The information was removed by Chinese officials from the database after for
eign scientists asked the CDC about it, but it had been copied by a French expert and shared with researchers outside China.A C
DC researcher, Zhou Lei, who worked in Wuhan, said Chinese scientists “shared all the data we had” and “adhered to principles o
f openness, objectivity and transparency.”Shen said scientists investigated the possibility of a laboratory leak and “fully sha
red our research and data without any concealment or reservation.”Shen said the source of COVID-19 had yet to be found, but he
noted it took years to identify the AIDS virus and its origin still is unclear.“Some forces and figures who instigate and parti
cipate in politicizing the traceability issue and attempting to smear China should not assume that the vision of the scientific
community around the world will be blinded by their clumsy manipulation,” Shen said.'
```

3.3.4.3 Removing Duplicate Instances

The scraping process is automated hence, there are chances of scraping the same articles more than once. Removing these duplicates is an important step. Once enough data has been collected, the duplicates will be dropped to avoid such redundancies using Python.

3.4 Data Transformation

3.4.1 AG News Dataset

After pre-processing the dataset, data transformation is needed to transform data into the format that the picked pre-trained models such as BERT, RoBERTa, and ERNIE can understand. The tokenization process for these transformers involves splitting the input text into tokens (can be subwords, words, punctuation, symbols, etc..), then mapping those tokens to an integer in-built with the model, which is called input_ids. However, there is a maximum length that each model can take. BERT, RoBERTa, and ERNIE can take the length of input text up to 512 tokens. If input text is longer than that, there is a need for the sequence to be handled. If the input text is shorter than maximum length, there is an option to pad in the list to make input length consistent. There are two special tokens, which are [SEP], and [CLS], which help the models understand meaning and structure of input sequences. In specific, [CLS] token represents the beginning of an input sequence, and [SEP] separates the first with second input sequence. Figure 72 shows the sample input before the transformation process begins. The below section will describe in detail the tokenization result from the three models.

Figure 72

Sample Content Before Transformation

```
('Wall St. Bears Claw Back Into the Black (Reuters) Reuters - '
 "Short-sellers, Wall Street's dwindling band of ultra-cynics, are seeing "
 'green again.')
```

3.4.1.1 BERT and ERNIE

BERT and ERNIE use WordPiece tokenization algorithms. WordPiece tokenization starts from a small vocabulary beginning with the alphabet. However, WordPiece identifies its subwords by adding a prefix # for its characters or subwords. Through its rules, the vocabulary

library learns more merged elements. Figure 73 shows the sample content above after being tokenized by the WordPiece algorithm. Figure 74 shows the tokenized being converted to integers called ID that are built and understood by the two models. Since the tokenization algorithm for BERT and ERNIE is different from RoBERTa, the resulting tokenization and IDs also differ.

Figure 73

Sample Content After BERT and ERNIE Tokenize

```
['wall', 'st', '.', 'bears', 'claw', 'back', 'into', 'the', 'black', '(',
 'reuters', ')', 'reuters', '-', 'short', '-', 'sellers', ',', 'wall',
 'street', "'", 's', 'd', '##wind', '##ling', '\n', 'band', 'of', 'ultra',
 '-', 'cy', '##nic', '##s', ',', 'are', 'seeing', 'green', 'again', '.']
```

Figure 74

Sample Content Tokenized IDs for BERT and ERNIE

```
[2813, 2358, 1012, 6468, 15020, 2067, 2046, 1996, 2304, 1006,
 26665, 1007, 26665, 1011, 2460, 1011, 19041, 1010, 2813, 2395,
 1005, 1055, 1040, 11101, 2989, 2316, 1997, 11087, 1011, 22330,
 8713, 2015, 1010, 2024, 3773, 2665, 2153, 1012]
```

Figure 75

Sample Transformed Data for BERT and ERNIE

	content_text	input_ids	attention_mask	targets
0	Wall St. Pullback Reflects Tech Blowout (Reute...	[101, 2813, 2358, 1012, 4139, 5963, 11138, 662...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0
1	Wall St. Bears Claw Back Into the Black (Reute...	[101, 2813, 2358, 1012, 6468, 15020, 2067, 204...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0
2	Oil and Economy Cloud Stocks' Outlook (Reuters...	[101, 3514, 1998, 4610, 6112, 15768, 1005, 176...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0
3	Iraq Halts Oil Exports from Main Southern Pipe...	[101, 5712, 9190, 2015, 3514, 14338, 2013, 236...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0
4	Stocks End Up, But Near Year Lows (Reuters) Re...	[101, 15768, 2203, 2039, 1010, 2021, 2379, 209...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0

3.4.1.2 RoBERTa

RoBERTa on the other hand uses Byte-Pair Encoding (BPE) tokenization algorithm. This algorithm has the base vocabulary that contains every ASCII character, and the characters will be

merged together based on the frequency to create new vocabulary through learning. Characters that are not in the base vocabulary will be tokenized as [UNK], meaning unknown characters.

Figure 76

Sample Content After RoBERTa Tokenize

```
['Wall', 'ĠSt', '.', 'ĠBears', 'ĠClaw', 'ĠBack', 'ĠInto', 'Ġthe', 'ĠBlack',
 'Ġ(', 'Reuters', ')', 'ĠReuters', 'Ġ-', 'ĠShort', '-', 'sell', 'ers', ',',
 'ĠWall', 'ĠStreet', "'s", 'Ġdwindling', 'Ġband', 'Ġof', 'Ġultra', '-',
 'cy', 'n', 'ics', ',', 'Ġare', 'Ġseeing', 'Ġgreen', 'Ġagain', '.']
```

Figure 77

Sample Content Tokenized IDs for RoBERTa

```
[28216, 312, 4, 6033, 44121, 3727, 20693, 5, 1378, 36, 1251, 43,
 1201, 111, 7787, 12, 5727, 268, 6, 2298, 852, 18, 25564, 1971,
 9, 9620, 12, 4469, 282, 2857, 6, 32, 1782, 2272, 456, 4]
```

Figure 78

Sample Transformed Data for RoBERTa

	content_text	input_ids	attention_mask	targets
0	Wall St. Bears Claw Back Into the Black (Reute...	[0, 28216, 312, 4, 6033, 44121, 3727, 20693, 5...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0
1	Carlyle Looks Toward Commercial Aerospace (Reu...	[0, 9518, 352, 459, 26646, 20232, 1120, 9501, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0
2	Oil and Economy Cloud Stocks' Outlook (Reuters...	[0, 32397, 8, 15735, 7941, 312, 6368, 108, 110...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0
3	Iraq Halts Oil Exports from Main Southern Pipe...	[0, 37590, 6579, 1872, 4541, 3015, 8303, 31, 4...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0
4	Oil prices soar to all-time record, posing new...	[0, 32397, 850, 27283, 7, 70, 12, 958, 638, 6,...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	0

Figure 79

Sample Data After Converting Target Variable

	category	content
0	0	Wall St. Pullback Reflects Tech Blowout (Reute...
1	0	Wall St. Bears Claw Back Into the Black (Reute...
2	0	Carlyle Looks Toward Commercial Aerospace (Reu...
3	0	Oil and Economy Cloud Stocks' Outlook (Reuters...
4	0	Iraq Halts Oil Exports from Main Southern Pipe...

3.4.2 CoNLL2003 Dataset

3.4.2.1 BERT

The sentences in the Hugging Face dataset are broken down to single word level, which does not take into account the context and meaning of words to generate a sequence of subword tokens. To get the best results, we decide to re-tokenize the dataset to best fit the models we choose for the entity classification task, which are BERT, ERNIE, and ELECTRA

For BERT and ERNIE, we use the AutoTokenizer class from the same bert-based-case pretrained model to create a tokenizer object. Then, we utilize the tokenizer to tokenize the pre-tokenized input. The figures below give an example of the first row before and after tokenizing. Observe that one more element is added in front and at the end of the list, “CLS” and “SEP”. Additionally, the word “lamb” is divided into “la” and “##mb”

Figure 80

First Element of the Tokens Column before BERT Tokenization

```
['EU', 'rejects', 'German', 'call', 'to', 'boycott', 'British', 'lamb', '.']
```

Figure 81

First Element of the Tokens Column after BERT Tokenization

```
['[CLS]', 'EU', 'rejects', 'German', 'call', 'to', 'boycott', 'British',
'la', '##mb', '.', '[SEP]']
```

Another thing that needs to be done is adjusting the length of the ner_tags elements to match with the length of the tokens elements. The rule is that we add (-100) to the beginning and the end of the sentence as an indicator and the tokens that are separated have the same label as their original words. For example, "##mb" is assigned the same label as "la". We conduct the tokenization and adjusting process for the entire dataset and the result is as follows.

Figure 82

Token View after BERT Tokenization and Conversion.

	tokens	labels
0	[[CLS], EU, rejects, German, call, to, boycott...	[-100, 3, 0, 7, 0, 0, 0, 7, 0, 0, 0, -100]
1	[[CLS], Peter, Blackburn, [SEP]]	[-100, 1, 2, -100]
2	[[CLS], BR, ##US, ##SE, ##LS, 1996, -, 08, -, ...	[-100, 5, 6, 6, 6, 0, 0, 0, 0, 0, -100]
3	[[CLS], The, European, Commission, said, on, T...	[-100, 0, 3, 4, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, ...
4	[[CLS], Germany, ', s, representative, to, the...	[-100, 5, 0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 1, ...
...	...	...
18726	[[CLS], That, is, why, this, is, so, emotional...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18727	[[CLS], ", It, was, the, joy, that, we, all, h...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18728	[[CLS], Charlton, managed, Ireland, for, 93, m...	[-100, 1, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18729	[[CLS], He, guided, Ireland, to, two, successi...	[-100, 0, 0, 5, 0, 0, 0, 7, 8, 0, 0, 0, 0, 0, ...
18730	[[CLS], The, la, ##nky, former, Leeds, United,...	[-100, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 5, 0, ...

18731 rows × 2 columns

We have been showing the dataset with word tokens, but in fact, the inputs should be input_ids, token_type_ids, attention_mask, and labels. input_ids is a tensor that represents the numerical representation of the tokenized input sequence, token_type_ids is a tensor that indicates which segment of the input sequence each token belongs to, and the attention_mask

column contains a binary value that indicates whether the corresponding token is a valid token (1) or a padding token (0). We extract the information from Figure 83 to get Figure 84.

Figure 83

BERT Tokenized Dataset

```
tokenized_datasets
```

```
DatasetDict({
    data: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 18731
    })
})
```

Figure 84

BERT Tokenized Dataset in Dataframe Format

	input_ids	token_type_ids	attention_mask	labels
0	[101, 7270, 22961, 1528, 1840, 1106, 21423, 14...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	[-100, 3, 0, 7, 0, 0, 0, 7, 0, 0, 0, -100]
1	[101, 1943, 14428, 102]	[0, 0, 0, 0]	[1, 1, 1, 1]	[-100, 1, 2, -100]
2	[101, 26660, 13329, 12649, 15928, 1820, 118, 4...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	[-100, 5, 6, 6, 6, 0, 0, 0, 0, 0, -100]
3	[101, 1109, 1735, 2827, 1163, 1113, 9170, 1122...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 3, 4, 0, 0, 0, 0, 0, 7, 0, 0, 0, ...
4	[101, 1860, 112, 188, 4702, 1106, 1103, 1735, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 5, 0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 1, ...
...	...	...	...	...
18726	[101, 1337, 1110, 1725, 1142, 1110, 1177, 6438...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18727	[101, 107, 1135, 1108, 1103, 8730, 1115, 1195,...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18728	[101, 19155, 2374, 2270, 1111, 5429, 2697, 117...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 1, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18729	[101, 1124, 8610, 2270, 1106, 1160, 11598, 129...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 5, 0, 0, 0, 7, 8, 0, 0, 0, 0, ...
18730	[101, 1109, 2495, 22792, 1393, 7383, 1244, 891...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 5, 0, ...

18731 rows × 4 columns

The models require fixed-length inputs, which means that all input sequences need to have the same length. Accordingly, the next step is to pad the input sequences, which can be done by the package DataCollatorForTokenClassification from library transformers. For example, we will compare the first two rows of labels column in Figure 85 with these after padding. Observe that the second row is padded with similar length of the first one by the (-100)s.

Figure 85

First Two Rows of Labels after Padding - BERT

```
tensor([[-100,    3,    0,    7,    0,    0,    0,    7,    0,    0,    0, -100],
        [-100,    1,    2, -100, -100, -100, -100, -100, -100, -100, -100, -100]])
```

The result of the padding process are the input tensors deriving from the tokenized dataset. Figure 86 and present how the tensors look like in data frame form. The length of all lists now are 173. Observe that the attention_mark column now has both values 0 and 1, where 0 means padding token.

Figure 86

Input tensors in Dataframe Form - BERT

	input_ids	token_type_ids	attention_mask	labels
0	[101, 7270, 22961, 1528, 1840, 1106, 21423, 14...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, ...	[-100, 3, 0, 7, 0, 0, 0, 7, 0, 0, 0, -100, -10...
1	[101, 1943, 14428, 102, 0, 0, 0, 0, 0, 0, 0...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[-100, 1, 2, -100, -100, -100, -100, -100, -10...
2	[101, 26660, 13329, 12649, 15928, 1820, 118, 4...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, ...	[-100, 5, 6, 6, 6, 0, 0, 0, 0, 0, -100, -100, ...
3	[101, 1109, 1735, 2827, 1163, 1113, 9170, 1122...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 3, 4, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, ...
4	[101, 1860, 112, 188, 4702, 1106, 1103, 1735, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 5, 0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 1, ...
...	...	...	...	...
18726	[101, 1337, 1110, 1725, 1142, 1110, 1177, 6438...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18727	[101, 107, 1135, 1108, 1103, 8730, 1115, 1195,...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18728	[101, 19155, 2374, 2270, 1111, 5429, 2697, 117...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 1, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18729	[101, 1124, 8610, 2270, 1106, 1160, 11598, 129...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 5, 0, 0, 0, 7, 8, 0, 0, 0, 0, 0, ...
18730	[101, 1109, 2495, 22792, 1393, 7383, 1244, 891...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 5, 0, ...

18731 rows × 4 columns

Figure 87

Input Tensors Length - BERT

```
<bound method NDFrame._add_numeric_operations.<locals>.max of 0        173
1        173
2        173
3        173
4        173
        ...
18726    173
18727    173
18728    173
18729    173
18730    173
Name: input_ids, Length: 18731, dtype: int64>
```

After data transformation, there are a couple of more duplicate values appearing in the input_ids column shown in Figure 88. Thus, we drop the duplicates one more time to avoid bias. Figure 89 shows that the transformed dataset has 18726 rows and 4 columns.

Figure 88

Duplicates after Tokenization and Padding - BERT

```
batch_df[batch_df['input_ids'].duplicated()].count()
```

```
input_ids         5
token_type_ids    5
attention_mask    5
labels            5
dtype: int64
```

Figure 89

ConLL2003 Sample Transformed Data for BERT

	input_ids	token_type_ids	attention_mask	labels
0	[101, 7270, 22961, 1528, 1840, 1106, 21423, 14...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, ...	[-100, 3, 0, 7, 0, 0, 0, 7, 0, 0, 0, -100, -10...
1	[101, 1943, 14428, 102, 0, 0, 0, 0, 0, 0, 0, 0...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[-100, 1, 2, -100, -100, -100, -100, -100, -10...
2	[101, 26660, 13329, 12649, 15928, 1820, 118, 4...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, ...	[-100, 5, 6, 6, 6, 0, 0, 0, 0, 0, -100, -100, ...
3	[101, 1109, 1735, 2827, 1163, 1113, 9170, 1122...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 3, 4, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, ...
4	[101, 1860, 112, 188, 4702, 1106, 1103, 1735, ...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 5, 0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 1, ...
...	...	...	...	...
18726	[101, 1337, 1110, 1725, 1142, 1110, 1177, 6438...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18727	[101, 107, 1135, 1108, 1103, 8730, 1115, 1195,...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18728	[101, 19155, 2374, 2270, 1111, 5429, 2697, 117...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 1, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18729	[101, 1124, 8610, 2270, 1106, 1160, 11598, 129...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 5, 0, 0, 0, 7, 8, 0, 0, 0, 0, 0, ...
18730	[101, 1109, 2495, 22792, 1393, 7383, 1244, 891...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 5, 0, ...

18726 rows × 4 columns

3.4.2.2 ERNIE

Applying the same logic to transform the data for ERNIE using a pre-trained ernie-2.0-base-en model. We have 24 input_id duplicates after tokenizing and 18707 rows and 4 columns at the end of the process. Figure 93 presents the sample transformed data that is used for the ERNIE model.

Figure 92

Duplicates after Tokenization and Padding - ERNIE

```
input_ids         24
token_type_ids    24
attention_mask    24
labels            24
dtype: int64
```

Figure 93

ConLL2003 Sample Transformed Data for ERNIE

	input_ids	token_type_ids	attention_mask	labels
0	[101, 7327, 19164, 2446, 2655, 2000, 17757, 23...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, ...	[-100, 3, 0, 7, 0, 0, 0, 7, 0, 0, -100, -100, ...
1	[101, 2848, 13934, 102, 0, 0, 0, 0, 0, 0, 0...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[-100, 1, 2, -100, -100, -100, -100, -100, -10...
2	[101, 9371, 2727, 1011, 5511, 1011, 2570, 102,...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, ...	[-100, 5, 0, 0, 0, 0, 0, -100, -100, -100, -10...
3	[101, 1996, 2647, 3222, 2056, 2006, 9432, 2009...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 3, 4, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, ...
4	[101, 2762, 1005, 1055, 4387, 2000, 1996, 2647...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 5, 0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 1, ...
...	...	...	...	...
18726	[101, 2008, 2003, 2339, 2023, 2003, 2061, 6832...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18727	[101, 1000, 2009, 2001, 1996, 6569, 2008, 2057...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18728	[101, 17821, 3266, 3163, 2005, 6109, 3503, 101...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 1, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
18729	[101, 2002, 8546, 3163, 2000, 2048, 11165, 208...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 5, 0, 0, 0, 7, 8, 0, 0, 0, 0, 0, ...
18730	[101, 1996, 17595, 4801, 2280, 7873, 2142, 829...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 0, 5, 0, ...

18707 rows × 4 columns

3.4.2.3 ELECTRA

Similar to BERT, we use the AutoTokenizer class from an electra-base-discriminator pre-trained model to create a tokenizer object. We also apply the same transformation steps on the preprocessed data. Figure 90 shows that ELECTRA tokenizer creates more duplicate values in the input_ids column than BERT tokenizer after data transformation. We drop the duplicates one more time to avoid bias. Figure 91 displays the sample of transformed dataset after dropping duplicates.

Figure 90

Duplicates after Tokenization and Padding - BERT

```
input_ids         24
token_type_ids    24
attention_mask    24
labels            24
dtype: int64
```

Figure 91

ConLL2003 Sample Transformed Data for ELECTRA

	input_ids	token_type_ids	attention_mask	labels
0	[101, 7327, 19164, 2446, 2655, 2000, 17757, 23...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, ...	[-100, 3, 0, 7, 0, 0, 0, 7, 0, 0, -100, -100, ...
1	[101, 2848, 13934, 102, 0, 0, 0, 0, 0, 0, 0...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[-100, 1, 2, -100, -100, -100, -100, -100, -10...
2	[101, 9371, 2727, 1011, 5511, 1011, 2570, 102,...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, ...	[-100, 5, 0, 0, 0, 0, 0, -100, -100, -100, -10...
3	[101, 1996, 2647, 3222, 2056, 2006, 9432, 2009...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 0, 3, 4, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, ...
4	[101, 2762, 1005, 1055, 4387, 2000, 1996, 2647...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ...	[-100, 5, 0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 0, 1, ...

3.4.3 CNN-Daily Mail Dataset

3.4.3.1 T5-11B

The transformation process for the CNN-Daily Mail dataset depends on the model being implemented. This project uses T5-11B and BART-base for the abstract summarization task, which means that both models require a different approach to how the data will be processed for tokenization.

The T5 model employs a T5 tokenizer that originates from the XLNet tokenizer, which itself is built upon the unsupervised text tokenizer known as SentencePiece (Raffel et al. 2019). T5 requires a prefix of “summarization:” for the input text to perform a summarization task. Additionally, it is essential to ensure that the model does not exceed the maximum context size; therefore, the parameter truncation is set to a max length of 1024. This means that the tokenization will prevent encoding lengths any longer than 1024. This is one of the primary reasons that T5-11B was selected over using the standard T5-base because T5-base only allows for a max input length of 512. The shorter input length of 512 for the T5-base equates to a model

that is less capable of producing abstract summarizations than T5-11B. Figure 94 shows the first article in the CNN-Daily Mail dataset before tokenization, and Figure 95 shows the same article after tokenization.

Figure 94

CNN-Daily Mail First Article Before T5 Tokenization

{'article': 'LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on Monday, but
he insists the money won\'t cast a spell on him. Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the Phoenix" To the disappointment of gossip
columnists around the world, the young actor says he has no plans to fritter his cash away on fast cars, drink and celebrity parties. "I don\'t plan to be one of those
people who, as soon as they turn 18, suddenly buy themselves a massive sports car collection or something similar," he told an Australian interviewer earlier this month.
"I don\'t think I\'ll be particularly extravagant. "The things I like buying are things that cost about 10 pounds -- books and CDs and DVDs." At 18, Radcliffe will be
able to gamble in a casino, buy a drink in a pub or see the horror film "Hostel: Part II," currently six places below his number one movie on the UK box office chart.
Details of how he\'ll mark his landmark birthday are under wraps. His agent and publicist had no comment on his plans. "I\'ll definitely have some sort of party," he
said in an interview. "Hopefully none of you will be reading about it." Radcliffe\'s earnings from the first five Potter films have been held in a trust fund which he
has not been able to touch. Despite his growing fame and riches, the actor says he is keeping his feet firmly on the ground. "People are always looking to say \'kid star
goes off the rails,\'" he told reporters last month. "But I try very hard not to go that way because it would be too easy for them." His latest outing as the boy wizard
in "Harry Potter and the Order of the Phoenix" is breaking records on both sides of the Atlantic and he will reprise the role in the last two films. Watch I-Reporter
give her review of Potter\'s latest » . There is life beyond Potter, however. The Londoner has filmed a TV movie called "My Boy Jack," about author Rudyard Kipling and
his son, due for release later this year. He will also appear in "December Boys," an Australian film about four boys who escape an orphanage. Earlier this year, he made
his stage debut playing a tortured teenager in Peter Shaffer\'s "Equus." Meanwhile, he is braced for even closer media scrutiny now that he\'s legally an adult: "I just
think I\'m going to be more sort of fair game," he told Reuters. E-mail to a friend . Copyright 2007 Reuters. All rights reserved.This material may not be published,
broadcast, rewritten, or redistributed.',

Figure 95

CNN-Daily Mail First Article After T5 Tokenization

{'input_ids': [301, 24796, 4170, 6, 2789, 41, 18844, 61, 1636, 8929, 16023, 2213, 4173, 6324, 12591, 15, 11391, 592, 12, 3, 9, 2196, 3996, 1755, 770, 8785, 591, 11039,
770, 61, 13462, 38, 3, 88, 5050, 507, 30, 2089, 6, 68, 3, 88, 10419, 7, 8, 540, 751, 31, 17, 4061, 3, 9, 10783, 30, 376, 5, 4173, 6324, 12591, 15, 38, 8929, 16023, 16,
96, 15537, 651, 16023, 11, 8, 5197, 13, 8, 12308, 121, 304, 8, 19142, 13, 29517, 6710, 343, 7, 300, 8, 296, 6, 8, 1021, 7556, 845, 3, 88, 65, 150, 1390, 12, 9030, 17,
449, 112, 1723, 550, 30, 1006, 2948, 6, 3281, 11, 17086, 2251, 5, 96, 196, 278, 31, 17, 515, 12, 36, 80, 13, 273, 151, 113, 6, 38, 1116, 38, 79, 919, 14985, 8247, 805,
1452, 3, 9, 3805, 2100, 443, 1232, 42, 424, 1126, 976, 3, 88, 1219, 46, 3746, 2772, 49, 2283, 48, 847, 5, 96, 196, 278, 31, 17, 317, 27, 31, 195, 36, 1989, 28887, 5, 96,
634, 378, 27, 114, 2611, 33, 378, 24, 583, 81, 335, 7051, 1636, 1335, 11, 3190, 7, 11, 5677, 7, 535, 486, 14985, 6324, 12591, 15, 56, 36, 3, 179, 12, 24068, 16, 3, 9,
2653, 6, 805, 3, 9, 3281, 16, 3, 9, 11943, 42, 217, 8, 12082, 814, 96, 4489, 7, 1625, 10, 2733, 2466, 976, 1083, 1296, 1747, 666, 112, 381, 80, 1974, 30, 8, 1270, 1367,
828, 5059, 5, 9487, 13, 149, 3, 88, 31, 195, 3946, 112, 15754, 3591, 33, 365, 6215, 7, 5, 978, 3102, 11, 452, 343, 141, 150, 1670, 30, 112, 1390, 5, 96, 196, 31, 195,
1728, 43, 128, 1843, 13, 1088, 976, 3, 88, 243, 16, 46, 2772, 5, 96, 12599, 5839, 13, 25, 56, 36, 1183, 81, 34, 535, 6324, 12591, 15, 31, 7, 8783, 45, 8, 166, 874,
16023, 4852, 43, 118, 1213, 16, 3, 9, 2019, 3069, 84, 3, 88, 65, 59, 118, 3, 179, 12, 1586, 5, 3, 4868, 112, 1710, 10393, 11, 20816, 6, 8, 7556, 845, 3, 88, 19, 2627,
112, 1922, 3, 16804, 30, 8, 1591, 5, 96, 24337, 33, 373, 479, 12, 497, 3, 31, 2168, 26, 2213, 1550, 326, 8, 6579, 7, 6, 31, 121, 3, 88, 1219, 19644, 336, 847, 5, 96,
11836, 27, 653, 182, 614, 59, 12, 281, 24, 194, 250, 34, 133, 36, 396, 514, 21, 135, 535, 978, 1251, 91, 53, 38, 8, 4940, 25027, 16, 96, 15537, 651, 16023, 11, 8, 5197,
13, 8, 12308, 121, 19, 7814, 3187, 30, 321, 4458, 13, 8, 9640, 11, 3, 88, 56, 21504, 8, 1075, 16, 8, 336, 192, 4852, 5, 4195, 27, 18, 1649, 1493, 49, 428, 160, 1132, 13,
16023, 31, 7, 1251, 1168, 3, 5, 290, 19, 280, 1909, 16023, 6, 983, 5, 37, 1524, 49, 65, 3, 25403, 3, 9, 1424, 1974, 718, 96, 7008, 7508, 4496, 976, 81, 2291, 17806,
6636, 4320, 102, 697, 11, 112, 520, 6, 788, 21, 1576, 865, 48, 215, 5, 216, 56, 92, 2385, 16, 96, 29835, 16575, 976, 1], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}

Figure 96 shows what the input_ids look like when the model decodes them. Notice that the prefix is added at the beginning for the T5 model to know it will perform summarization, and the ‘</s>’ is added at the end to mark the eos token. Additionally, dynamic padding was utilized but can not be seen in the figure below because the article length is long enough not to require padding.

Figure 96

CNN-Daily Mail First Article After T5 Tokenization Decoded


```
array('summarize: LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on
Monday, but he insists the money won\'t cast a spell on him. Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the Phoenix" To the disappointment of
gossip columnists around the world, the young actor says he has no plans to fritter his cash away on fast cars, drink and celebrity parties. "I don\'t plan to be one of
those people who, as soon as they turn 18, suddenly buy themselves a massive sports car collection or something similar," he told an Australian interviewer earlier this
month. "I don\'t think I\'ll be particularly extravagant. "The things I like buying are things that cost about 10 pounds -- books and CDs and DVDs." At 18, Radcliffe
will be able to gamble in a casino, buy a drink in a pub or see the horror film "Hostel: Part II," currently six places below his number one movie on the UK box office
chart. Details of how he\'ll mark his landmark birthday are under wraps. His agent and publicist had no comment on his plans. "I\'ll definitely have some sort of party,"
he said in an interview. "Hopefully none of you will be reading about it." Radcliffe\'s earnings from the first five Potter films have been held in a trust fund which he
has not been able to touch. Despite his growing fame and riches, the actor says he is keeping his feet firmly on the ground. "People are always looking to say \'kid star
goes off the rails,\'" he told reporters last month. "But I try very hard not to go that way because it would be too easy for them." His latest outing as the boy wizard
in "Harry Potter and the Order of the Phoenix" is breaking records on both sides of the Atlantic and he will reprise the role in the last two films. Watch I-Reporter
give her review of Potter\'s latest ». There is life beyond Potter, however. The Londoner has filmed a TV movie called "My Boy Jack," about author Rudyard Kipling and
his son, due for release later this year. He will also appear in "December</s>',
      dtype='<U2074')
```


3.4.3.2 BART

Performing tokenization with the BART model varied compared to T5. Unlike T5, BART uses an encoder similar to BERT and a decoder that is like GPT. BART shares merges and vocab with the model RoBERTa, allowing BART to handle max input lengths of 1024. This equates to an equal length compared to the T5-11B model. The tokenization process for BART was similar to T5, except there was no need for a “summarize:” prefix to cue BART for performing a summarization task. Padding was also applied during the tokenization process, but due to the lengths of the articles will not be necessary. Figure 95 shows the first article from the CNN-Daily Mail dataset after BART tokenization. Figure 96 shows the first article after BART decodes it.

Figure 97

CNN-Daily Mail First Article After BART Tokenization


```
{'input_ids': [0, 574, 4524, 6, 1156, 36, 1251, 43, 480, 3268, 10997, 999, 3028, 7312, 20152, 3077, 899, 7, 10, 431, 984, 844, 153, 1358, 4006, 4, 134, 153, 43, 13016,
25, 37, 4072, 504, 15, 302, 6, 53, 37, 9838, 5, 418, 351, 75, 2471, 10, 8921, 15, 123, 4, 3028, 7312, 20152, 25, 3268, 10997, 11, 22, 29345, 10997, 8, 5, 9729, 9, 5,
5524, 113, 598, 5, 10208, 9, 20445, 6730, 1952, 198, 5, 232, 6, 5, 664, 2701, 161, 37, 34, 117, 708, 7, 856, 3961, 1334, 39, 1055, 409, 15, 1769, 1677, 6, 4076, 8, 6794,
1799, 4, 22, 100, 218, 75, 563, 7, 28, 65, 9, 167, 82, 54, 6, 25, 1010, 25, 51, 1004, 504, 6, 6017, 907, 1235, 10, 2232, 1612, 512, 2783, 50, 402, 1122, 60, 37, 174, 41,
2059, 33242, 656, 42, 353, 4, 22, 100, 218, 75, 206, 38, 581, 28, 1605, 31879, 4, 22, 133, 383, 38, 101, 2159, 32, 383, 14, 701, 59, 158, 2697, 480, 2799, 8, 32570, 8,
37206, 72, 497, 504, 6, 7312, 20152, 40, 28, 441, 7, 23104, 11, 10, 10297, 6, 907, 10, 4076, 11, 10, 8881, 50, 192, 5, 8444, 822, 22, 40534, 523, 35, 4657, 3082, 60,
855, 411, 2127, 874, 39, 346, 65, 1569, 15, 5, 987, 2233, 558, 5966, 4, 10574, 9, 141, 37, 581, 2458, 39, 10043, 4115, 32, 223, 18166, 4, 832, 2936, 8, 285, 661, 56,
117, 1129, 15, 39, 708, 4, 22, 100, 581, 2299, 33, 103, 2345, 9, 537, 60, 37, 26, 11, 41, 1194, 4, 22, 19204, 4146, 9, 47, 40, 28, 2600, 59, 24, 72, 7312, 20152, 18,
1107, 31, 5, 78, 292, 10997, 3541, 33, 57, 547, 11, 10, 2416, 1391, 61, 37, 34, 45, 57, 441, 7, 2842, 4, 2285, 39, 1197, 9444, 8, 33884, 6, 5, 2701, 161, 37, 16, 2396,
39, 1730, 10523, 15, 5, 1255, 4, 22, 4763, 32, 460, 546, 7, 224, 128, 30704, 999, 1411, 160, 5, 26717, 10076, 37, 174, 1865, 94, 353, 4, 22, 1708, 38, 860, 182, 543, 45,
7, 213, 14, 169, 142, 24, 74, 28, 350, 1365, 13, 106, 72, 832, 665, 9913, 25, 5, 2143, 32660, 11, 22, 29345, 10997, 8, 5, 9729, 9, 5, 5524, 113, 16, 3433, 2189, 15, 258,
2380, 9, 5, 5038, 8, 37, 40, 769, 22627, 5, 774, 11, 5, 94, 80, 3541, 4, 1437, 3075, 38, 12, 22026, 10679, 492, 69, 1551, 9, 10997, 18, 665, 9313, 479, 345, 16, 301,
1684, 10997, 6, 959, 4, 20, 928, 254, 34, 10571, 10, 1012, 1569, 373, 22, 2387, 5637, 2722, 60, 59, 2730, 19257, 2675, 11488, 20418, 8, 39, 979, 6, 528, 13, 800, 423,
42, 76, 4, 91, 40, 67, 2082, 11, 22, 17704, 8732, 60, 41, 2059, 822, 59, 237, 2786, 54, 5111, 41, 21297, 1580, 4, 3322, 42, 76, 6, 37, 156, 39, 1289, 2453, 816, 10,
20464, 7044, 11, 2155, 840, 16717, 18, 22, 28568, 687, 72, 2276, 6, 37, 16, 5378, 8988, 13, 190, 2789, 433, 7731, 122, 14, 37, 18, 7818, 41, 4194, 35, 22, 100, 95, 206,
38, 437, 164, 7, 28, 55, 2345, 9, 2105, 177, 60, 37, 174, 1201, 4, 381, 12, 6380, 7, 10, 1441, 479, 1894, 3010, 1201, 4, 404, 659, 1875, 4, 713, 1468, 189, 45, 28, 1027,
6, 2308, 6, 4599, 6, 50, 3802, 4, 2], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
```


Figure 98

CNN-Daily Mail First Article After BART Tokenization Decoded

```
array('<s>LONDON, England (Reuters) -- Harry Potter star Daniel Radcliffe gains access to a reported £20 million ($41.1 million) fortune as he turns 18 on Monday, but he
insists the money won\'t cast a spell on him. Daniel Radcliffe as Harry Potter in "Harry Potter and the Order of the Phoenix" To the disappointment of gossip columnists
around the world, the young actor says he has no plans to fritter his cash away on fast cars, drink and celebrity parties. "I don\'t plan to be one of those people who,
as soon as they turn 18, suddenly buy themselves a massive sports car collection or something similar," he told an Australian interviewer earlier this month. "I don\'t
think I\'ll be particularly extravagant. "The things I like buying are things that cost about 10 pounds -- books and CDs and DVDs." At 18, Radcliffe will be able to
gamble in a casino, buy a drink in a pub or see the horror film "Hostel: Part II," currently six places below his number one movie on the UK box office chart. Details of
how he\'ll mark his landmark birthday are under wraps. His agent and publicist had no comment on his plans. "I\'ll definitely have some sort of party," he said in an
interview. "Hopefully none of you will be reading about it." Radcliffe\'s earnings from the first five Potter films have been held in a trust fund which he has not been
able to touch. Despite his growing fame and riches, the actor says he is keeping his feet firmly on the ground. "People are always looking to say \'kid star goes off the
rails,\'" he told reporters last month. "But I try very hard not to go that way because it would be too easy for them." His latest outing as the boy wizard in "Harry
Potter and the Order of the Phoenix" is breaking records on both sides of the Atlantic and he will reprise the role in the last two films.  Watch I-Reporter give her
review of Potter\'s latest ». There is life beyond Potter, however. The Londoner has filmed a TV movie called "My Boy Jack," about author Rudyard Kipling and his son,
due for release later this year. He will also appear in "December Boys," an Australian film about four boys who escape an orphanage. Earlier this year, he made his stage
debut playing a tortured teenager in Peter Shaffer\'s "Equus." Meanwhile, he is braced for even closer media scrutiny now that he\'s legally an adult: "I just think I\'m
going to be more sort of fair game," he told Reuters. E-mail to a friend. Copyright 2007 Reuters. All rights reserved.This material may not be published, broadcast,
rewritten, or redistributed.</s>',
      dtype='<U2532')
```

3.4.4 Corpus

3.4.4.1 Tokenization

The corpus created will go through the tokenization process. This process depends on the models selected as each model goes through a specific tokenization method so that the model can understand the data. Hence, the tokenization methods of the best model for each task will be applied to the corpus.

3.4.4.2 Merging of the Dataset

After the desired amount of news articles have been scraped from the sources, the corpus will be combined together using the merge function in Python to obtain a single dataset.

3.5 Data Preparation

3.5.1 AG News Dataset

After data transformation, data preparation is done on AG's news data with a 70-15-15 splitting ratio in training, validation, and testing datasets. Figure 99 shows the size of each dataset. Training set has the most data, with 382749 rows. Validation and test set has an equal number of data, with 82018 rows.

Figure 99

Training/Validation/Testing set

```
DatasetDict({
    train: Dataset({
        features: ['content_text', 'input_ids', 'attention_mask', 'targets'],
        num_rows: 382749
    })
    validation: Dataset({
        features: ['content_text', 'input_ids', 'attention_mask', 'targets'],
        num_rows: 82018
    })
    test: Dataset({
        features: ['content_text', 'input_ids', 'attention_mask', 'targets'],
        num_rows: 82018
    })
})
```

3.5.2 CoNLL2003 Dataset

For data preparation, we split the dataset into training, testing, and evaluation dataset into the portion of 80%, 10%, and 10% respectively. Figure 100, 101, and 102 depict the BERT, ERNIE, and ELECTRA datasets after splitting. BERT has 14980 rows and 4 columns for training dataset, 1873 rows and 4 columns for testing dataset, and 1873 rows and 4 columns for validation. In the meantime, ELECTRA and ERNIE have slightly lower numbers of rows in each dataset.

Figure 100

Transformed Dataset Splitting - BERT

```
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 14980
    })
    test: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 1873
    })
    validation: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 1873
    })
})
```

Figure 101

Transformed Dataset Splitting - ERNIE

```
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 14965
    })
    test: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 1871
    })
    validation: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 1871
    })
})
```

Figure 102

Transformed Dataset Splitting - ELECTRA

```
DatasetDict({
    train: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 14965
    })
    test: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 1871
    })
    validation: Dataset({
        features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
        num_rows: 1871
    })
})
```

3.5.3 CNN-Daily Mail Dataset

Data preparation for the CNN-Daily Mail dataset is straightforward since Hugging Face has already performed training, validation, and a test split. Therefore, when loading the dataset from Hugging Face, it is important to specify to read in all three datasets. The dataset split is 287113 training, 13368 validation, and 11490 testing. The split translates to about 92%, 4%, and 4%, which is standard for summarization fine-tuning. Figure 103 shows the dataset split.

Figure 103

CNN-Daily Mail Dataset Transformed Split

```
DatasetDict({
    train: Dataset({
        features: ['article', 'highlights'],
        num_rows: 287113
    })
    validation: Dataset({
        features: ['article', 'highlights'],
        num_rows: 13368
    })
    test: Dataset({
        features: ['article', 'highlights'],
        num_rows: 11490
    })
})
```

3.6 Data Statistics

3.6.1 AG News Dataset

Table 3 below shows the summarization of AG's dataset throughout the entire data engineering process. The number of rows, columns, and storage size show the statistics of the data at the end of the process.

Table 3

Statistics of AG's News

Stage	Dataset	# of Rows	# of Columns	Storage Size
Raw	Original: AG's news	1281103	9	88.0 MB
Pre-Processing	Preprocessed AG's news	546785	2	28.6 MB
Transformation	Transformed AG's news	546785	4	16.7 MB
Preparation	Training	382749	4	11.7 MB

	Testing	82018	4	2.5 MB
	Validation	82018	4	2.5 MB

3.6.2 CoNLL2003 Dataset

Table 4 below shows the summarization of the ConLL2003 dataset throughout the data engineering process including the number of rows, columns, and storage size of the data at the end of each step.

Table 4

Statistics of Datasets

Stage	Dataset	# of Rows	# of Columns	Storage Size
Raw	- Original: CoNLL2003 - Hugging Face: CoNLL2003	23616 20744	4 4	Hugging Face: 648.4 KB
Pre-Processed	CoNLL2003	18731	4	585.5 KB
Transformation	Tokenization, Extract word ID and Padding	18726	4	731.5 KB
Preparation	Training	14980	4	468.2 KB
	Testing	1873	4	58.7 KB
	Validation	1873	4	58.7 KB

Figure 107, 108, and 109 illustrates the statistics of the raw, preprocessed, transformed, and splitted datasets. Raw dataset has 18731 unique values, therefore 2013 duplicates, and "Scorers:" is the most frequent sentence.

Figure 107

Hugging Face CoNLL2003 Raw Dataset Statistics

	tokens	pos_tags	chunk_tags	ner_tags
count	20744	20744	20744	20744
unique	18731	13126	11282	8047
top	[Scorers, :]	[22, 11]	[11, 12]	[5, 0]
freq	30	611	1290	955

On the other hand, the preprocessed dataset has no duplicated sentences with 18731 unique values. The features are similar to the raw dataset.

Figure 108

CoNLL2003 Preprocessed Dataset Statistics

	tokens	pos_tags	chunk_tags	ner_tags
count	18731	18731	18731	18731
unique	18731	13126	11282	8044
top	[EU, rejects, German, call, to, boycott, Briti...	[22, 11, 11, 11, 11, 11, 11, 11]	[11, 12]	[0, 1, 2, 0, 5, 0, 0]
freq	1	352	696	580

The transformed dataset now has different attributes than the raw and preprocessed datasets and has no duplicated sentences with 18726 unique values in the input_ids column.

Figure 109

CoNLL2003 Transformed Dataset Statistics

	input_ids	token_type_ids	attention_mask	labels
count	18726	18726	18726	18726
unique	18726	1	99	9757
top	[101, 7270, 22961, 1528, 1840, 1106, 21423, 14...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, ...	[-100, 3, 0, 0, 0, 0, 0, 0, 0, -100, -100, -10...
freq	1	18726	1063	280

The training dataset has 14980 unique values and no duplicates in the input_ids column and no duplicates.

Figure 110

CoNLL2003 Training Dataset Statistics

	input_ids	token_type_ids	attention_mask	labels
count	14980	14980	14980	14980
unique	14980	1	96	8013
top	[101, 1109, 13484, 117, 1152, 2936, 11579, 113...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, ...	[-100, 3, 0, 0, 0, 0, 0, 0, 0, -100, -100, -10...
freq	1	14980	850	226

The testing dataset has 1873 unique values and no duplicates in the input_ids column and no duplicates.

Figure 111

CoNLL2003 Testing Dataset Statistics

	input_ids	token_type_ids	attention_mask	labels
count	1873	1873	1873	1873
unique	1873	1	70	1261
top	[101, 121, 119, 8636, 4974, 121, 118, 121, 119...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, ...	[-100, 3, 0, 0, 0, 0, 0, 0, 0, -100, -100, -10...
freq	1	1873	111	28

The validation dataset has 1873 unique values and no duplicates in the input_ids column and no duplicates.

Figure 112

CoNLL2003 Validation Dataset Statistics

	input_ids	token_type_ids	attention_mask	labels
count	1873	1873	1873	1873
unique	1873	1	71	1232
top	[101, 107, 10109, 1620, 8940, 1127, 2602, 1187...	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, ...	[-100, 0, 0, 0, 0, 0, 0, -100, -100, -100, -10...
freq	1	1873	103	29

3.6.3 CNN-Daily Mail Dataset

The table presented below displays a summary of the CNN-Daily Mail dataset across the complete data engineering process, including statistics on the number of rows, columns, and storage size at the conclusion of the process.

Table 5

Statistics of CNN-Daily Mail Dataset

Stage	Dataset	# of Rows	# of Columns	Storage Size
Raw	CNN-Daily Mail	311971	3	7.1 MB
Pre-Processed	CNN-Daily Mail	308870	2	5.2 MB
Transformation	Tokenization	308870	2	4.4 MB
Preparation	Training	284162	2	4 MB
	Validation	12534	2	200 KB
	Testing	12534	2	200 KB

3.6.4 Corpus

The scrapping of the news articles for the corpus is a continuous process that will take months to finish. Hence, the number of rows is not fixed as the articles are being scrapped in batches. Table 6 shows the statistics of the corpus.

Table 6

Data Statistics for the Corpus

Stage	Dataset	# of Rows	# of Columns

Raw	- InShorts Scrapped	150+	6
	- Yahoo Scrapped	150+	6
Pre-Processed	- InShorts Scrapped	150+ (Assuming No Duplicates)	2
	- Yahoo Scrapped	150+ (Assuming No Duplicates)	2
Transformation	Corpus	300+	2

3.7 Data Analytics Results

Figure 104 shows the distribution of the target attribute in the original AG's news dataset. Figure 105 shows the distribution after the transformation process. Notice that in Figure 64, 0 stands for *Business*, 1 is for *Sci/Tech*, 2 is for *Entertainment*, 3 is for *Entertainment*, and *4 is for World* category.

Figure 104

AG's News Target Distribution Before Data Engineering

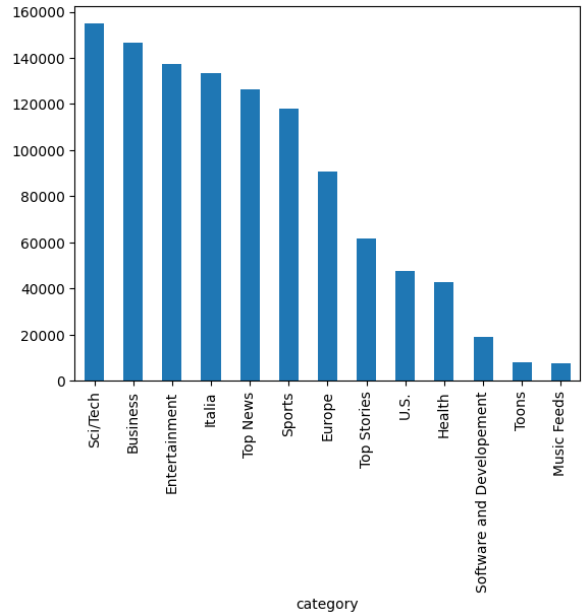


Figure 105

AG's News Target Distribution After Data Engineering

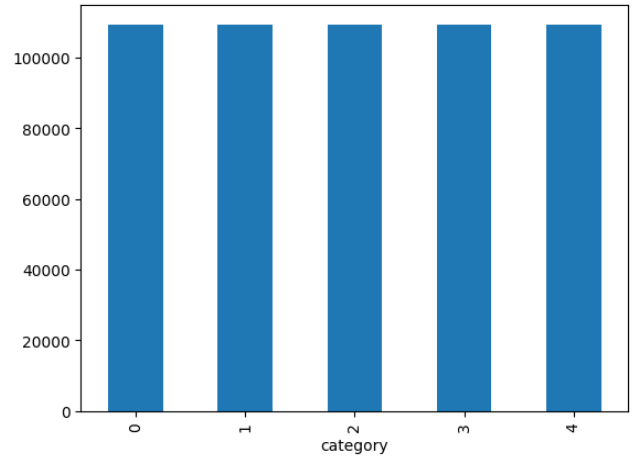
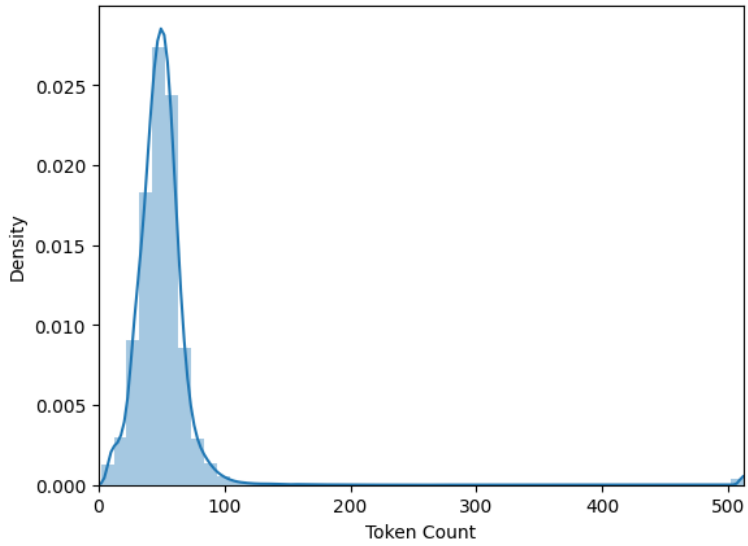


Figure 106 shows the density distribution of the content feature after tokenization.

Figure 106

AG's News Tokens Distribution After Tokenization



The word count for PAD token is 2816521 (Figure 113), which is so many compared with other token counts . Therefore it is reasonable to ignore the PAD tokens for now and investigate the next thirty words. Most of the next thirty words in Figure 114 are non-noun or non-verb words.

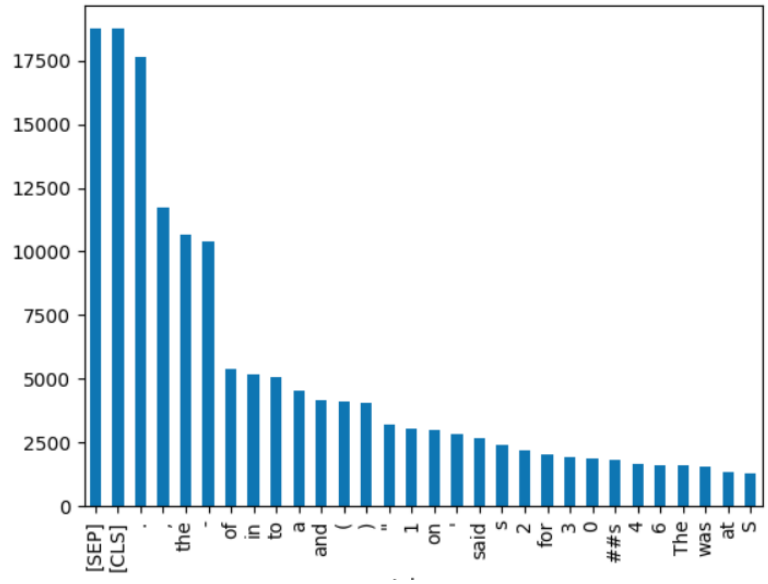
Figure 113

Transformed dataset word count

```
tokens
[PAD]          2816521
[SEP]            18726
[CLS]            18726
.                17614
,                11739
                 ...
##rated              1
programming          1
programs             1
credits              1
contestant           1
Length: 16340, dtype: int64
```

Figure 114

Thirty most frequent tokens in transformed dataset without [PAD] chart



For labels, the word count for the PAD label is also very high, 2853973 (Figure 115). Thus, we ignore the PAD labels and investigate the others. In figure 116, "O" stands for "Outside" and has the highest number of label counts. The next most frequent label is I-PER, which stands for "Inside a Person's name". It is used to tag tokens that are part of a named entity representing a person's name, following a "B-PER" (Beginning of a Person's name) tag that marks the start of the entity.

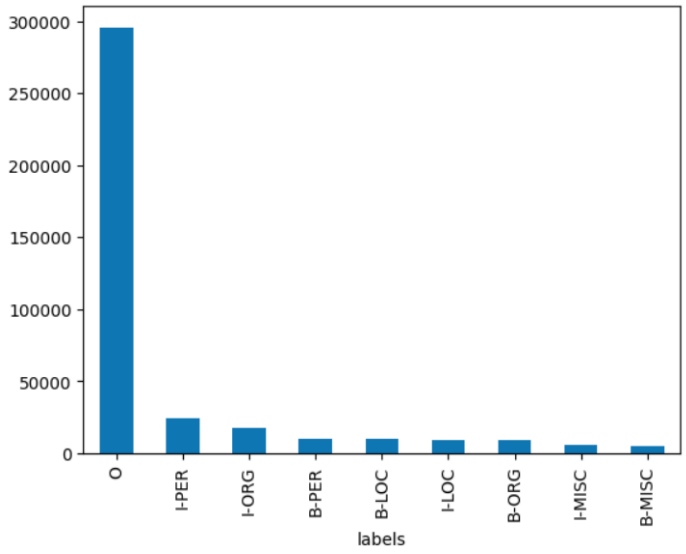
Figure 115

Transformed dataset label count

```
labels
PAD       2853973
O          295616
I-PER       24231
I-ORG       17525
B-PER        9911
B-LOC        9836
I-LOC        9404
B-ORG        8806
I-MISC       5460
B-MISC       4836
dtype: int64
```

Figure 116

Label count in transformed dataset without [PAD] chart



4. Model Development

4.1 Model Proposals

4.1.1 BERT

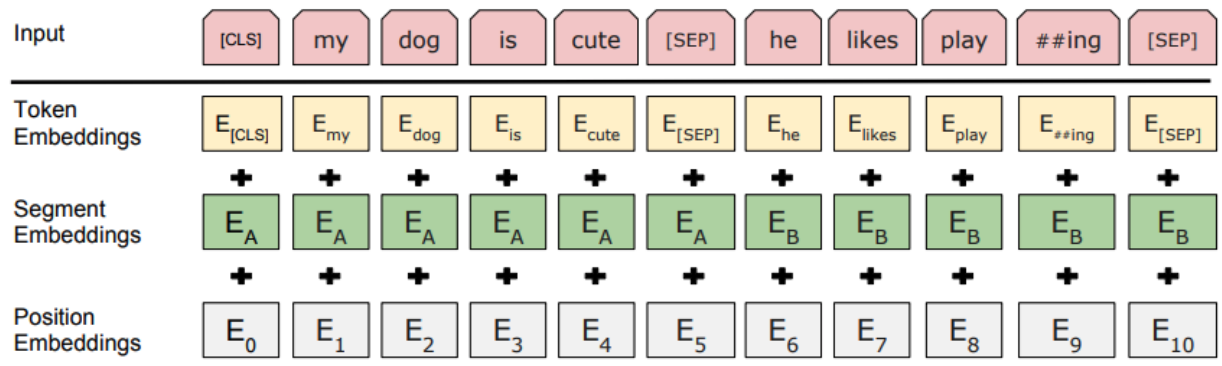
Devlin et al. (2018) introduced BERT as the first model to apply transformer architecture on bidirectional context to out-perform prior works. BERT was pre-trained on huge sources of data with 2.5B words on English Wikipedia, and 800M words on BooksCorpus (Zhu et al., 2015). This allows the application by fine-tuning BERT with an additional output layer on a specific downstream task with a few number of learning parameters to achieve state-of-the-art performances. On top of the massive training data, BERT stands out as a Masked Language Model (MLM), which enables bidirectional learning by forcing BERT to use surrounding words to predict the masked word. Next Sentence Prediction (NSP) helps BERT understand if there is a relationship between a given sentence and the previous one. According to the authors, NSP contributes to the achievement of the BERT model on sentence-level tasks such as natural language inference and paraphrasing. BERT is trained on both MLM and NSP at the same time.

BERT's model architecture is based on Transformer architecture described in Vaswani et. al (2017). However, it is an encoder only model with a multi-layer bidirectional learning mechanism. The Transformer architecture facilitates massive parallelization to enable BERT training with huge amounts of data in a fairly short amount of time.

Figure 1 explains how BERT models take an input sequence and transform it into an input for BERT model. Delvin et al. (2018) gives an example of the input sequence "My dog is cute. He likes playing". This sequence after going through token embeddings, segment embeddings, and position embeddings layer will return an output, and this output will be input for the BERT model. Figure 2 shows the general architecture of an encoding only model based on the framework in the paper and transformer architecture in Vaswani et al. (2017). Input representation is taken care of by the encoder blocks to generate final contextualized embeddings. The encoder block for BERT models consists of multiple multi-head self-attention layers, followed by feed forward layers. Depending on the size of the BERT model, there is a difference in the number of encoder blocks. The output of the BERT pre-trained model, which is the contextualized embeddings, will be used as input for the additional layer when it comes to the fine-tuning process with a specific task.

Figure 1

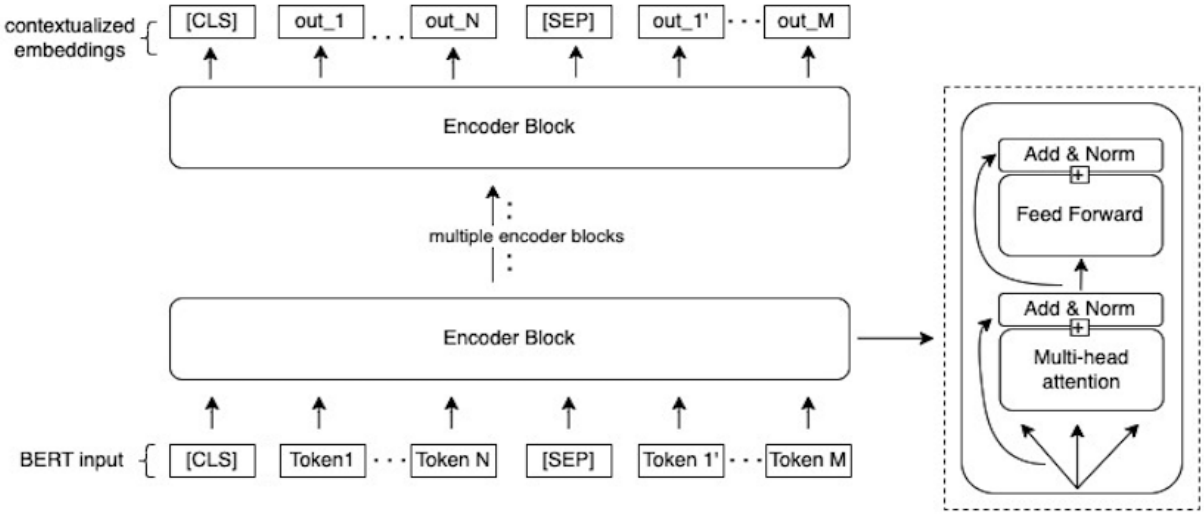
BERT Input Representation



Note. BERT Input representation by embedding layers. Adapted from “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, by Devlin, J., Chang, M., Lee, K., & Toutanova, K., 2018, *ArXiv (Cornell University)*. Copyright 2018 by ArXiv.

Figure 2

Input tokens Flow Diagram - BERT Model



4.1.2 RoBERTa

Liu et al. (2019) introduces RoBERTa (Robustly Optimized BERT Pre-training Approach) as a transformer-based neural network architecture for natural language processing tasks, including text classification. The paper mentions that Roberta was developed by Facebook AI and is an improvement on the BERT (Bidirectional Encoder Representations) model. Accordingly, the basic concept of RoBERTa is quite similar to BERT: a neural network architecture that uses self-attention mechanisms to process input data. Self-attention is the model’s ability to work on different parts of the input sequence while learning contextual relationships between the tokens. This allows the model to capture both short-term and long-term dependencies and results in better performance on language tasks.

The paper also brings up the features of RoBERTa:

- Masked language modeling: RoBERTa masks a certain portion of the input text and let the model predict the masked tokens based on context
- Dynamic masking: a masking pattern is generated every time a sentence is fed, meaning that for the same sentence inputting in the model, the masks are different each time.
- Full sentences without next sentence prediction: each document is a combination of contiguous sentences. The next document is created from sampling.
- Pre-training: RoBERTa is a pre-trained model on large batches of text data.
- Fine-tuning: the model is fine-tuned after pre-training to meet the requirements of the specific NLP tasks (Liu et al., 2019).

Liu et al. (2019) mention that RoBERTa uses a transformer architecture with multi layers, self-attention heads, and hidden dimensions that has several encoder layers and no decoder. As RoBERTa is an improved version of BERT, it includes a bidirectional encoder which can take into account the context of each token in the input sequence from both directions. left-to-right and right-to-left. An encoder layer takes in a sequence of input tokens and produces a sequence of hidden representations. To be more specific, the model uses Byte-Pair Encoding. Byte-Pair Encoding relies on subwords units that are extracted from text corpus analysis. RoBERTa also uses residual connections and layer normalization in each layer, which helps the model to learn more efficiently and avoid the vanishing gradient problem.

RoBERTa follows a transformer-based algorithm with an encoder system, and it is also a type of unsupervised learning model for natural language processing (NLP). Liu et al. (2019) describe the setup of RoBERTa as follows: applying transformer architecture with self-attention heads, training objectives with masked language model with dynamic masking, training

objectives with Full Sentences approach, training the model with significantly large mini-batches and increased learning rates, and fine-turning. After the pre-training process is completed, it is possible to fine-tune for a specific task using supervised learning techniques.

This is an example of how the model works on a text classification task. For a classification task, we need to fine-tune the pre-trained model on a labeled dataset. During this process, the model maps the input text with the provided label, which adds a linear layer on top of the mechanism. When we put a new corpus into a pre-trained RoBERTa model for a classification task, the model uses its knowledge of language to classify the text into predefined categories as it has learned to understand the relationships between words and sentences. This new dataset should already be tokenized and padded in a form that the machine can understand. Then, the model will generate a prediction for each input text, indicating which label the text belongs to.

4.1.3 ERNIE

Enhanced Representation through kNowledge IntEgration (ERNIE) is a language representation model which is enhanced by knowledge, developed by Baidu Inc. It improves natural language processing by integrating external knowledge sources such as Wikipedia into the pre-training process of language models. The purpose of ERNIE is to improve language representation by incorporating knowledge masking techniques, specifically entity-level masking and phrase-level masking. This is achieved through the design of the pre-training approach of the language model (Sun et al., 2019). Hence, it is highly beneficial for Classification,Question Answering and Sentiment Analysis tasks.

The authors start by pointing out the limitations of existing language models, such as BERT, which are based on pre-training on large amounts of text data, but do not incorporate

external knowledge sources. They argue that integrating external knowledge can improve the performance of NLP tasks, especially in cases where the training data is limited or the language is complex.

ERNIE is based on a transformer encoder architecture similar to BERT, but with several modifications to incorporate external knowledge. Specifically, ERNIE uses two additional pre-training tasks: entity masking and entity linking. In the entity masking task, ERNIE randomly masks some of the entity mentions in the input text and requires the model to predict the masked entities based on the context. In the entity linking task, ERNIE requires the model to link the entity mentioned in the input text to the corresponding entities in a knowledge base. the architecture of the ERNIE model:

- Tokenization: The input text is first split into individual tokens or words.
- Embedding Layer: Each token is then converted into a continuous vector representation using pre-trained word embeddings.
- Transformer Encoder Layers: The token embeddings are then processed by a stack of transformer encoder layers. These layers are responsible for capturing the context and dependencies between the tokens in the input sequence. ERNIE uses a modified version of the transformer architecture that includes several enhancements, such as cross-attention and relative position representations.
- Knowledge Integration: ERNIE incorporates external knowledge from various sources, such as Wikipedia, to enrich the token embeddings. This is done by aligning the tokens in the input text with the corresponding entities or concepts in the knowledge base, and then updating the token embeddings based on the information from the relevant entities. The

two types of masking strategies used in ERNIE are entity-level masking and phrase-level masking. Figure 3 shows the different types of masking.

  - Entity-level masking is a type of basic level masking that involves randomly masking entity mentions in the input text and requiring the model to predict the masked entities based on the context.
  - Phrase-level masking, on the other hand, is a more complex form of masking that involves masking entire phrases or clauses in the input text. This allows ERNIE to capture more complex relationships between words and concepts, such as idiomatic expressions and collocations.

- Pooling Layer: The output of the transformer encoder layers is then fed into a pooling layer, which aggregates the information from all the tokens in the input sequence into a fixed-size representation. ERNIE uses several pooling strategies, such as mean pooling and max pooling, to capture different aspects of the input.
- Classification Layer: Finally, the pooled representation is passed through a classification layer to make predictions on the given task. ERNIE can be fine-tuned on a wide range of natural language processing tasks, such as sentiment analysis, text classification, and question-answering.

Figure 3

Different Types of Masking

Note. Sun, Y., Shuohuan, W., Li, Y., Feng, S., Chen, X., Zhang, H., Tian, X., Danxiang, Z., Tian, H., & Wu, H. (2019). ERNIE: Enhanced Representation through Knowledge Integration. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1904.09223

The authors also conduct ablation studies to analyze the contributions of different components of ERNIE. They find that both the masking tasks contribute to the performance improvement, and that using a large external knowledge base is also important.

4.1.4 ELECTRA

Clark et al. (2020) proposed an efficient pre-training task called replaced token detection, which improves upon the inabilities of BERT where the input is replaced by [MASK] during pre-training. This method requires a lot of computation power and large amounts of data to produce significant results. Whereas, in a replaced token detection instead of using a masking technique on the input, the method alters it by substituting certain tokens with believable alternatives that are randomly selected from a limited generator network. Rather than teaching a model to anticipate the initial identities of the altered tokens, we instruct a discriminative model to predict whether each token in the changed input was swapped with a sample from the generator network or not.

The paper also proposes a novel pre-training method for natural language processing (NLP) tasks called Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA) using the above pre-training task. The goal of pre-training is to train a model on a large amount of unlabeled text data, so that it can be fine-tuned on specific downstream tasks, such as sentiment analysis or text classification.

The ELECTRA architecture consists of a generator network and a discriminator network, both of which are trained jointly in a self-supervised manner. The generator network is a small

neural network that takes in a masked input sequence and generates a set of fake tokens to replace the masked tokens. The goal of the generator is to create fake tokens that are as realistic as possible, so that the discriminator network is unable to distinguish between the real and fake tokens.

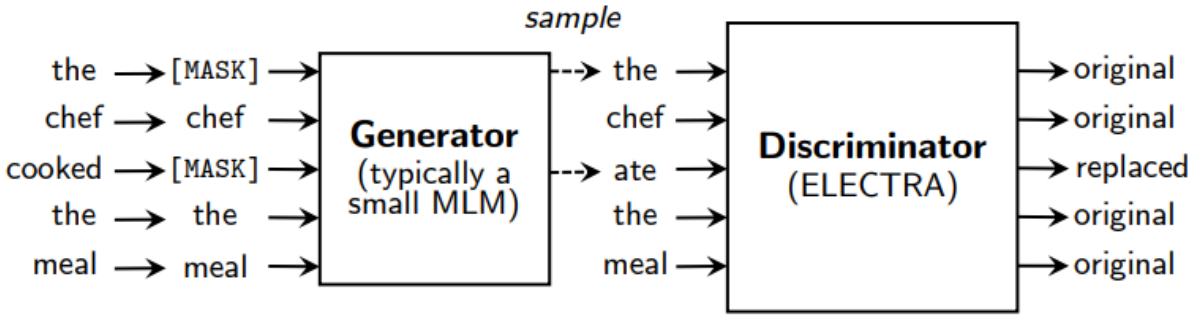
The discriminator network is a larger neural network that takes in an input sequence, and is trained to classify whether each token in the sequence is real or fake. Specifically, the discriminator receives as input a sequence that has been modified in one of two ways:

- "Real" input sequence: some of the tokens in the sequence are randomly masked out, and the discriminator is trained to predict the original, unmasked token at each masked position.
- "Fake" input sequence: the same tokens are masked out, but instead of predicting the original tokens, the discriminator is trained to predict whether each masked token is real or fake.

Figure 4 shows the sample of how the replaced token detection works.

Figure 4

Overview of Replaced Token Detection



Note. Clark, K. R., Luong, M., Le, Q. V., & Manning, C. D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2003.10555

The generator network is trained to minimize the binary cross-entropy loss between its predictions and the discriminator's predictions on the fake input sequence, while the discriminator network is trained to maximize the binary cross-entropy loss between its predictions and the true labels (real or fake) on both the real and fake input sequences.

The authors also introduce a novel training technique called "contrastive adversarial training", which encourages the generator network to produce diverse sets of fake tokens that are difficult for the discriminator to distinguish. In this approach, the generator is trained not only to minimize the binary cross-entropy loss, but also to maximize the cosine similarity between the embeddings of different fake tokens generated from the same input sequence.

Overall, the ELECTRA architecture is designed to efficiently learn a discriminative representation of the input sequence by leveraging the generator network to create challenging fake tokens that the discriminator must learn to distinguish from real tokens. This approach has shown to be effective in achieving state-of-the-art performance on a wide range of natural language processing tasks such as NER.

4.1.5 BART

BART (Bidirectional and Auto-Regressive Transformers) was launched in a research paper by Lewis et al. (2020) and is a pre-trained model that includes two objects: bidirectional denoising autoencoder and sequence-to-sequence pre-training. Lewis et al. (2020) mention that the first stage of pre-training is corrupting the text with an arbitrary noise function and the second stage is reconstructing the text by a sequence-to-sequence model. Accordingly, the

denoising autoencoder used in BART is a type of neural network that is trained to reconstruct a clean input from a noisy version of that input. The authors point out that BART works well for text generation tasks and comprehension tasks when fine-tuning and the model has the ability to reconstruct any kind of text corruption, even when all source's information is lost.

The following features are listed in the BART introductory research paper:

- The denoising autoencoder is one of BART's outstanding features. The special point of the denoising autoencoder in BART is that it is trained on corrupted data, which means the data is intentionally or unintentionally altered or perturbed in some way.
- Autoregressive decoder generates output in the order of left to right. A characteristic of the autoregressive decoder is that it predicts the next token based on the meaning of the previous ones.
- Pre-training: BART model is trained on a massive amount of text data to reconstruct the original input sequence from a corrupted version of the sequence.
- BART can be fine-tuned for several NLP tasks, such as sequence classification tasks, token classification tasks, abstractive question answering and summarization, English translation, and so forth (Lewis et al, 2020).

The architecture of BART is encoder-decoder, which is based on the transformer architecture. Lewis et al. (2020) apply six encoder-decoder layers for the base model and 12 layers for the large model. BART is also known as a bidirectional transformer because its encoder can understand the input tokens in both directions: left to right and right to left. This bidirectional encoding technique helps BART better capture the relationship between tokens and create more comprehensive text. On the other hand, the autoregressive decoder generates output

tokens in a left-to-right order, which is unidirectional. The decoder also performs cross-attention and self-attention over the encoder's final hidden layer.

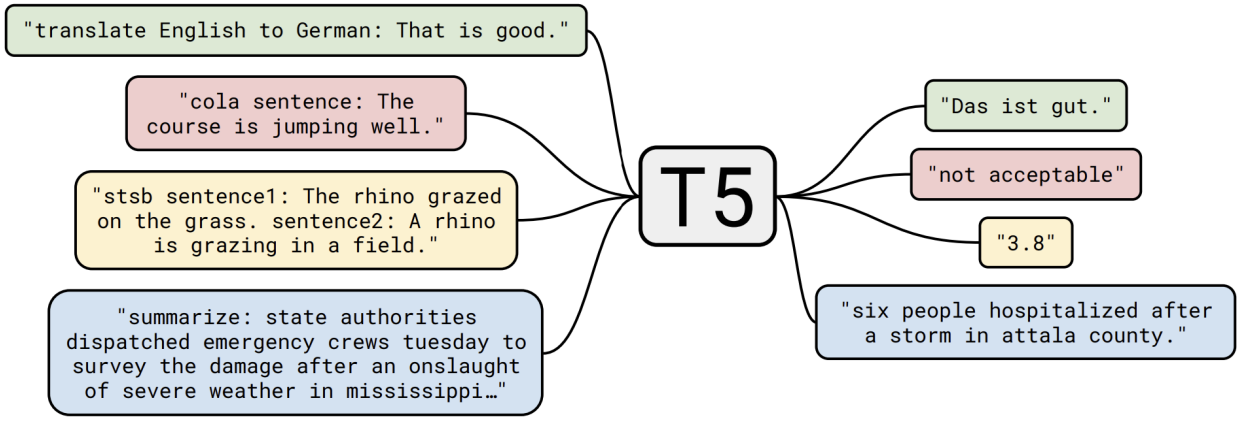
To put it in a nutshell, BART is set up as follows: using sequence-to-sequence transformer architecture with several encoder-decoder layers, masking random tokens, deleting random tokens, infilling text, permutate sentences, rotate document, training the model with large datasets, and fine-turning the pre-trained model for specific tasks. For the algorithm, the encoder first takes input as a sequence of tokens and maps them to a set of hidden representations for each token. After that, the decoder takes those representations as input and generates a sequence of output tokens. Assume we are using BART for an articles summarization task. First, the bidirectional encoder creates a fixed-length vector representation for each input sequence. Next, the autoregressive transformer decoder generates the summary sequence one token at a time given the conditions of the encoded input representation and the previously generated tokens.

4.1.6 Flan-T5

Text-to-text Transfer Transformer T5 is a transformer created by Raffel et al. (2019). The research team created a unified framework that allows the model to tackle various language problems with a text-to-text approach. The concept of text-to-text is where a model is provided with some text to cue it for the context of the particular task it should be performing. For example, if the model needs to perform abstractive summarization, then the model would need a prefix of "summary:" to perform the task. Once the context is provided, the model is then expected to provide an output text. This teacher-forcing approach of providing context is a key concept for T5. The figure below shows an example of using a prefix for various tasks.

Figure 5

Visual of T5 Text-to-Text Examples



Note. Visual depiction of T5 text-to-text. Adapted from "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," by Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J., 2019, *The Journal of Machine Learning Research, Volume 21*, p. 5485. Copyright 2020 by JMLR.org.

The benefit of using a text-to-text framework is that it allows the model to use similar hyperparameters and loss function to solve various NLP tasks. A feature that T5 has is that it is pre-trained on its own large novel dataset called Colossal Clean Crawled Corpus (C4) (Raffel et al., 2019). C4 is 745 GB of English text that implements a Common Crawl approach to scrape text from the internet. T5 benefits from using this dataset because of its sheer size, which allows for an unsupervised pre-training technique.

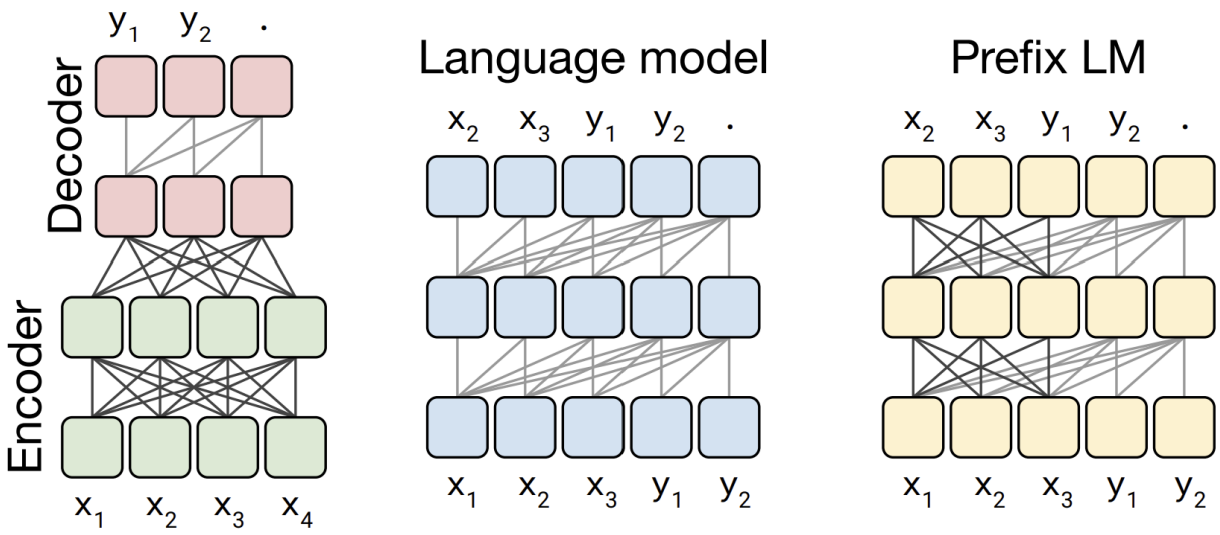
The architecture for T5 consists of a sequence of tokens that are converted into embeddings and then fed into the encoder (Raffel et al., 2020). The encoder comprises blocks with two subcomponents: a self-attention layer and a small feed-forward network. Each subcomponent's input is exposed to layer normalization. After the layer normalization process, a residual skip connection is used to merge the input and output of each subcomponent. Dropout is used at points throughout the stack, including on the attention weights, within the feed-forward

network, on the skip connection, and at the input and output. The decoder's structure is similar to that of the encoder but features an attention mechanism after each self-attention layer that focuses on the encoder's output. The decoder's self-attention mechanism makes use of an autoregressive or causal self-attention that allows for past outputs to receive attention. The final decoder output is placed into a dense layer with a softmax output that shares weights with the input embedding matrix.

Figure 6 below depicts a sequence of elements using blocks and attention visibility using lines, where different colored blocks represent different Transformer layers (Raffel et al., 2019). Dark grey lines denote the fully-visible masking, while light grey lines denote the causal masking. The end of the sequence token is represented by ".". X and y represent the input and output. The encoder-decoder architecture illustrated in the left figure utilizes full-visible masking in the encoder and encoder-decoder attention while employing causal masking in the decoder. In the middle, a language model is shown with a single Transformer layer stack, which takes the concatenation of the input and target and uses causal masking throughout. In the right figure, the prefix to a text-to-text model allows fully-visible masking over the input.

Figure 6

T5 Architecture Visual



Note. Architecture of T5 for encoder and decoder. Adapted from "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," by Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J., 2019, *The Journal of Machine Learning Research, Volume 21*, p. 5485. Copyright 2020 by JMLR.org.

T5 uses a masked language modeling approach by randomly dropping 15% of tokens from the input sequence (Raffel et al., 2020). All the tokens that are dropped are replaced with a single token. Finally, the variant is a much larger model than its base model. The T5 uses a feed-forward upwards projection size of 65,536 and a large 128-head attention model with 11 billion parameters. T5 equates to a model that is capable of SOTA performance for abstract summarization tasks.

4.2 Model Supports

4.2.1 Description of the Platform and Environment

The project is carried out on Google Colab and Jupyter Notebook. Our collected datasets are loaded in different forms: the AG dataset is downloaded as an xml file from the online source and saved on our shared Google Drive, the CoNLL2003 and CNN-Daily Mail datasets are

loaded from Hugging Face library to Google Colaboratory, our corpus dataset is scraped on a weekly basis, loaded and updated in csv form on our shared Google Drive. We use Python as the main coding language for the project and Google Colab as the main environment to write scripts. Python is picked because it has a variety of libraries that the project requires and Google Colab is chosen because it supports Python and team members can easily coordinate with each other in a cloud-shared environment. Additionally, Jupyter Notebook is used based on team members' preference.

Hugging Face is a big support for our models development. Hugging Face tokenization libraries make it easy for us to preprocess and tokenize our data on a variety of embedding models, such as BERT, BART, T5, ERNIE, and so on. After the tokenization process, thanks to the transformers library, we are able to load pre-trained models based on specific tasks, fine-tune, and evaluate the performance of the models on our own corpora. Eventually, after model evaluation, we can save and push our fine-tuned model on the Hugging Face hub for model deployment in the next phase.

4.2.2 Tools Used

Table 1

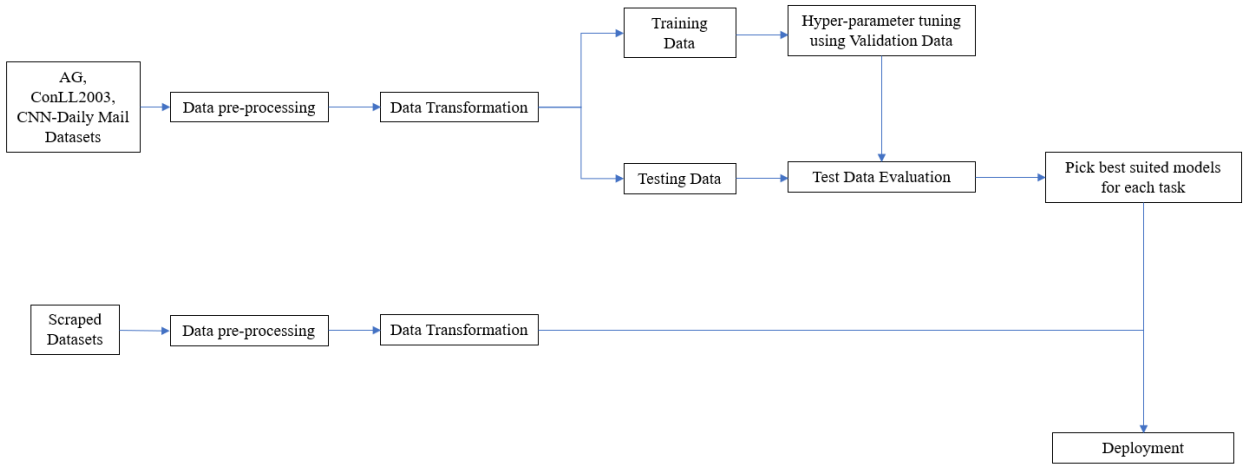
Model Support Tools

Tool	**Library**	**Method**	**Usage**
Google Colab	google.colab	drive.CreateFile	Create file using file_id locally to load data
HuggingFace	datasets	load_dataset	Return dictionary to get dataset ready to use in a dataloader for ML modelsda

	transformers	AutoTokenizer	Use with specific checkpoints for tokenization process
		DataCollatorForTokenClassification	Build batches, and apply padding to input set
Python	pandas	read_csv	Load .csv file data into Data Frames
		head, info, sample, shape	Inspect and show sample data
	matplotlib.pyplot	figure, show, xlabel, ylabel	Plot visualization on datasets statistics
	sklearn.model_selection	train_test_split	Split data into training and testing set
	seaborn	displot	Density plot

Figure 7

Data Flow Diagram



4.3 Model Comparison and Justification

4.3.1 Text Classification Task

4.3.1.1 BERT

A pre-trained BERT model makes it possible to perform text classification tasks by fine-tuning the model by adding an additional classifier layer. This approach is done in Zheng & Yang (2019) by proposing BERT-CNN model based on $BERT_{BASE}$ model, which has 12 transformer layers with 12 attention heads in each layer, hidden size of 768, and total parameters of 110M (Devlin et al., 2018). According to the authors, by applying CNN to the last layer of BERT with a local CNN encoder, the paper has achieved better performance on $BERT_{BASE}$ on multiple datasets on text classification tasks. BERT has some limitations when it comes to text classification tasks. Since its max length input is 512, there requires long sequence handlings for input with length longer than 512 tokens. Handling long sequences can result in computational expense, and catastrophic forgetting problems. For long input, the model will likely have to deal with gradient vanishing, and losing important information in the previous parts of the input. In Sun et al. (2019), the authors have investigated approaches to further fine-tune the BERT model for text classification tasks. They concluded that the top layer of BERT has a strong impact on text classification, and catastrophic forgetting problems can be addressed by having an appropriate layer-wise decreasing learning rate. Even though there are some potential difficulties in applying the BERT pre-trained model for text classification tasks, it is still considered a strong approach for this specific NLP task. By understanding the model architecture and applying research approaches, it is possible to leverage advantages of BERT and minimize disadvantages of this pre-trained model.

4.3.1.2 RoBERTa

RoBERTa pretrained model can be fine-tuned for different NLP tasks. For this project, it is used for the news and articles classification task, or also understood as text classification task. In the project of Liu et al. (2019), RoBERTa achieved state-of-the-art performance in several NLP tasks, such as GLUE, RACE, and SQuAD. The purpose of GLUE is to evaluate the natural language understanding system, the aim of RACE is to test the ability of the model on answering multiple-choices questions, and the target of SQuAD is to answer a question in complete sentences (Liu et al., 2019).

RoBERTa has multiple strengths in the articles classification task:

- It uses a bidirectional attention mechanism that allows it to consider the input text in both directions: right to left and left to right. This is suitable for text classification tasks that require comprehensive understanding like our project.
- The model uses dynamic instead of static masking, which prevents it from memorizing specific patterns in the data.
- The transformer is highly scalable and capable of classifying a large number of articles at a time, which aligns with the target of our project: we expect that the users input a variety of articles and the application returns all the articles' categories at once.
- We can fine-tune the pretrained model to fit the project's needs with huge improvements on the accuracy score.
- RoBERTa understands many languages if it is trained on these languages. Therefore, if we want to extend our project to include foreign language articles, it is flexible enough (Liu et al., 2019)

On the other hand, RoBERTa also has some limitations:

- As mentioned in section 4.1, RoBERTa should be trained with large datasets and training the model with a small dataset can lead to overfitting. To avoid overfitting, hyperparameters and regularization techniques need to be tuned carefully.
- As the model size is large (should pre-train with big datasets), RoBERTa can be expensive in terms of computational resources such as powerful hardware and large amounts of memory.
- Similar to other transformers, it is difficult to explain how the algorithm works in the black box to give the predictions. This might be a disadvantage for the projects that require transparency for decision making purposes (Liu et al., 2019).

4.3.1.3 ERNIE

The ERNIE architecture for the task of text classification is a novel language model that has proven to to perform well for NLP tasks including text classification. It uses transformer based architecture along with knowledge from multiple sources and integrates them to provide enhanced representation of the input. This makes it much more appealing than other transformer based models that require larger corpus to fine-tune the models. ERNIE is specifically designed to handle long texts, such as news articles, and is being used extensively in text classification tasks, including news article classification.

Zhaoye et al. (2021), implemented a hybrid text classification model by combining CNN (Convolutional Neural Network) and BILSTM (Bidirectional Long Short Term Memory) which utilized ERNIE in the embedded layer. The ERNIE model was used to pretrain the input and

annotate the corpus. The knowledge encoder in the ERNIE model fuses the knowledge information and the semantics of the token which eliminates the need to train the word vector.

Similarly, Li et al. (2021), also implemented ERNIE alongside RNN to classify text in the Chinese language. Instead of using the conventional method of generating word vectors, the authors employed ERNIE to acquire the full semantic representation of a semantic unit. Moreover, they integrated an RNN model with the ERNIE model, which enhances the processing of sequence information, particularly when the input preceding the current one is linked to the upcoming input, as required in text classification tasks.

ERNIE is powerful because it is designed to capture the meaning of the text and is able to understand the context of the text, which are very important for the task of text classification. Also, it can be trained on multiple languages, making it more effective in case the news articles are multilingual. This model also has certain limitations such as the computational cost. ERNIE requires significant computational resources to train and run. Since ERNIE relies on external knowledge sources, it can limit the domain specific knowledge for certain domains which have limited external sources to transfer the learnings from. Also, ERNIE is a black box, which makes it difficult to understand the underlying process and to interpret its predictions. This makes the model less transparent.

4.3.1.4 Text Classification Model Comparison

RoBERTa and BERT are similar in architecture since they are both pre-trained encoding only models. However RoBERTa is an improved pre-trained model of BERT with modified parameters, significantly larger trained mini batches and learning rates, and no next-sentence pre-training objective. In specific, BERT was trained on 3.3B words on English Wikipedia and BooksCorpus. RoBERTa was also trained on these sources, with the addition of other sources.

RoBERTa$_{BASE}$ has 125M parameters, which is 15M more than BERT$_{BASE}$, and RoBERTA$_{LARGE}$ has 355M parameters, which is more than BERT$_{BASE}$ and BERT$_{LARGE}$ models. ERNIE is designed to incorporate domain-specific knowledge into the language model. For example, ERNIE can be pre-trained on medical texts or legal documents to improve its performance on tasks related to those domains. BERT, on the other hand, is not specifically designed to incorporate domain-specific knowledge. Also ERNIE has been pre-trained in multiple languages. Whereas, BERT and RoBERTa are pre-trained only in the English language. RoBERTa with more training data and is a larger size pre-trained model, will have the ability to perform better than BERT in multiple NLP tasks, including text classification. However, it can be more computationally expensive compared to BERT. ERNIE is highly computationally intensive when compared to BERT and RoBERTa, since it incorporates knowledge from external sources.

4.3.2 Named Entity Recognition

4.3.2.1 BERT

In Devlin et al. (2018), BERT architecture has outperformed more than 11 NLP tasks compared to prior works, and Named Entity Recognition (NER) is one of the tasks. The authors proposed two approaches to apply pre-trained BERT model for this task, that are fine-tuning and feature-based. According to the authors, the fine-tuning approach for BERT$_{LARGE}$ achieves the highest performance on the CoNLL-2003 dataset. BERT$_{LARGE}$ has 24 transformer layers, 16 attention heads in each layer, with hidden size of 1024 making the total number of parameters is 340M. Zhao et al. (2021) have also applied pre-trained BERT to fine-tune on their materials NER task on biomedical related data. BERT was not trained on scientific biomedical data, but reached an accuracy of 82.9%. The accuracy score is achieved after fine-tuning with learning rate of $5e^{-5}$, warm-up proportion as 0.1, maximum token length of 128, and Adam optimizer with

batch size of 16. BERT's state-of-the-art performance on NER tasks is possible thanks to the ability to learn contextualized word embeddings during the pre-training process. This enables the model to understand surrounding context, and is helpful with recognizing entities based on input content. However, there are some limitations in the BERT pre-trained model. Even though BERT is trained on a huge amount of data, the way that it learns is through a fixed vocabulary. Therefore, if there is a named entity that is not present in the training process, it can reduce the performance during fine-tuning. Overall, BERT is considered a good pre-trained model for NER tasks and has achieved state-of-the-art performance.

4.3.2.3 ERNIE

ERNIE is specifically designed to integrate external knowledge into its pre-trained representations, and the model has demonstrated competitive or state-of-the-art performance on various natural language processing benchmarks. ERNIE models can be fine-tuned on specific downstream tasks. This adaptability allows users to customize the model for specific applications. The common use cases for ERNIE includes text classification, question answering, machine translation, NER, and so forth. We are proposing ERNIE 2.0 for the NER task whose new constructed unsupervised language processing tasks outperform BERT and XLNet (Sun et al, 2020).

Sun et al. (2020) indicate that ERNIE has three different kind of pre-trained tasks:

- Work-aware pre-training tasks: use phrase-masking, capitalize words with specific semantic information, and predicts the tokens if they are present in another segment of the original text
- Structure-aware pre-training tasks: learn the relationships among all sentences and learn sentence distance.

- Semantic-aware pre-training tasks: predict the semantic among sentences and acquire knowledge about the relevance of short texts in information retrieval.

In a project, ERNIE was implemented on nine classical Chinese NLP tasks, including MSRA-NER where MSRA is a simplified Chinese dataset for NER provided by Microsoft Research Asia, and ERNIE 2.0 made further improvements than ERNIE 1.0 and BERT-base on all nine tasks (Sun et al, 2020). The effectiveness of ERNIE on the Chinese-language dataset suggests potential outstanding performance with English-language datasets.

4.3.2.3 ELECTRA

ELECTRA is a SoTA model that has performed extremely well in a number of NLP tasks recently. It is a pre-training method for natural language processing tasks that involves training a generator model to replace certain words in a sentence with random replacements, while a discriminator model is trained to distinguish between the original sentence and the modified sentence. This process helps the generator to learn better representations of the language.

While Electra was not originally designed specifically for named entity recognition (NER), it has been used as a pre-training method for various NLP tasks, including NER. In fact, several recent studies have shown that Electra outperforms other pre-training methods on NER tasks. This makes it a novel approach for the project since not much work has been done on NER using this model.

To use Electra for NER, we would first need to fine-tune the Electra model on a NER dataset, such as CoNLL-2003 or OntoNotes. The model will be then trained to anticipate the identified entities in the text depending on the input sentence during the fine-tuning process. The Electra model can be used to perform NER on fresh text after some fine-tuning. To accomplish this, we would first tokenize the text into sentences, and then each sentence would be tokenized

into words. After that, we would run each sentence through the Electra model to get a forecast for each word. The predictions would list the terms that are named entities together with the category of named entity they belong to (e.g., person, group, or place). To acquire the final NER findings for the entire text, we would then combine the predictions for each sentence.

Reasons for choosing ELECTRA for NER is that it is a more effective pre-training method for NER than other pre-training methods because it uses a discriminator model to distinguish between original and modified sentences. This improves the generator's ability to learn more accurate and useful representations. Also, it can be easily pre-trained on domain specific datasets. ELECTRA has also been shown to be robust to noisy text. Major disadvantages of this model include high computational expense, requirements of specialized domain dataset for improved NER performance and limited interpretability on how the predictions were arrived upon.

4.3.2.4 Named Entity Recognition Model Comparison

BERT is pre-trained using a Masked Language Model objective, which gives the model the ability to predict randomly masked tokens by looking at the surrounding context. With this pre-training process, BERT can easily be fine-tuned for NER tasks, where it has to predict the entity label for the target token in the input sequence. On the other hand, ERNIE 2.0 uses phrase-masking and entity-level masking and introduces task-specific schemas during pre-training to guide the model in understanding specific relationships and patterns relevant to downstream tasks. BERT is relatively larger than ELECTRA. BERT's two variant model sizes are 110M and 340M parameters, while ELECTRA has two versions, with the small one having only 14M parameters, and the base one having the same size as BERT's. In the meantime, ERNIE 2.0 base has 110 parameters. BERT, ELECTRA, and ERNIE achieve SOTA results on

NLP tasks. While ERNIE is not popular for NER tasks on English datasets, ELECTRA outperforms BERT in certain NER tasks such as recognizing named entities in noisy text or identifying rare entities. This is due to the fact that the ELECTRA model uses a discrimination system to differentiate between original token and the synthetic ones, which is better at modeling the distribution of rare and out-of-vocabulary words from the noisy text. Last but not least, ERNIE has been pre-trained in multiple languages whereas BERT and ELECTRA are pre-trained only in the English language.

4.3.3 Abstractive Summarization

4.3.3.1 Flan-T5 Abstract Summarization

The Flan-T5 transformer is a variant of the T5 model, which is a SOTA model that has achieved excellent performance on a variety of NLP tasks, including abstract summarization (Raffel et al., 2019). T5 has been trained on a massive 745 GB pre-training corpus, which includes a diverse range of text sources. Due to the size of the training corpus, the model has been exposed to various texts that allow for improved generalization for creating abstract summaries. T5 uses an encoder-decoder architecture, which is particularly well-suited for abstract summarization. The encoder component of the model can understand the input text and extract important features, while the decoder component can generate a concise summary that captures the most salient information. T5 can be fine-tuned for a specific summarization task like news summarization using a relatively small amount of training data, which equates to a model that is very adaptable. T5's multi-task learning approach allows for it to be implemented for many NLP Tasks such as text classification, translation, and question answering. This allows the model to leverage knowledge from these related tasks to improve its performance on abstract

summarization. Overall, T5 is a very capable model that offers SOTA performance for performing abstract summarization tasks.

4.3.3.2 BART Abstract Summarization

BART was designed for NLP generation tasks, including abstract summarization. BART is pre-trained using both auto-regressive and denoising objectives, which allows it to generate highly accurate summaries (Lewis et al., 2020). The denoising objective requires the model to reconstruct corrupted input text, which helps it to learn robust representations that can handle noisy input. Like T5, BART uses an encoder-decoder architecture that is well-suited for abstract summarization tasks. The encoder component of the model can understand the input text and extract important features, while the decoder component can generate a concise summary that captures the most important information. For fine-tuning, BART can be fine-tuned for a specific summarization task using a relatively small amount of training data. BART has demonstrated its ability to perform well on both short and long input sequences because of its ability to encode an input of 1024, which is important for abstract summarization tasks where the input text can vary in length. Overall, BART is a powerful transformer for performing abstract summarization.

4.3.3.3 Abstract Summarization Model Comparison

T5 and BART transformers offer SOTA performance for performing abstract summarization, but there are differences between the models. T5 is pre-trained using a range of NLP understanding and generation objectives, while BART is pre-trained using both auto-regressive and denoising objectives. Both models use an encoder-decoder architecture that is well-suited for summarization tasks. However, the specifics of the architecture differ between the two models. T5 uses a self-attention mechanism in its encoder and decoder, while BART uses a combination of self-attention and cross-attention (Lewis et al., 2020). Performing fine-tuning

differs for both models due to the size of training data required for each model. T5 has demonstrated the ability to achieve efficient performance even with relatively small amounts of training data, while BART may require more training data to achieve equivalent performance. The computational requirement differs, as T5 is a larger model, which means that it may require more computational resources to train and deploy (Raffel et al. 2019) .

In the paper by Qi et al. (2020), the team demonstrated that both models performed well on abstract summarization based on the ROGUE metric for the CNN-Daily Mail dataset. BART performed with a slight edge in its capability. The figure below shows the performance.

Figure 8

Performance of T5 and BART on CNN-Daily Mail Dataset

Dataset	Method	Corpus	R-1	R-2	R-L
CNN/DailyMail	T5 (Raffel et al., 2019)	750GB	43.52	**21.55**	40.69
	PEGASUSLARGE (C4) (Zhang et al., 2019)	750GB	43.90	21.20	40.76
	PEGASUSLARGE (HugeNews) (Zhang et al., 2019)	3800GB	44.17	21.47	41.11
	BART (Lewis et al., 2019)	160GB	44.16	21.28	40.90
	ProphetNet	160GB	**44.20**	21.17	**41.30**

Note. Abstract summarization results for T5 and BART on CNN-Daily Mail dataset. Adapted from ProphetNet: Predicting Future N-gram for Sequence-to-SequencePre-training, by Qi, W., Yan, Y., Gong, Y., Liu, D., Duan, N., Chen, J., Zhang, R., & Zhou, M, 2020, *Empirical Methods in Natural Language Processing*. p. 2407. Copyright 2020 by Association for Computational Linguistics.

In summary, both T5 and BART are very capable models that can achieve SOTA performance on summarization tasks.

4.4 Model Evaluation Methods

4.4.1 Text Classification Metrics

The following metrics will be used for all models that are implemented for the text classification task (BERT, RoBERTa, ERNIE). Text classification for news articles is a method for classifying articles based on their content. Since the AG News dataset implemented will be balanced, the primary metric used will be *accuracy.*

4.4.1.1 Accuracy

Accuracy is a widely used metric to evaluate the performance of text classification models in NLP tasks. Accuracy is the ratio of correctly classified documents to the total number of documents in the dataset (Kelleher et al., 2015, p. 465). Accuracy can be expressed mathematically, as depicted in the figure below.

Figure 9

Metric Accuracy Equation

$$classification\ accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)}$$

Note. Mathematical equation for accuracy. Adapted from Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms (p. 465), by Kelleher, J. D., Namee, B. M., & D'Arcy, A, 2015, Amsterdam University Press. Copyright 2015 by the MIT Press.

For example, if a text classification model correctly classifies 85 out of 100 documents, its accuracy would be 85%. Accuracy is a valuable metric when the classes in the dataset are balanced, meaning they have roughly equal numbers of targets. When the classes are imbalanced, accuracy may not be the best metric to use as it can be misleading. The F1 score

may be more appropriate in such cases because it can help identify which classes are not performing well.

4.4.1.2 F1

F1 score will also be used to evaluate the text classification models. Even in a balanced dataset, it can still be beneficial to use the F1 score, as it can provide more granular information about the model's performance. In news text classification, the goal is to classify news articles into one or more predefined categories, such as politics, sports, entertainment, technology, etc. The F1 score is particularly useful in this context because it can provide insights into how well the classifier is performing for each category because it provides the harmonic mean of recall and precision, which can provide more information into what the model is doing compared to other metrics (Géron, 2019, p. 141). The figure below shows the equation for calculating F1.

Figure 10

Metric F1 Equation

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

Note. Mathematical equation for F1 score. Adapted from Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 2nd ed (p. 141), by Géron, A., 2019, O'Reilly Media. Copyright 2019 by O'Reilly Media.

4.4.1.3 Precision

For NLP news text classification, precision measures the proportion of correctly predicted positive instances, which would be the number of news articles that belong to a

specific category among all instances that are predicted as positive (Géron, 2019, p. 139). The equation for precision can be seen in the figure below.

Figure 11

Metric Precision Equation

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

Note. Mathematical equation for precision. Adapted from Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.)(p. 139), by Géron, A., 2019, O'Reilly Media. Copyright 2019 by O'Reilly Media.

Precision is used in news text classification because it is a metric to judge how well the model delivers relevant and reliable information to readers. For example, in a news classification task where the goal is to identify articles related to sports, high precision would ensure that all the articles that are classified as sports news are indeed about sports without falsely labeling other types of news as sports. Precision can help improve the user experience of a news app by providing more personalized and relevant content to readers. Precision has the ability to identify articles related to specific topics or categories, and a news app can make use of it to recommend articles to readers that are more likely to be of interest to them.

4.4.1.4 Recall

Recall is a crucial metric for text classification because it measures the proportion of correctly predicted positive instances, meaning news articles that belong to a specific category among all instances that actually belong to that category in the dataset (Géron, 2019, p. 139). The equation for recall can be seen in the figure below.

Figure 12

Metric Recall Equation

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

Note. Mathematical equation for recall. Adapted from Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.)(p. 139), by Géron, A., 2019, O'Reilly Media. Copyright 2019 by O'Reilly Media.

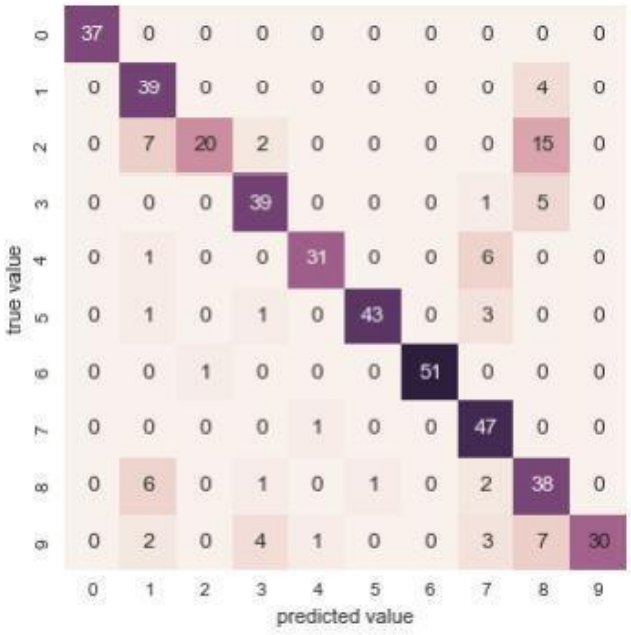
Recall is essential in news text classification because it ensures that all the relevant articles are identified, even if some irrelevant articles are also included. For example, in a news classification task where the goal is to identify articles related to politics, a high recall would ensure that all articles that are related to politics are correctly identified, even if some articles that are not related to politics are also included.

4.4.1.5 Confusion Matrix

The confusion matrix provides an intuitive visual for the user to assess the model's performance. The concept involves tallying the number of cases in which the model incorrectly classified its predictions (Géron, 2019, p. 138). When examining the confusion matrix, the rows correspond to the actual values, while the columns represent the predicted values (Géron, 2019, p. 139).

Figure 13

Example of Confusion Matrix



Note. Visual depiction of a confusion matrix. Adapted from Python Data Science Handbook: Essential Tools for Working with Data (p. 357), by Vanderplas, J., & VanderPlas, J., 2016, Van Duuren Media. Copyright 2016 by O'Reilly Media.

The purple line in the horizontal direction in the depicted figure above represents the correct predictions made from the model, while the values outside the purple line represent the incorrect ones. This visual representation helps in promptly identifying the values or classes which the model has trouble predicting correctly. For instance, in this case, if we focus on the true value of two, we can observe that the model faces difficulty in making accurate predictions for this value. It tends to incorrectly predict the value of 8, with as many as 15 instances.

4.4.2 NER Metrics

The following metrics will be used for all models that are implemented for the NER task (BERT, Electra). NER for news articles is a task of NLP that involves identifying and classifying named entities in the article. The primary metric used will be the F1 score.

4.4.2.1 F1

F1 is the most commonly used metric for NER because it considers both precision and recall, which are measures for evaluating the performance of a NER model (Géron, 2019, p. 141). Precision measures the model's accuracy in identifying named entities, while Recall measures the completeness of the model in identifying named entities. Both precision and recall are important because a NER model should accurately identify as many entities as possible while minimizing the number of false positives. The equation for F1 can be seen above in Figure 86 .

4.4.2.1 Precision

Precision is often used to evaluate the performance of a NER model. It measures the proportion of identified named entities that are correct among all the named entities predicted by the model (Géron, 2019, p. 139). The importance of precision in NER is that it helps to understand how reliable the model is in identifying named entities. If the precision is high, it means that the model is accurately identifying the named entities. Additionally, precision allows for a NER model to identify where false positives or incorrectly named entities identified by the model can potentially be very harmful. For instance, consider a news app that processes news articles to extract information about events and their locations. If the NER model incorrectly identifies a person's name as a location, this error might propagate to downstream applications, leading to incorrect information. The equation for Precision can be viewed above in Figure 87 .

4.4.2.1 Recall

The Recall metric can be used for evaluating the performance of a NER model. Recall measures the completeness of the model in identifying named entities that belong to an entity (Géron, 2019, p. 139). In the context of NER, this means the percentage of all named entities in the text that were correctly identified by the model. The importance of Recall for NER is due to

the fact that it is crucial to identify all the named entities in a text, especially in applications such as information extraction or information retrieval, where the aim is to extract relevant information from the text. If a NER model misses some named entities, it could lead to incomplete or inaccurate information extraction. Therefore, a high Recall score is desirable for NER models, as it indicates that the model is able to correctly identify a high proportion of named entities in the new article. The mathematical equation for recall is shown in Figure 88.

4.4.3 Abstract Summarization Metrics

The following metrics will be used for all models that are implemented for the abstract summarization task (T5-11B, BART). Abstract summarization for news articles is the task of generating a concise and informative summary of a news article. The primary metric used will be ROGUE.

4.4.3.1 BLEU

The Bilingual Evaluation Understudy (BLEU) is a metric that was created by Papineni et al. (2002) for evaluating the quality of NLP text summaries. The abstractive summarization task involves generating a shorter summary of a longer text by comprehending its meaning and producing a summary. BLEU measures how well a model-generated summary matches a human-generated summary. BLEU first computes the precision of the predicted summary with respect to the reference summary for n-grams of different lengths, as well as the brevity penalty, which penalizes translations that are shorter than the reference summary (Papineni et al., 2002). The precision of the n-grams is computed by counting the number of times they occur in the predicted summary and comparing it to the number of times they occur in the reference summary. The overall BLEU score is then computed as the geometric mean of the individual

n-gram precisions, weighted by their respective lengths, and multiplied by the brevity penalty. The figure below shows the equation for BLEU.

Figure 14

Metric BLEU Equation

$$\mathrm{BP} = \begin{cases} 1 & \text{if } c > r \\ e^{(1-r/c)} & \text{if } c \leq r \end{cases}.$$

Then,

$$\mathrm{BLEU} = \mathrm{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

Note. BLEU mathematical equation. Adapted from BLEU: a method for automatic evaluation of machine translation, by Papineni, K., Roukos, S., Ward, T. J., & Zhu, W., 2002, *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. p.311-315. Copyright 2002 by Association for Computational Linguistics.

The metric then assigns a score between 0 and 1, where a score of 1 means the predicted summary is an exact match to the human-generated summary. BLEU provides a uniform way to evaluate summaries generated by different models, and it also helps to ensure that the generated summaries are accurate and understandable. BLEU is an essential metric for applications like news article summarization, where readers rely on the summary to understand the key points of the article.

4.4.3.2 ROGUE-1/2/L

ROUGE is currently the most popular metric for evaluating the performance of abstractive summarization models in NLP. It has become the standard for summarization evaluation, and it is widely used in research papers, benchmark datasets, and evaluation

campaigns. It was first introduced by Lin (2004) and it evaluates the similarity between the generated summary and the reference summary by computing the overlap between the n-grams, or contiguous sequences of n words in the two summaries. ROUGE can be used with different values of n, depending on the specific task and the length of the summaries. The figure below shows the equation for computing different n values of ROUGE.

Figure 15

Metric ROUGE-N Equation

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \quad (1)$$

Note. Mathematical depiction for ROGUE-N grams. Adapted from ROUGE: A Package for Automatic Evaluation of Summaries, by Lin, C, 2004, *Annual Meeting of the Association for Computational Linguistics*. p.74-81. Copyright 2004 by Association for Computational Linguistics.

ROUGE-1, ROUGE-2, and ROUGE-L are variants of the ROUGE metric that differ in the size of the n-grams they consider and the way they compute the overlap between the generated and reference summaries. ROUGE-1 (unigram ROUGE) evaluates the overlap of single words between the generated and reference summaries (Lin 2004). It measures the precision and recall of the unigrams in the generated summary that match those in the reference summary. ROUGE-1 is useful for evaluating summaries that are focused on the most important keywords and concepts of the source document. ROUGE-2 is similar to ROUGE-1, but instead of considering unigrams it considers bigrams. ROUGE-2 also measures precision and recall for

the purpose of rendering a score. ROUGE-2 evaluates summaries that capture more complex phrases and sentence structures. ROUGE-L, also known as the longest common subsequence (LCS) based ROUGE, computes the LCS between the generated and reference summaries and measures the precision and recall of this LCS. It considers the length of the LCS, which is the longest contiguous sequence of words that appear in both summaries and the positions of the LCS in the reference and generated summaries. ROUGE-L is useful for evaluating summaries that are longer and more fluent than the source document. The equation for computing ROGUE-L can be seen in the figure below.

Figure 16

Metric ROUGE-L Equation

$$R_{lcs} = \frac{LCS(X,Y)}{m} \quad (2)$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \quad (3)$$

$$F_{lcs} = \frac{(1+\beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \quad (4)$$

Note. Mathematical equation depiction of ROUGE-L. Adapted from ROUGE: A Package for Automatic Evaluation of Summaries, by Lin, C, 2004, *Annual Meeting of the Association for Computational Linguistics*. p.74-81. Copyright 2004 by Association for Computational Linguistics.

ROUGE-1, ROUGE-2, and ROUGE-L will all be used together to provide a more comprehensive evaluation of the quality of the generated abstract news summaries.

4.5 Model Validation and Evaluation Results

4.5.1 Text Classification

To understand the impact of fine-tuning pretrained models for the text classification task using the preprocessed and transformed AG's News dataset, we decided to first evaluate the validation set on the baseline pretrained models.

Table 2

Baseline Model Evaluation Before Fine Tuning

Model	Accuracy	F1	Precision	Recall
BERT (base-uncased)	0.2220	0.1332	0.1484	0.2220
RoBERTa (base)	0.1991	0.0670	0.0634	0.1991
ERNIE (base-en-2.0)	0.1906	0.0798	0.1710	0.1906

As shown in Table 2 above, the performance of the base model is very poor on the validation set for the task. The highest accuracy score is only 0.22. To solve this problem, we fine-tuned all three models and chose the same configuration for all models to appropriately compare and evaluate their performance.

Table 3

Model Evaluation After Fine-tuning

Model	Accuracy	Precision	F1	Recall

BERT (base-uncased)	0.8445	0.8415	0.8436	0.8445
RoBERTa (base)	0.7969	0.7925	0.7953	0.7969
ERNIE (base-en-2.0)	0.8534	0.8507	0.8522	0.8534

Using purely the pretrained models on a specific task is not a good approach. Fine-tuning the pretrained models for the task on designated and correctly transformed dataset can result in a much better performance.

From Table 3, the accuracy reflects the ability of the model to classify categories correctly. In this case, ERNIE has the highest accuracy of 0.8534, slightly higher than BERT. Recall score is sharing the same value as accuracy. With the value of 0.8522, ERNIE also has the highest precision. ERNIE has the highest F1 value of 0.8507. Precision, Recall, and F1 scores are calculated using the weighted average among the category classes. To understand better how each model predicts for each class, we used confusion matrix as one of the selected evaluation model methods.

Figure 17

BERT(base-uncased) Confusion Matrix Evaluation

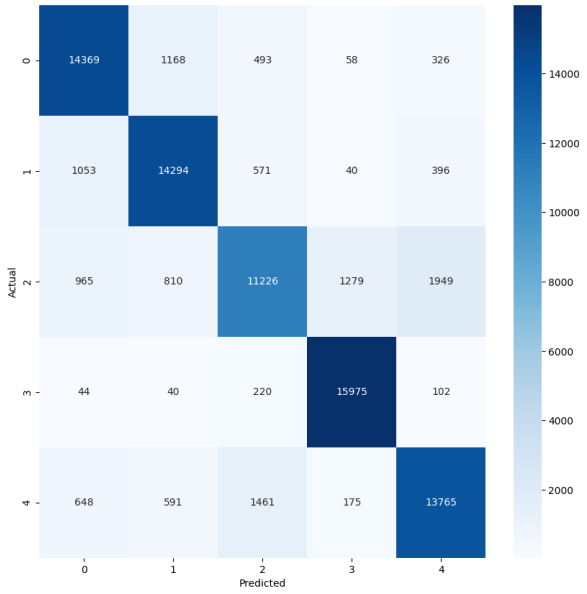


Figure 18

RoBERTa(base) Confusion Matrix Evaluation

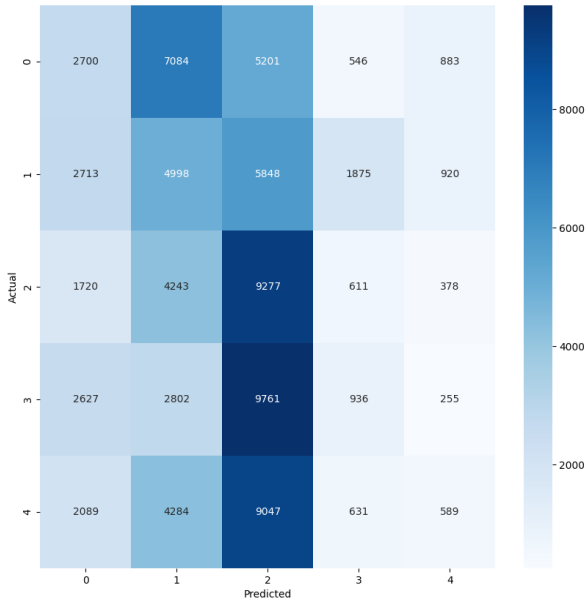
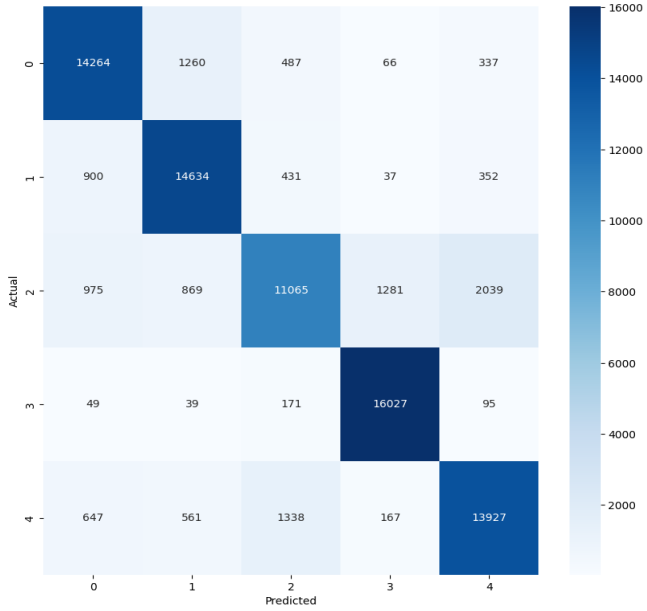


Figure 19

ERNIE(base-en-2.0) Confusion Matrix Evaluation



BERT and ERNIE are performing well on predicting each category, and yield similar values on the confusion matrices. RoBERTa, on the other hand, does not perform well in predicting news categories.

Figure 20

Sample Fine-tuned ERNIE and BERT Output - Text Classification

```
Input text:
 China: Smuggling Still Threat to Economy (AP) AP — Smuggling of goods ranging from oil to frozen food
 is costing China's treasury billions of dollars each year,
 threatening the national economy and
 possibly even communist rule despite a marathon crackdown, officials said Tuesday.
--------------------------------------------------------------------------------
Actual category:
 Business
--------------------------------------------------------------------------------
Predicted category:
 Business
```

Figure 21

Sample Fine-tuned RoBERTa Output - Text Classification

```
Input text:
 China: Smuggling Still Threat to Economy (AP) AP — Smuggling of goods ranging from oil to frozen food
 is costing China's treasury billions of dollars each year,
 threatening the national economy and
 possibly even communist rule despite a marathon crackdown, officials said Tuesday.
--------------------------------------------------------------------------------
Actual category:
 Business
--------------------------------------------------------------------------------
Predicted category:
 Sci/Tech
```

Figure 20 and 21 show sample inputs from the validation set with the labeled target and the prediction from the three models. In the sample input, ERNIE and BERT are capable of predicting the right category, while RoBERTa is not.

After evaluating and comparing the performance of the fine-tuned models on this task, we decided to proceed with the model with highest performance, which is BERT, for optimizing the hyper-parameter for a potential better performance.

4.5.2 Named Entity Recognition

Table 4 presents the model performance comparison for NER task using accuracy score, precision score, F1 score, and recall score. The models include BERT, ELECTRA, and ERNIE baseline and fine-tuned.

For baseline evaluation, we use the validation datasets in which we split 10% from the transformed data. The validation dataset of BERT has 1873 rows and 4 columns while that of ELECTRA and ERNIE both have 1871 rows and 4 columns (figure 22, 23, and 24). The baseline pre-trained models are underperforming for the task with significantly low index values; for example, BERT baseline returns only 6.12% accuracy and 0.92% precision, ERNIE baseline underperforms with 2.93% accuracy score and 1.64% precision, and ELECTRA baseline is slightly better with 10.23% accuracy and 2.6% precision. These outcomes are understandable as BERT, ERNIE, and ELECTRA cannot really support downstream tasks without fine tuning.

Figure 22

Validation transformed data of BERT for NER task

```
validation: Dataset({
    features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
    num_rows: 1873
})
```

Figure 23

Validation transformed data of ELECTRA for NER task

```
Dataset({
    features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
    num_rows: 1871
})
```

Figure 23

Validation transformed data of ERNIE for NER task

```
validation: Dataset({
    features: ['input_ids', 'token_type_ids', 'attention_mask', 'labels'],
    num_rows: 1871
})
```

For fine-tuning, we use the transformed training and validation dataset that are mentioned in section 3. Fine-tuned models display more than 98 percent accuracy rates, which are remarkably better than those of baseline models. After assessing the results, we decided to move forward with ELECTRA for deployment as it has the highest accuracy, precision, F1, and recall scores.

Table 4

Baseline and fine-tuned performances of NER-task models

Model	Accuracy	Precision	F1	Recall
BERT (baseline)	0.0612	0.0092	0.0118	0.0103

ERNIE (baseline)	0.0293	0.0098	0.0164	0.0491
ELECTRA (baseline)	0.1023	0.0262	0.0465	0.2101
BERT (fine-tuned)	0.9839	0.9325	0.9352	0.9378
ERNIE (fine-tuned)	0.9852	0.9323	0.9368	0.9413
ELECTRA (fine-tuned)	**0.9865**	**0.9451**	**0.9472**	**0.9494**

The next three figures show the confusion matrices of BERT, ERNIE, and ELECTRA fine-tuned models on the validation dataset. Notice that the total data points of ERNIE and of ELECTRA are smaller than BERT's, which derives from the differences in the tokenization process. For example, each model has different numbers of padding and special tokens, and these elements are eliminated in the calculations of the metrics. In overall, the three models work well in predicting NER labels.

Figure 24

Confusion matrix of BERT fine-tuned model on validation dataset for NER task

		y_pred								
		O	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC	B-MISC	I-MISC
y_true	O	29292	1	7	6	31	2	7	12	23
	B-PER	9	971	0	14	0	1	0	1	0
	I-PER	24	2	2364	0	25	0	6	0	4
	B-ORG	12	7	0	811	8	10	0	7	0
	I-ORG	45	0	19	4	1713	0	20	0	10
	B-LOC	2	5	0	25	1	959	1	9	0
	I-LOC	11	1	15	0	43	0	838	0	9
	B-MISC	14	5	0	22	0	10	0	406	4
	I-MISC	53	1	9	1	46	1	19	3	371

Figure 24

Confusion matrix of ERNIE fine-tuned model on validation dataset for NER task

		y_pred								
		O	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC	B-MISC	I-MISC
y_true	O	15512	531	76	7337	810	234	3033	95	577
	B-PER	567	4	0	128	17	9	225	0	38
	I-PER	1291	24	0	270	51	44	435	0	61
	B-ORG	509	0	0	156	10	2	158	0	13
	I-ORG	680	2	1	119	17	2	351	0	13
	B-LOC	438	3	0	241	19	0	295	0	12
	I-LOC	316	2	0	74	4	4	147	0	3
	B-MISC	270	0	0	78	3	1	111	0	15
	I-MISC	224	3	0	50	7	1	82	0	8

Figure 25

Confusion matrix of ELECTRA fine-tuned model on validation dataset for NER task

		y_pred								
		O	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC	B-MISC	I-MISC
y_true	O	28061	6	6	14	35	4	15	19	45
	B-PER	7	971	3	5	0	1	0	1	0
	I-PER	1	1	2161	0	11	0	1	0	1
	B-ORG	13	7	0	808	2	9	0	9	0
	I-ORG	30	1	9	1	1128	0	2	1	13
	B-LOC	6	3	0	19	0	970	3	5	2
	I-LOC	10	1	6	0	17	1	498	1	16
	B-MISC	15	1	0	13	0	5	0	436	8
	I-MISC	34	1	2	0	32	0	5	4	297

There are two ways the fine-tuned model can return outputs. First is showing the entities with prefix tags like B and I; second is not displaying the prefix tag. Figure 25 and 26 are examples of the ELECTRA model outputs.

Figure 26

ELECTRA model output for deployment with prefix tags

```
test_token_classifier("Donald Trump was the president of the United States")
```

```
[{'entity': 'B-PER',
  'score': 0.99795556,
  'index': 1,
  'word': 'donald',
  'start': 0,
  'end': 6},
 {'entity': 'I-PER',
  'score': 0.9985619,
  'index': 2,
  'word': 'trump',
  'start': 7,
  'end': 12},
 {'entity': 'B-LOC',
  'score': 0.9954181,
  'index': 8,
  'word': 'united',
  'start': 38,
  'end': 44},
 {'entity': 'I-LOC',
  'score': 0.9939116,
  'index': 9,
  'word': 'states',
  'start': 45,
  'end': 51}]
```

Figure 27

ELECTRA model output for deployment without prefix tags

```
token_classifier("Donald Trump was the president of the United States")
```

```
[{'entity_group': 'PER',
  'score': 0.9982587,
  'word': 'donald trump',
  'start': 0,
  'end': 12},
 {'entity_group': 'LOC',
  'score': 0.99466485,
  'word': 'united states',
  'start': 38,
  'end': 51}]
```

4.5.3 Text Summarization

For the abstract summarization task, it was paramount to evaluate the initial performance of the BART-base and Flan-T5-small models. To evaluate the models accurately, we adopted a multi-faceted approach. Our primary metrics were the ROUGE suite—specifically ROUGE-1, ROUGE-2, and ROUGE-L—to measure the overlap of unigrams, bigrams, and the longest common subsequence, respectively, between the generated and the reference summaries. We

complemented this with the BLEU score, traditionally used in machine translation but applicable here to assess the precision of the generated summaries against multiple references.

Our primary metric of focus was the ROUGE-L metric due to its emphasis on the longest common subsequence, which is particularly relevant for assessing the fluency and structure of the summaries produced by our models. The ROUGE-L score is the primary indicator of how well each summary captures the essence and flow of the original news articles in a coherent and cohesive manner.

The benchmark for this evaluation was the CNN/DailyMail validation set, consisting of 13,060 instances. We ensured that the dataset was preprocessed consistently for both models to facilitate a fair comparison, which included standard tokenization practices.

Table 5

Baseline Model Evaluation of Abstract Summarization Before Fine Tuning on Validation Data

Model	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
BART-base (baseline)	0.2968	0.1320	0.1854	9.81
Flan-T5-small (baseline)	0.3241	0.1378	0.2368	10.87

From Table 5, the initial performance of our baseline models can be observed, and the BART-base achieved a Rouge-L score of 0.1854. The performance suggests that there's room for enhancement in terms of capturing the sequence structure of the summaries. In essence, the model might sometimes struggle with maintaining a coherent flow or could omit essential details. The Flan-T5-small baseline exhibited a higher Rouge-L score, indicating that even in its baseline configuration, it possesses a better capability to generate summaries that are both content-rich and coherent.

Table 6

Model Evaluation of Abstract Summarization After Fine Tuning on Validation Data

Model	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
BART-base (fine-tuned)	0.2503	0.1253	0.2087	10.22
Flan-T5-small (fine-tuned)	0.3484	0.1434	0.3280	13.07

The fine-tuning was conducted with a consistent methodology across both models, carefully controlling for hyperparameters, training epochs, and other variables to ensure any variation in performance was due to the models' inherent capabilities rather than external factors. Table 6 displays the post-fine-tuning results. The Flan-T5-small model's Rouge-L score saw a substantial improvement, reaching 0.3280 on the validation set, reinforcing its enhanced ability to create summaries that capture the essence of news articles.

Figures 28 and 29 illustrate the results of the summarization process using fine-tuned BART-base and Flan-T5-small models, respectively, applied to the validation dataset. Within the given example, the summary generated by the Flan-T5-small model is of superior quality.

Figure 28

Sample Fine-tuned BART-base Output - Abstract Summarization

```
Input text:
(CNN)Five Americans who were monitored for three weeks at an Omaha, Nebraska,
hospital after being exposed to Ebola in West Africa have been released, a
Nebraska Medicine spokesman said in an email Wednesday. One of the five had a
heart-related issue on Saturday and has been discharged but hasn't left the
area, Taylor Wilson wrote. The others have already gone home. They were exposed
to Ebola in Sierra Leone in March, but none developed the deadly virus. They are
clinicians for Partners in Health, a Boston-based aid group. They all had
contact with a colleague who was diagnosed with the disease and is being treated
at the National Institutes of Health in Bethesda, Maryland. As of Monday, that
health care worker is in fair condition. The Centers for Disease Control and
Prevention in Atlanta has said the last of 17 patients who were being monitored
are expected to be released by Thursday. More than 10,000 people have died in a
West African epidemic of Ebola that dates to December 2013, according to the
World Health Organization. Almost all the deaths have been in Guinea, Liberia
and Sierra Leone. Ebola is spread by direct contact with the bodily fluids of an
infected person.
------------------------------------------------------------
Target summary:
17 Americans were exposed to the Ebola virus while in Sierra Leone in March .
Another person was diagnosed with the disease and taken to hospital in Maryland
. National Institutes of Health says the patient is in fair condition after
weeks of treatment .
------------------------------------------------------------
Predicted summary:
Five Americans who were monitored for three weeks at an Omaha, Nebraska
hospital. One of the five had a heart-related issue on Saturday. The others have
already gone home.
```

Figure 29

Sample Fine-tuned Flan-T5-small Output - Abstract Summarization

```
Input text:
(CNN)Five Americans who were monitored for three weeks at an Omaha, Nebraska,
hospital after being exposed to Ebola in West Africa have been released, a
Nebraska Medicine spokesman said in an email Wednesday. One of the five had a
heart-related issue on Saturday and has been discharged but hasn't left the
area, Taylor Wilson wrote. The others have already gone home. They were exposed
to Ebola in Sierra Leone in March, but none developed the deadly virus. They are
clinicians for Partners in Health, a Boston-based aid group. They all had
contact with a colleague who was diagnosed with the disease and is being treated
at the National Institutes of Health in Bethesda, Maryland. As of Monday, that
health care worker is in fair condition. The Centers for Disease Control and
Prevention in Atlanta has said the last of 17 patients who were being monitored
are expected to be released by Thursday. More than 10,000 people have died in a
West African epidemic of Ebola that dates to December 2013, according to the
World Health Organization. Almost all the deaths have been in Guinea, Liberia
and Sierra Leone. Ebola is spread by direct contact with the bodily fluids of an
infected person.
------------------------------------------------------------
Target summary:
17 Americans were exposed to the Ebola virus while in Sierra Leone in March .
Another person was diagnosed with the disease and taken to hospital in Maryland
. National Institutes of Health says the patient is in fair condition after
weeks of treatment .
------------------------------------------------------------
Predicted summary:
17 Americans were exposed to Ebola in Sierra Leone in March. They were monitored
for three weeks at an Omaha, Nebraska, hospital. One of the five had a heart-
related issue and was discharged, a hospital spokesman says. None of them
developed the deadly virus, but they had contact with a colleague with Ebola.
```

After careful analysis and comparison, our decision was to deploy the Flan-T5-small fine tuned model for our application. This decision is rooted in its superior Rouge and BLEU scores, which indicate its enhanced capability to generate more accurate and coherent summaries, making it the best choice for the overarching goal of delivering high-quality news summaries to our users.

5. Data Analytics System

5.1 System Requirements Analysis

5.1.1 System Boundaries, Actors, and Use Cases

The system boundary defines the project's scope, which also establishes what is within and outside the system. Our news analysis system essentially consists of a web application that works inside this confinement. It is built around three specialized components, each of which is

tuned for the subtle interpretation and deconstruction of news narratives. The Multi-label Text Classification component, at the center of this system, divides news items into five key categories: World, Science-Technology, Business, Entertainment, and Sports with the use of cutting-edge models like BERT, RoBERTa, and ERNIE. To enhance this capacity, the Named Entity Recognition (NER) component uses the strength of ELECTRA, ERNIE, or BERT to comb through text and highlight important entities such as people, places, and institutions, among others. The text summarization component completes this suite by using the knowledge of BART or T5 to condense articles into clear, concise summaries. As a whole, these elements combine to create a potent system that makes it possible to thoroughly examine news information. This system acts as a compass for scholars and voracious readers traversing the vast expanse of daily news. The news system has two main external interfaces. First, it provides readers and scholars with an easy-to-use online interface that enables users to input news stories and quickly access informative results like classifications and summaries. Second, it links to multiple news data sources, assuring a steady stream of up-to-date information. Direct data imports, APIs, and RSS feeds are some of the techniques used to do this. These linkages to the outside world not only keep the system current but also give hints about prospective future integrations for more functionally extensive features.

Three main players play crucial roles in our news analysis system's operating environment. The User, which includes both academic readers and general readers, is the most engaging of them. They interact with the system actively by contributing articles and asking for information such as classifications, entity recognitions, and summaries. The System Administrator, a behind-the-scenes orchestrator who oversees everything from normal system maintenance to trickier system changes, comes next. Last but not least, the Data Source serves as

a reservoir, supplying the system with a steady stream of news items. This actor, despite being more passive, is essential for the system's ongoing content analysis and current relevance. Each actor improves the system's usability and user experience to some extent.

For colleges and schools, the news article classification system has enormous promise for facilitating research and fostering media literacy. It makes it easier for academics and students to go through massive amounts of news data, enabling effective domain-specific research. It may be included into journalism, media studies, or political science classes by teachers to provide students a practical understanding of the news and help them distinguish between different types of news. Beyond academia, this technique can help debating groups, current affairs organizations, and school project efforts. Students, teachers, researchers, librarians, journalism students, and content curators in educational contexts would be the system's main users. They would utilize it to promote educated, structured news consumption and study.

Businesses and the corporate world benefit from the news item classification system, which improves market information and strategic decision-making. The system effectively filters news for businesses watching market trends, offering insights unique to fields like technology or finance. It may be used by marketing teams to comprehend alterations in the behaviors of consumers throughout the world. To keep current on trends and rules affecting the workplace, HR departments may use categorized news. Additionally, corporate communication teams may use the technology to track public opinion about their business, which will help with reputation management.

5.1.2 Data analytics and Machine Learning capabilities

The system is designed to offer the following advanced data analytics and machine learning:

The system employs the ERNIE transformer model in its article classification module to automatically categorize incoming news articles. Articles are sorted into predefined categories: Sports, World, Entertainment, Business, and Sci/Tech. This transformer-based categorization helps streamline the content and makes it easier to manage and analyze. The output is displayed as a category label in a text box, providing immediate clarity on the article's main focus.

The Electra transformer model handles the Named Entity Recognition (NER) task. This module scans through the text of each article to identify and label key entities, which could range from people and organizations to locations.

For condensing lengthy articles into brief, coherent summaries without losing essential information, the system utilizes the T5 transformer model. This abstract summarization module works to distill the article's key points into a format that is both quick to read and informative. The output of this module is a text-based summary that serves as a preview or quick-read version of the full article within a text box.

The outputs from the text classification, NER, and abstract summarization models are stored in the AWS RDS database. The database will be instrumental for fine-tuning and enhancing the system's capabilities. Using Amazon Quicksight, it will allow to visualize a range of analytics, such as the frequency distribution of article types and the most mentioned entities. These insights will help better understand user behavior and provide a glimpse into trending topics and influential figures in the news. Plus, it offers the ability to gain valuable resources for ongoing system improvement and feature development by archiving these model outputs and leveraging them for analytics. This data-driven approach enables the developers to make more informed decisions, creating a more effective and reliable news analysis platform.

5.2 System Design

5.2.1 System Architecture and Infrastructure

Figure 30 shows the frontend and backend architecture of our project. On the frontend side, the users enter an article link of interest through the user interface. Once the input is entered, the system scraps the article's title and content, and the hosting server calls our three models saved as checkpoints on Hugging Face. The backend system consists of a scraping work, data preprocess function, models, storage, and visualization analytics.

Before being fed to the models, the article content is preprocessed by a function called pipeline from Hugging Face Transformers library which is capable of cleaning and transforming the data based on our models' configuration. The pipeline function can tokenize, pad, add special tokens, create an attention mask, and organize batches if providing multiple inputs. Moreover, pineline can be used for text generation, zero-shot classification, and NER tasks.

Hugging face checkpoints are utilized to fine tune deep learning models for all the tasks, and one best model of each task is picked for deployment. After the model development stage, we choose one best model for each of the main tasks: Text Classification, Named Entity Recognition, and Abstract Summarization. All models are saved as new checkpoints in Hugging Face and can be retrieved when the hosting server sends the signal. The three models run in parallel once they receive the pre-processed data and their outputs go to two places: our database storage and UI.

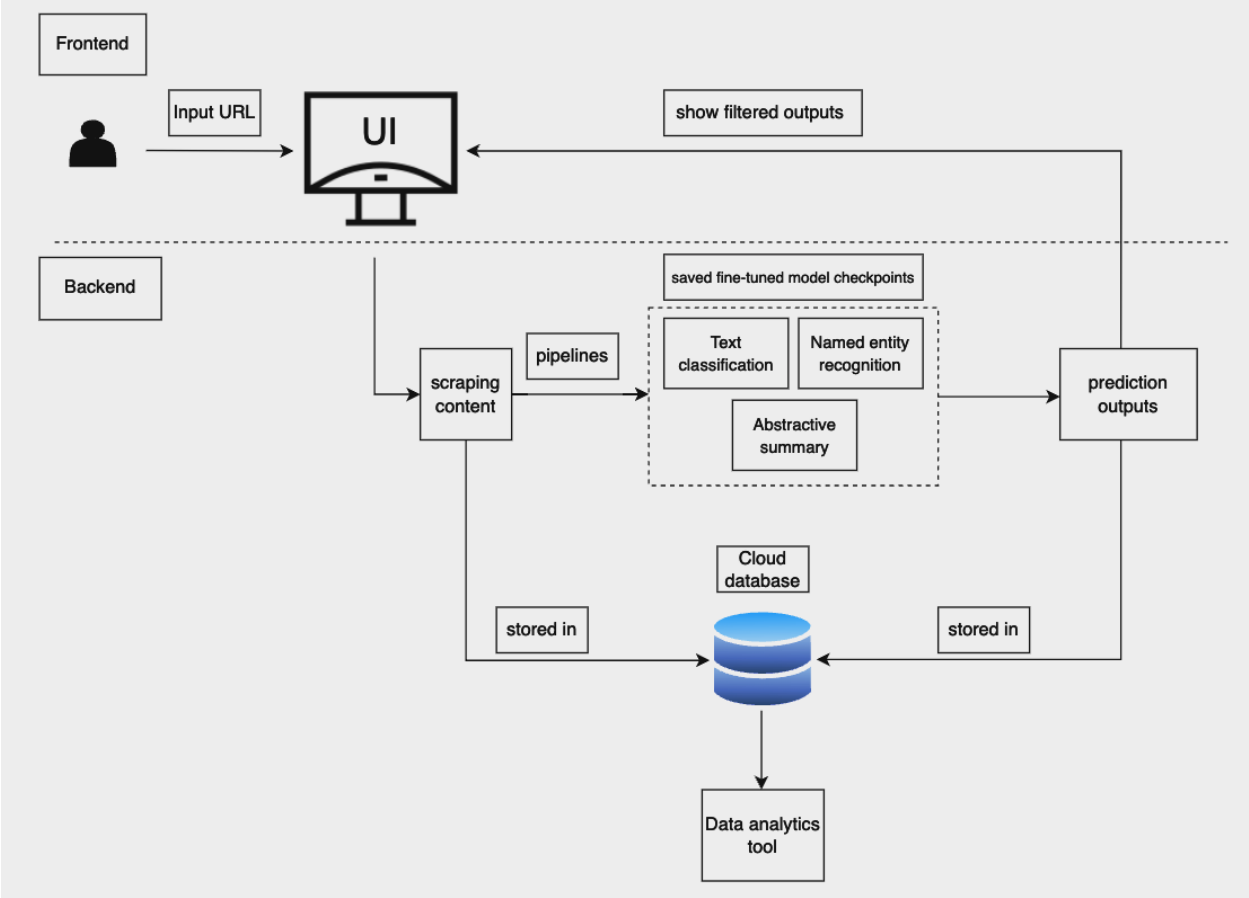
A cloud storage is designed to save the models' deliverables, including the article's raw content, category (e.g, business, entertainment), named entity recognition tags, and an article summary. After that, the storage feeds the historical data to a tool that provides data analytics through visualizations.

The final outputs going through the UI are the title of the article (scraping from article's input link) and the results of our three models. On the other hand, the cloud-based business intelligence tool plays an important role to improve the system and models' performances by giving user inputs' analytics. The dashboards are for internal use only and are not publicly available.

In general, the frontend process goes as follows: a user input an article link through the user interface and the application returns the article's title, the predicted category, location tags, person tags, organization tags, a summary, and a graph that shows entities and their relationships.

Figure 30

System Architecture



5.2.2 System Supporting Platform and Cloud Environment

The figure below illustrates our supporting platforms and cloud environment. The diagram has the same structure as the system architecture. Our user interface is Gradio, which provides a high-level API and allows users to interact with the application. Gradio is implemented through its own Python library and easy to use. Users can input data, visualize results, and see model predictions in real-time with Gradio. This interactivity can enhance the user experience and help users understand model behavior. Currently we run Gradio UIs on our local machine, and they can be accessed by anyone who has access to our machine or network.

Our storage is Amazon Relational Database Service (AWS RDS), which is a fully managed database service and can take care of routine database administration tasks for us, such as scaling, backups, monitoring, provisioning, and patching. This allows us to focus on our application development and data modeling rather than database maintenance. Furthermore, AWS RDS provides a reliable and scalable storage solution for our relational data. We can simply scale the data horizontally as the amount of data increases significantly.

Amazon Quicksight is our data analytics tool choice. AWS RDS and Amazon QuickSight is a convenient solution for data storage, management, analysis, and visualization. AWS RDS can seamlessly integrate with Amazon QuickSight and easily connect our data sources to the visualization and analytics platform without the need for complex data pipelines. Additionally, we can perform real-time or near-real-time data analysis as QuickSight can connect directly to the AWS RDS database to create dynamic and interactive dashboards.

We use Hugging Face as the platform for model development due to several reasons. First, it is easy to use, especially for the transformer library. Second, it has a large community and knowledge base of its own. The system encourages a collaborative environment where

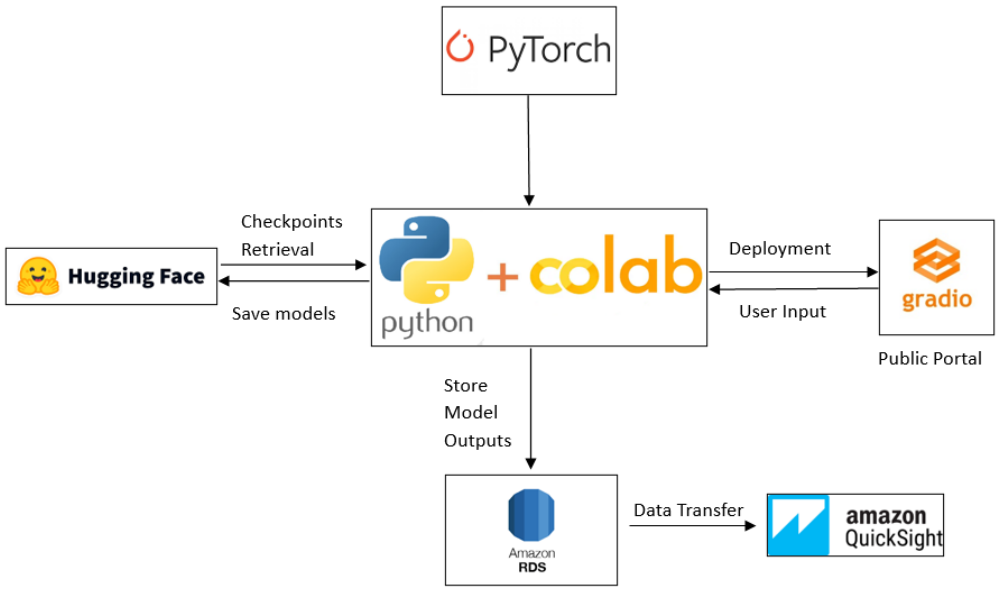
everyone can share knowledge and innovative solutions. As a result, Hugging Face has extensive documentation, tutorials, and example codes for the latest techniques, which is very helpful for developers. Last but not least, Hugging Face provides a vast collection of pretrained models for various NLP tasks, which is highly beneficial for our project in terms of model searching and fine tuning.

Google Collab using Python is the main hosting server for our application, whose responsibilities are scraping the input articles' title and content, retrieve models from Hugging Face, and push the models' outputs back to Gradio and AWS RDS. The main library supporting our models is Pytorch, which connects directly to the Google Collab hosting server in the diagram. Pytorch is a popular deep learning framework that supports Python interface - our primary coding language. In this project, Pytorch is used to process multidimensional arrays, also known as tensors .

Once a user inputs an article URL, Gradio sends a signal to Google Collab. Then, Google Collab retrieves the checkpoints from Hugging Face and pushes back the models' predictions to Gradio. On the other hand, the article's content and model outputs are stored at AWS RDS then transferred to Amazon QuickSight for data analytics purposes.

Figure 31

Supporting Platforms in the Framework

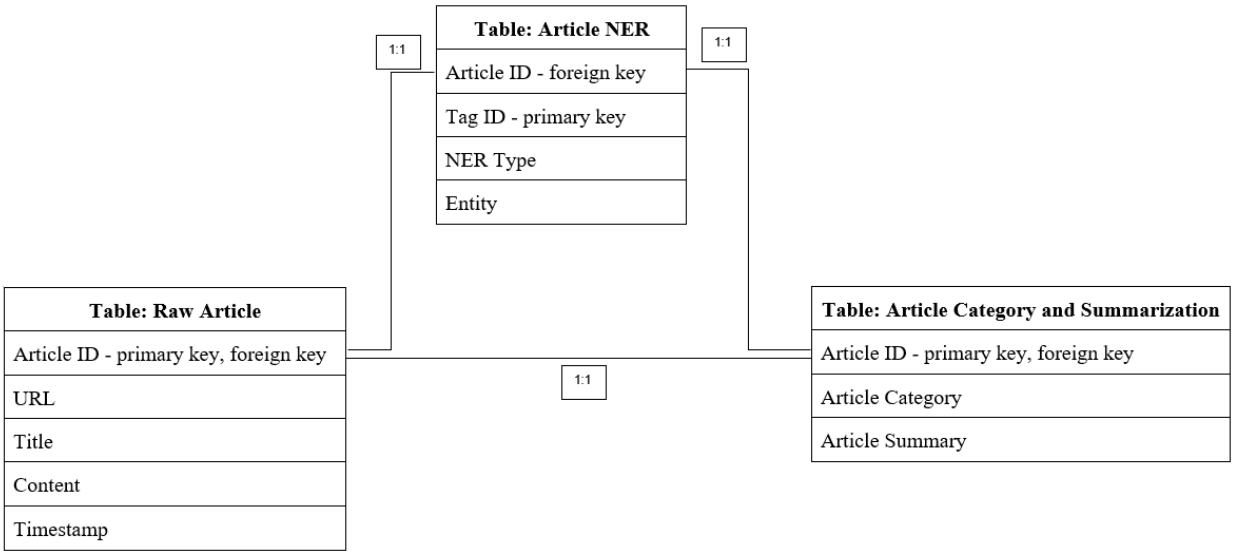


5.2.3 System Data Management Solution

Our data is stored in AWS RDS and can be retrieved by PostgreSQL. Performance optimization is one reason for storing the data: it will avoid recomputation, ensuring quick and efficient content delivery to users. We pick horizontal scaling for the storage setting, meaning the system will add additional RDS instances for reads and writes as the amount of data increases. In addition, we take the advantage of Amazon Cloudwatch service to monitor the CPU usage, performance, and health of our database. Amazon S3 is utilized to save a copy of our database as a backup and recovery plan in case the database is down. The backup is refreshed weekly to keep the cost low and the data not outdated. Figure 32 briefly presents the schema of our database.

Figure 32

Database Schema



We have one table storing the articles' raw data and two tables storing the outputs of our three models. All connect with each other in a one-to-one relationship. Table Raw Article keeps the scraped data, including inputs' urls, titles, contents, and timestamp. We also create an article ID attribute as a primary and foreign key to distinguish different news.

The next two tables below save our model outputs. The Article Category and Article Summarization tables have the same structure so we decided to merge these two into an Article Category and Summarization table. We don't combine Raw Article due to the inefficiency in terms of query complexity, running time, and cost. The design for the Article NER table is slightly different as its primary key and foreign key are distinguished attributes. A sample of how the table looks is shown below.

Table 7

NER table

ner id	article id	ner type	entity
1	1	location	California
2	1	location	San Jose
3	1	person	Elon Musk

4	2	person	Trump
5	2	organization	Apple

5.2.4 System User Interface and Data Visualization

Figure 33 shows the current version of the user interface for the project when the user first reaches the application. Users can input the article link in the input box on the left. On the right side, there are four output boxes (with NER output box being hidden since it is treated as HTML component), which show the news article content when the user input an url, the category that the article is classified, named entities recognized, and the abstractive text summarization.

Figure 33

User Interface Before Input

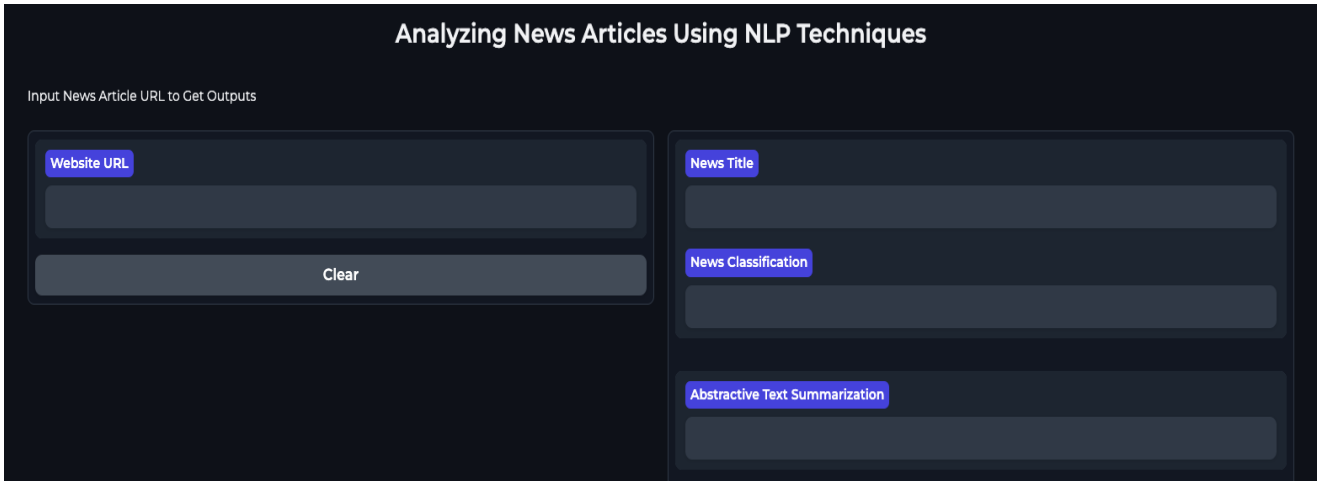
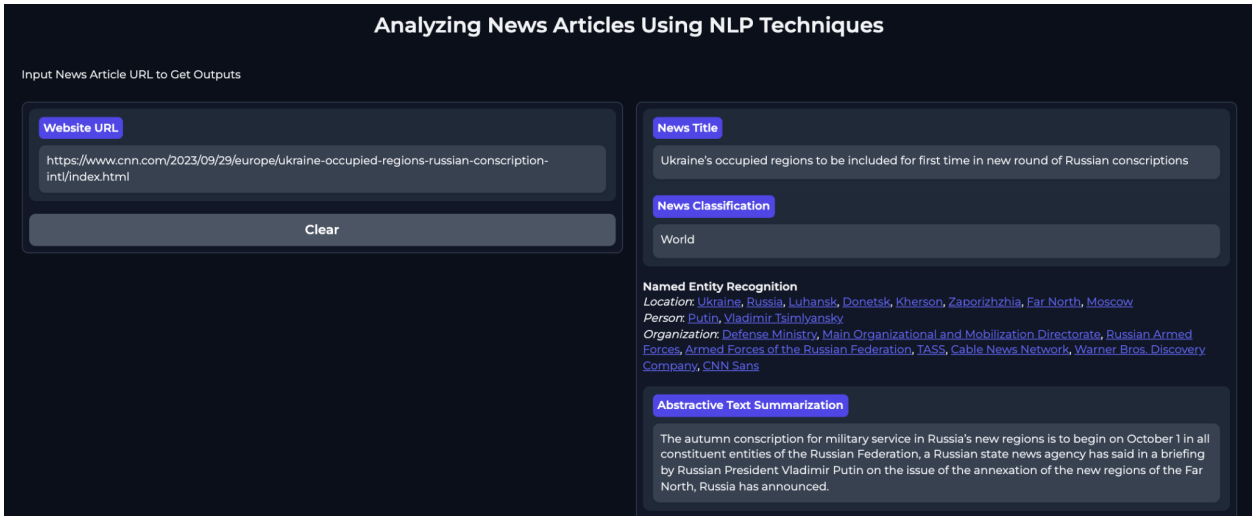


Figure 34 shows the interface after the user puts an url input.

Figure 34

User Interface With Output



We collect the user's input and the corresponding output and run analytics on the stored information. These analytics can be beneficial for developers to understand the distribution of categories users have input, the ongoing trending news, or entities. We will create visualizations such as line plots, scatter plots, bar charts, etc,.. to explore the analytics we can derive from the front end data. This will be further discussed in chapter 6.

5.3 Intelligent Solution

5.3.1 Integrated solutions, ensemble, developed and applied machine learning models

For the news classification model, we have fine-tuned three models, including BERT, RoBERTa, and ERNIE on AG's News dataset, which includes five labeled news categories. To understand and compare performance among the three models, we have examined metrics like accuracy, precision, recall, and F-1 score for evaluation. As of current results, ERNIE is the best performing model to predict the news category. To reduce processing time and still be able to maintain the performance of this model, when a user inserts a url, we will feed the summarization output to the classification model.

For the NER task, Electra and BERT model are fine-tuned on the CoNLL-2003 dataset. We have compared and validated the performance of the two models using F1-score and

entity-level metrics. Electra outperformed BERT in recognizing named entities in the dataset. NER enriches news articles by returning key entities, helps users to understand and extract key information. In order for the model to capture long articles and not lose information, we integrated a chunking function to divide the articles into chunks, so the model can classify all entities in each chunk. Besides that, to better filter the results, we have integrated merging entities, and assigned default google search pages for each of the clicked entities. For example, if Trump and Donal Trump are recognized as entities from the model, the merging entities function helps us reduce redundant information and store only Donald Trump.

For the abstract summarization task, pretrained T5-11B and BART are fine-tuned on CNN-Daily mail dataset. After evaluating the two models' outputs based on the ROUGE metric, we have found out that the T5-11B model has higher performance. The model is capable of providing users with quick and informative article summaries. The process of optimizing the hyper-parameters such as number of beams, temperature value, number of repeat ngram, and length penalty is still being conducted for a more capable model for the final product.

The project leverages machine learning models to help users analyze news articles. The three models are embedded models for three specific tasks. To better fine-tune and validate models, we have separate training processes for each task, and proceed with hyper-parameter tuning after picking the best performing models as mentioned. All models are integrated into the application's backend. After the user hits the submit button with a valid article link, three best models are rendered in the backend through a unified pipeline.

5.3.2 Project input datasets, expected outputs, supporting system contexts, and solution APIs.

- Input datasets: Users input a URL of a news article they are interested in, and the app employs web scraping techniques using BeautifulSoup to fetch the article's title, content,

and date published. The scraped information is the input data that the transformer models will utilize.

- Expected outputs: The app relies on several pre-trained transformer models for text classification, named entity recognition (NER), and text summarization. These models are loaded during the initialization phase of the app, making them readily available for processing user inputs. Each model will input the data from the scraped article and perform their respective task. The expected outputs of the app are multifaceted; the app classifies the news article into several categories: Sports, World, Entertainment, Business, or Sci/Tech. The app also identifies named entities like locations, persons, and organizations while providing Wikipedia links for these entities for further exploration. The app also concisely summarizes the news article, offering users a quick overview. The output of the models is then stored in an AWS RDS Postgres SQL database.
- Supporting system: In terms of supporting systems, the app leverages a range of technologies. BeautifulSoup is utilized for web scraping, enabling the app to fetch the content of news articles. The app uses Hugging Face's Transformers library for the natural language processing of the input article text. Finally, a PostgreSQL database is utilized to store all the processed information.
- Solution APIs: As for the solution APIs, the app integrates Gradio for the user interface, allowing users to input a URL for analysis. The Transformers library supplies the pre-trained models for the NLP tasks. The Requests library makes HTTP requests to fetch news articles when the user provides a URL. Lastly, psycopg2 is used for database interactions, enabling the app to store and retrieve processed information in an AWS RDS PostgreSQL database.

5.4 System Support Environment

We have carefully chosen a variety of tools for the project that serve as the foundation for the creation, implementation, and scalability of our news analysis system. Even though these parts aren't the main focus while the product is in use every day, they are crucial supporting resources. An overview of the main tools used is below:

Google Colab for Accelerated Model Training

The main platform for developing and refining machine learning models is Google Colab. We significantly speed up the model training process by utilizing the processing power of the GPUs supplied by Colab. This effectiveness plays a key role in our work on optimizing text categorization, named entity recognition, and text summarization models.

Google Drive for Seamless Dataset Management

The main center for handling our sizable dataset is Google Drive. The training, validation, and testing data are readily accessible because of its smooth interaction with Google Colab. This synergy makes it much easier for the team members to work together and share data.

AWS RDS with PostgreSQL

Amazon RDS with PostgreSQL is the data management and archiving solution we choose. This relational database service in the cloud offers strong administration, storage, and security options for data. It contains crucial project data such as metadata, model weights, and user interactions. As the project develops, this infrastructure ensures the accuracy and scalability of the data.

Gradio

Gradio contributes significantly to the technical functionality of our solution by offering a simple online interface. It streamlines user interactions, making it simple for users to enter news

items and get results. Our system's technical aspects are simplified by this integration, making it usable by people with basic technical knowledge. Gradio's clear methodology makes it easier to integrate machine learning models into web apps, which helps to create a news analysis platform that is more user-centric and effective.

Hugging Face Hub for Model Management

The Hugging Face Hub is used as a key element in the project for effective model management and system integration. Natural language processing (NLP) platform known as The Hugging Face Hub gives users access to a significant collection of pre-trained NLP models, including the most advanced transformer model.

Our system and The Hugging Face Hub's seamless integration let us quickly access these pre-trained models. By utilizing this hub, we have access to a large number of weights and configurations that have already been trained, greatly decreasing the time and effort required to create complicated NLP models from the start. This not only expedites the development process and saves significant computing resources, but it also enables us to concentrate on perfecting these models to meet our needs.

The Hugging Face Hub also makes it easier for the team to collaborate and share models. We can quickly post our improved models to the hub so that other team members and the larger NLP community may use them. This encourages knowledge exchange and guarantees that our system keeps up with the most recent NLP developments since we can take in and modify updated pre-trained models as they become accessible on the hub.

In essence, the support ecosystem that enables the creation, implementation, and scalability of this news analysis system is made up of these carefully selected tools and parts. Even though they might not be at the forefront of our users' everyday interactions, they are

crucial to maintaining the effectiveness, dependability, and user-centric design of our system throughout the course of its entire existence. We are able to produce accurate and perceptive news analysis findings while protecting the integrity of our data and models thanks to this all-encompassing environment.

6. System Evaluation and Visualization

6.1 Analysis of Model Execution and Evaluation Results

6.1.1 Text Classification

ERNIE is our best fine-tuned model for the news classification task. We have tried to find the optimized hyper-parameter to improve the performance of the ERNIE model. Table 8 shows all the hyper-parameters of the deployed model.

Table 8

ERNIE Model Hyper-parameter

Parameter	**Values**
Learning rate	2e-5
Epochs	3
Weight decay	0.01
Train batch size	16
Evaluation batch size	16

To evaluate that the model can handle generalization of the task well, we now apply the best model on the test dataset.

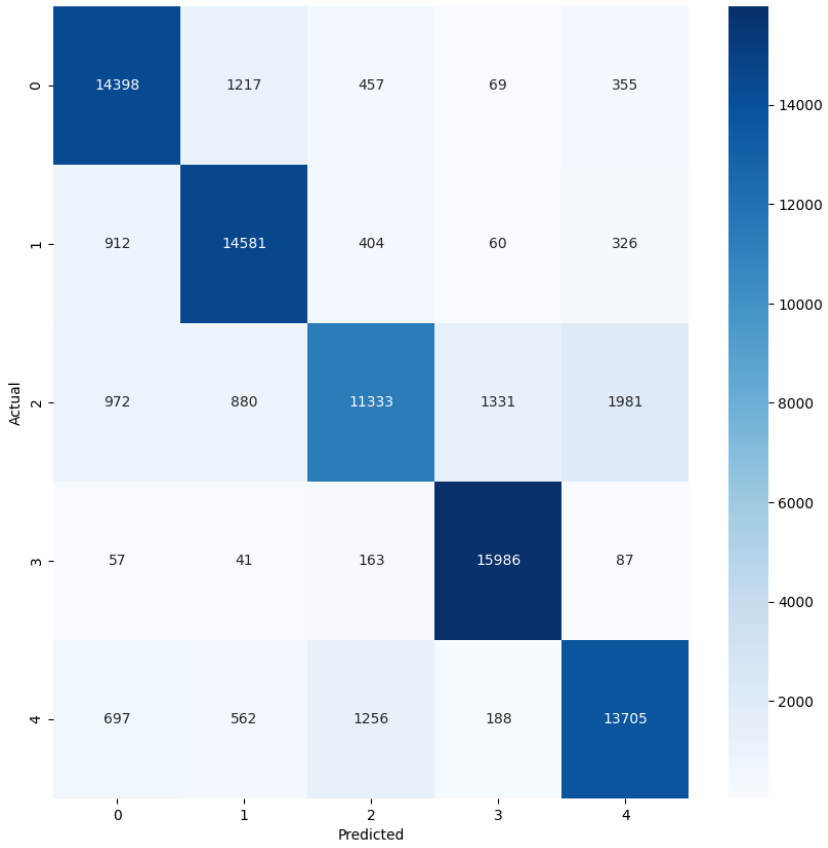
Table 9

Model Evaluation on Test Set - Text Classification

Model	Accuracy	Precision	F1	Recall
ERNIE (fine-tuned)	0.8515	0.8495	0.8501	0.8515

Figure 35

Confusion Matrix on Test Set - Text Classification



The model performance on the test set is similar to the validation set on 4.5, which is the result we look for.

6.1.2 Name Entity Recognition

As shown in section 4.5, ELECTRA outperforms BERT and ERNIE in terms of four metrics: accuracy, precision, recall, and F1 score and we move forward with ELECTRA for our deployment; therefore, this section only analyzes the ELECTRA's pretrained model output. Table 10 illustrates the hyperparameters of the ELECTRA model. The learning rate is 2e-5, which is a common choice for fine-tuning large pre-trained language models, especially in the context of Natural Language Processing. Train batch size and evaluation batch size are set as 8, they take more time to converge but are ideal with limited GPU memory.

Table 10

ELECTRA Hyperparameter Values

Parameter	**Values**
Learning rate	2e-5
Epochs	4
Weight decay	0.01
Train batch size	8
Evaluation batch size	8

We run the pre-trained ELECTRA model on the test dataset to better evaluate if the output matches tagged targets. Table 11 illustrates the performance of the model on the test

dataset with the same four metrics. The model achieved high proficiency with 0.9868 accuracy, 0.9369 precision, 0.9424 F1, and 0.9479 recall score.

Table 11

ELECTRA fine-tuned model on test dataset for NER task

Model	Accuracy	Precision	F1	Recall
ELECTRA (fine-tuned on test dataset)	0.9868	0.9369	0.9424	0.9480

Figure 36 provides the confusion matrix of the fine-tuned model on the test dataset. Observe that the true positive values are relatively high. The accuracy, precision, F1, and recall scores are measured based on nine different labels: O, B-PER, I-PER, B-ORG, I-ORG, B-LOC, I-LOC, B-MISC, I-MISC. Padding tokens are eliminated in the calculations. In deployment, we remove the prefixes B and I and group the labels such that only five categories are visible, including PER, ORG, LOC, MISC, and O.

Figure 36

Confusion matrix of the ELECTRA fine-tuned model on test dataset for NER task

		y_pred								
		O	B-PER	I-PER	B-ORG	I-ORG	B-LOC	I-LOC	B-MISC	I-MISC
y_true	O	28086	1	9	25	47	5	6	23	31
	B-PER	6	899	2	6	0	2	0	3	0
	I-PER	9	2	2047	1	9	0	1	0	1
	B-ORG	14	6	0	830	3	7	0	5	0
	I-ORG	43	0	7	4	1223	2	5	0	8
	B-LOC	10	2	0	13	1	1017	1	9	0
	I-LOC	4	0	0	0	18	3	516	0	11
	B-MISC	17	1	0	12	0	6	1	431	3
	I-MISC	39	2	4	0	11	0	5	6	310

6.1.3 Text Summarization

For deployment within our system, the Flan-T5-small model was selected not only for its superior performance but also for its deployment readiness, characterized by quick summary generation and seamless integration capabilities. The hyperparameters for fine-tuning, detailed in the following table, were carefully chosen to balance efficiency and efficacy.

Table 12

Flan-T5-small Hyperparameter Values

Parameter	**Values**
Learning rate	3e-5
Epochs	4
Train batch size	auto_find_batch_size

To mitigate the computational costs associated with fine-tuning for abstract summarization tasks, we employed Parameter-Efficient Fine-Tuning (PEFT) coupled with Quantized Low-Rank Adapters (QLoRA). This innovative approach reduced the number of trainable parameters to 688,128, a significant decrease from the traditional 77 million parameters typically required for the Flan-T5-small model; curbing computational and financial expenses without compromising model performance. This reduction maintains the model's performance integrity while ensuring that computational and financial expenditures are within practical bounds. It is a testament to the efficacy of our fine-tuning strategy that we can achieve comparable performance with a fraction of the parameter count. The specific LoRA configuration used during training is outlined in the subsequent table.

Table 13

Flan-T5-small Low Rank Adapter Values

Parameter	**Values**
r	16
lora_alpha	32
lora_dropout	0.05
Target_modules	“q” , “v”

Evaluating model performance on the CNN Dailymail test set, which includes 11,185 instances, has demonstrated that the Flan-T5-small’s efficiency translates into real-world applicability. The performance metrics in Table 14, are consistent with those discussed in section 4.5 of this report. These results confirm the model's robustness and validate our choice for deployment, ensuring that users will benefit from rapid and accurate summary generation.

Table 14

Model Evaluation of Abstract Summarization After Fine Tuning on Test Data

Model	ROUGE-1	ROUGE-2	ROUGE-L	BLEU
Flan-T5-small (fine-tuned)	0.3364	0.1404	0.3230	13.01

6.2 Achievements and Constraints

6.2.1 Achievements

We have accomplished a number of noteworthy milestones in the development of our news analysis system, which highlight its usefulness and efficiency. Despite the computational demands of our models, we optimize our training pipeline to make it efficient. For example, model checkpoints are saved at some specific intervals, allowing us to continue optimizing from the last checkpoint in case of interruptions.

One of the greatest achievements is the precise multi-class text classifications that we were able to accomplish by using sophisticated pre-trained models such as ERNIE. We have overcome challenges when solving the text classification task. First, by handling imbalance target features, we have avoided the model's bias towards over-represented classes. Feeding the actual article content to this model led to a longer processing time to get the output since the model needs to divide the input into sequences for a more accurate prediction. Feeding the truncated article by the model's max input length can cause misleading prediction because the model is losing the input content. Facing this issue, we have decided to put the summarized output as the input for the classification task. By doing this, we have improved the processing time of the model, and still maintained the quality of the output.

Furthermore, our system achieves outstanding results in Named Entity Recognition (NER) tasks, extracting and classifying persons, companies, and places from news content using state-of-the-art models such as Electra. For the NER task, we use ConLL2003 which is not considered a large dataset in the context of training and fine-tuning large language models like BERT or ELECTRA. Our models leverage their pre-trained knowledge to excel the task with relatively small amounts of labeled data. The fine-tuned BERT and ELECTRA models also

achieve a competitive performance on NER benchmarks with a relatively high accuracy score (both are greater than 98 percent).

In pursuing an advanced abstract summarization model, we made significant progress, particularly with the Flan-T5-small model, which now surpasses larger models in both performance and computational speed, condensing what used to be a 40-second task into one that takes less than 10 seconds— establishing the Flan-T5-small as our top-performing model. Our innovative strides didn't stop there; we also enhanced training efficiency by implementing PEFT alongside QLoRA, which slashed the necessary training time by half—from 40 hours to 20—thereby reducing both computational load and expense.

Additionally, we have designed a user-friendly online interface that streamlines user interactions and makes accessing classifications, entity recognitions, and summaries simple for both general users and academic readers. Because of its real-time integration with several news data sources, the system is always up to date and a great resource for consumers looking for the most recent information. Together, these accomplishments establish our news analysis system as a potent, user-focused instrument for navigating the intricacies of news data.

6.2.2 Constraints

Even though our news analysis system has accomplished a great deal, it is important to recognize the difficulties and limitations that it faced when it was being developed. The need for significant computer power is one significant barrier, especially when it comes to training and optimizing intricate large language models. These models need a lot of GPU power, which might make large-scale deployments very difficult. Furthermore, the system is somewhat vulnerable to variations and irregularities in the availability and structure of acquiring data due to its dependency on outside news data sources. Maintaining robust data integration and high-quality

data is still a difficulty. Additionally, the diversity and quality of training data might affect the accuracy of NER and text summarization models, therefore ongoing attempts to improve model performance are required. Last but not least, achieving robust system response time requirements necessitates effective resource allocation and optimization, which can be difficult in circumstances with evolving utilization. Given these limitations, we are able to provide our readers with a useful and trustworthy news analysis tool.

For the NER task, the performance after fine-tuning BERT and ELECTRA are not significantly different, therefore deciding whether to use BERT or ELECTRA for deployment is difficult. Another constraint of our NER models is that some named entities may have ambiguous labels or context-dependent interpretations, which can make it difficult to create accurate annotations for fine-tuning. Resolving such ambiguity is a constraint during data preparation.

During the abstract summarization process, a significant constraint for fine-tuning the model was the financial and computational resources. Initial training on a Google Colab T4 GPU quickly proved inadequate, compelling an investment in the much more robust A100 GPU to fine-tune our model for abstract summarization effectively. The balance between computational requirements and budgetary restrictions has been delicate, necessitating planning and strategic allocation of resources. Despite these hurdles, the enhancements in our model's performance justify the investments made in computational resources.

6.3 System Quality Evaluation of Model Functions and Performance

To guarantee the reliability and efficiency of our news analysis system, we must assess the models' accuracy and runtime performance in relation to system reaction time objectives. We meticulously use a variety of measures, including accuracy, precision, recall, and F1-score, to

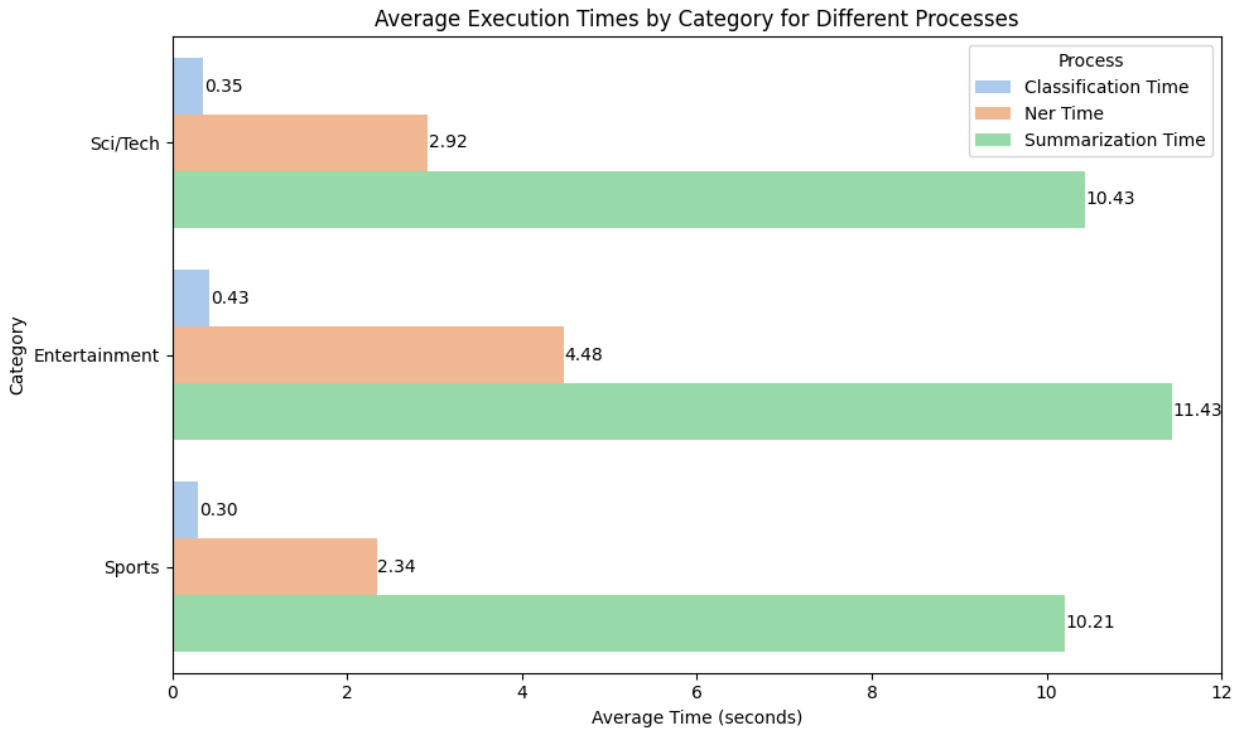
evaluate the models' correctness in classifying news items into categories and correctly identifying named entities. Furthermore, the ROUGE and BLEU scores offer significant insight into the effectiveness of text summarization. At the same time, runtime performance evaluation includes real-time system monitoring, resource consumption and tracking response times for different activities. This holistic approach enables us to fine-tune models and system components, ensuring not only the correctness of results but also meeting our stringent system response time targets, thus delivering an efficient and dependable tool for news analysis to our users.

The input and the output are the two main parts of this section. The end-user's raw text or the article URL incorporating the tasks to be completed makes up the input. The results of every model on its tasks make up the output component. AWS RDS PostgreSQL database is used to store the input and output components, which aids in system performance analysis. The system inputs and outputs can be seen in Appendix A.

A table in the database is made that records how long it takes a model to run to keep track of each model's reaction timings. Every time a submission is sent to the input field, tasks including summarization, NER, and classification are performed. There are differences in response times for every job. It is crucial to record this, for that reason. Figure 37 illustrates the model performance for various categories.

Figure 37

Average Execution Times by Category for Different Processes



6.4 System Visualization

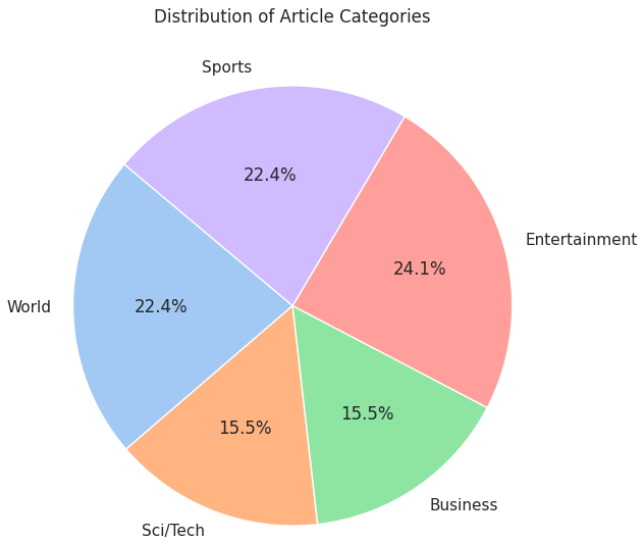
When it comes to news analysis, system visualization plays a crucial role since there is an endless flow of information. The news analysis system, which skillfully combines natural language processing and machine learning, is excellent at classifying, analyzing, and summarizing news stories. Nonetheless, the efficacy of any analytical system depends not only on its data processing capabilities but also on its ability to provide the findings understandably and intuitively. The use of visualization techniques is essential to reaching this goal. They act as a link between the end users and the data analysis backend, facilitating a more thorough comprehension of the project's conclusions.

The visualization covers a wide range of topics related to news analysis. These visualizations capture the essence of our system's capabilities, from the distribution of news categories to the dynamics of named entities inside articles, from temporal patterns in news volume to the effect of summarization on article length.

The distribution of news stories among the five categories World, Science and Technology, Business, Entertainment, and Sports is depicted in the pie chart in Figure 38. It gives viewers a concise summary of how often each category is in the news data that has been examined.

Figure 38

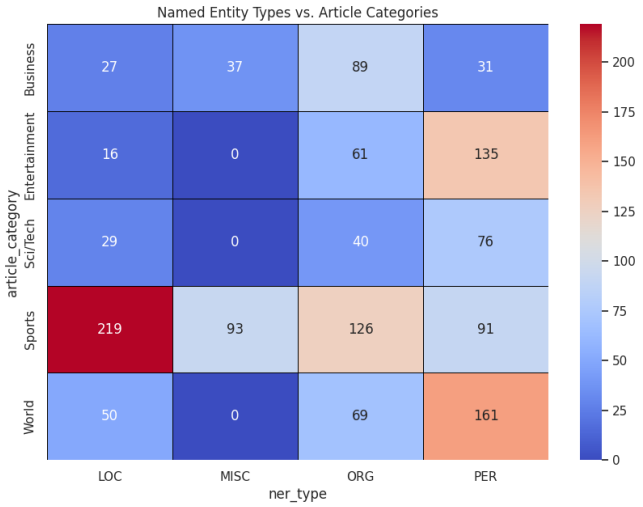
Distribution of Article Categories



The link between named entity types and article categories is displayed in the heatmap in Figure 39. To get insight into the contextual importance of entities inside stories, viewers can investigate the distribution of various entity kinds (persons, locations, and organizations) across news categories. The heatmap gives insights such as the NER location for category sports has a higher ner count compared to other categories. This allows the developers to figure out whether it is an anomaly or not.

Figure 39

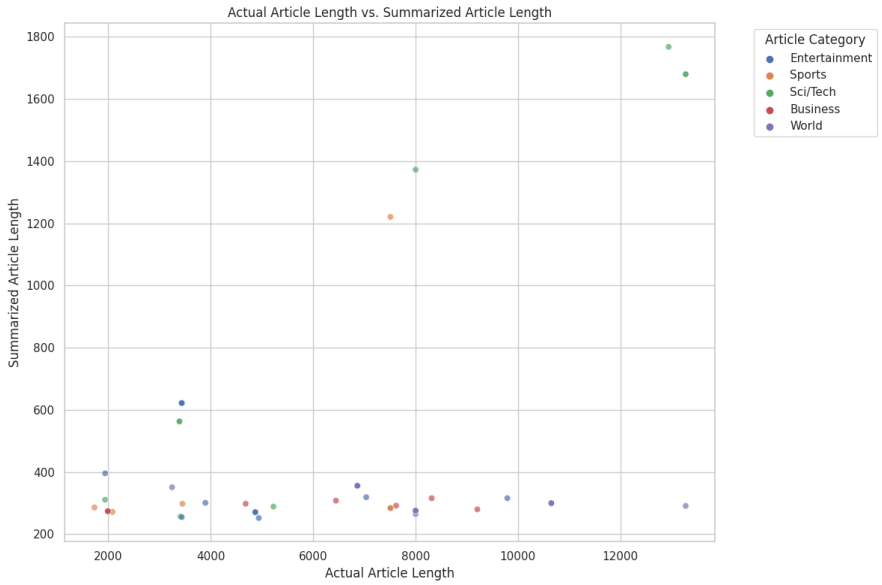
Heatmap between Named Entity Types and. Article Categories



The Scatter Plot in Figure 40 shows the difference between news article summaries and real lengths. Viewers may investigate the relationship between article length and the summary process, since distinct hues correspond to various categories. Compared to comparable articles in other categories, it is evident that a small number of Sci/Tech articles have significantly longer summaries.

Figure 40

Scatter Plot to Show Actual Length and Summarized Length

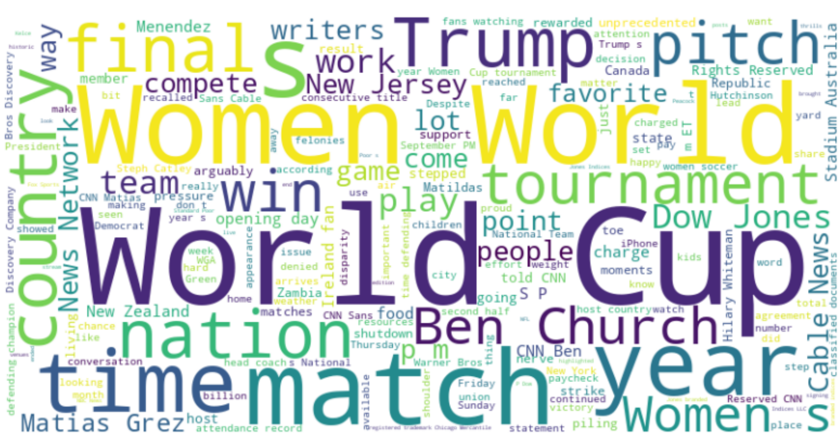


We use a word cloud to examine the linguistic richness of news articles in this visually appealing presentation. The Figure 41 immerses viewers in the literary landscape of news items, in contrast to the preceding representations. The word cloud provides a clear image of the most common themes and keywords by compiling and displaying the words that appear most frequently in the news data analysis.

Viewers may understand the most popular subjects and keywords in the news corpus in a novel and engaging way with the help of this visualization. By emphasizing the most powerful words and phrases, it distills the core of the news stories. The word cloud is a crucial component of our system's data display since it is an effective tool for rapidly recognizing important themes, trends, and issues within news items.

Figure 41

Wordcloud to the Show Most Occurring Words

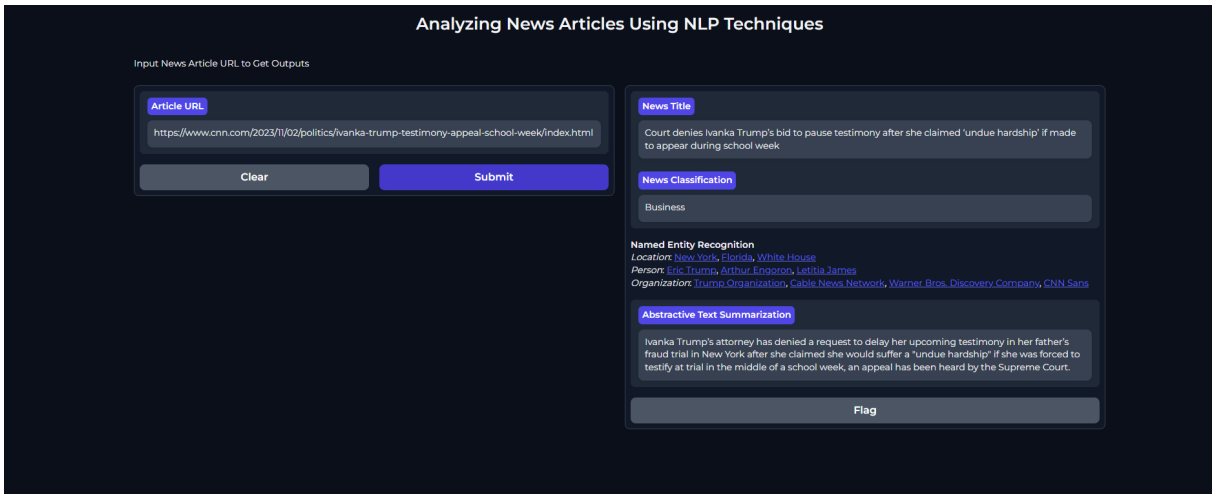


These are not the only visualizations offered by the system. Developers make use of the visualizations, which are concealed from users. This is done to assist developers in understanding the patterns that underlie the data for each of the jobs, allowing them to delve deeper and make informed judgments. The end-users engage with the Gradio-developed front end, where they may enter the article's URL or its raw content. After a series of preprocessing

steps, the input is fed into the models. Finally, the models' output is presented on the front end. Figure 42 shows the UI of the front end.

Figure 42

UI Developed Using Gradio



7. Conclusion

7.1 Summary

The project has achieved significant milestones across various facets, successfully optimizing the training pipeline to handle computational demands and implementing a resilient checkpoint system. We have achieved the desired business requirements by implementing three successful and efficient tasks to analyze news articles. During the data collection and engineering process, we have learned and performed necessary processing, including handling missing values, encoding target features, mitigating bias issues in multi-class text classification tasks, and tokenizing and transforming data to a designed structure for each of the tasks. The modeling phase required an understanding of how to fine-tune for each task to implement and evaluate them. We have succeeded in delivering outputs for Text Classification tasks with 85 percent accuracy by trying multiple pre-trained models as BERT, ERNIE, and RoBERTa. Besides, we

have demonstrated precise and outstanding results in Named Entity Recognition using advanced models like ERNIE and Electra. Moreover, the Flan-T5-small model excelled in abstract summarization, surpassing larger counterparts, and the implementation of PEFT and QLoRA significantly reduced training time. Our system implementation resulted in a user-friendly interface with real-time integration, establishing the news analysis system as a potent, efficient, and up-to-date tool for navigating news data intricacies, with implications for resource optimization and model adaptability in real-world applications.

7.2 Benefits and Shortcoming

7.2.1 Benefits

Our news analyzing project, featuring three integral tasks—news classification, Named Entity Recognition (NER), and abstractive summarization—offers a plethora of benefits in navigating the vast landscape of information. Through news classification, the project efficiently categorizes articles, providing users with organized and accessible content. NER enhances the depth of understanding by extracting and categorizing entities, shedding light on the key players and topics within articles. The abstractive summarization task distills the essence of lengthy articles into concise and informative summaries, facilitating quick comprehension and saving valuable time for users. Collectively, these tasks empower users to stay well-informed, navigate through diverse news categories, and gain nuanced insights into complex stories, contributing to a more efficient and comprehensive news consumption experience.

7.2.2 Shortcomings

While our News Analyzing Project excels in its multifaceted approach to news analysis, it does encounter certain limitations. One notable constraint lies in access limitations to specific news sources, potentially restricting the diversity of input data. The project's current runtime,

pegged at around one minute, may pose challenges in handling a high volume of user requests simultaneously, affecting real-time responsiveness. Furthermore, the system currently categorizes news into five predetermined categories, which might limit its adaptability to emerging or niche topics that fall outside these predefined classifications. Additionally, the accuracy of the Named Entity Recognition (NER) and abstractive summarization tasks could be impacted by the quality and diversity of training data. To address these shortcomings, continuous efforts are required to expand access to diverse news sources, enhance real-time processing capabilities, and refine the categorization framework to accommodate evolving news trends and themes.

7.3 Potential System and Model Applications

The algorithms and technology have more potential uses than only analyzing news. Although providing precise and timely insights from articles has been the main emphasis, there are many other uses for machine learning models due to their adaptability. These models may be applied to other domains because they were trained for tasks including text categorization, named entity recognition, and text summarization.

The system can be quite useful for automatically classifying and summarizing academic publications in the field of academic research. This improves researchers' productivity and knowledge retrieval skills in addition to streamlining the time-consuming process of finding relevant papers. It may also be used for customer feedback processing and sentiment analysis in the business environment. They are proficient in the analysis and summarization of customer reviews, which helps businesses understand consumer sentiment in-depth, spot new patterns and effectively address criticism. Moreover, the named entity recognition (NER) capabilities of the system have wide knowledge management applications when combined with Wikipedia page tagging. This capability may be used to create complex knowledge graphs and information

retrieval systems, which are essential in the legal, banking, and healthcare industries for managing contracts and documents, fraud detection, and risk assessment, among other sectors.

In essence, the system is a flexible tool that provides insightful analysis of articles and can transform information processing and comprehension in a wide range of fields and sectors. In the age of data-driven decision-making and knowledge discovery, their accuracy and agility make them invaluable assets.

7.4 Experience and Lessons Learned

The project triggered the learning of the latest and most handful pre-trained models and approaches as of the time the project was conducted, such as ERNIE, ELECTRA, T4, and so forth. Spending time to research and understand the models helped save a lot of time in later steps as we had some ideas of why the models would work well for our project.

Another lesson learned is that building solid and organized codes is beneficial, especially in case we want to try more models for a task. In the case of our project, we considered implementing some more models at the last minute as we found out the compatibility between the model and our tasks. Since we built an organized pre-processed code with clear explanation code, it was much easier to test a new model and we actually added ERNIE to the NER task and were able to comprehensively compare three models: BERT, ERNIE, and ELECTRA.

Pre-processing the data properly is crucial in our project because one missing or incorrect step might drive our result remarkably. A typical NLP project like ours requires many pre-processed techniques involved, such as removing stop-words, removing URLs, removing punctuations, expanding contraction, stemming, or lemmatizing, etc. All these are standard steps to reduce the noise in the text data and convert it into a structured format that can be analyzed by the models.

Deep learning models are known to be more dependent on the computing power as they usually have more data points and parameters, and one of the biggest obstacles of the project was the computation limitation. We fine-tuned the models to the best of our knowledge and available resources, but they might even be more optimal if we had a more resourceful and faster machine.

7.5 Recommendations for Future Work

To enhance the comprehensiveness of our news analyzing application, we envision the expansion of the news categories. This includes incorporating expanded categories on top of the existing five. To embrace a more sophisticated classification system, we aim to shift from multiclass to multilabel classification. This advancement will allow news articles to be associated with multiple categories simultaneously, reflecting the complexity and multi-faceted nature of real-world news.

Implementing knowledge-based graphs for NER is a strategic initiative to improve the precision and context awareness of named entities in news articles. This approach leverages existing knowledge bases to enhance entity recognition accuracy and foster a more nuanced understanding of entities' relationships within the news context. Knowledge graphs can be continuously updated to reflect real-time changes and ensure the system remains current and relevant to the world.

Advancements in summarization techniques will be a focal point for the future. We are currently implementing abstractive summarization, but for the future, users can have the ability to choose between abstractive and extractive options.

Last but not least, we want to explore the ability to enhance the user interface of our news analyzing application. The focus is on improving the overall aesthetic appeal, intuitiveness, and functionality of the UI to ensure a seamless and enjoyable user interaction. We plan to implement

a more visually engaging design that aligns with modern UI/UX principles, incorporating user-friendly navigation and a responsive layout.

7.6 Contributions and Impacts on Society

Our article classification app helps in organizing and categorizing vast amounts of information available on the internet. It enables efficient retrieval of relevant articles or documents based on specific topics, making it easier for users to find the information they need. It is applicable in a variety of industries, such as finance, education, public sector, and so forth. For example, policymakers can stay updated on scientific developments and technological trends to formulate policies that align with the current state of these fields. This ensures that regulations are adaptive and supportive of innovation.

NER identifies and classifies names of people, organizations, locations, and other, within text helps in extracting valuable information. This can contribute to knowledge discovery, trend analysis, and understanding relationships between entities, which is valuable for researchers, businesses, and policymakers. Search engines can use NER tags to enhance search results, and users can receive more accurate and relevant information, improving their overall online experience. Businesses and policymakers can use the application to analyze large volumes of textual data, helping them make more informed decisions. This is especially crucial in areas such as policy-making, risk assessment, and market analysis.

Our summarization application saves time for readers and helps them quickly grasp the key points of a document without going through the entire text. For education and learning, students and researchers can benefit from automatic summarization to quickly understand and review extensive articles, facilitating learning and research processes. In addition, for people

with language barriers or those who have difficulty reading lengthy texts, summarization can provide a more accessible way to understand the main ideas and information.

References

Baevski, A., Edunov, S., Liu, Y., Zettlemoyer, L., & Auli, M. (2019). Cloze-driven Pretraining of Self-attention Networks. *Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/d19-1539

Chen, X., Cong, P., & Lv, S. (2022). A Long-Text Classification Method of Chinese News Based on BERT and CNN. *IEEE Access*, *10*, 34046–34057. https://doi.org/10.1109/access.2022.3162614

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N.M., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B.C., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., García, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A.M., Pillai, T.S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Díaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K.S., Eck, D., Dean, J., Petrov, S., & Fiedel, N. (2022). PaLM: Scaling Language Modeling with Pathways. *ArXiv, abs/2204.02311*.

Clark, K., Luong, M., Le, Q.V., & Manning, C.D. (2020). ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. *ArXiv, abs/2003.10555*.

Deping, L., Hongjuan, W., Mengyang, L., & Pei, L. (2021). News text classification based on Bidirectional Encoder Representation from Transformers. *2021 International Conference on Artificial Intelligence, Big Data and Algorithms (CAIBDA)*. https://doi.org/10.1109/caibda53561.2021.00036

Devlin, J., Chang, M., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv (Cornell University)*. https://arxiv.org/pdf/1810.04805v2

Dou, Z., Liu, P., Hayashi, H., Jiang, Z., & Neubig, G. (2021). GSum: A General Framework for Guided Neural Abstractive Summarization. *North American Chapter of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/2021.naacl-main.384

Edmundson, H. P. (1969). New methods in automatic extracting. Journal of the ACM, 16(2), 264-285. https://dl.acm.org/doi/10.1145/321510.321519

Gehrmann, S., Ziegler, Z. M., & Rush, A. M. (2019). Generating Abstractive Summaries with Finetuned Language Models. *International Conference on Natural Language Generation*. https://doi.org/10.18653/v1/w19-8665

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems (2nd ed.). O'Reilly Media.

Grishman, R., & Sundheim, B. (1996). Message understanding conference - 6: A brief history. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING 96), Copenhagen, August 1996* (pp. 466-471).

Gulli, A. (n.d.). *AG's corpus of news articles* [Dataset]. http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

Hermann, K., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., & Blunsom, P. (2015). Teaching Machines to Read and Comprehend. *NIPS'15: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. https://arxiv.org/pdf/1506.03340.pdf

Hu, A., Dou, Z., Nie, J. Y., & Wen, J. R. (2020). Leveraging Multi-Token Entities in Document-Level Named Entity Recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(05), 7961–7968. https://doi.org/10.1609/aaai.v34i05.6304

Hutchins, W. J. (2003). Early years in machine translation: memoirs and biographies of pioneers. John Benjamins Publishing.

Jiang, Y., Hu, C., Xiao, T., Zhang, C. L., & Zhu, J. (2019b). Improved Differentiable Architecture Search for Language Modeling and Named Entity Recognition. *Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/d19-1367

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In European conference on machine learning (pp. 137-142). Springer. https://link.springer.com/chapter/10.1007/bfb0026683

Joshi, M. S., Levy, O., Zettlemoyer, L., & Weld, D. S. (2019). BERT for Coreference Resolution: Baselines and Analysis. *Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/d19-1588

Kelleher, J. D., Namee, B. M., & D'Arcy, A. (2015). Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies. Amsterdam University Press.

Kong, J., Wang, J., Zhang, X. (2022). Hierarchical BERT with an adaptive fine-tuning strategy for document classification. https://doi.org/10.1016/j.knosys.2021.107872

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., & Zettlemoyer, L. (2020). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/2020.acl-main.703

Li, J., Zhang, D., & Wulamu, A. (2021). Chinese Text Classification Based on ERNIE-RNN. In *2021 2nd International Conference on Electronics, Communications and Information Technology (CECIT)*. https://doi.org/10.1109/cecit53797.2021.00072

Lin, C. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. *Annual Meeting of the Association for Computational Linguistics*. https://aclanthology.org/W04-1013.pdf

Liu, Y., & Liu, P. (2021). SimCLS: A Simple Framework for Contrastive Learning of Abstractive Summarization. *Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/2021.acl-short.135

Liu, Y., Liu, P., Radev, D., & Neubig, G. (2022). BRIO: Bringing Order to Abstractive Summarization. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). https://doi.org/10.18653/v1/2022.acl-long.207

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. & Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach (cite arxiv:1907.11692)

Liu, Z., Lin, W., Shi, Y. P., & Zhao, J. (2021). A Robustly Optimized BERT Pre-training Approach with Post-training. *Springer International Publishing EBooks*, 471–484. https://doi.org/10.1007/978-3-030-84186-7_31

Maribeth, B. (1968). PERT and CPM: a selected bibliography. Archive.org. https://archive.org/details/pertcpmselectedb53bren/page/n5/mode/2up

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. https://arxiv.org/abs/1301.3781.

Nallapati, R., Zhou, B., Santos, C. N. D., Gulcehre, C., & Xiang, B. (2016). Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond. *ArXiv (Cornell University)*. https://doi.org/10.18653/v1/k16-1028

Ni, J., Hernandez Abrego, G., Constant, N., Ma, J., Hall, K., Cer, D., & Yang, Y. (2022). Sentence-T5: Scalable sentence encoders from pre-trained text-to-text models. *Findings of the Association for Computational Linguistics: ACL 2022*. https://doi.org/10.18653/v1/2022.findings-acl.146

Papineni, K., Roukos, S., Ward, T. J., & Zhu, W. (2002). BLEU: a method for automatic evaluation of machine translation. *Meeting of the Association for Computational Linguistics*. https://doi.org/10.3115/1073083.1073135

Potamias, R. A., Siolas, G., & Stafylopatis, A.-. (2020). A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, *32*(23), 17309–17320. https://doi.org/10.1007/s00521-020-05102-3

Qi, W., Yan, Y., Gong, Y., Liu, D., Duan, N., Chen, J., Zhang, R., & Zhou, M. (2020). ProphetNet: Predicting Future N-gram for Sequence-to-SequencePre-training. *Empirical Methods in Natural Language Processing*. https://doi.org/10.18653/v1/2020.findings-emnlp.217

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2019). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. https://doi.org/10.48550/arXiv.1910.10683

Sanh, V., Debut, L., Chaumond, J., Wolf, T. (2020). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. https://arxiv.org/abs/1910.01108

Schweter, S., Akbik, A. (2021). FLERT: Document-Level Features for Named Entity Recognition. https://arxiv.org/abs/2011.06993

Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to fine-tune bert for text classification? *Lecture Notes in Computer Science*, 194–206. https://doi.org/10.1007/978-3-030-32381-3_16

Sun, Y., Shuohuan, W., Li, Y., Feng, S., Chen, X., Zhang, H., Tian, X., Danxiang, Z., Tian, H., & Wu, H. (2019). ERNIE: Enhanced Representation through Knowledge Integration. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1904.09223

Tjong Kim Sang, E. F., & De Meulder, F. (2003). Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 (pp. 142-147).

Ushio, A., & Camacho-Collados, J. (2021). T-NER: An all-round python library for Transformer-based named entity recognition. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*. https://doi.org/10.18653/v1/2021.eacl-demos.7

Vanderplas, J., & VanderPlas, J. (2016). Python Data Science Handbook: Essential Tools for Working with Data. Van Duuren Media.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. (2017). Attention is all you need. *In Advances in neural information processing systems.* https://papers.nips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Vychegzhanin, S., & Kotelnikov, E. (2019). Comparison of Named Entity Recognition Tools Applied to News Articles. *2019 Ivannikov Ispras Open Conference (ISPRAS)*. https://doi.org/10.1109/ispras47671.2019.0001

Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., & Tu, K. (2021). Automated Concatenation of Embeddings for Structured Prediction. *Meeting of the Association for Computational Linguistics*. https://doi.org/10.18653/v1/2021.acl-long.206

Wang, X., Jiang, Y., Bach, N., Wang, T., Huang, Z., Huang, F., Tu, K. (2022). Improving Named Entity Recognition by External Context Retrieving and Cooperative Learning. https://doi.org/10.48550/arXiv.2105.03654

Wu, C., Wu, F., Yu, Y., Qi, T., Huang, Y., & Liu, Q. (2021). NewsBERT: Distilling Pre-trained Language Model for Intelligent News Application. *Findings of the Association for Computational Linguistics: EMNLP 2021*. https://doi.org/10.18653/v1/2021.findings-emnlp.280

Yamada, I., Asai, A., Shindo, H., Takeda, H., Matsumoto, Y. (2020). Luke: Deep Contextualized Entity Representations with Entity-aware Self-attention. https://doi.org/10.48550/arXiv.2010.01057

Yan, H., Deng, B., Li, X., & Qiu, X. (2019). TENER: Adapting Transformer Encoder for Named Entity Recognition. ArXiv, abs/1911.04474. Yan, H., Deng, B., Li, X., & Qiu, X. (2019). TENER: Adapting Transformer Encoder for Named Entity Recognition. ArXiv, abs/1911.04474.

Yang, P., Li, W., & Zhao, G. (2019). Language Model-Driven Topic Clustering and Summarization for News Articles. *IEEE Access*, *7*, 185506–185519. https://doi.org/10.1109/access.2019.2960538

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. https://doi.org/10.48550/arXiv.1906.08237

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. (2020). PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization. *International Conference on Machine Learning*, *1*, 11328–11339. http://proceedings.mlr.press/v119/zhang20ae/zhang20ae.pdf

Zhang, T. (2023, January 31). *Benchmarking Large Language Models for News Summarization*. arXiv.org. https://doi.org/10.48550/arXiv.2301.13848

Zhang, X., Zhao, J., LeCun, Y. (2015) Character-level Convolutional Networks for Text Classification. *Neural Information Processing Systems. https://doi.org/10.48550/arXiv.1509.01626*

Zhao, X., Greenberg, J., An, Y., & Hu, X. T. (2021). Fine-tuning Bert Model for materials named entity recognition. *2021 IEEE International Conference on Big Data (Big Data)*. https://doi.org/10.1109/bigdata52589.2021.9671697

Zhaoye, X., Xiaoqun, L., & Peijie, S. (2021). Hybrid Chinese text classification model based on pretraining model. *Journal of Physics: Conference Series*, *1961*(1), 012002. https://doi.org/10.1088/1742-6596/1961/1/012002

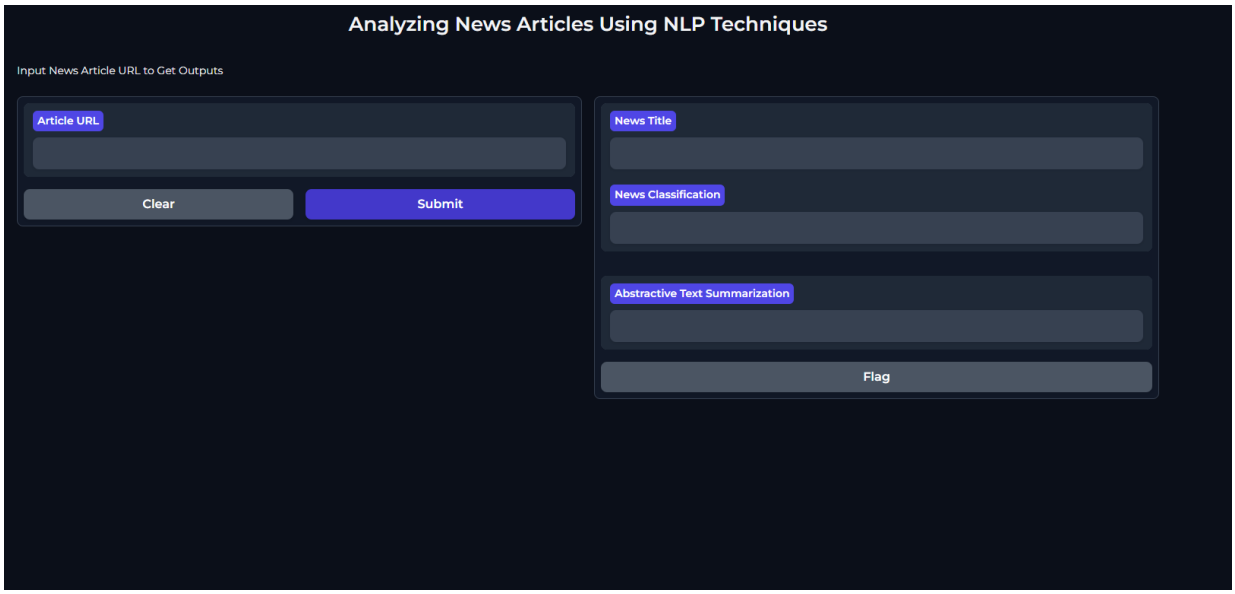
Zheng, S., & Yang, M. (2019). A new method of improving Bert for text classification. *Intelligence Science and Big Data Engineering. Big Data and Machine Learning*, 442–452. https://doi.org/10.1007/978-3-030-36204-1_37

Zhu, Y., Kiros, R., Zemel, R., Salakhutdinov, R., Urtasun, R., Torralba, A., & Fidler, S. (2015). Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/iccv.2015.11

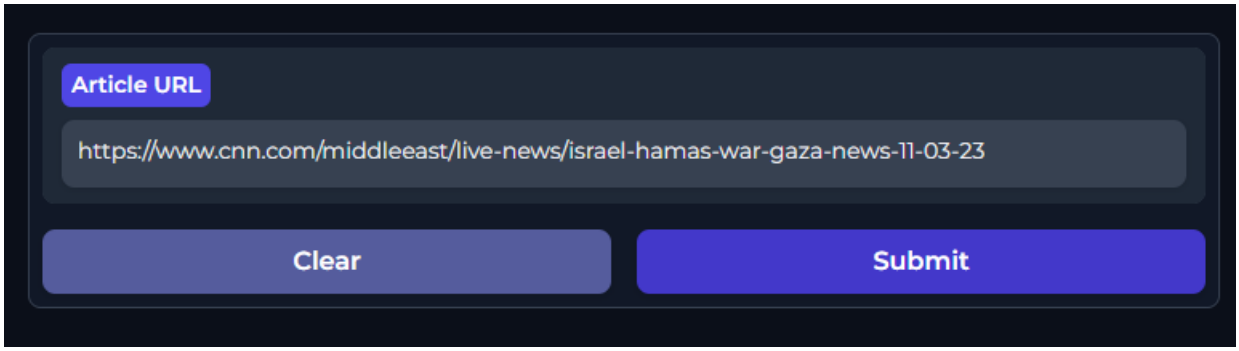
Appendix A

Appendix A of the report provides the screenshots of the user interface. It consists of the inputs, processing and output GUI.

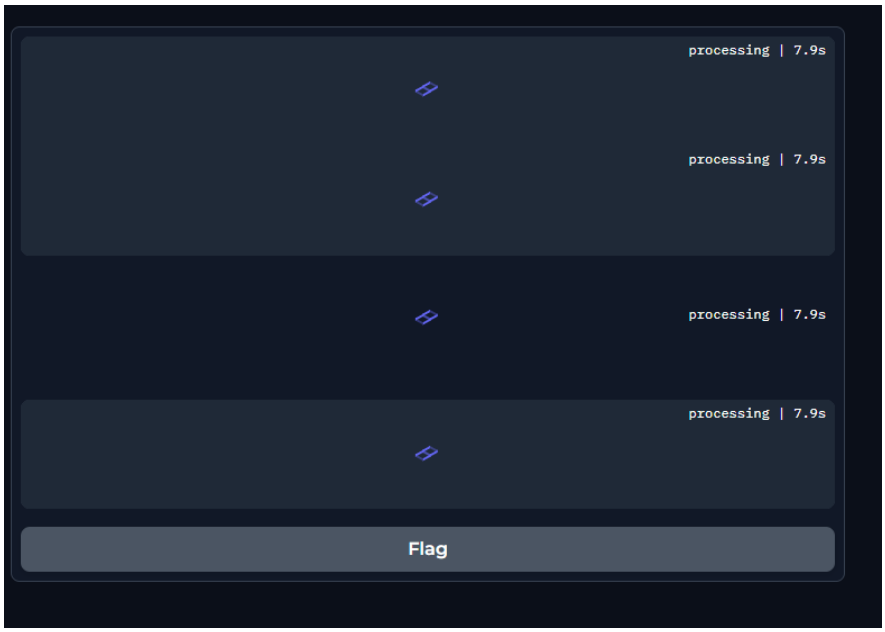
System UI without inputs and outputs



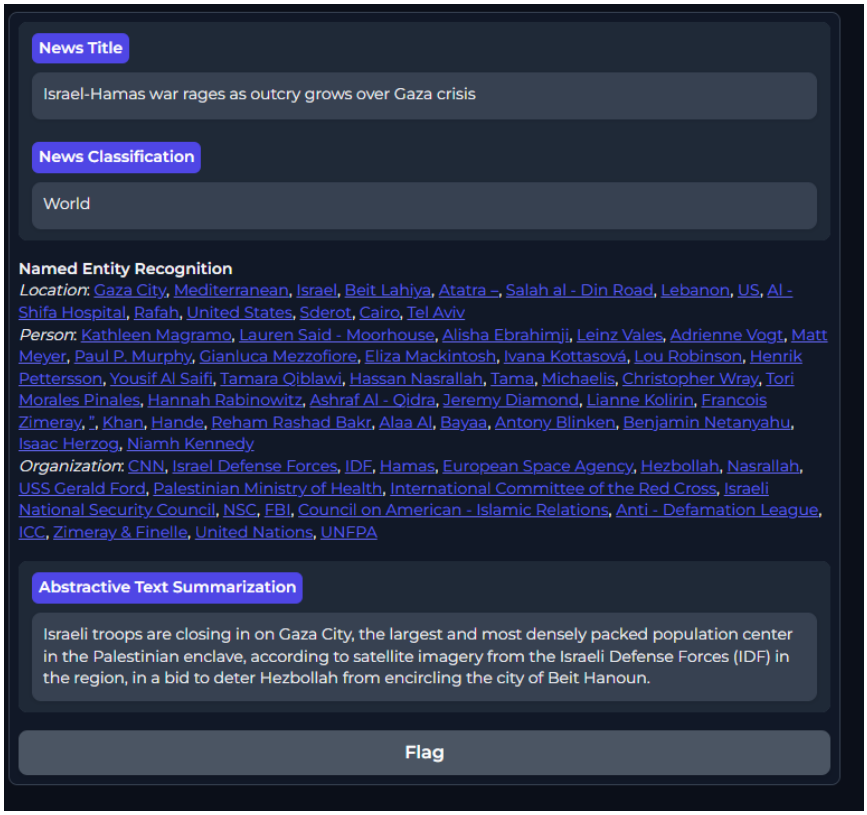
System Inputs



System Processing



System Output



Quality of Service (QoS) Report

Overview

This report presents the Quality of Service (QoS) results for the News Article Analysis System, which performs three key tasks: classification, Named Entity Recognition (NER), and summarization. The testing focused on measuring the time taken by the system to complete each task.

Testing Results

Classification Time (seconds)	**NER Time (seconds)**	**Summarization Time (seconds)**
0.273702	3.227790	12.956416
0.291723	2.588286	11.333945
0.316841	2.154046	8.410232
0.284414	2.282001	10.890079
0.434787	2.619441	7.907824
0.425477	4.477119	11.430696
0.279232	7.522642	17.000388

0.552027	3.422284	12.600062

Individual Task Performance

Classification Time Analysis.

The system demonstrated an average classification time of 0.332 seconds across the tested articles. The maximum classification time observed was 0.552 seconds.

NER Time Analysis.

The NER task exhibited an average execution time of 3.318 seconds. The maximum NER time observed during testing was 7.522 seconds.

Summarization Time Analysis.

Summarization tasks were completed with an average time of 11.741 seconds. The longest summarization time recorded was 17.000 seconds.

Observations

The system's performance in classifying articles was consistent, with minimal variation in classification times. NER tasks demonstrated good performance, with reasonable execution times, although some articles required more time due to variations in content complexity. Summarization tasks generally maintained acceptable response times, but certain articles with longer content required more time for summarization.

Appendix B

Appendix B of this report contains the datasets essential for fine-tuning models in Text Classification, Named Entity Recognition (NER) classification, and abstract summarization tasks. In addition to the datasets, Appendix B also incorporates screenshots of the AWS RDS PostgreSQL database.

Snapshot of AWS RDS PostgreSQL Database

The schema for this database can be found in this report under section 5.2.3, which offers a visual overview of the database structure and organization. This appendix also includes snapshots of various returned queries executed against the database.

Snapshot of Raw Article Table

article_id	article_url	article_title	news_content	timestamp
10	https://www.cnn.com/2023/09/24/business/wga-st...	Writers Guild and studios reach tentative deal...	Markets \n\n\n\nFear & Greed Index \n\n\n\n\n...	2023-09-25 04:36:32.742100+00:00
11	https://www.cnn.com/2023/09/24/business/wga-st...	Writers Guild and studios reach tentative deal...	Markets \n\n\n\nFear & Greed Index \n\n\n\n\n...	2023-09-25 04:38:44.792324+00:00
14	https://www.cnn.com/2023/09/24/business/wga-st...	Writers Guild and studios reach tentative deal...	Markets \n\n\n\nFear & Greed Index \n\n\n\n\n...	2023-09-25 04:49:28.455779+00:00
15	https://www.cnn.com/2023/09/24/entertainment/t...	Taylor Swift cheers on Travis Kelce at Kansas ...	\n It's a love story, perhaps, for NFL t...	2023-09-25 04:59:17.459335+00:00
16	https://www.cnn.com/2023/09/24/politics/cassid...	Cassidy Hutchinson defends herself in first po...	\nCassidy Hutchinson, the former Trump White ...	2023-09-25 05:13:29.018312+00:00

Snapshot of Article Category and Summarization Table

article_id	article_category	article_summary
10	Entertainment	Writers Guild of America has ratified a deal w...
11	Entertainment	Writers Guild of America has ratified a deal w...
14	Entertainment	Writers Guild of America has ratified a deal w...
15	Sports	Jason Kelce has said he was a "little bit hurt...
16	Entertainment	Former White House aide Cassidy Hutchinson has...
...	...	...

Snapshot of Article NER Table

ner_id	article_id	ner_type	entity_type
1248	66	LOC	Mexico
1249	66	LOC	Zacatecas
1250	66	LOC	Malpaso
4	14	ORG	Writers Guild of America
5	14	LOC	Hollywood

These visual aids not only contextualize the data environment but also serve as a practical reference for the database's application within the project. For ease of access and to support future research, a link to the folder containing these resources is provided below.

Folder Link

Appendix C

Appendix C of this report serves as a comprehensive repository of the project's technical and presentation materials. It includes a detailed collection of program artifacts and the complete source code related to all three tasks and application, ensuring a deeper understanding of the system's functionality. Additionally, this appendix contains the PowerPoint presentations that summarize the project's datasets, pre-processing, preparation, database, and performance of the models on all three tasks. Demonstration videos dynamically and practically showcase the project's features and capabilities. This multimedia appendix presents a holistic view of the project's development and operational nuances. A link to access the folder containing all these materials is provided below for easy reference and in-depth review.

Folder Link