

AN INTELLIGENT EMAIL RESPONSE SYSTEM (IERS)

by

ZILI LUO

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Master of Science

Queen's University
Kingston, Ontario, Canada
June 2023

Copyright © Zili Luo, 2023

Abstract

Email is one of the most common methods of official and personal communication to exchange information. For the administration department, dealing with hundreds of emails with the same type of inquiries or requests results in a huge operational overhead. In this work, we propose an intelligent email response system that performs three main steps: (i) automatically classify emails, (ii) detect questions or requests for information, and (iii) generate a response from a FAQ database or a knowledge base.

In the first stage, we explore email classification models. An email classification system should understand the topics in the email content for categorizing emails and indicating if an incoming email should be handled by the mailbox owner. Email categorization based on topics is a multi-label classification task. Most existing email categorization models perform binary classification to identify spam, phishing, or malware attacks. We propose a CNN-BiLSTM model for multi-class email classification. Our experiments show that compared to CNN (76.19%) and BiLSTM (61.9%) models, the CNN-BiLSTM (83.33%) model has much better performance.

The second stage in our pipeline addresses question detection. We explore applying machine learning (ML) approaches to the International Computer Science Institute (ICSI) Meeting Recorder Dialogue Act (MRDA) corpus dataset. In the dataset

with question marks, the Random Forest (RF) model and the CNN model give a 0.99 F1-score and 1.00 F1-score respectively while the Rule-based model gives a worse result (0.89 F1-score). If the question marks are removed the F1-score decreases to 0.85 for both RF and CNN models. Moreover, using parser tree graph embedding with Feather and GL2VEC fails to detect questions. Finally, using TF-IDF weighted GloVe embedding gives worse results compare to GloVe embedding for both RF and CNN models.

In response generation, we use the SOTA BART model on the Eli5 dataset as the baseline. We find using complete sections and including a feature selection section provides better results in response generation. Using complete sentences gives a 0.2929 F1-score compared to 0.2729 F1-score with the baseline. Moreover, using a pre-trained extractive QA model in feature selection gives the best performance (0.2932 F1-score).

Acknowledgments

This research is funded by BAM lab. Special thanks to my supervisor Professor Dr. Farhana Zulkernine for her guidance and great patience during each stage of this research and inspiration in this field and Professor Dr. Brian Harrington for explaining the detail of the ASKnet Algorithm and suggestions on implementation. Thanks to my family for supporting and encouraging me to complete this work.

Contents

Abstract	i
Acknowledgments	iii
Contents	iv
List of Tables	vii
List of Figures	viii
List of Algorithms	x
Chapter 1: Introduction	1
1.1 Motivation	1
1.2 Problem Description	2
1.2.1 Challenges	2
1.3 Research Objective	4
1.4 Proposed Solution	4
1.5 Contributions	5
1.6 Thesis Organization	6
Chapter 2: Background and Literature Review	7
2.1 Feature Extraction	8
2.1.1 POS and NER	8
2.1.2 Parser Tree	9
2.2 Text Encoding Methods	9
2.2.1 One-hot Encoding	9
2.2.2 Word Encoding	9
2.2.3 Contextualized embedding	10
2.2.4 TF-IDF	11
2.3 Machine Learning (ML) Techniques	11
2.3.1 Supervised Classical ML	11

2.3.2	Supervised Deep Learning	13
2.3.3	Unsupervised ML	14
2.4	Measurements	15
2.4.1	Accuracy and F-Measure	15
2.4.2	ROUGE	16
2.5	Literature Review	17
2.5.1	Email Classification	17
2.5.2	Related Work	18
2.6	Question Detection	19
2.7	Response Generation	21
2.8	Summary	25
Chapter 3: Email Extraction and Classification		26
3.1	Background	27
3.1.1	Email Classification	27
3.2	Email Extraction and Preprocessing	27
3.2.1	Limitations	28
3.2.2	Email Processing	28
3.3	Email Classification	30
3.3.1	Classification Models	30
3.3.2	BiLSTM model	34
3.3.3	Dataset	36
3.3.4	Experiment Setup	37
3.3.5	Training and Validation	37
3.3.6	Result and Discussion	39
3.4	Real World Use Case Scenario	40
3.5	Summary	42
Chapter 4: Question Detection		43
4.1	Dataset	44
4.2	NLP Pipeline	44
4.3	Using Part-Of-Speech Tags and Parser Tree Graph as Features For Question Detection	46
4.3.1	Transform Sentence Into a Parser Tree Graph	46
4.4	Experiments	48
4.4.1	Preprocessing	49
4.4.2	Model Implementation	50
4.4.3	Experimental Setup	52
4.4.4	Results and Discussion	53
4.5	Summary	56

Chapter 5: Response Generation	57
5.1 Background	57
5.1.1 Extractive Question-Answering with NQ	58
5.1.2 Extractive Question-Answering with SQuAD V2	58
5.1.3 Abstractive Question-Answering with ELI5	59
5.1.4 BERT, Sentence-BERT, and BART	59
5.2 Problem Description	61
5.2.1 Proposed Solution	64
5.3 Implementation	65
5.3.1 Datasets	66
5.3.2 Data Preprocessing	67
5.3.3 Deep Learning Model	67
5.4 Experiments	67
5.4.1 Experiment 1: Full Section vs Slices	68
5.4.2 Experiment 2: Feature Selection and Representation	68
5.4.3 Results	69
5.5 Future work with Local Knowledge Network	70
5.6 Summary	71
Chapter 6: Conclusion and Future Work	72
6.1 Future Work	74
6.1.1 Email Classification	74
6.1.2 Question Detection	74
6.1.3 Response Generation	74
Bibliography	76
Appendix A: List of all possible name abbreviations in Penn Treebank	86
Appendix B: Graph Constructor	89
B.1 Advantages	89
B.1.1 Merge Same Representation by Merging Nodes	89
B.2 Asknet Algorithm	90
B.3 Design and Implementation Detail	90
B.3.1 Transform OIE Tuple into a Hierarchical Tree Structure	91
B.3.2 Transform Tree to Graph	92
B.4 Spreading Activation	93
B.5 Merging Nodes and Graphs	97
B.6 Linearzation Challenges	99

List of Tables

3.1	Results on classification models	39
4.1	Categories in the ICSI MRDA Dataset	44
4.2	Python Libraries	46
4.3	Results of model performance on well-formatted data	53
4.4	Results of model performance on data without the question mark. RF refers to Random Forest and CNN refers to the Convolutional Neural Network model	54
5.1	Comparative performance of the different data preprocessing and embedding approaches.	70
A.1	Penn Treebank POS tagset	87
A.2	Penn Treebank syntactic tagset	88

List of Figures

1.1	A schematic diagram of the component modules of the proposed automated email response generation system.	4
2.1	Examples of GloVe Clusters A:Traffic related words, B: Countries . . .	10
2.2	Confusion Matrix	15
2.3	Example of meaning representation input as a structured database and its corresponding natural language expression	22
2.4	Multi-Document Input to Linearized Graph Color indicates coreference resolution.	24
3.1	Email Splitters A:Splitter 1, B: Splitter 2	29
3.2	CNN model structure	33
3.3	BiLSTM model structure	34
3.4	CNN-BiLSTM model structure	34
3.5	Hierarchical CNN-BiLSTM model structure	35
3.6	Example data for email classification	36
3.7	Accuracy curves during training process	38
3.8	Validation result of real-world data	41
4.1	NLP pipeline	45

4.2	Example of sentence to graph result Left: original parser tree from NER result, Right: generated graph representation from the parser tree.) SBAR: Clause introduced by subordinating conjunction or 0, IN: Preposition/subordinating conjunction, NP: Noun phase, PRP: Personal Pronoun, ADVP: Adverb phrase, RB: Adverb, VBP: Verb Phase, NNS: Noun, plural, VBG: Verb, gerund/present participle. . .	48
4.3	Preprocessed data	49
5.1	Example data from NQ database	58
5.2	ELI5 example	59
5.3	BERT, Sentence-BERT, and BART model structures	60
5.4	Token slices V.S. full section.	62
5.5	An overview of our response generation pipeline	64
5.6	Example of attention aid selection	69
B.1	Graphs generated using (A) ASKnet algorithm and (B) Fan et al.'s local graph algorithm from the sentence "In December, John decided to join the party."	91
B.2	Example OIE slice	92
B.3	Edge-like node example	96
B.4	Multiple representations in the same node	99
B.5	Rule-based linearization examples	100

List of Algorithms

1	Email Data preprocessing	31
2	Create a graph from tree string	47

Chapter 1

Introduction

Despite the emerging communication technology and the existence of a variety of meeting platforms, emails remain to be the most used communication method in all organizations. In 2019, 293.6 billion emails were sent and received each day, which is expected to exceed 347.3 billion daily by 2022 [44]. Despite the availability of a FAQ (Frequently Asked Questions) database, or information posted on websites, people often do not spend enough time searching for necessary information, and instead send emails to inquire about the information they need. With the growth of Machine Learning (ML) and Deep Learning (DL) techniques, intelligent systems can be built to offload some of this work from the administration to automatically categorize emails [5, 42, 32, 39, 65], detect questions in a piece of text or the intent of the sender of the text [15, 8], to generate answers from supporting documents or dialog [59, 30, 51].

1.1 Motivation

In most organizations, the administration section deals with many emails every day. Many of those emails contain one or more general inquiries or common requests. If

there exists an efficient and reliable algorithm to classify those emails, detect questions and inquiries, and generate a response by retrieving an answer from the FAQ automatically, then that would facilitate and expedite the administrative work. Especially at the university, many emails contain similar questions during the time of admission, the beginning of the term, exams, and graduation. An intelligent email response system can reduce the workload of answering these similar questions by categorizing emails based on the topic, detecting questions in emails (an email can contain multiple questions), and automatically generating responses using an information corpus or FAQ while administrative personnel can reply to the more critical emails.

1.2 Problem Description

Question detection is a well-known problem in Natural Language Processing (NLP). The general approach is to simply detect specific words that are commonly used to ask questions ('wh*' words), search for specific characters in sentences including question marks, or templates to look for defined patterns of text [59, 72]. These are efficient techniques but they perform poorly if a question mark is missing or the email contains typos (misspelled words) and incorrect grammar.

1.2.1 Challenges

There are several challenges in creating an intelligent and automated email response system. Emails can be written in different languages, in different structures, and received in different email management tools. Extracting the text from the various

email management tools requires different plug-ins. Similarly, generating and sending responses requires the response system to be integrated with the email system. Furthermore, one email can contain multiple queries and can be a spam email or a phishing email. Emails can also contain sensitive content, for example, personal information about the sender which must be handled carefully. An intelligent email response system should therefore take all of the above scenarios into consideration when generating an automated response. Another issue is to address possible grammar errors, inappropriate use of question marks or punctuation, and abbreviations in question detection tasks. Rule-based algorithms are weak in those situations.

Based on our study of the existing literature on email classification, question detection, and answer generation, we observe the following.

1. Email classification mainly addresses spam, and phishing detection, but does not categorize emails based on the intent of the email.
2. Intent detection tasks focus on the general predefined intents of the email such as request, delivery, positive or negative. [15]. They do not focus on finding the specific sentence that shows the intent or detects multiple different intents. Although rule-based methods give very high accuracy [59], they perform poorly in the presence of corrupted data, malformed sentences, and typing errors. which we discuss in the next section.
3. Answer generation tasks already achieve very high accuracy in finding answers given in matching context as the question [3, 66, 73]. However, if the answer is scattered across a document, or in multiple supporting documents, or not presented in the matching context as the question, the existing systems struggle to produce the correct.

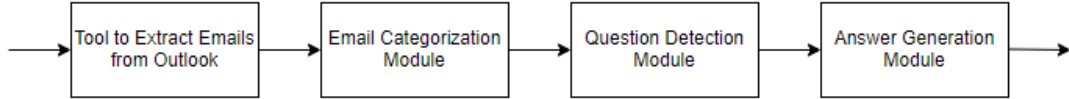


Figure 1.1: A schematic diagram of the component modules of the proposed automated email response generation system.

1.3 Research Objective

In this study we propose an intelligent email response system that addresses the following research objectives.

1. Identify specific aspects and intents of emails and categorize them appropriately.
2. Detect and extract questions from unstructured email text considering multiple questions may appear in the same email.
3. Generate a response for extracted questions.

1.4 Proposed Solution

We propose an intelligent email response system for organizations to reduce the mundane task of replying to frequent questions and wasting valuable time and resources to attend to more distinct and critical emails. The proposed system as shown in Figure 1.1 consists of multiple components as described below to address the above research problems.

1. We propose a CNN-BiLSTM combined model and a hierarchical CNN-BiLSTM model to handle the classification task to categorize emails based on topics

to facilitate answer generation. Due to the short length of emails, traditional LSTM models cannot provide satisfactory results. A more stable performance can be achieved by using a CNN to extract features in the text before applying LSTM. To ensure the process is fast, we do not use contextualize models such as BERT to generate embedding.

2. We propose a question detection algorithm by using more sentence features such as parts of speech tags, and sentence structure features such as parser trees to boost question detection performance. We develop machine learning methods that can be much more stable against corruption and mistyping.
3. We propose an answer generation algorithm using an abstractive question-answering model that can handle situations where the answer is scattered across the supporting document or multiple supporting documents, or the answer is not directly pointed out in the documents.

1.5 Contributions

In this research we:

1. Explore existing intelligent answer generation system ,email classification, question detection and finally, answer generation system.
2. We propose an intelligent answer generation system that combines:
 - (a) An email content extraction tool that can extract emails from Outlook with anonymous processing.
 - (b) An email classification tool to categorize emails based on their intents.

- (c) A question detection tool that is capable of detecting multiple questions in the email using parser tree structure as a graph.
- (d) An abstractive answer generation tool with the state-of-the-art model in ELI5 dataset [19] and different preprocessing methods.

The technical contributions of our research are summarized as follows.

1. We design and implement an LSTM-CNN model to solve multi-label email classification tasks with a view to determine if the email should be responded to or be forwarded to other authority.
2. A question detection algorithm which applies machine learning models and text-to-graph features. The algorithm is validated using the International Computer Science Institute (ICSI) Meeting Recorder Dialogue Act (MRDA) dataset [60] and specifically the text data features.
3. By exploring different preprocessing methods, we improve the state-of-the-art BART model by up to 10% in ROUGE-1, 2% in ROUGE-2 on the ELI5 dataset.

1.6 Thesis Organization

The remaining parts of the thesis are organized as follows.

Chapter 2 presents the background and literature review.

Our email extraction tool and the proposed email classification models using real-world email data are illustrated in Chapter 3.

Chapter 4 presents the question detection framework.

The response generation module is demonstrated in Chapter 5.

Chapter 6 summarizes our work and concludes the thesis.

Chapter 2

Background and Literature Review

Our Intelligent Email Response System (IERS) contains multiple components which apply Natural Language Processing (NLP) techniques to process the text data in the email content. The data can be tagged with Parts of Speech (POS) and Named Entity Recognition (NER) to extract informative content. Text data needs to be transformed into vectors to be fed into machine learning models for further processing as described below. We first discuss the concepts related to text feature extraction, encoding, and machine learning using the extracted features as input which are important and relevant for email classification, question and intent detection, and response generation. Later in this chapter, we present the literature review on the three key contributions namely email categorization, question detection, and response generation.

Existing IERS generally use templates which are filled to generate responses [1], or use retrieving based system to respond with pre-composed responses from a database by comparing string similarities [43]. Very little research exist on IERS in the literature, which calls for the need for a modern IERS with high efficiency and the ability to generate a response.

2.1 Feature Extraction

In NLP, data items are tagged, redundant or uninformative items are removed, and features are extracted from text or audio data. For text-based features, the most popular features are word-level embeddings as discussed in the next section. In this section, we discuss other feature extraction methods such as Term Frequency-Inverse Document Frequency (TF-IDF), Part of Speech (POS), Named Entity Recognition (NER), word n-grams, parser tree, and number of occurrence of words [8, 74, 4, 55, 32, 69, 58]. Audio features can include pitches, loudness, speaker changes, and length of utterance [8, 63].

2.1.1 POS and NER

POS tagging process labels each word in the text with its corresponding part of speech such as nouns, verbs, adjectives, and other grammatical categories. POS tagging is usually performed with ML algorithms such as Stanford NER tagger [20], or Spacy [28]. There are some popular POS tagsets, and each of them has its own set of tags and rules, such as Penn Treebank tagset [41] and Universal Dependencies tagset [13]. NER process helps identify proper nouns and names using a dictionary approach or rule based approach or both.

POS tagging is a useful tool in NLP since it describes the grammatical structure of a sentence and distinguishes words that have multiple meanings for the ML algorithms and models. POS is usually applied in Named Entity Recognition (NER), text classification, and Natural Language Generation (NLG).

2.1.2 Parser Tree

A parser tree represents the Generative Grammar of a sentence. In NLP, the parser tree can be used to determine the relationship between two terms in the sentence or represent the general structure of the sentence. Boakye et al. [8] show that the parser tree is a good feature to be used for detecting questions in the text.

2.2 Text Encoding Methods

All the text encoding methods convert each word to a simple number. The embedding methods convert words to vectors. The conversion is needed to transform words into a value that can be used in machine learning and other analytical processes. The three most common embedding methods are one-hot encoding, word embedding, and contextualized embedding [33]. There is also word frequency based encoding method called TF-IDF as described below.

2.2.1 One-hot Encoding

An advantage of one-hot encoding is its simplicity of implementation. Each data set is transformed into a vector of 0s and a single 1. It has two main drawbacks: it cannot deal with high cardinality of data and mapping of data is uninformed of the context. So semantically similar but syntactically different data are encoded differently and cannot be grouped together.

2.2.2 Word Encoding

In NLP, word embedding is a learned representation of words for text analysis. Word embedding, unlike one-hot encoding, results in words with similar meanings having

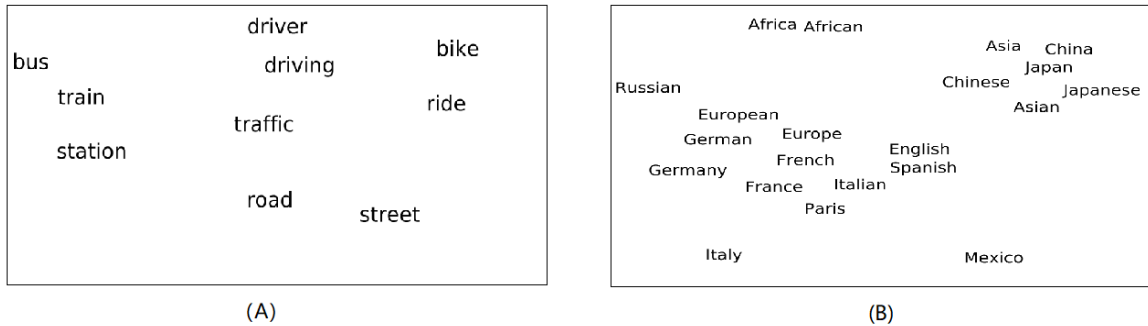


Figure 2.1: Examples of GloVe Clusters
 A: Traffic related words, B: Countries

similar vector representations, which suggests that words whose embedding vectors are close together in the vector space, have similar meanings *Figure 2.1*. Since word embedding maps the same word into the same vector, it is difficult to deal with polysemy (the same word having multiple meanings) and homonymy (different words having similar spelling or pronunciation but different meanings). The most popular word embedding method is GloVe [46].

2.2.3 Contextualized embedding

A contextualized embedding scheme is motivated by the need to accommodate multiple meanings per word and to map them into different vectors (multi-sense embeddings). In recent years, contextually-meaningful embedding models such as ELMo [47] and BERT [14] have been developed. To disambiguate polysemy, these embeddings use the context of a word. This maps words with similar meanings into similar vector representations, and into different embeddings when the same word has different meanings based on the context. The training of contextualized embedding models, however, requires significantly more time and resources than the training of the other two embedding methods: one-hot embedding and word embedding [48].

2.2.4 TF-IDF

TF-IDF is calculated by multiply TF and IDF as Equations 2.1 and 2.2 below.

$$TF = \frac{\text{Number of occurrence of the word in the document}}{\text{Total Number of words in the document}} \quad (2.1)$$

$$IDF = \log \frac{\text{Number of documents/sentences}}{\text{Number of documents/sentences containing the word term}} \quad (2.2)$$

TF-IDF is used to determine the importance of each word in the document, by comparing the word occurrence frequency within a specific document to its frequency in the entire corpus. The hypothesis behind this is a word that occurs more frequently is more important to the entire corpus [11].

2.3 Machine Learning (ML) Techniques

There are two commonly used ML techniques for email classification in the literature, namely supervised ML and unsupervised ML. Supervised ML approach is the most prevalent one which requires labeled data to train the models using error correction learning while unsupervised approach applies similarity calculation to group data into clusters [4].

2.3.1 Supervised Classical ML

a) Decision Tree (DT)

The DT is a commonly used supervised ML algorithm that can be applied for regression and classification. The DT is in a flowchart-like structure, where each node in the tree represents a "test" on an attribute.

b) Random Forest (RF)

The RF is an ensemble classifier that makes predictions by applying various DTs to different sub-samples in the dataset. Each DT in the RF is constructed by random selection of the best attributes.

c) Naive Bayes (NB)

The NB classifier utilizes the Bayes rule of conditional probability and all data features. The probabilities and features are individually analyzed based on the assumption that they are independent and have the same importance level. The NB classifier takes advantage of quick convergence and simplicity, while it is possible to understand the associations and interactions among the features of each sample.

d) Support Vector Machine (SVM)

The SVM is commonly used for classification and regression activities. SVM points each data item within it to an n -dimensional space, where n is the feature number of each data item. The SVM aims to maximize the gap between different classes when plotting data items to the n -dimensional space. The non-linearly separable data can be classified by SVM with the help of a kernel function that indicates the separating hyperspace.

e) Neural Networks (NN)

The NN contains a set of interconnected units (neurons) with weights. The NN is referred to as feed-forward since the information flows through a single direction and does not skip or flow back to any units. During training, the weights between units are updated, and the non-linearity provides the NN with

the power and the ability to learn complex mappings.

f) Linear Regression (LIR)

The LIR is a supervised ML technique. It carries out a regression task to model values to be predicted based on independent variables. It is commonly used in predicting and determining the link between data items.

g) K-Nearest Neighbors (KNN)

The KNN usually helps in classification, with the assumption that similar data items maintain close proximity. The KNN applies similarity measures such as Euclidean distance to check for the degree of similarity.

2.3.2 Supervised Deep Learning

Deep learning is an ML branch that uses multilayer Artificial Neural Networks (ANNs). Deep learning models have achieved very high accuracy in complicated problems such as text classification [74], extractive response generation [73, 67], and Sentiment Analysis [2]. Deep learning differs from classical ML methods with regards to the unique capacity to learn depictions from a variety of data types such as audio, video, text, or images by non-linear modeling of the data features which are extracted and refined through multiple data processing layers. With adaptable designs, Deep Neural Networks (DNN) can learn model parameters and the target outcomes straight from raw data through an iterative optimization process and improve their accuracy as more information is provided. The most often used deep learning models are CNNs and RNNs.

a) Convolutional Neural Network (CNN)

The CNN contains various convolutional layers with non-linear activation functions such as ReLU. The CNN carries out the convolution of the input to compute the output and provides the outcome of a local connection. For each layer in the CNN, there are many kernels of different sizes applied and their outputs are combined to attain the output. The CNN learns by updating filter values during the training phase.

b) Recurrent Neural Network (RNN)

The RNN learns hidden sequential associations in variable-length input sequences. However, the RNN has a long-term dependency issue, which could exacerbate the gradient exploding and vanishing issues. Long Short-Term Memory (LSTM) [27] applies polymorphism on the RNN, which utilizes gates on the input and recurrent input to influence the state and also the output at multiple intervals, to encounter the long-term dependency issues.

2.3.3 Unsupervised ML

a) K-Means Clustering

The K-Means Clustering is applied for partitioning or clustering a dataset into k groups. It first randomly selects k data points as cluster centroids from the dataset. Then it uses finite iterations to place the centroids at the centres of k densely located data clusters by calculating distances of the centroids from each data point in the corresponding cluster. The iterative training process continues until the centroids are positioned at the cluster centre with little repositioning required in subsequent cycles or epochs. When clustering a new word w in a text data, the word is assigned to the closest cluster center C_j , where $1 \leq j \leq k$.

2.4 Measurements

There are several measurement metrics that are commonly used in NLP tasks, such as accuracy, F-score, and Recall-Oriented Understudy for Gisting Evaluation (ROUGE).

2.4.1 Accuracy and F-Measure

Accuracy and F-Measure use True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) to calculate.

Truth Prediction	True	False
	True	TP
False	FN	TN

Figure 2.2: Confusion Matrix

As shown in Figure 2.2, TP indicates True is predicted as True, TN indicates False is predicted as False, FP indicates False is predicted as True, FN indicates True is predicted as False.

The Equation of Accuracy, Precision, Recall, and F1-score are listed below as 2.3, 2.4, 2.5, 2.6 respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.3)$$

$$Precision = \frac{TP}{TP + FP} \quad (2.4)$$

$$Recall = \frac{TP}{TP + FN} \quad (2.5)$$

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * TP}{2 * TP + FP + FN} \quad (2.6)$$

2.4.2 ROUGE

ROUGE measures the quality of generated context by counting and comparing the overlap of machine-generated context with the human-created context [40].

ROUGE Measures has five evaluations as listed below [68].

1. ROUGE-N: Overlap of n-grams.
2. ROUGE-L: Longest Common Subsequence (LCS) based statistics.
3. ROUGE-W: Weighted LCS-based statistics that favors consecutive LCSes.
4. ROUGE-S: Skip-bigram based co-occurrence statistics.
5. ROUGE-SU: Skip-bigram and unigram-based co-occurrence statistics.

ROUGE measures are commonly used in NLG tasks as it is focused on recall [38].

2.5 Literature Review

We discuss the most relevant work under three subsections based on the three key research contributions.

2.5.1 Email Classification

Email remains as the primary medium for official communication, despite a proliferation of applications for communication (e.g., LinkedIn, social media, and messaging systems). The number of emails sent and received each day in 2019 was 293.6 billion. In 2022, they are expected to exceed 347.3 billion per day [44]. A method of email classification is necessary in light of the extensive use of email.

There are several ways to classify emails, mostly phishing, spam, junk, and legit emails [5, 32, 42, 4, 55].

Taxonomy of Email Messages

One way to classify emails is to apply a rule-based filter or learning-based filter to email messages [4, 56, 55].

A rule-based filter usually focuses on the features as listed below appearing in different parts of the emails [55].

1. Email body: The features that are extracted from the email body such as shapes, HTML, hyperlinks, and specific words or phrases. These features are usually in a binary form to indicate presence or absence of the words in the email.
2. Subject: Data features that are extracted from the email subject to indicate

if the email is a reference to a previous email, or intent of the email such as confirm or request.

3. Script: Data features that indicate the source platform of the email such as JavaScript, pop-up window codes, on-click activities, and other scripts in the email.
4. Sender: Extracted information about the sender of the email such as the email address of the sender.

2.5.2 Related Work

Michailoff et al. [42] apply the NB classifier and Deep Neural Network (DNN) to classify emails under different root folders such as Mobile and Fixed telephony, and other. The result shows that the DNN (84.2% Accuracy) performs better than the NB classifier (76.21% Accuracy) but it is ($2.5215 * 10^{11}$ complexity) more compute-intensive than the NB classifier ($1.28 * 10^7$ complexity).

Yasin et al. [69] compare performances of the RF (0.991 F1-score), the NB classifier (0.945 F1-score), the SVM (0.969 F1-score), the NN (0.977 F1-score), and the DT (0.984 F1-score) using F-measure for detecting phishing emails. The result shows that the RF achieves the best performance.

Zhang et al. [74] adopt CNN at the character level to classify news by its context. They compared character-level CNN with LSTM, Bag of Words (BOW), TF-IDF, and word n-gram methods using errors in classification. The character level CNN achieves the lowest errors in one half of the dataset while word n-gram achieves the lowest error in the other half. They state that character-level CNN is an effective method for text classification as it does not require words to work.

Kiritchenko et al. [32] improves the learning speed of the SVM (94.01% Accuracy) and the NB (91.51% Accuracy) classifier by applying a co-training algorithm. The algorithm trains two classifiers based on two different sets of features and each time one model only adds the most confident prediction to the class. They report that the SVM performs very well with the co-training algorithm while the NB classifier performs very poorly.

Alsmadi et al. [5] use three different SVMs to detect spam emails. Each of SVM is formulated on the top 100 frequent words, the top 100 frequent words after removing stop words, and N-Gram terms. They report that all three SVMs show a very high True Positive(TP) rate while the SVM with N-Gram shows the best False Positive Rate.

2.6 Question Detection

Compared to email classification problems, question detection problems present more challenges. It is a common practice to use rule-based expressions in the question detection domain since sentences posing questions always share some characteristics, such as wh-words and question marks. It is true that rule-based methods perform well with well-formatted data, but they do not perform well with corrupted data [8, 64]. Moreover, existing research on question sentences mostly focused on question classification [64], which tries to classify question sentences into different question types instead of detecting if a sentence is a question. Or using acoustic features instead of text features [63].

A semi-rule-based method can also be used to handle random data. Zafari et al. [72] present an intelligence text processing tool for medical data which can highlight

important keywords by calculating a score based on different data features in the chat data from an online medical advising service using cTAKES [57]. cTAKES is an NLP system for the extraction of information from electronic medical records clinical free-text.

The part of speech (POS) tags are utilized by Yu, D et al. [70] in order to detect questions. By modifying a vanilla CNN network, they were able to calculate normal word embeddings as well as POS tags. They proposed three different models, PCNN-CON, PCNN-MAT, and PCNN-TEN, which use both embeddings differently. The vanilla CNN model is considered to be the baseline model. According to their result, adding POS tags as a feature does not result in a significant improvement in performance.

Tang et al.[63] build different RNN based on Gated Recurrent Unit (GRU). they use GRU, Bi-directional GRU, and Double Bi-directional GRU with acoustic features with the hypothesis that pitch carries the major information of the question.

Tayyar et al. [64] propose a rule-based question detection system that achieves 97.2% Accuracy which is close to 6% improvement to the previous SOTA of 91.6%. They extract parser tree structures, looking for specific words and structure strings to detect questions. They state that combining the rule-based system with ML algorithms can potentially further increase accuracy. Failure cases can be found due to mistyping, polymorphism of words such as round and rounded, and rule-failure to specific cases.

Additionally, Boakye [8] presents a comparison of different text features such as word n-grams, Part Of Speech (POS), and parser tree. The word n-grams feature and the parse tree feature score the highest in F1-score (67.48 F1-score and 66.78

F1-score), while the POS tag feature scores significantly lower (50.11 F1-score).

2.7 Response Generation

Google has compiled a dataset of Natural Question (NQ) [35] that contains questions posed by real-world individuals and their answers can be found on Wikipedia. The benchmark requires two types of answers: a short answer and a long answer. As an example, if the question is “Where is the world’s largest ice sheet located today?”, the short answer would be “Antarctica”. The long answer would be as follows. “The Antarctic ice sheet is the largest single mass of ice on Earth. It covers an area of almost 14 million km and contains 30 million km of ice. Around 90 % of the Earth’s ice mass is in Antarctica which, if melted, would cause sea levels to rise by 58 meters. The continent-wide average surface temperature trend of Antarctica is positive and significant at >0.05 °C / decade since 1957 .”

The NQ benchmark contains a number of models that perform well. In the long answer question, poolingformer [73] achieves state-of-the-art performance. This model revises the full self-attention approach into a two-level attention model, slide-window attention, and pooling attention. The slide-window approach was previously proposed by Alberti et al. [3]. Poolingformer’s slide-window attention applied to the long document only attends to neighborhood tokens within the window, whereas second-level attention increases the window size and computes attention weights based on pooled key and value matrices.

ReflectionNet-ensemble model [67], another state-of-the-art model for the short answer task, adopts the slide-window and two-step approach with the distinction of using two transformer models: a machine reading comprehension (MRC) model as

	Action1	Object1	Value1	Value2	Action2	Object2	Value21	Value22
Meaning Representation	Confirm	Person	100	200	Request	Data	Null	Null
Text	与会人数在100人到200人之间，请问您在哪天开会？ (The number of participants is between 100 and 200, when is the meeting scheduled?)							

Figure 2.3: Example of meaning representation input as a structured database and its corresponding natural language expression

well as a reflection model. It is based on a pre-trained transformer that can answer all types of questions in the NQ dataset (multi-span answer, no answer, and yes/no answer). Based on the output of the MRC model, the Reflection model calculates a confidence score. The MRC model’s role is to predict the right answer, while the reflection model computes the confidence score of that answer in order to verify whether the answer is correct. Accordingly, the reflection model utilizes a two-step prediction procedure, in which the first step addresses no-answer cases, either by determining the correct answer or by generating a no-answer response, while the second step determines yes/no questions for generating yes/no responses.

Yuan et al. [71] suggest that some of the generation tasks are based on utilizing predefined parser trees or grammars, while others are based on abstract question-answering tasks. They propose an algorithm for producing responses based on probabilistic context-free grammars (PCFGs). A template of answer parser trees and labeled semantic representation-text pairs were used to train their model. There are several fields in each semantic representation, and each field has a value (Figure 2.3). In order to generate a response, the PCFG parser attempts to learn how to extract the semantic representations from the text. As a result of the model, the following

response can be generated:“ 初步定在北京,好的 (scheduled in Beijing, right)” (1-best model) and “ 会议将在北京召开,对吗 ? (The meeting will be held in Beijing, right?)” (k-best model with k=20), based on their research, parser tree and grammar have a robust impact on the generation of a response to a question.

In another study by Du et al. [16], words are generated according to the order in which they appear in the lexicalized PCFG (L-PCFG) parse tree in left to right depth-first order. In order to remove unary rules, they are using Stanford NER tagger to obtain the initial parser tree and then applying content-based lexicalization to it. The model is designed in accordance with three requirements: determining the structure of children, determining the heir of a node, and sequentializing a tree. Research findings indicate that lexicalized grammars outperform unlexicalized grammars, and dependency-based lexicalization performs best when adhering to personas. There is also another research by Ford et al. [21] that follows a two-step approach by generating templates and then filling in the blanks using a translation model.

In the machine learning field, some studies apply transformer models to generate a natural response. Fan et al. [18] use an open information extraction model [62] to shorten sentences to “who does what to whom at when where how” format and merge similar units together by using a coreference resolution model [37] to construct a graph to find potential relations between different sentences within different documents. Then they linearize the graph back to a new sentence and use a transformer model with top-k attention to generate an answer using the abstractive approach as shown in Figure 2.4.

Lewis et al. introduced a transformer model called the BART model [38], which

Question : What is Albert Einstein famous for?

Document 1:

Albert Einstein, German-born theoretical physicist, best known for developing the theory of relativity.

.....

The theory of relativity is one of the two pillars of the modern physics.

.....

He won the physics Nobel Prize

Document 2:

Albert Einstein developed the theory of relativity.

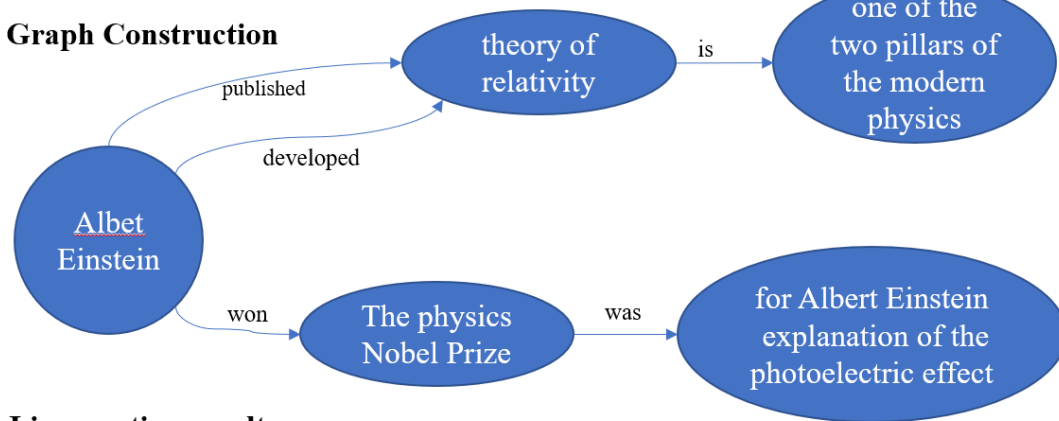
.....

He won the Nobel Prize

.....

The great prize was for his explanation of the photoelectric effect.

Graph Construction



Linearization result

- 1.<sub> Albert Einstein<obj> theory of relativity<pred>published<s>developed<obj>the physics Nobel Prize<s>won
- 2.<sub> theory of relativity<obj>one of the two pillars of the modern physics<pred>is
- 3.<sub> the physics Nobel Prize<obj> for Albert Einstein explanation of the photoelectric effect<pred> was

Figure 2.4: Multi-Document Input to Linearized Graph

Color indicates coreference resolution.

combines the encoder from the Bidirectional Encoder Representations from Transformers (BERT) [14] model and the decoder from the Generative Pre-trained Transformer (GPT) model to generate, translate, and comprehend natural language more efficiently. This encoder-decoder structure can be used to predict an arbitrary sequence. In generative tasks, the use of GPT as the decoder enhances the performance

of the BART model.

ChatGPT [45] based on the GPT-3 model introduced by OpenAI can produce high-quality human-like answers. However, it cannot distinguish whether the source information is accurate and so occasionally provides outdated or fake answers, which also raises ethical concerns [12].

2.8 Summary

Prior work has been conducted on email classification, question and intent detection, and answer generation, but the challenges depend on the complexity of the question and the difficulty of matching the context and semantics of the question to the answers in a text corpus. A complex research problem is the presentation of the answers, which may range from a simple yes/no to a paragraph containing extracted sentences from the knowledge base. It becomes increasingly difficult when the problem is to generate a short, to-the-point custom response based on the structure and semantics of the question, especially if the question is a multi-part question requiring extraction of answer components from multiple knowledge bases. Considering the context of generating an email response, we were interested in exploring the research domain of question answering.

In the following chapters, we present our approach to categorizing emails and extracting email data for testing our system.

Chapter 3

Email Extraction and Classification

As shown in Figure 1.1, we propose an Intelligent Email Response System (IERS) that has three major components in the pipeline. The first challenge is to extract emails from the email application and apply anonymization. Next, retrieved emails need to be classified into different categories, and the more specific those categories are, the better the emails can be processed or forwarded to the appropriate authority.

This chapter presents our algorithm for retrieving full conversation dialogues from email inboxes and replacing names of people with a random names from a pre-defined name database in order to make the email anonymous. Then, we discuss possible solutions for email classification based on CNN, BiLSTM, and CNN-BiLSTM models with experimental details and validations. Finally, a real-world example is presented that demonstrates our model and tool, along with the validation results.

3.1 Background

3.1.1 Email Classification

Historically, email classification algorithms were based primarily on probabilistic methods, with only two categories required: spam and non-spam [5]. Machine learning methods perform better than traditional probabilistic methods and have become increasingly popular in recent years. In the word semantic vector space, a word embedding scheme such as GloVe [46] enables computers to understand natural language by linking a word or character to its corresponding vector (usually unique). A computer could use this vector to understand and calculate the meaning and correlation between words, find the nearest neighbors (synonyms), and visualize concepts and relationships between words and characters. It is possible to classify emails into multiple categories using the vectors as inputs to machine learning and deep learning models.

3.2 Email Extraction and Preprocessing

We develop a tool for extracting emails from a popular email management system such as Outlook to extract and use real emails to validate our proposed models.

The "pypff" library is used to extract emails from the Outlook email data file (.pst) and export them to a CSV file for further processing. This way the tool can run locally on the users' machine which would extract, anonymize, and package the data to send to our proposed IERS that runs locally or on a back-end server. The code can be packaged as an executable file to transfer and deploy on the user's machine. The entire system uses "os.outlook" library since it can process email internally and

is able to send responses immediately.

3.2.1 Limitations

The tool we developed can only handle .pst and .ost files from the Outlook application and currently process only the English language.

The email extraction process can be extended to other languages with modifications to the regex, using different splitters and NER models matching the target language.

3.2.2 Email Processing

The data is stored in Pandas data frame format after being extracted from the Outlook email data file or directly from Outlook.

Processing Subject Line:

We remove "re:" from the subject of the conversation.

Cleaning Email Body:

The email body is processed using five regex expressions:

1. Remove embedded hyperlinks
2. Remove unembedded hyperlinks
3. Remove redundant newline
4. Remove any invalid characters (anything other than alphabets, numbers, @, commas, colons, and dots)
5. Remove redundant spaces

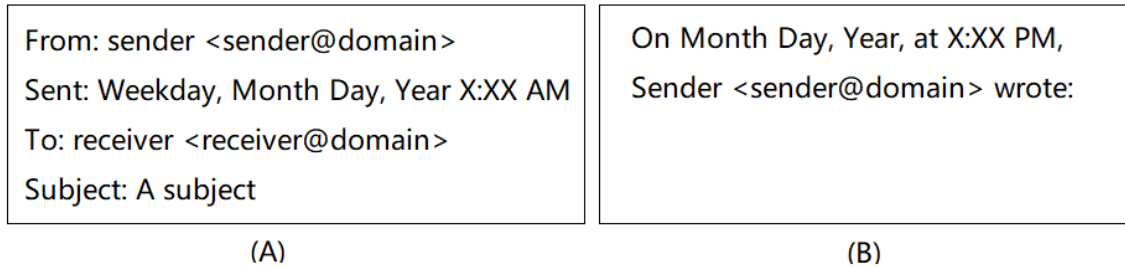


Figure 3.1: Email Splitters
 A: Splitter 1, B: Splitter 2

Extracting Complete Conversation from Multiple Emails:

Replies of emails often include the content from the previous email between two parties. To extract conversations between the same parties from subsequent emails, we select emails having the same subject line, and then extract the longest email, which contains all the dialogues denoting a complete conversation between two parties.

Splitting Conversation:

Next, we split different topics of conversation based on the context with two splitters as shown in Figure 3.1.

1. Email header (email header contains 4 new lines with “property:” and ending in “subject: subject of the whole conversation”)
2. On [date] sender wrote:

A description of our algorithm is shown in Algorithm 1. Each conversation is assigned a unique “conversation_index” property that indicates whether an email with the same subject belongs to the same conversation.

Anonymization:

Anonymization of email text is done using Stanford named entity recognition tagger (Stanford NER tagger) [20], which can identify words that have the “person” property. A temporal dictionary is established for each conversation to store the occurrence of the same name in that conversation. The names are then replaced with random names from a name database. In the case the NER tagger fails to detect all the names, we apply a backtrack step that searches the email text for the same word in the temporal name dictionary and replaces it. It is possible to ensure that all the same names in the conversation will be replaced in the same way by utilizing the longest email in the conversation. However, NER tags cannot recognize non-English names or names within sentence fragments.

The processed data can be exported as a CSV file or stored in memory for the next processing step as described below.

3.3 Email Classification

We develop and validate several classification models to determine the categories of emails after they have been extracted.

3.3.1 Classification Models

The recent Convolutional Neural Networks (CNNs) ConvNets [74] achieved superior performance in topic classification when applied at the character level. In Natural Language Processing (NLP), the Long Short-Term Memory (LSTM) [27] model has demonstrated superior performance because it can not only extract data features from the input text but can also learn long sequential patterns in the text. Researchers have reported that CNN-based NLP models can also be used to classify documents,

Algorithm 1 Email Data preprocessing

Input : A pandas data frame (DF_in) contains all the data from outlook data file**Output** : A pandas data frame (DF_r) with emails in each conversation split down

- 1: Remove all re: in the subject of the input data frame DF_in
 - 2: Create an empty data frame DF_r to store the result
 - 3: Initialize a subject index $i = 0$
 - 4: **for** each subject S in data frame DF_in : **do**
 - 5: Create a data frame DF containing all emails with that subject sorted by DateTime while the top-most email is the latest email under that subject.
 - 6: **while** DF is not empty **do**:
 - 7: Get the first email FE in DF
 - 8: Split FE by the email splitters mentioned in Section 3.2.2 to get a list $dialogue_list$ that contains each dialogue in this conversation.
 - 9: Create a temporary dictionary $nameDict$ to store names
 - 10: Processing each dialogue of the conversion with NER tagger, Replace all words with person entity with a random name. Depending on the length of continuous existence, will be replaced with first name (middle name) (last name) in the $nameDict$, if the name replace pair does not exist, replace with random and store the origin name: replaced name pair into the dictionary
 - 11: Append each dialogue to the DF_r with the subject index i
 - 12: Remove other email conversations that contain the same dialogues in $dialogue_list$ in DF
 - 13: **end while**
 - 14: $i = i + 1$
 - 15: **end for**
 - 16: Return the result data frame DF_r
-

including emails, and achieve competitive results. But CNN has a parallel structure that will ignore the sequence information from the input text. Therefore, we combine these two models into a CNN-BiLSTM model that can perform better in multi-label topic classification. CNN performs well on concise email data, but can not process long sequences, where LSTM assists in tracing sequence information.

Model Configuration

We use the categorical cross-entropy loss as equation 3.1 as the loss function and use the Adam optimizer as the model optimizer. The learning rate for all the models is set to 3e-4 to avoid the model converging too fast to a sub-optimal solution, except for the Hierarchical CNN-BiLSTM which uses a decaying learning rate. All the layers in the model have a dropout of 0.2, and all the dense layers in the model have a bias. All the BiLSTM layers use the Tanh activation function as equation 3.2 All the dense layers except the last before output use the Relu activation function as shown in equation 3.3. The last dense layer before the output uses the Softmax activation function as shown in equation 3.4.

$$-\sum_{c=1}^M y_{o,c} \log(p_{o,c}) \quad (3.1)$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad (3.2)$$

$$\text{Relu}(z) = \max(0, z) \quad (3.3)$$

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad \text{for } i = 1, 2, \dots, K \quad (3.4)$$

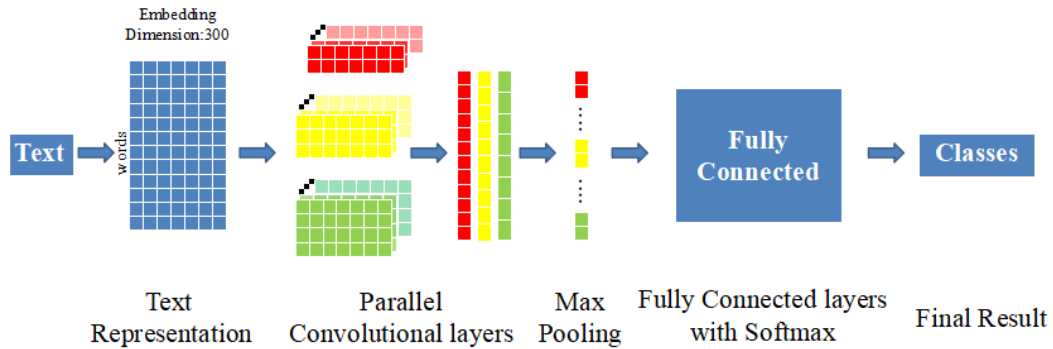


Figure 3.2: CNN model structure

CNN model

Rather than using a sequential architecture, CNN on the word level uses a parallel architecture similar to that used for image processing. By applying different filter sizes to the text representation, CNN models are able to extract features of varying lengths from the subtext of the email content. Following the max-pooling layer, these features reconstruct a new embedding of the entire email content to feed to the fully connected layer. There are 100 filters of each size of 3, 4, and 5. Therefore, the embedding feed to a fully connected layer has a dimension of 300. As a result, there are three fully connected layers with dimensions of (300, 128), (128, 32), and (32, 14) respectively. All fully-connected layers except the last layer are activated using the ReLu activation function. All dense layers have a bias. A visualization of the model structure can be found in Figure 3.2.

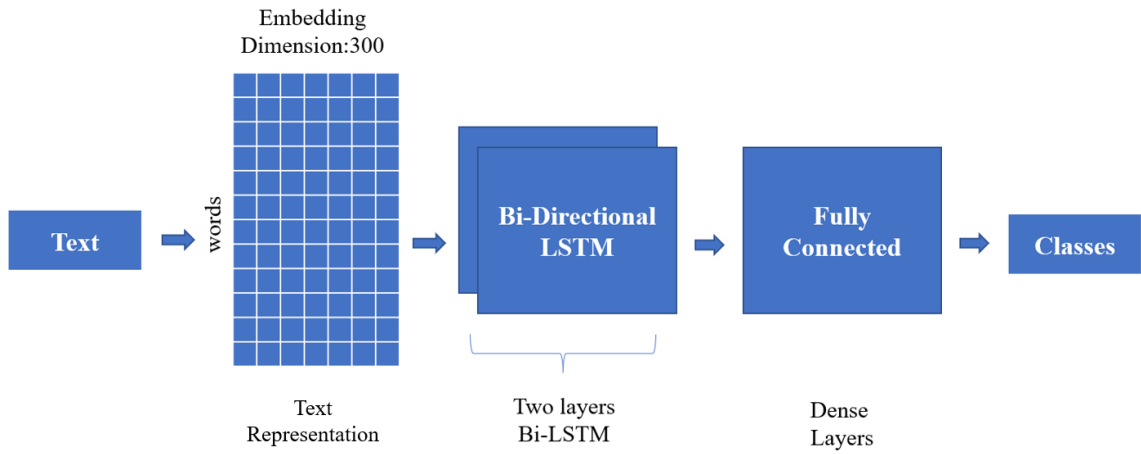


Figure 3.3: BiLSTM model structure

3.3.2 BiLSTM model

To compare the performance, we also implement a BiLSTM model as shown in Figure 3.3. In this model, we use two layers of BiLSTM with a hidden size of 64 for each BiLSTM layer. There are two Fully connected layers before prediction with sizes of (128,32) and (32,14) respectively.

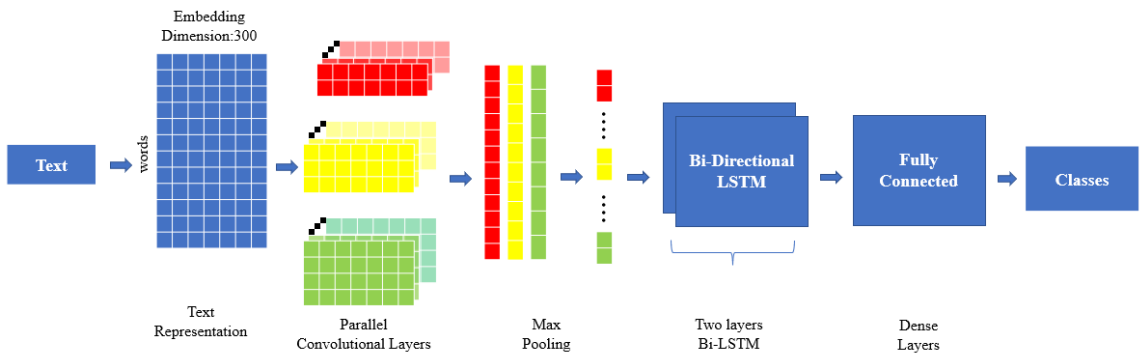


Figure 3.4: CNN-BiLSTM model structure

CNN-BiLSTM model

In the CNN model, we add BiLSTM layers between the max-pooling layer and the fully connected dense layer. BiLSTM models are fed directly with data from the max-pooling layer. In this study, two BiLSTM layers with a hidden layer of size 64 were used to avoid over-fitting because the dataset is quite small. In the CNN-BiLSTM model, the BiLSTM replaces one of the dense layers with an input size of 300 and an output size of 128. A visualized model structure is shown in Figure 3.4.



Figure 3.5: Hierarchical CNN-BiLSTM model structure

Hierarchical CNN-BiLSTM model

As the dataset is structured hierarchically, we apply a hierarchical classification model containing two parallel CNN-BiLSTM branches inspired by Kolisnik et al. [34], we have modified our CNN-BiLSTM model into a hierarchical model, which will attempt to classify the main-category and subcategories simultaneously. One CNN-BiLSTM

Text_Of_Email	Category	Sub-Category
Dear Sir/Ma'am, The wifi in my room is not working. I live in Room 7 of SH-3. Please help me with this. Thanks, XYZ	ITD	WFO
Dear All, I can't access the wifi through my phone. Please help. With best regards, Adit	ITD	WFO
Dear Sir, I can't access the wifi from my laptop. Please help me fix this. Best, Himanshu	ITD	WFO
Dear Sir/Ma'am, The wifi has been very erratic on my device. I need this urgently. Please help me fix this. Regards, Deb	ITD	WFO
My internet has not been working properly for the past few days. Please solve the issue. Thanking you, Jai	ITD	WFO
Hi, I stay on the 6th floor of SH-2 residence. I have not been able to connect to the wifi for 2 hours now. Can you please help fix this asap. Thank you Gauri R	ITD	WFO

Figure 3.6: Example data for email classification

branch (teacher branch) predicts the main-category in the same manner as the combined model (main pipeline) while the other CNN-BiLSTM pipeline incorporates input from the hidden state of the BiLSTM layer of the main pipeline into the hidden state of the BiLSTM layer in this pipeline. The hierarchical structure of the CNN-BiLSTM model is shown in Figure 3.5.

We use a decaying weight w to control the bias of the loss function. Its initial value is 0.99, decays by 0.03 for each epoch, and has a minimum value of 0.1.

3.3.3 Dataset

We use the dataset proposed by Singh et al. [61] in this study as Dataset_1. The database contains 255 emails which are divided into three main-categories and 14 sub-categories separating emails from an administrative section of a university, with nearly equal numbers of emails in each subcategory. For example, an IT department receiving requests for assistance with computer problems will have the department name as the main-category and the specific request type such as a clicker problem or a WiFi outage as a sub-category. Dataset_1 contains only the email body. Other information such as the subject, time, and sender is not included. The labels indicate the recipient of the email and the request type. A few examples from Dataset_1 are shown in Figure 3.6.

Since Dataset_1 only contains the email body, we do not apply our Algorithm 1 described in Section 3.2.2. However, we apply our algorithm to a real-life dataset,

Dataset_2, as explained later in this chapter. We created Dataset_2 for testing our method for a real-life use case scenario.

3.3.4 Experiment Setup

The email texts are padded with pad tokens to make all the emails have the same maximum length of 128 tokens. The text is tokenized using a tokenizer from the NLTK library [7], and each token is converted into a vector embedding of 300 dimensions using the GloVe library [46].

We select the same number of samples from each category as the testing set, which makes the leftover training data unbalanced. Thus, We use SMOTE (Synthetic Minority Oversampling Technique) library [9], on the training Dataset_2 to upsample and balancing the training set.

3.3.5 Training and Validation

A portion of the same Dataset_2 is used to test the models. In the training process, the testing subset is not used. The data was split into training and testing data at a 80-20 ratio before training. We tested other ratios, but increasing the testing set size would result in performance loss. For the CNN, BiLSTM, CNN-BiLSTM, and hierarchical CNN-BiLSTM models, training and testing were conducted. For the BiLSTM model, CNN and pooling layers were removed and the embedding was fed directly into the BiLSTM model. Each model except the BiLSTM is trained for 50 epochs while the BiLSTM model is trained for 100 epochs. We measure the performance of the model using accuracy as our validation method, the equation is

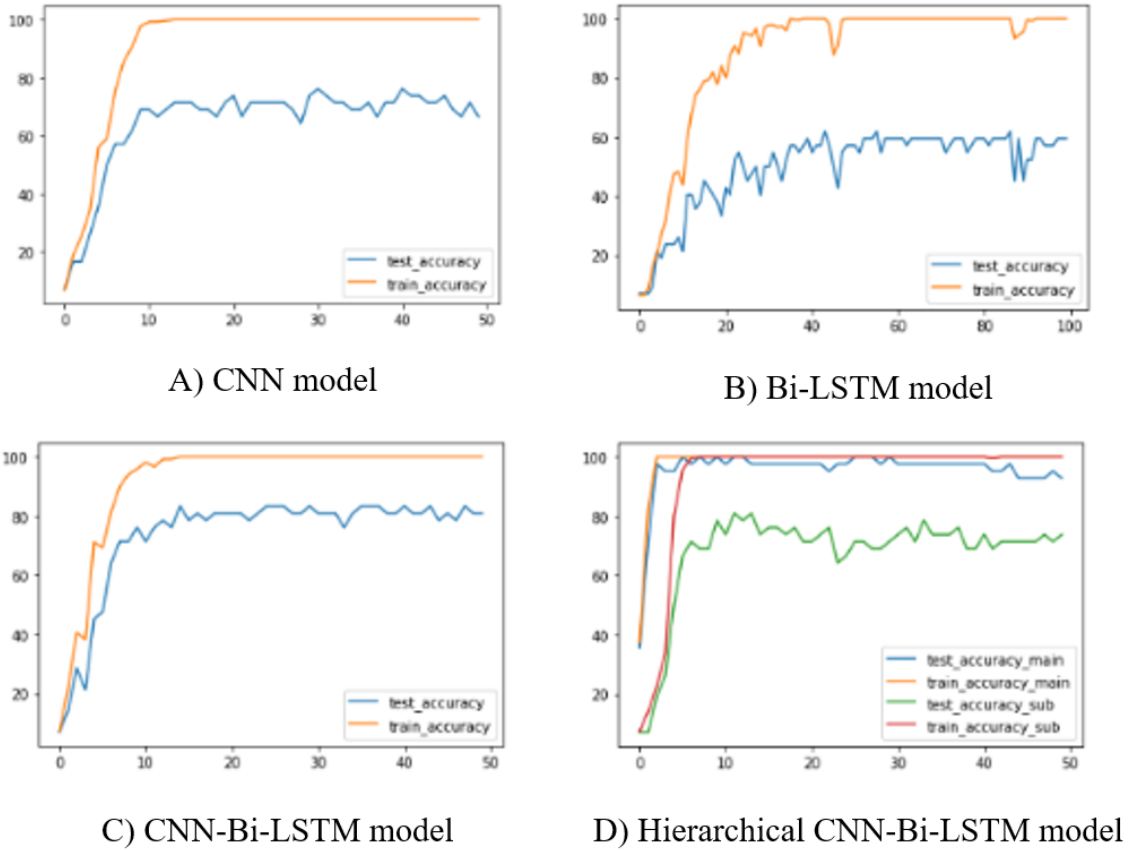


Figure 3.7: Accuracy curves during training process

shown in equation 3.5.

$$[H]Accuracy = \frac{Number\ of\ correct\ predictions}{Number\ of\ total\ predictions} \quad (3.5)$$

Our study compares the performance of LSTM and Bidirectional LSTM, with one LSTM layer and two LSTM layers, as well as two Bidirectional LSTM giving the best results. Our training curve in Figure 3.7 for CNN and CNN-BiLSTM models indicates that, in general, CNNs and CNN-BiLSTMs converge very quickly, reaching the best performance within 30 epochs, while the BiLSTM models have a slow convergence

Model	Training Accuracy	Testing Accuracy
CNN	100%	76.19%
Bidirectional-LSTM	99.2%	61.9%
CNN-BiLSTM	100%	83.33%
Hierarchical CNN-BiLSTM	100% main 100% sub	96% main 85.33% sub

Table 3.1: Results on classification models

speed, and require many more epochs to achieve optimal performance.

3.3.6 Result and Discussion

We run the trained model on the training data to obtain the training accuracy. From our results in Table 3.1, all the models have a huge gap between their training and testing accuracy, which might be due to overfitting issues. We also found that the LSTM model does not learn from the dataset under some conditions, resulting in very low accuracy for both training and testing. We tested using the direct output and final hidden state, changing the number of layers, and the BiLSTM model only work on this dataset under the condition that BiLSTM’s final hidden states were used as the input to dense layers. We also tested the LSTM model, which gives 17% training accuracy and 13% testing accuracy. The bad performance of the BiLSTM model may be caused by the short input data or the embedding layers feeding embeddings in an unexpected manner.

Compared with CNN or BiLSTM alone, the combined model produces better

results. Additionally, we examine how SMOTE affects the training dataset; in general, SMOTE will result in approximately 40 more training examples, however, it will not have a significant impact on the result while the maximum length of each sentence is set to 128 characters. By setting the maximum length to a large number, SMOTE will result in even worse results.

We investigated the accuracy for each category, and it appears that the model always has difficulties with ADC (Adding Course), COF (Course Offered), and WFO (WiFi Outage). By reviewing the dataset, we discovered that most of the emails in the WFO category mention “WiFi”. Although a very small percentage in this category do not have this indicator, they have similar indicators to emails in other categories. The other two categories experience similar results.

In general, since the dataset is so small and has too many categories, there are only 18 emails in each category, and only 80% of them are used for training, it is possible that there will be no similarity in features between emails within the same category.

3.4 Real World Use Case Scenario

As a real-world application of the work, we construct a simple dataset as `dataset_2` by extracting emails from Outlook using the algorithm described in section 3.2 and divide the same into informative and non-informative emails. Emails that contain information about device maintenance, conference activities, meetings notice, and class broadcast are considered informative emails received from the Senior Systems Analyst, Graduate Program Assistant, and thesis Supervisor. In addition to informative emails, we label non-informative emails as those containing newsletters, event

invitations, and surveys from the school.

Dataset_2 contains 346 informative emails and 423 non-informative emails. Since Dataset_2 only contains two categories and is quite different from the dataset we used for model training, we propose two methods to validate the model as explained below.

1. We use our pre-trained CNN-BiLSTM model in Section 3.3.2 and set a threshold on softmax result. If the result is above the threshold we classify the email as informative and otherwise as non-informative.
2. We modify the architecture and retrain the model to test if the model can accurately classify this dataset.

	precision	recall	f1-score	support		precision	recall	f1-score	support
other	0.57	0.47	0.51	153	other	0.85	0.88	0.86	153
info	0.55	0.64	0.59	153	info	0.87	0.85	0.86	153
accuracy			0.56	306	accuracy			0.86	306
macro avg	0.56	0.56	0.55	306	macro avg	0.86	0.86	0.86	306
weighted avg	0.56	0.56	0.55	306	weighted avg	0.86	0.86	0.86	306

A) Validation result of scenario 1

B) Validation result of scenario 2

Figure 3.8: Validation result of real-world data

For validation 1, we fine-tuned our pre-trained CNN-BiLSTM model for another 50 epochs on our simple Dataset_2 using the same loss function, learning rate, and optimizer as stated in Section 3.3.1 on the real-world data. Then we try to find the best threshold by iterating the threshold value from 0 to 1 with an increment of 0.05 for each step. The model gives the best performance when the threshold is set to 0.45, which gives a 0.56 F1 score, the validation result is shown in Figure 3.8.A

With validation method 2, We change the shape of the final dense layer to (32,2) and train the model from scratch on Dataset_2 for 50 epochs using the same loss function, learning rate, and optimizer as stated in Section 3.3.1. The model gives 99.56% training accuracy and 86% testing accuracy, 0.86 F1-score. The validation result is shown in Figure 3.8.B

The huge difference in performance can be attributed to the different vocabulary sizes. The vocabulary size in Section 3.3.2 is only 1,441 while Dataset_2 in this section has a vocabulary size of 15,067. With the aid of our extraction algorithms, we validate that our model is capable of handling email classification tasks in real-life use-case situations.

3.5 Summary

This chapter discusses the implementation of an email extraction algorithm as well as our exploration, design, implementation, and validation of email classification models. As part of our system pipeline, these two components constitute the very first step, and they demonstrate the capability of our system to be applied in real-world situations.

The next chapter will present our exploration of several approaches to question detection, our hypothesis, and answer generation.

Chapter 4

Question Detection

After email categorization, some specific categories of emails can be selected to be handled automatically by a question-answering system. Specifically, emails with inquiries about procedures, dates, and personnel/contacts are often received by the administration, can be handled by an Intelligent automated Email System (IERS). The first step to answering questions is to detect the questions in emails. Although rule-based methods already perform very well with very low run-time [59, 36], they suffer from mistype and corruption of the text, spelling errors, absence of '?', 'wh' words, and ill-formatted sentences. In such situations, the rule-based system fails to detect questions. One email can also contain multiple questions or a mixed content of inquiries and other matters, which challenges pose additional the automated system.

This chapter presents the exploration and design of a question detection framework using sentence features and graphs structure to detect questions using machine learning approaches.

Dialogue Act	Labels	Count	%
Statement	S	64233	59.36
BackChannel	B	14620	13.51
Disruption	D	14518	13.45
FloorGrabber	F	7818	7.23
Question	Q	6983	6.45

Table 4.1: Categories in the ICSI MRDA Dataset

4.1 Dataset

We are using the International Computer Science Institute (ICSI) ICSI Meeting Recorder Dialogue Act (MRDA) dataset [60] from Boakye et al. [8] to detect questions in meeting conversations as the email benchmark dataset used in email classification is very small. The original data is in audio file format. We retrieve preprocessed plain text format of the data from a GitHub repository [17]. The dataset contains 108,202 sentences in total with the distribution as shown in Table 4.1. In this part of the study, we only focus on question identification. Therefore, we regroup the data into two categories: question, and non-question.

4.2 NLP Pipeline

The general NLP pipeline consists of a number of processing steps as shown in Figure 4.1. For example, if the sentence “because I really appreciate people coming.” is given as an input, the output would be a graph as shown in Figure 10. A graph embedding is generated with GL2vec [10] and Feather-G Embedding [54] from karateclub library [53], including a tokenized word list with the corresponding POS tags: [(“because”, ”IN”), (“i”, ”PRP”), (“really”, ”RB”), (“appreciate”, ”VBP”), (“people”, ”NNS”),

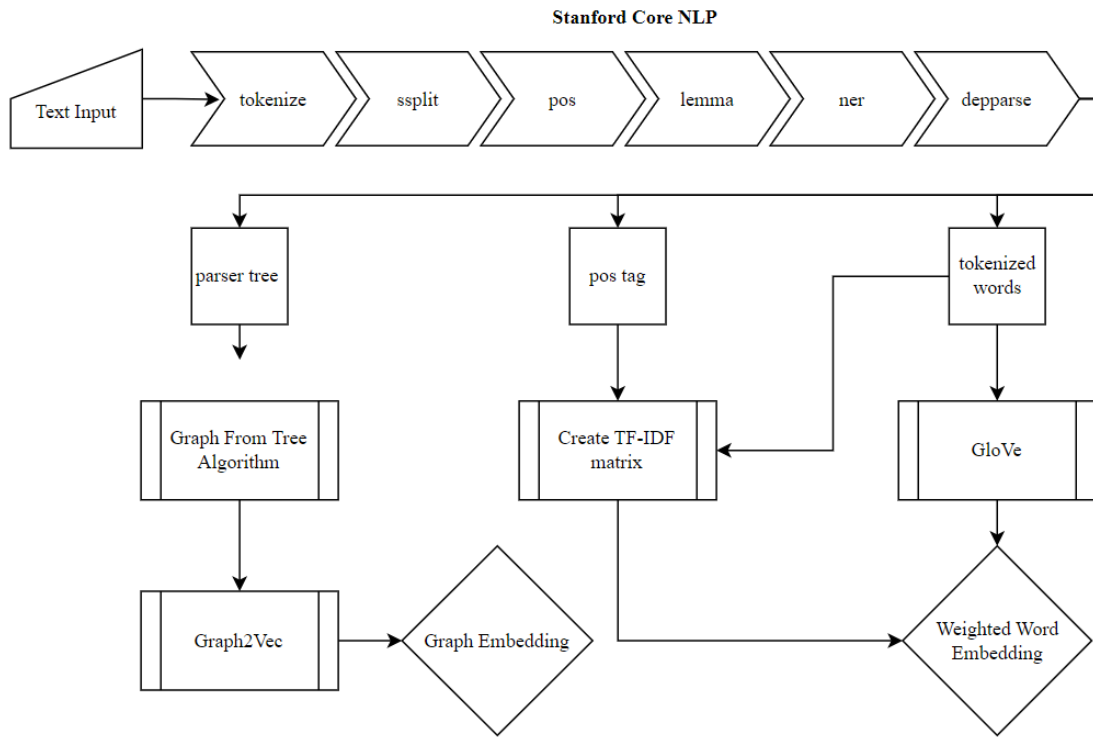


Figure 4.1: NLP pipeline

(“coming”, ”VBG”)], here IN, PRP, RB, VBP, NNS, and VBG stands for preposition, personal pronoun, adverb, verb (non-3rd ps, sing, present), noun (plural), and verb (gerybd/present participle) respectively. The list is then used to generate Term Frequency-Inverse Document Frequency (TF-IDF) matrix and GloVe word embeddings. By multiplying the matrix and word embeddings we created a TF-IDF weighted GloVe word embedding. We use the python libraries as shown in Table 4.2 in our pipeline.

Library	Version	Usage
NLTK	3.6.2	Tokenizer, parser tree generation
Karateclub	1.3.3	Graph embedding generation
Networkx	2.5.1	Graph generation
Pycorenlp	0.3.0	Named entity recognition

Table 4.2: Python Libraries

4.3 Using Part-Of-Speech Tags and Parser Tree Graph as Features For Question Detection

Yu et al. [70] utilize part-of-speech (POS) tags to extract more information from sentences and achieve better sentence classification results. Their experiment shows that involving POS tags of the sentence is an easy but powerful way to boost the model performance in text classification. Other research studies [8, 71, 16] also find the parser tree of the sentence as a useful feature to use in NLP tasks. Based on these evidences, instead of using a linear structure of a parser tree (e.g. parser tree string), we propose using a graph representation of a parser tree of the sentence to aid the model with question detection.

Each node in the tree is labeled with the NER tagger and then transformed into a graph embedding vector.

4.3.1 Transform Sentence Into a Parser Tree Graph

The graph is converted by the output string of Stanford NER tagger, convert to an NLTK parser tree first using constituency parsing, and then pass the tree through an algorithm to build the graph representation with the NetworkX library. We also remove the leaf node, which is the word itself, from the graph. The conversion is done by initializing an empty graph and adding each node from the parser tree into

the graph. We also establish the edges between the nodes by using the tree position attributes. Tree position is a list representing the node position in the tree, the length of this list for a node indicates the parent-child relationship as the length of the child is always one unit longer than the parent while the parent is a sub-list of its children. We use Algorithm 2 for the sentence-to-graph conversion and show an example in Figure 4.2. We include examples of graphs that contain all possible abbreviations for syntactic and POS tagsets resulting from the Stanford NER tagger in Appendix A.

Algorithm 2 Create a graph from tree string

Input : A tree string TS from Stanford NER tagger

Output : A graph G

- 1: Convert TS into NLTK parser tree PT
- 2: Create an empty graph G for output
- 3: **for** each node position in PT and the node position is not the leaves **do**
- 4: Add a node N to the graph and label it with the same label as in the parser tree with its tree position
- 5: **end for**
- 6: **for** each node N in graph if tree position of N == 1 **do**
- 7: Add an edge E from the root to N
- 8: **end for**
- 9: **for** $i = 1$ to $\max_position_length$ **do**
- 10: Parent \leftarrow tree positions with length equal to i
- 11: Possible_child \leftarrow tree positions with length equal to $i + 1$
- 12: Add an edge between P in parent and C in $Possible_child$ if $C[0:-1] == P$
- 13: **end for**
- 14: Return G

In Algorithm 2, the tree position is a list of strings that indicates the branch path of the node. As shown in Figure 4.2, the tree position of SBAR, IN, S is [0], [0,0], [0,1] respectively. In 12 of Algorithm 2, we take all but the last element from the child node's tree position string list which is slice [0:-1], and compare them with their possible parent's tree position string list. If those two lists are equal, it can conclude

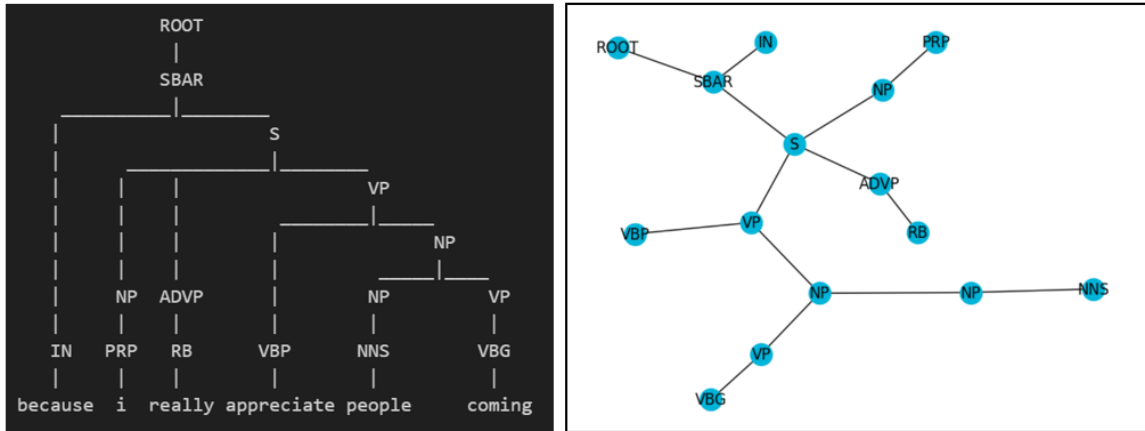


Figure 4.2: Example of sentence to graph result

Left: original parser tree from NER result, Right: generated graph representation from the parser tree.)

SBAR: Clause introduced by subordinating conjunction or 0, IN: Preposition/subordinating conjunction, NP: Noun phrase, PRP: Personal Pronoun, ADVP: Adverb phrase, RB: Adverb, VBP: Verb Phase, NNS: Noun, plural, VBG: Verb, gerund/present participle.

the child-parent relation.

4.4 Experiments

We conduct several experiments to validate the impact of POS tags and graph embedding of sentences on question detection with machine learning. We implemented two models with different approaches: The Random Forest (RF) model and the Convolutional Neural Network (CNN) model. For each model, we are trying to use text-only features, graph-only features, and a combination of them. We expect that with the TF-IDF weighted text embedding and graph embedding, the model would perform better than when using only text embeddings.

	Sentence	Question		Sentence_pos	treeString
0	okay	False		NN	(ROOT\^n (INTJ (UH okay)))
1	so um	False		RB JJ	(ROOT\^n (S (CC so)\^n (INTJ (UH um))))
2	i was going to try to get out of here like in ...	False	NN VBD VBG TO VB TO VB IN IN RB IN IN PDT DT NN		(ROOT\^n (S\^n (NP (PRP i))\^n (VP (...
3	um	False		NN	(ROOT\^n (INTJ (UH um)))
4	because i really appreciate people coming	False		IN NN RB JJ NNS VBG	(ROOT\^n (SBAR (IN because)\^n (S\^n ...

Figure 4.3: Preprocessed data

4.4.1 Preprocessing

Removing Duplicates and Balancing the Dataset

The dataset has many duplication short sentences such as “uh”, and “ah”. We removed all duplicates and reduced the number of sentences from 106,692 to 66,124. In our experiments, we treat labels such as statement, backchannel, disruption, floorgrabber in Table 4.1 as non-questions. The processed dataset has 5,178 sentences labeled as questions. To make sure the training and testing data are not highly unbalanced while retaining some portion of the whole dataset, we picked 15,534 sentences from the non-question group to form the dataset for our experiments which is around 33% of the processed dataset.

Annotating and Encoding

We performed the processing steps as shown in the NLP pipeline in Figure 4.1 and created a TF-IDF matrix for each word with their POS tags. Instead of using a single word (e.g., Appreciate), we attached their POS tags to them (e.g., appreciate/VBP) to indicate words that have different effects in sentences. For each sentence, we use the Stanford NLP library to generate a parser tree string which indicates the parser tree structure of that sentence. We show some examples of processed data in Figure 4.3. In this stage, we pad each sentence to 128 tokens.

Generate Embedding

For text embedding, we use GloVe [46] to generate the word embedding as done in our email classification task as described in section 3.3.4. We multiply the vector with the TF-IDF score of the same word to get their weighted embedding in each sentence based on Schmidt [58], which demonstrate that the TF-IDF weighted embedding can improve the model performance in a text classification task. For graph embedding, we use two different embedding methods GL2vec and Feather-G embedding as described in Section 4.2.

4.4.2 Model Implementation

We implement the Random Forest (RF) model as our baseline model since it is fast and gives high accuracy in classification tasks, and a simple rule-based model as a comparison. Next, we implement the CNN model, which is good at processing spatial features since we use graph representation, the CNN model can extract features from distributed graph nodes.

Simple rule-based model

The simple rule-based model uses two feature rules to determine if a sentence is a question.

1. In-text Features: wh-words and question marks.
2. Parser Tree Features:
 - (a) SBARQ for direct question introduced by a wh-word or a wh-phrase

- (b) SQ for inverted yes/no questions, or main clause of a wh-question, following the wh-phrase in SBARQ

The in-text features rule is applied to the input sentence directly, and the parser tree features rule is applied to the parser tree structure string. If either rule is satisfied, the sentence is classified as a question, otherwise the sentence is classified as non-question.

Baseline Random Forest Model

We implement an RF model with the NLTK library and set `n_estimator` to 10 and use it as our baseline model. This model is tested on the TF-IDF matrix on text, text embedding, graph embedding, and a combination of weighted text embedding and both graph embeddings. The purpose of this model is to check if those embedding methods could have a decent capability to classify texts.

CNN model

We also implement some CNN models for those embedding methods. Before we combine the embeddings of text and graph, we build two models to take one of the embeddings and combine them together to take both embeddings as input.

The text model is built on the Conv1D layer, after the max-pooling layer and flattening layer, the data is fed to the dense layers to get the predicted result. The Conv1D layer has 128 kernels with size 5.

The graph model has a similar structure, but it is built on the Conv2D layer.

We train the CNN model for 20 epochs with the Adam optimizer and a learning rate of 0.001.

The combined model concatenates the result from the flattened layer of both the text model and graph model and fed it to the dense layers to get the prediction.

4.4.3 Experimental Setup

Since we want to explore the model performance on mis-typed and corrupted data, we keep the question marks in sentences in only one experiment to compare the performance with the rule-based model and the machine learning model. We expect that question marks would have a huge impact on the success of question classification. It is possible to achieve a very satisfactory result using the rule-based method if all the text has correct grammar and organization.

For each model, we validate the performance using weighted GloVe word embedding, using graph embedding, and using combined embedding of word and graph.

4.4.4 Results and Discussion

Model	Precision		Recall		F1-score
	Question	Non-question	Question	Non-question	
Rule-based					
In-text & PT	0.69	1.00	1.00	0.85	0.89
RF					
TF-IDF	0.88	0.84	0.45	0.98	0.84
+ POS	0.98	0.99	0.96	0.99	0.99
GloVe	0.96	0.97	0.91	0.99	0.97
+ Weighted	0.98	0.91	0.70	0.99	0.92
CNN					
GloVe	0.99	1.00	1.00	1.00	1.00
+ Weighted	0.98	0.99	0.98	0.99	0.99

Table 4.3: Results of model performance on well-formatted data

Model	Precision		Recall		F1-score
	Question	Non-question	Question	Non-question	
Rule-based					
In-text & PT	0.54	0.85	0.55	0.84	0.77
RF					
TF-IDF	0.77	0.82	0.37	0.96	0.81
+POS	0.78	0.82	0.36	0.96	0.81
GloVe	0.80	0.86	0.51	0.96	0.85
+Weighted	0.76	0.83	0.40	0.96	0.82
Graph	0.31	0.75	0.08	0.94	0.73
+W GloVe	0.79	0.82	0.36	0.97	0.82
CNN					
GloVe	0.71	0.89	0.66	0.91	0.85
+Weighted	0.69	0.89	0.64	0.91	0.84
Graph	0.45	0.76	0.03	0.99	0.75
+W GloVe	0.63	0.89	0.68	0.87	0.82

Table 4.4: Results of model performance on data without the question mark. RF refers to Random Forest and CNN refers to the Convolutional Neural Network model

From our results in Table 4.3, we observe simple rule-based model gives good accuracy, but it fails to classify sentences such as “I’m not really sure which is more frequent.” or “but I don’t know how we can do that.”. The RF classifier and the CNN model, on the other hand, give much better performance on those kinds of sentences.

In the data with question marks, we find the random forest model performs better with the TF-IDF feature when POS tags are included while the CNN model using GloVe embedding gives the best performance on formatted data. Moreover, using TF-IDF weighted GloVe embedding results in performance loss in both RF and CNN

models.

In the results from data without question marks, all models have a significant performance loss, and the rule-based method is still worse than the RF model and CNN model. The RF and CNN models using GloVe embedding give the best performance. The RF model performs better in predicting questions while the CNN model performs better in predicting non-questions. Using TF-IDF weighted Glove embedding still results in worse performance. Finally, The model using Graph embedding fails to predict questions.

We tested two different graph embedding methods. Results presented in Table 4.4 are obtained using GL2vec graph embedding. Models using Feather-G embedding classified all testing data as non-questions. The following are our observations and take away from the study of question detection.

1. Rule-based model performs worse than ML models with simple corruption (absence of question mark).
2. Graph embedding using GL2VEC or Feather performs very poorly by itself and has nearly no impact when combined with text embeddings.
3. Feather embedding seems to have no effect on this classification task, as it is different from GL2VEC, GL2VEC uses edge features to generate graph embedding while feather uses node features while randomly traveling through the graph.
4. Question marks have a robust effect on question classification but may also be a strong corruption.

From our experiment in the machine learning approach, we found that it might be

easier and more effective to simply detect certain features in the sentence. Although the rule-based model is fast and easy to implement with good accuracy, ML models are needed as they can perform better in both well-formatted data and corrupted data.

4.5 Summary

In this chapter, we explore, design, and present our question detection module. Question marks have a huge impact on both rule-based methods and machine-learning methods. Machine learning methods can still give some results with the absence of question marks. The next chapter presents the answer generation tool. Due to the absence of a good email dataset for the study, we tried to extract emails from the administrative system but since there is privacy concern and the emails would be unlabeled, we decided to use a benchmark dataset for this part of the study.

Chapter 5

Response Generation

In this chapter, we present the background of response generation tasks, followed by our exploration of a benchmark dataset for abstractive response generation task namely Explain Like I am 5 (ELI5) [19]. We discuss the state-of-the-art model BART [38] and our hypothesis to improve its performance. Experiments are presented to validate our hypothesis.

5.1 Background

Question-answering is commonly performed using two approaches: extractive or retrieval-based and abstractive or generative-based approach. Extractive question answering focuses on extracting the correct answer from the given context while abstractive question answering focuses on generating an answer from the context that correctly answers the question. Two popular benchmark databases are commonly cited in literature for extractive and abstractive question answering namely the Natural Questions (NQ) and the ELI5 as described below.

<p>Question:</p> <p>where does the last name hogan come from?</p> <p>Short Answer:</p> <p><i>None</i></p>	<p>Long Answer:</p> <p>Hogan is an Irish surname . If derived from the Irish Gaelic , Ó hÓgáin , it is diminutive of Og meaning " young " . If it is derived from Cornish , it means " mortal " . This youthful definition of the name is also reflected in the Welsh , where Hogyn means stripling . The word Hogen means high in Dutch .</p>
<p>Question:</p> <p>where is the world s largest ice sheet located today?</p> <p>Short Answer:</p> <p>Antarctica</p>	<p>Long Answer:</p> <p>The Antarctic ice sheet is the largest single mass of ice on Earth . It covers an area of almost 14 million km and contains 30 million km of ice . Around 90 % of the Earth's ice mass is in Antarctica , which , if melted , would cause sea levels to rise by 58 meters . The continent - wide average surface temperature trend of Antarctica is positive and significant at > 0.05 ° C / decade since 1957 .</p>

Figure 5.1: Example data from NQ database

5.1.1 Extractive Question-Answering with NQ

Natural Questions (NQ) database [35] is a new Question Answering benchmark released by Google, which contains real user questions. For question answering, the entire Wikipedia page much be searched to determine if a satisfactory answer exists and then the answer can be extracted from that given page, as well as a short-form and a long-form answer as shown in Figure 5.1.

5.1.2 Extractive Question-Answering with SQuAD V2

Stanford Question Answering Dataset (SQuAD) [50] is a reading comprehension dataset. It consists of questions based on Wikipedia articles. The answer can be a segment of text, or span, from the relevant article pages and sometimes the question is unanswerable. SQuAD V2 [49] add over 50,000 questions that are unanswerable but look similar to those answerable questions in the SQuAD v1.1 dataset. Squad v2 brings the challenge of determining if the question can be answered while providing an answer if possible.

<p>Question : Why does water heated to room temperature feel colder than the air around it?</p> <p>Answer : It doesn't feel colder than the air around it, it feels colder than your body temperature. Water is a better conductor of heat than air, so it takes more energy to heat it up than air is to cool it down. <u>So</u> when you heat water to room temperature, it takes away more energy from your body than air does.</p>	<p>Documents : 1.when the skin is completely wet. The body continuously[.....].</p> <p>2. heat will flow from the hotter to the colder, by whatever pathway[.....].</p> <p>3. air condition and moving along a line of constant enthalpy toward a state of higher humidity[.....].</p> <p>.....</p> <p>10 . Conduction On a microscopic scale, heat conduction occurs as hot, <u>apidly</u> moving or vibrating atoms and molecules interact with neighboring atoms [.....].</p>
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Figure 5.2: ELI5 example

5.1.3 Abstractive Question-Answering with ELI5

Explain like I am 5 (ELI5) dataset [19] is a benchmark dataset for abstractive question-answering tasks. It contains questions and answers from the Reddit "Explain Like I am 5" threads as shown in Figure 5.2. For question answering, multiple documents must be searched and a multi-sentence answer must be generated.

5.1.4 BERT, Sentence-BERT, and BART

BERT [14], Sentence-BERT [52] and BART [38] are the state-of-the-art (SOTA) models commonly applied to text embedding and text analytics tasks to achieve superior performance. We explore, apply and extend these models for response generation. The models are briefly described below.

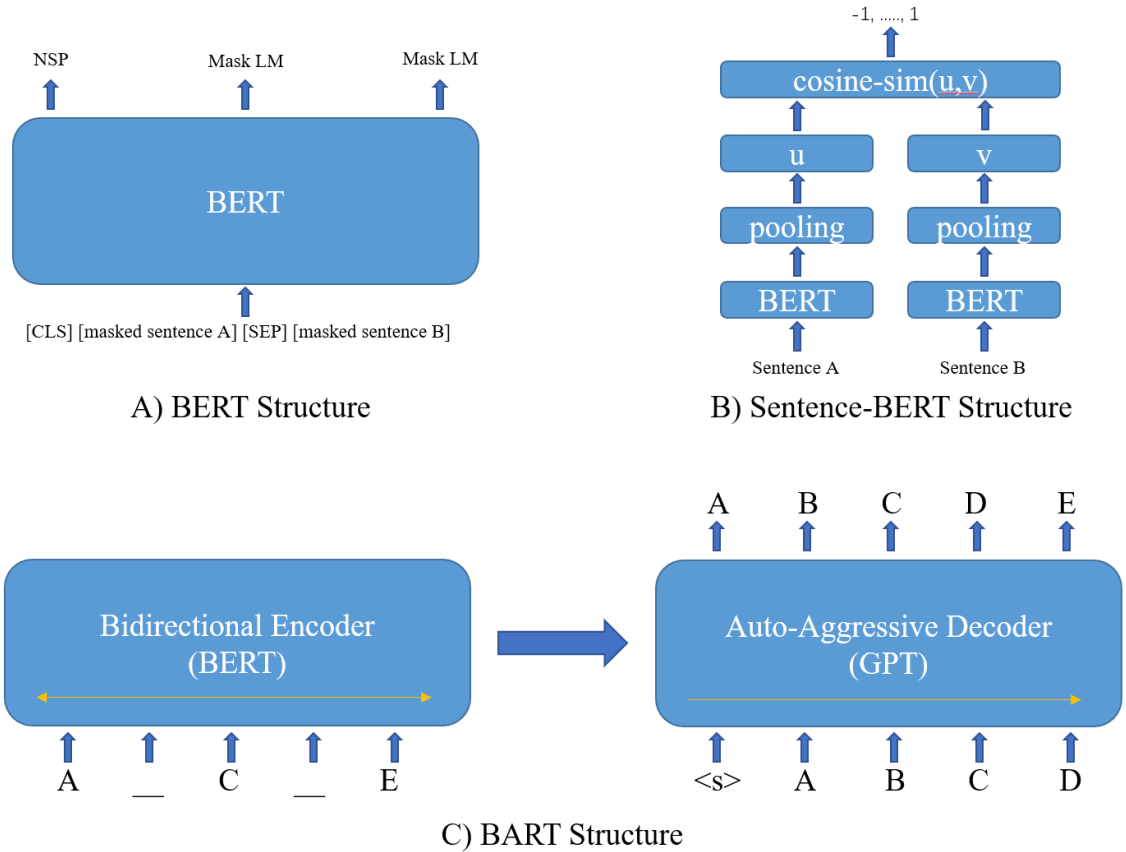


Figure 5.3: BERT, Sentence-BERT, and BART model structures

Architectures of the BERT, Sentence-BERT and BART models are shown in Figure 5.3.

BERT [14] is designed to pre-train deep bidirectional representations of context, which makes it easy to be fine-tuned for specific tasks by adding and training just one additional output layer.

Sentence-BERT [52] applies siamese and triplet network structures to produce meaningful embeddings of sentences that can be directly compared with cosine similarity. Sentence-BERT can reduce the time cost to find most similar pairs in 10,000 sentences from 65 hours using BERT to 5 seconds with similar accuracy.

BART [38] applies an auto-aggressive decoder after the BERT model. Since the BERT model predicts masked and missing tokens independently, which makes BERT not suitable for generation tasks. GPT, which is used as the decoder for the BART model, is an auto-aggressive decoder that can be used for generation tasks. It makes predictions based on only the left-ward context. The BART model connects the bidirectional encoder with the auto-aggressive decoder namely BERT and GPT to allow the arbitrary noise transformations.

5.2 Problem Description

Although models like the Poolingformer [73] and the ReflectionNet-ensemble [67] have already achieved very high accuracy on extractive question-answering tasks, they do not address the problem where answers are not directly given in the document or multiple potential answers exist across multiple documents. Further study on abstractive question-answering is needed to improve the general performance on abstractive question-answering tasks.

Extractive Q adapts answers to match questions. Therefore, instead of just extracting one or more lines from documents to provide an answer, a proper QA system should compose sentences as humans do from multiple pieces of information. We aim to address this abstract answer-generation problem based on supporting documents. To achieve this goal, we started exploring the state-of-the-art model BART [38] on the ELI5 dataset with a pre-trained model [29], based on the following hypotheses to improve the model performance.

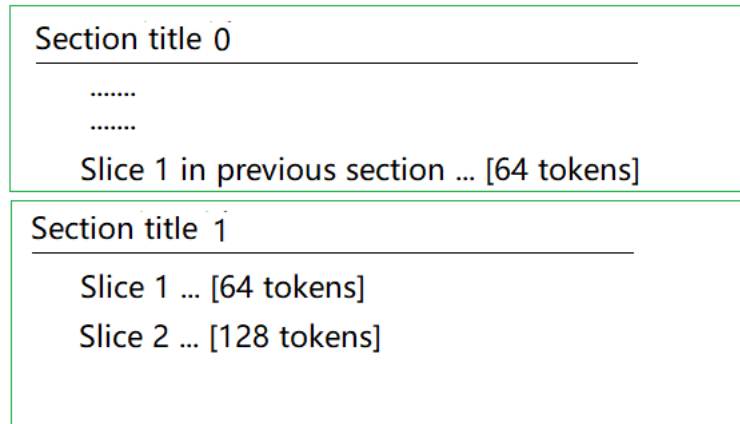


Figure 5.4: Token slices V.S. full section.

1. *Using the full document (complete information) is better than using document slices (incomplete information) to generate an answer.*

During our exploration of the pre-trained model, we found that the model uses fixed document slices containing 128 tokens from the relevant Wikipedia page. The approach is cost-efficient as it uses the most relevant documents as supporting documents. However, it is possible that information scattered across the same document or other documents may be missed because of the restriction on the maximum number of supporting documents that can be used by the model. Also, the existing approach sometimes truncates two sections which is prone to information loss. We propose using the whole section of a document instead of a slice of 128 tokens to avoid information loss and thereby, improve the model's performance. We show an example in Figure 5.4, where tokens within each green box indicate a full section that might contain multiple slices.

2. *Using the important features of the sentence could improve performance*

We propose the following three ways to extract important information data features from the whole document.

- (a) Using Open Information Extraction (OIE) model [62].
- (b) Using an extractive QA model.
- (c) A combination of the above two approaches.

Inspired by Fan et al.'s work [18], we use the OIE model which has the potential to greatly shorten the length of the sentence to a representation that can answer "who does what to whom at when where how" questions while retaining the essential information. Using the extractive approach in the beginning can make our approach cost-effective by eliminating unimportant and irrelevant parts of the document while keeping information left corresponding to question terms. Extractive QA models also have great performance in finding answers from the context. A combination of extractive and abstractive approaches can reduce the workload of the OIE and the final QA model. As models can have a constraint on the maximum length of the input data, shortening the input can give the model access to more information.

3. *Using sentence to graph approach to create a knowledge network representation which can improve QA performance*

Fan et al.'s work gives insights about extracting information from a sentence by using a graph approach. The authors form a local knowledge graph or network

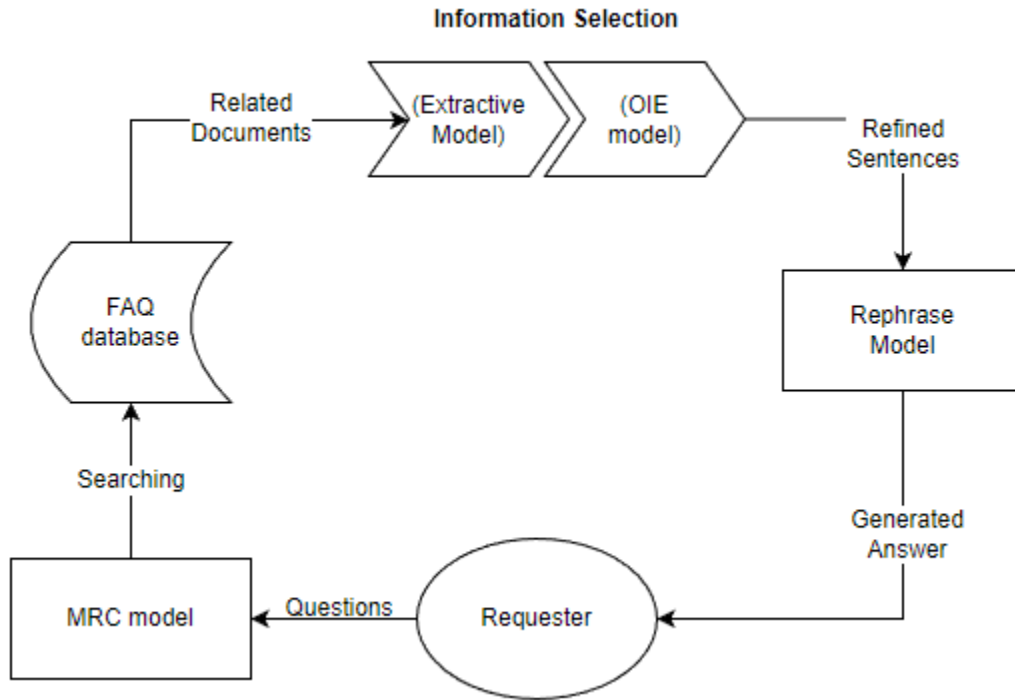


Figure 5.5: An overview of our response generation pipeline

from multiple supporting documents that can connect similar entities across documents to find potential relations between entities and also further reduce the size of the input to the model.

5.2.1 Proposed Solution

A general pipeline displaying our proposed approach is shown in Figure 5.5. It includes three main components as explained below.

1. Supporting Documents Selection: The first step involves taking the question and the FAQ database as inputs and selecting the most relevant documents to extract or generate the answer. The existing MRC model [52] reads the question

and compares it with supporting documents in the FAQ database, retrieving the top-k most relevant supporting documents. We test the existing approach against multiple extensions that we implemented to further optimize the document selection and embedding approach as explained next in Section 5.3.

2. **Information Selection From Documents:** The selected documents from step 1 are processed further to match the user’s question to retrieve an answer. If a good match is found, exact lines from the FAQ documents can be used as a response as in the retrieval-based approach. Otherwise, the matched documents can be used to generate an abstract answer using the generation-based approach. We explore the use of keywords instead of full document segments in our generation-based approach by implementing a knowledge graph. As before, we use the MRC model to retrieve the top-k documents from the FAQ database and apply an OIE method to extract the key tokens from the supporting documents. We implement a graph constructor to create a knowledge graph from the key tokens which is used in Step 3 as input to search and generate the answer.
3. **Answer Generation:** The answer generation model generates an abstract answer using the knowledge graph. Linearizing the knowledge graph provides the answer generation model with a span of rich information which is used to compose the answer.

5.3 Implementation

In this section, we present our choice of the dataset, preprocessing of the data, and the DNN model.

5.3.1 Datasets

ELI5 dataset

The ELI5 dataset represents the first large-scale dataset for abstract long-form question-answering tasks. The dataset is derived from a Reddit forum thread entitled “Explain Like I’m Five (ELI5)”, which is an online community that provides answers that are understandable by children aged five and under. In this dataset, answers are typically provided in the form of a multi-sentence response.

The question answers in this dataset resemble scenarios in organizations or communities where people asking questions generally want a detailed explanation. Therefore, we use this dataset in building and demonstrating our generative question-answering system.

Wiki40b snippet

Wiki40b snippet [24] contains Wikipedia [22] pages split into plain text snippets of a length of 128 tokens for dense semantic indexing. Each entry contains the following features:

1. `_id`: a string feature.
2. `datasets_id`: a int32 feature.
3. `wiki_id`: a string feature.
4. `start_paragraph`: a int32 feature.
5. `start_character`: a int32 feature.

6. end_paragraph: a int32 feature.
7. end_character: a int32 feature.
8. article_title: a string feature.
9. section_title: a string feature.
10. passage_text: a string feature.

We use this dataset as our FAQ database for the BART model.

5.3.2 Data Preprocessing

Unlike slicing documents into 128 tokens applied by Jernite [29], we concatenate all the document pieces that belong to the same article_title feature and then split it into sections since the same slices can exist in multiple sections, it can introduce duplicate information but helps capture more context.

5.3.3 Deep Learning Model

We start with the pre-trained BART model[29] on the Eli5 dataset as our baseline, which has a similar performance to the BART model in the paper that reaches SOTA performance on the ELI5 dataset.

5.4 Experiments

We conduct multiple experiments presented in this section to validate our hypothesis about applying a different data preprocessing method and observe its effect on the BART model performance. Next we try to explore a graph representation of key information content to reduce the space and time needed to generate an answer.

5.4.1 Experiment 1: Full Section vs Slices

In this experiment, we aim to validate the hypothesis 1 that full document contains more context than document slices for response generation. As we put the slices together into full sections. We regenerate the embedding for each section via the same pre-trained Sentence-BERT model. The default max-length is 128, but we observed that some sections exceed that length. So generate embeddings using both 128 (Method 1a) and 512 (Method 1b) as maximum-token-length. We compare the performance of the two different maximum-token-length to generate section embedding of the section-based approach with that of the baseline slice-based approach.

5.4.2 Experiment 2: Feature Selection and Representation

We conduct two experiments matching hypotheses 1 and 2 as mentioned in Section 5.2.

Experiment 2a: Open Information Extraction

We use the OIE model from the AllenNLP library [23] as our information extraction in two modes.

In partial information extraction mode, we extract “who does what to whom” (Subject, Verb, Object) information (Method $2a_1$), and in full information extraction mode we extract “who does what to whom at when where how” (subject, verb, object, and parameters) (Method $2a_2$).

The Amazon rainforest (Portuguese : Floresta Amazônica or Amazônia Spanish : Selva Amazónica , Amazonía or usually Amazonia French : Forêt amazonienne Dutch : Amazoneregenwoud), , also known in English as Amazonia or the Amazon Jungle , is a moist broadleaf forest that covers most of the Amazon basin of South America . This basin encompasses 7,000,000 square kilometres (2,700,000 sq mi) , of which 5,500,000 square kilometres (2,100,000 sq mi) are covered by the rainforest . This region includes territory belonging to to nine nations . The majority of the forest is contained within Brazil , with 60 % of the rainforest , followed by Peru with 13 % , Colombia with 10 % , and with minor amounts in Venezuela , Ecuador , Bolivia , Guyana , Suriname and French Guiana . States or departments in four nations contain " Amazonas " in their names . The Amazon represents over half of the planet 's remaining rainforests , and comprises the largest and most biodiverse tract of tropical rainforest in the world , with an estimated 390 billion individual trees divided into 16,000 species .

Figure 5.6: Example of attention aid selection

Experiment 2b: Sentence Extraction Approach

We use the LongFormer[6] model which is pre-trained on SQUAD V2 dataset and has the capability to handle very long sequences. Instead of letting the model output its prediction on the answer, we use its hidden attention to mark important tokens and then select several tokens around attended tokens. We select top k attended tokens where k represents a ratio of tokens selected with respect to the total number of tokens in the document. In our experiment, k is set to 0.1 with a minimum bound of 10 and a maximum bound of 100.

In Figure 5.6, words with a blue background are the tokens the model attended. In this section, we only selected tokens that are attended as input (Method 2b₁). We select at least 10 tokens being attended to and 10 tokens around each of the attended tokens. Selecting tokens around the attended tokens can shorten the sentence while keeping important information.

5.4.3 Results

To evaluate our approach to select supporting documents and document embedding through data preprocessing, we use ROUGE as our evaluation metric and the original BART model as our baseline. We present our experiment result in Table 5.1

Experiment	Precision			Recall			F1-score		
	R1	R2	RL	R1	R2	RL	R1	R2	RL
Baseline	0.3254	0.0680	0.3251	0.3118	0.0631	0.2560	0.2729	0.0551	0.2583
Method 1a	0.3685	0.0707	0.3278	0.2892	0.0617	0.2563	0.2929	0.0561	0.2596
Method 1b	0.3684	0.0710	0.3278	0.2891	0.0619	0.2561	0.2929	0.0563	0.2596
Method 2a ₁	0.2910	0.0391	0.2608	0.2240	0.0313	0.2001	0.2287	0.0296	0.2044
Method 2a ₂	0.3693	0.0706	0.3289	0.2879	0.0612	0.2554	0.2925	0.0558	0.2596
Method 2b ₁	0.3692	0.0712	0.3289	0.2891	0.0619	0.2564	0.2932	0.0564	0.2602

Table 5.1: Comparative performance of the different data preprocessing and embedding approaches.

Discussion

As a result of the concatenation of the reference documents, we achieved a significant increase in precision with reduced recall. It resulted in a slightly improved F1 score for ROUGE-1, ROUGE-2, and ROUGE-L.

In general, using an input of length 512 does not result in a significant improvement in performance. As most sections are still within 128 tokens, and 95% of sections have fewer than 512 tokens. The first 128 tokens already contain sufficient information to be selected as a supporting document most of the time. Therefore, there is not much difference between using 128 and 512 as the maximum input length.

We observe a significant performance loss by using only subject, verb, and object in partial OIE extraction mode, and a slight increase in precision-recall bias with full OIE extraction mode. However, the F1-score is similar to using full-text document embedding.

5.5 Future work with Local Knowledge Network

Since hypothetically the model is learning keyword-answer pairs, when the relevant keywords are fed into the model, a more precise answer should be generated. When a model is fed non-relevant keywords, it is expected to generate non-relevant answer

sentences, which would result in a loss of recall. Also if too many tokens are fed into the model, it would likely result in less attention being paid to the relevant keywords.

We designed our graph constructor based on the above hypothesis to maximize the extraction of relevant keywords in the input while restricting the input length within a specified limit as dictated by the model’s capability. However, more efficient and robust models must be developed to linearize the graph back to a well-formatted sentence for answer generation. The graph cannot be linearized back into a full sentence via a rule-based method. We present this part of the exploration and problems in Appendix B.

5.6 Summary

In this chapter, we present our hypothesis and experiment to improve SOTA abstractive answer generation model in preprocessing steps, we validate that using complete information will improve the model performance at preprocessing step while not all the tokens in the sentence contribute to good performance, “who does what to whom at when where how” is enough for sentence-based information extraction. We highlight shortcomings of the OIE linearization method to be addressed as future work. Despite the challenge of linearization, local knowledge networks can help find potential relations between similar things in different documents. The next chapter presents a summary of our research and presents potential future works directions.

Chapter 6

Conclusion and Future Work

In this research, we design an Intelligent Email Response System (IERS) that can be deployed in companies and organizations to ease the email handling workload. Upon receiving an email, our system should automatically generate a response within one second. Moreover, our system differs from chatbot systems in that our systems generate responses independently, whereas chatbots provide responses based on previous conversations.

We present literature reviews on general NLP tasks, email classification, question detection, and answer generation tasks in Chapter 2. We also discuss the challenges of building an automatic email response system, and compare extractive question-answering tasks and abstractive question-answering tasks.

Chapter 3 presents our design of an email extraction algorithm that can overcome the anonymization issue and extract full conversations instead of individual emails to make it easier to collect training data for the model. We design, implement, and compare the performance among CNN, BiLSTM, CNN-BiLSTM, and Hierarchical CNN-BiLSTM models using a university departmental email dataset. We found

CNN-BiLSTM model and the Hierarchical CNN-BiLSTM models have a similar performance which is much superior to CNN and BiLSTM models. At the end of that chapter, we showcase a real-world application of our email pipeline which includes email extraction and classification. Our model achieves a satisfactory performance of 86% accuracy with real-life email data.

We explored question detection system using machine learning models in Chapter 4. The rule-based approach performs poorly on question detection in malformed sentences. We demonstrate that missing question marks in the text affect both rule-based and ML model performances negatively in terms of accuracy, but machine learning models achieve greater accuracy without question marks compared to the rule-based approach. We also explore the potential of using a graph embedding approach to transform sentences to graph representation and apply a parser tree algorithm to the graph. However, the graph embedding approach did not provide satisfactory or expected outcomes by itself and also when used with text embedding approach.

Finally, we explore the state-of-the-art abstractive question-answering model, BART, on the ELI5 dataset. We proposed three hypotheses to preprocess the input data and use the processed data as input to the BART model to improve the model's performance. We design and implement several experiments to validate our hypothesis and achieve a slight improvement of 0.2932 F1-score in Rouge-1 compared to the 0.2729 F1-score in Rouge-1 with the baseline BART model. We also present our exploration and the problems we encountered while extending the model to use a local knowledge net approach as shown in Appendix B.

6.1 Future Work

Potential future work directions are listed below.

6.1.1 Email Classification

Although our CNN-LSTM model reaches good performance, the accuracy may be further improved by changing the model structure, using different embedding methods, or including the subject line of the email. The dataset 1 described in Section 3.3.3 does not include the subject line of the email, which can potentially lead to better performance.

6.1.2 Question Detection

We plan to investigate more sentence-to-graph embedding techniques using a machine-learning model to convert sentences to parser trees instead of using the OIE model which does not perform well when the sentence is not well formatted or has mistakes. Also, the graph embedding method we use either focuses on node features or edge features while traveling through the graph. A better graph embedding method or sentence analyzer can improve the result.

6.1.3 Response Generation

The recent introduction of the ChatGPT [45] model shows superior performance on abstractive question-answering tasks, although it may generate outdated information or sentence with ethical issues. Integrating fine-tuned ChatGPT in our response generation part can potentially provide sensible responses despite ChatGPT relies on very large data and may also pose concerns due to bias in generating information,

lack of creativity, and ensuring ethics in implementing machine cognition [12]. There are several challenges with the local knowledge net approach. New models can be developed to construct the local knowledge net in a fast and precise manner to produce sensible and correct responses directly from the local knowledge net with an improved algorithm.

Bibliography

- [1] Abdulkareem Al-Alwani. A novel email response algorithm for email management systems. *Journal of Computer Science*, 10:689–696, 04 2014.
- [2] Abdulaziz Alayba, Vasile Palade, Matthew England, and Rahat Iqbal. *A Combined CNN and LSTM Model for Arabic Sentiment Analysis: Second IFIP TC 5, TC 8/WG 8.4, 8.9, TC 12/WG 12.9 International Cross-Domain Conference, CD-MAKE 2018, Hamburg, Germany, August 27–30, 2018, Proceedings*, pages 179–191. 08 2018.
- [3] Chris Alberti, Kenton Lee, and Michael Collins. A bert baseline for the natural questions, 01 2019.
- [4] Ammar Almomani, B. B. Gupta, Samer Atawneh, A. Meulenberg, and Eman Almomani. A survey of phishing email filtering techniques. *IEEE Communications Surveys & Tutorials*, 15(4):2070–2090, 2013.
- [5] Izzat Alsmadi and Ikdam Alhami. Clustering and classification of email contents. *Journal of King Saud University - Computer and Information Sciences*, 27(1):46–57, 2015.

-
- [6] Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv:2004.05150*, 2020.
- [7] Edward Loper Bird, Steven and Ewan Klein. O’Reilly Media Inc., 2009.
- [8] Kofi Boakye, Benoit Favre, and Dilek Hakkani-Tur. Any questions? automatic question detection in meetings. pages 485 – 489, 01 2010.
- [9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, jun 2002.
- [10] Hong Chen and Hisashi Koga. *GL2vec: Graph Embedding Enriched by Line Graphs with Edge Features*, pages 3–14. 12 2019.
- [11] Raymond Cheng. <https://towardsdatascience.com/understanding-tf-idf-a-traditional-approach-to-feature-extraction-in-nlp-a5bf>
- [12] Naem Chowdhury and Sagedur Rahman. A brief review of chatgpt: Limitations, challenges and ethical-social implications. 02 2023.
- [13] Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. Universal Dependencies. *Computational Linguistics*, 47(2):255–308, 07 2021.
- [14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*

- (*Long and Short Papers*), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
- [15] Mingzhe Du, Jose Vidal, and Zaid Al-Ibadi. Using pre-trained embeddings to detect the intent of an email. 02 2020.
- [16] Wenchao Du and Alan Black. Top-down structurally-constrained neural response generation with lexicalized probabilistic context-free grammar. pages 3762–3771, 01 2019.
- [17] Nathan Duran. <https://github.com/NathanDuran/MRDA-Corpus>.
- [18] Angela Fan, Claire Gardent, Chloe Braud, and Antoine Bordes. Using local knowledge graph construction to scale seq2seq models to multi-document inputs. pages 4177–4187, 01 2019.
- [19] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. Eli5: Long form question answering, 2019.
- [20] Jenny Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. 01 2005.
- [21] Nicolas Ford, Daniel Duckworth, Mohammad Norouzi, and George Dahl. The importance of generation order in language modeling. pages 2942–2946, 01 2018.
- [22] Wikimedia Foundation. Wikimedia downloads. "<https://dumps.wikimedia.org>".

- [23] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. Allennlp: A deep semantic natural language processing platform. 03 2018.
- [24] Mandy Guo, Zihang Dai, Denny Vrandečić, and Rami Al-Rfou. Wiki-40b: Multilingual language model dataset. In *LREC 2020*, 2020.
- [25] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.
- [26] Brian Harrington and Stephen Clark. Asknet: Automated semantic knowledge network. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pages 889–894. AAAI Press, 2007.
- [27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.
- [28] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020.
- [29] Y. Jernite. <https://yjernite.github.io/lfqa.html>.
- [30] Shiney Jeyaraj and Raghuveera Tripurarihatla. A framework for automatic generation of faqs from email repositories. pages 1035–1041, 11 2018.
- [31] Steven J. Jones. Efficient computation of spreading activation using lazy evaluation. 2016.

- [32] Svetlana Kiritchenko. Email classification with co-training. *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, 04 2002.
- [33] Will Koehrsen. Neural network embeddings explained. <https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526>, Oct 2018.
- [34] Brendan Kolisnik, Isaac Hogan, and Farhana Zulkernine. Condition-cnn: A hierarchical multi-label fashion image classification model. *Expert Systems with Applications*, 182:115195, 2021.
- [35] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Christopher Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 03 2019.
- [36] Helen Kwong and Neil Yorke-Smith. Detection of imperative and declarative question-answer pairs in email conversations. volume 25, pages 1519–1524, 01 2009.
- [37] Kenton Lee, Luheng He, and L. Zettlemoyer. Higher-order coreference resolution with coarse-to-fine inference. In *NAACL-HLT*, 2018.
- [38] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. Bart: Denoising

- sequence-to-sequence pre-training for natural language generation, translation, and comprehension. pages 7871–7880, 01 2020.
- [39] Chenliang Li, Haoran Wang, Zhiqian Zhang, Aixin Sun, and Zongyang Ma. Topic modeling for short texts with auxiliary word embeddings. *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016.
- [40] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.
- [41] Mitchell Marcus, Mary Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19:313–330, 07 2002.
- [42] John Michailoff. Email classification: An evaluation of deep neural networks with naive bayes, 2019.
- [43] Ankita Mittal and Aparna Ramamurthy. Automatic email response system in e-learning. *Proceedings of the International Conference on Advances in Information Communication Technology & Computing*, 2016.
- [44] Maryam Mohsin. 10 email marketing statistics you need to know in 2023. <https://www.oberlo.com/blog/email-marketing-statistics>, Jan 2023.
- [45] openAI. Introducing chatgpt. <https://openai.com/blog/chatgpt>. Accessed: 2023-3-19.

- [46] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. volume 14, pages 1532–1543, 01 2014.
- [47] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.
- [48] Andreas Pogiatzis. Nlp: Contextualized word embeddings from bert. <https://towardsdatascience.com/nlp-extract-contextualized-word-embeddings-from-bert-keras-tf-67ef29f60a7b>, Mar 2019.
- [49] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.
- [50] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text, 2016.
- [51] Muhammad Rana. Eaglebot: A chatbot based multi-tier question answering system for retrieving answers from heterogeneous sources using bert. 2019.
- [52] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. pages 3973–3983, 01 2019.
- [53] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Karate club: An api oriented open-source python framework for unsupervised learning on graphs. pages 3125–3132, 10 2020.

- [54] Benedek Rozemberczki and Rik Sarkar. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. pages 1325–1334, 10 2020.
- [55] Said Salloum, Tarek Gaber, Sunil Vadera, and Khaled Shaalan. Phishing email detection using natural language processing techniques: A literature survey. *Procedia Computer Science*, 189:19–28, 2021. AI in Computational Linguistics.
- [56] Said Salloum, Tarek Gaber, Sunil Vadera, and Khaled Shaalan. A systematic literature review on phishing email detection using natural language processing techniques. *IEEE Access*, 10:65703–65727, 2022.
- [57] Guergana Savova, James Masanz, Philip Ogren, Jiaping Zheng, Sunghwan Sohn, Karin Kipper-Schuler, and Christopher Chute. Mayo clinical text analysis and knowledge extraction system (ctakes): Architecture, component evaluation and applications. *Journal of the American Medical Informatics Association : JAMIA*, 17:507–13, 09 2010.
- [58] Craig W. Schmidt. Improving a tf-idf weighted document vector embedding. *ArXiv*, abs/1902.09875, 2019.
- [59] Lokesh Shrestha and Kathleen McKeown. Detection of question-answer pairs in email conversations. COLING '04, page 889–es, USA, 2004. Association for Computational Linguistics.
- [60] Elizabeth Shriberg, Raj Dhillon, Sonali Bhagat, Jeremy Ang, and Hannah Carvey-Essenburt. The icsi meeting recorder dialogue act (mrda) corpus. *SIG-dial*, 04 2004.

-
- [61] Aditya Singh, Dibyendu Mishra, Sanchit Bansal, Vinayak Agarwal, Anjali Goyal, and Ashish Sureka. Email Dataset for Automatic Response Suggestion within a University. "https://figshare.com/articles/dataset/Email_Dataset_for_Automatic_Response_Suggestion_within_a_University/5853057", 2018.
- [62] Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and I. Dagan. Supervised open information extraction. In *NAACL-HLT*, 2018.
- [63] Yaodong Tang, Yuchen Huang, Zhiyong Wu, Helen Meng, Mingxing Xu, and Lianhong Cai. Question detection from acoustic features using recurrent neural network with gated recurrent unit. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6125–6129, 2016.
- [64] Harish Tayyar Madabushi and Mark Lee. High accuracy rule-based question classification using question syntax and semantics. 12 2016.
- [65] Andre Vellino and Inge Alberts. Article assisting the appraisal of e-mail records with automatic classification. *Records Management Journal*, 26, 11 2016.
- [66] Xuguang Wang, Linjun Shou, Ming Gong, Nan Duan, and Daxin Jiang. No answer is better than wrong answer: A reflection model for document level machine reading comprehension, 09 2020.
- [67] Xuguang Wang, Linjun Shou, Ming Gong, Nan Duan, and Daxin Jiang. No answer is better than wrong answer: A reflection model for document level machine reading comprehension. pages 4141–4150, 01 2020.
- [68] Wikipedia. [https://en.wikipedia.org/wiki/ROUGE_\(metric\)](https://en.wikipedia.org/wiki/ROUGE_(metric)).

-
- [69] Adwan Yasin and Abdelmunem Abuhasan. An intelligent classification model for phishing email detection. *International Journal of Network Security & Its Applications*, 8:55–72, 07 2016.
- [70] Dong Jin Yu. *Part-of-speech tag embedding for modeling sentences and documents*. PhD thesis, 2016.
- [71] Caixia Yuan, Xiaojie Wang, and Qianhui He. Response generation in dialogue using a tailored pcfg parser. pages 81–85, 01 2015.
- [72] Hasan Zafari and Farhana Zulkernine. Chatsum: An intelligent medical chat summarization tool. pages 1–2, 07 2021.
- [73] Hang Zhang, Yeyun Gong, Yelong Shen, Weisheng Li, Jiancheng Lv, Nan Duan, and Weizhu Chen. Poolingformer: Long document modeling with pooling attention, 2022.
- [74] Xiang Zhang, Junbo Zhao, and Yann Lecun. Character-level convolutional networks for text classification. 09 2015.

Appendix A

List of all possible name abbreviations in Penn

Treebank

The Table A.1 and A.2 is created according to Penn Tree Bank by Mitchell et al. [41].

The Penn Treebank POS tagset			
CC	Coordinating conjunction	TO	to
CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential there	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund/present participle
IN	preposition/subordinating conjunction	VBN	Verb, past participle
JJ	Adjective	VBP	Verb, non-3rd ps, sing, present
JJR	Adjective, comparative	VBZ	Verb, 3rd ps, sing, present
JJS	Adjective, superlative	WDT	wh-determiner
LS	List item marker	WP	wh-pronoun
MD	Modal	WP\$	Possessive wh-pronoun
NN	Noun, singular or mass	WRB	wh-adverb
NNS	Noun, plural	#	Pound sign
NNP	Proper noun, singular	\$	Dollar sign
NNPS	Proper noun, plural	.	Sentence-final punctuation
PDT	predeterminer	,	Comma
POS	Possessive ending	:	Colon, semi-colon
PRP	Personal pronoun	(Left bracket character
PP\$	Possessive pronoun)	Right bracket character
RB	Adverb	"	Straight double quote
RBR	Adverb, comparative	'	Left open single quote
RBS	Adverb, superlative	"	Left open double quote
RP	Particle	'	Right close single quote
SYM	Symbol (mathematical or scientific)	"	Right close double quote

Table A.1: Penn Treebank POS tagset

The Penn Treebank syntactic tagset	
ADJP	Adjective phrase
ADVP	Adverb phrase
NP	Noun phrase
PP	Prepositional phrase
S	Simple declarative clause
SBAR	Clause introduced by subordinating conjunction or θ
SBARQ	Direct question introduced by wh-word or wh-phase
SINV	Declarative sentence with subject-aux inversion
SQ	Subconstituent of SBARQ excluding wh-word or wh-phase
VP	Verb phrase
WHADVP	wh-adverb phrase
WHNP	wh-noun phrase
WHPP	wh-prepositional phrase
X	Constituent of unknown or uncertain category
Null elements	
*	"Understood" subject of infinitive or imperative
0	Zero variant of that in subordinate clauses
T	Trace—marks position where moved wh-constituent is interpreted
NIL	Marks position where preposition is interpreted in pied-piping contexts

Table A.2: Penn Treebank syntactic tagset

Appendix B

Graph Constructor

We present our exploration of the local knowledge net constructor in this appendix.

B.1 Advantages

Although both extractive and abstractive question-answering models can work with plain text input, local knowledge net in graph form has the advantage to connect potential relations before feeding the model to create more meaningful and precise support sentences containing as little redundancy as possible. The local knowledge net itself may have the ability to generate a simple answer to the question [26]. Moreover, using a local knowledge net can provide more relevant entities once an entity in the net is selected as a potential answer.

B.1.1 Merge Same Representation by Merging Nodes

Two different sentences can have entities that represent the same meaning. For example, consider the two sentences below.

1. "Alice likes dogs"

2. "Alice likes cats"

if we use graph representation, we can simply merge entities "Alice" and "likes" together and reproduce a sentence "Alice likes dogs and cats". The re-write sentence has 5 tokens while the original two sentences have 6 tokens in total. The ability to merge similar representations can eliminate duplicates while retaining context information.

B.2 Asknet Algorithm

Instead of using Fan et al's local graph approach [18], we modify Brian et al's ASKnet algorithm[26] to build the knowledge graph because the ASKnet algorithm provides an advantage over the local graph approach in dealing with nested OIE information. It delves into the details of the sentence structure and components, to further reduce redundancy (Figure B.1 on the next page). Moreover, the linearized representation of the graph can be restricted not to exceed the input length of the answer generation model. ASKnet uses a NER model and a rule-based filter using prolog to extract logical relations, then initializes the graph based on those relations.

B.3 Design and Implementation Detail

ASKnet's original implementation was based on a NER model and a box filter implemented in Java. We use the OIE model to extract the tokens which we implemented in Python. Since we use OIE relations, we do not need an additional filter to help with relation recognition. Due to the differences, we needed to apply multiple adaptations to the original ASKnet algorithm as described below.

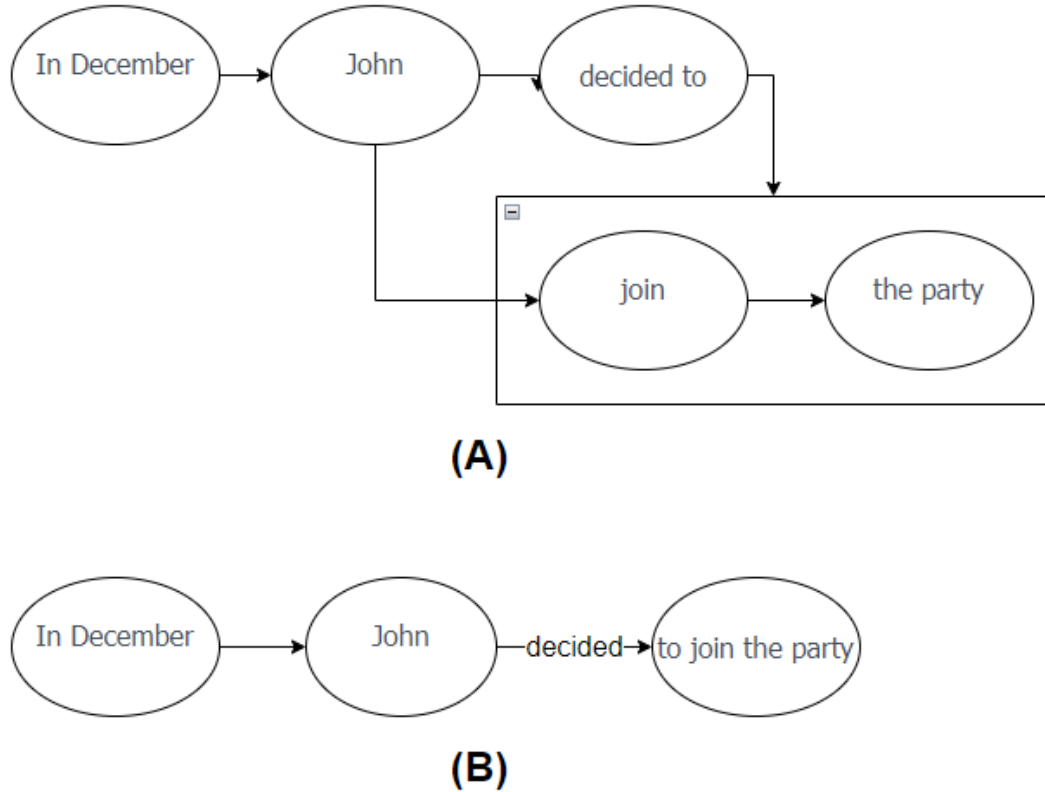


Figure B.1: Graphs generated using (A) ASKnet algorithm and (B) Fan et al.'s local graph algorithm from the sentence "In December, John decided to join the party."

B.3.1 Transform OIE Tuple into a Hierarchical Tree Structure

The OIE predictor can provide multiple levels of extraction. Fan et al.'s work on the other hand, only uses the top level of the extraction provided by the predictor, which eliminates the need for merging entities. However, it also results in loss of information that could be used to potentially merge entities at a deeper level. We show an example in Figure B.2. In the figure, there are two levels of extraction from the OIE predictor where (B) is the child of ARG2 in (A). Since (B) is in passive voice, the extraction of (B) does not provide ARG1 or ARG0 which is the subject.

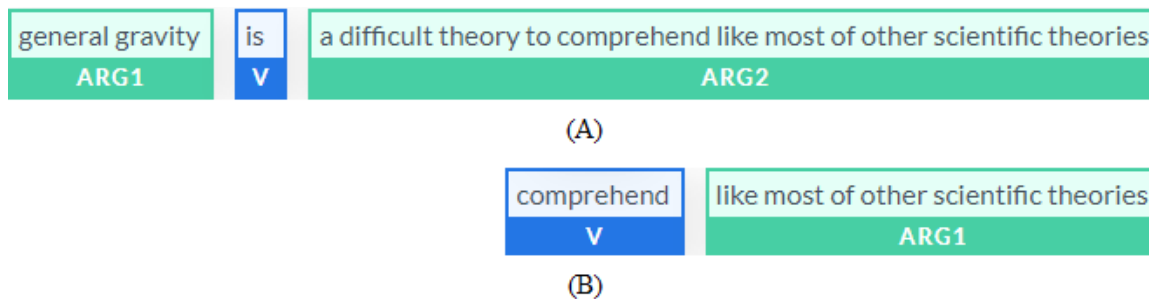


Figure B.2: Example OIE slice

The extraction creates a dictionary datatype of key-value storage structure, which adds the entity type as a key and the corresponding list of token indexes as the value (e.g. “ARG1”:[0,1,2,3]). We design an algorithm to convert the dictionary data into a tree structure while adding the missing subjects for extractions in the passive voice.

B.3.2 Transform Tree to Graph

We use networkX [25] library to implement the knowledge graph. Once the tree structure is ready, we add a node to the graph for each node in the tree in a bottom-up order. Each node in the graph contains attributes indicating its current activation, activation threshold, span, document ID, Elmo embedding, and co-occurrence reference, as well as if it is a parent node. As we build a tree structure before forming a graph, a graph can be easily built in three steps:

1. Adding all nodes in the tree to the empty graph.
2. Linking all nodes as tuples with reference to ARG0 \rightarrow V \rightarrow ARG1 in the order of (Subject \rightarrow Verb \rightarrow Object) where M nodes (ARGM) can be linked to V.

3. Checking parent-child relation in the tree structure, and thereby, adding parent-child link and giving parent nodes *is_parent* attribute.

By building the graph nodes in this manner, it is possible to break the sentence structure down into the smallest pieces possible so that the OIE predictor will have the maximum effect. Besides using the relational link described above, we also have another group of links called “ordered links” which link those nodes by their order of occurrence in the sentence from the original document. This group of links does not contribute to the upcoming Spreading Activation stage but can potentially make it much easier to linearize the graph back to text.

B.4 Spreading Activation

We apply the idea of spreading activation from Cognitive Science to identify the words that are contextually relevant in the graph structure with regard to the original sentence. In our implementation, the subject (parent) and verb constitute the pipeline. They do not fire naturally, just let the activation spread through them. There are two different types of spreading activations (SA): within a single graph, and transfer to distributed graphs. While merging graphs, the updated graph is merged into the main graph. The updated graph starts SA within itself. Since the graph itself is static and the existence of a decay factor, we need to manually add activation to the graph node to calculate a distribution matrix. At this stage, we pick several nodes and manually add the same activation amount as their activation threshold, and then let them fire in a round-table order. We record how many times each node fires within a finite number of iterations and compute a normalized distribution based on how many time each node have fired outside its turn.

We transfer the distribution to the main graph and calculate embedding similarity and node attributes similarity. We also check a node in the update graph has co-occurrence reference with nodes in the main network. Based on the score and the existence of co-occurrence reference, the nodes would take different proportions of total activation transferred from update graph when fired manually. Despite the graphs being directed, in the SA process, the activation flow ignores the direction. The SA process self-refines in a finite number of iterations. If the similarity score between two nodes is above a certain threshold, they are merged.

Instead of changing the weights of links on certain criteria, we take the number of links between two nodes to determine how many activation will flow through the links to the nodes. We follow the criteria listed below to perform the weight change operation.

1. Since links have directions at this stage, we want more attention going forward (subject to object) instead of backward. Therefore, we add a penalty parameter p_b where we set $p_b \leq 1$ on the backward links.
2. If some links are going to a parent node, the parent node will drain more activation from the source node. Therefore, we set another parameter p_p where we set $p_p \geq 1$ on the group of links to the parent node.

Edge-like Nodes

As we mentioned earlier, we use the verb and parent nodes to build the activation flow pipeline (edge-like nodes), which means they have their activation thresholds set to zero. In the same pulse fire algorithm as Asknet [26], we use two lists to store “nodes that fire in the current pulse” and “nodes that need to fire in the next pulse”.

The edge-like nodes can easily cause infinite loops with the same implementation as the original Asknet which is similar to neural networks, since they are firing all the time.

To solve this problem, we use the lazy activation approach [31] and develop an algorithm to calculate how many activations will flow to other entity nodes directly from a group of edge-like nodes. The core ideas behind lazy spreading activation are: only a small proportion of nodes in the graph will be merged at each update. If no merge operation or update happens to a node and its neighbors, the weights are not changed.

Inspired by the lazy spreading activation approach, we build a dictionary for each graph as “node in the graph”: “neighbor”: “entity node to go”: “proportion”. This enables fast calculation of activation flow during the spreading activation stage since we only have to calculate the weights once unless the node or its neighbors have been updated.

In the activation flow algorithm for the edge-like nodes, we use recursion to calculate the flow of activation to each entity node while recording which nodes have been visited already in this recursion branch. Activation in this branch will not flow to already visited nodes. When the recursion ends, we sum all the records and calculate the proportion of activation flow to each entity node.

Example of Spreading Activation process with Edge-like node

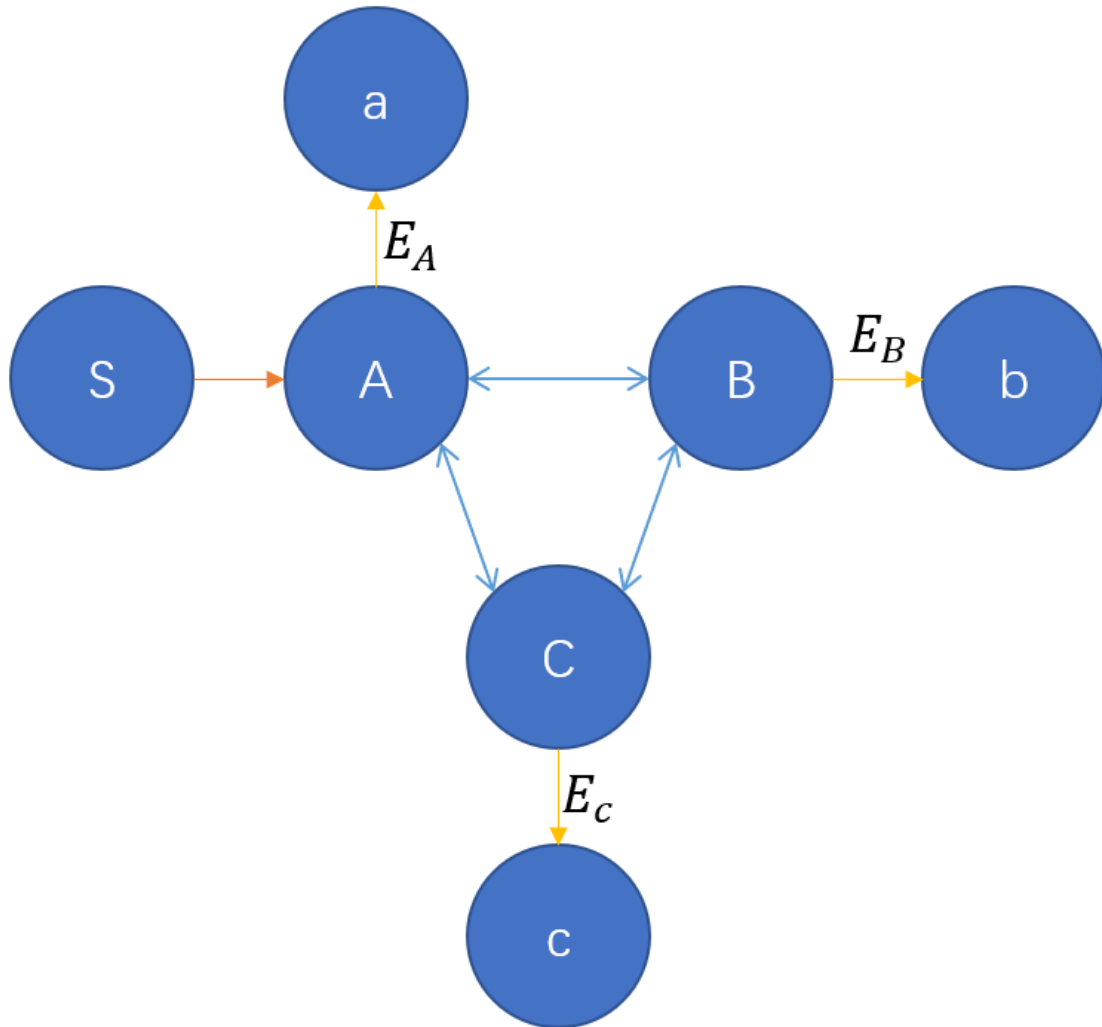


Figure B.3: Edge-like node example

Here is an example that explains our algorithm. In Figure B.3, assume there is a 3.0 activation flow from node S to node A via the red link, where node A , node B , and node C are all edge-like nodes which have outer links or edges to connect to the neighboring entity nodes node a , node b , and node c via the yellow links E_A , E_B , and

E_C . Node a , node b , and node c are all non-edge-like nodes.

Activation in node A will split evenly to nodes a , B , and C . In this step, node a receives 1 activation, and node B , and node C will continue recursion.

The next step is recursion in node B . Since in this branch, node A has been visited already, activation of 1 in B will split evenly to node b and node C each getting an activation of 0.5. In the next step, the 0.5 activation in node C goes to node c only since node A and node B have been visited already.

A similar process occurs during recursion in node C branch. By the end, Nodes a , b , and c each receive 1.0 activation in total. So we build a dictionary {"Node S":{"Node A":{"Node a": 0.33,"Node b": 0.33, "Node c": 0.33}}} for future reference. When a node or its neighbor is updated, this process will repeat to update the weights in the dictionary to reflect the change of weights.

B.5 Merging Nodes and Graphs

When merging two different graphs, we calculate a similarity score for each pair of non-edge-like nodes from each graph. In our design, we use average Elmo [47] embedding for each token in the node as the node embedding. During our exploration, we found Elmo might produce low similarity for two words that have the same meaning and even have the same characters. So besides Elmo embedding, we add a proportion of string similarity scores to calculate the final similarity score.

The similarity score S between two nodes is calculated based on three aspects: cosine similarity of embedding S_e , tf-idf score S_s , and the result of the co-reference resolution model S_c . The final similarity score is calculated from $S = \alpha \times S_e + \beta \times S_s + \theta \times S_c$. If S is above a certain threshold, then the score is saved in a score matrix

M_{score} for the next step.

Once the initial similarity score matrix is calculated, the update graph starts the self SA process. In a finite number of iterations, we manually add an activation value to each node and let them fire in a round-table order. We record the number of firings for other nodes that are not manually activated in this iteration. After this step, we compute a distribution matrix M_{self_SA} based on how many times each node fires in the self SA process.

In the next step, we transfer activation to the main network. We do a similar thing as self-SA, but the activation we add to candidate nodes in the main network is distributed to the distribution matrix M_{self_SA} , and the activation each node receives is based on the score matrix M_{score} .

For example, if there are two non-edge-like nodes N_A and N_B in the update network where node N_A has two candidate nodes in the main network node N_{a1} and node N_{a2} , during the self-SA process, node N_A fires 90 times and node N_B fires 10 times, node N_{a1} will have a similarity score of 0.8 while node N_{a2} will have a similarity score of 0.4. If the transferred activation would be 100 units, node N_{a1} will receive $100 \times \frac{90}{90+10} \times \frac{0.8}{0.8+0.4} = 60$ units of activation while node N_{a2} will receive $100 \times \frac{10}{90+10} \times \frac{0.4}{0.8+0.4} = 3.33$ units of activation.

During the transfer-SA process, we use the transferred activation of one node to refine the score of other nodes. Let's assume that node N_B has two corresponding nodes N_{b1} and N_{b2} . During transfer-SA of node N_A , if node N_{b1} fires and node N_{b2} does not fire, the similarity score of node N_{b1} will increase while the similarity score of node N_{b2} will decrease. After a finite number of iterations or if only one node still has a high score above the merge threshold, those two nodes will be merged

together. Otherwise, the update network will be added to the main network without any merging.

B.6 Linearization Challenges

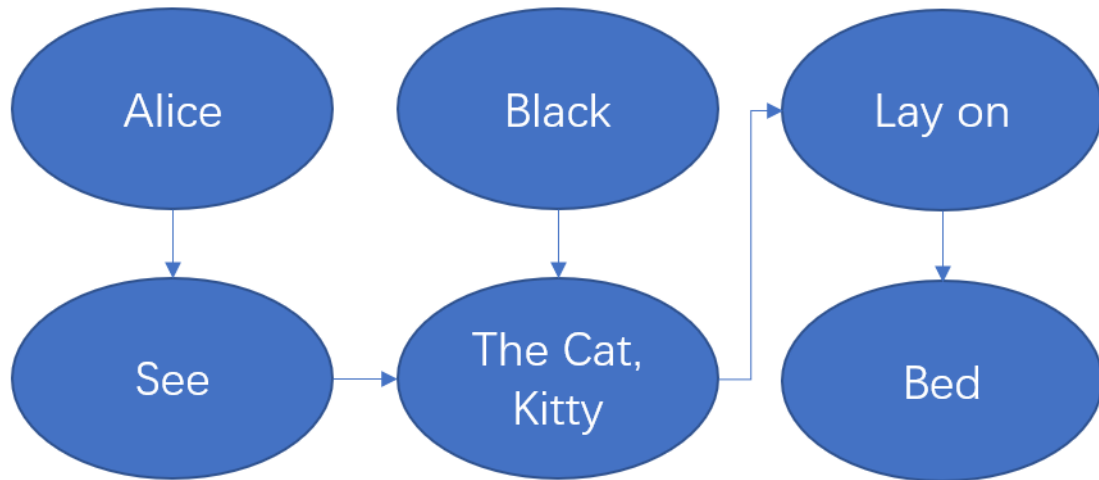


Figure B.4: Multiple representations in the same node

Linearization can bring a knowledge graph or network back to a text representation, which brings challenges to summarize the different representations in the same node to a complete and meaningful representation. As shown in Figure B.4, there are two representations “Cat” and “Kitty” in the same node. Usually, the different representations in the same node indicate to the same thing, in this case, the cat is named Kitty. When producing a text representation from the graph, the expected output should be “Alice sees Kitty the cat lay on the bed” to provide evidence for potential questions such as “What is lay on the bad?”, “What is the name of the cat?”. Instead of just select one representation in the node.





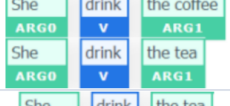

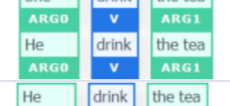



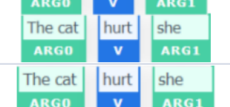
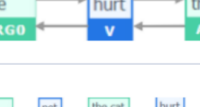




Source Sentences	Merge Type	Graph Structure	Expected Linearization
	Subject(same type)		She drink the coffee and pet the cat
	Object(same type)		She pet and he abandon the cat
	Subject and Verb(same type)		She drink the coffee and the tea
	Object and Verb(same type)		He and She drink the tea
	Subject and Object(same type)		He drink and praise the tea
	Subject, Object and Verb (different type)		She and the cat hurt each other
	Subject to Object (different type)		She pet the cat which hurt he
	Merge under parent (Subject)		He said she pet the cat, she drink the tea

Figure B.5: Rule-based linearization examples

We propose a simple rule-based method as Figure B.5 to linearize the graph with a restriction such that it can only select one representation in the node. With the graph growing, it starts losing information and composing wrong representations as the process becomes more complicated. Additionally, the rule-based method can not fully avoid duplicates. In the last row, the method provides two sentences, where the expected outcome was “He said she pet the cat and drink the tea”.

These challenges raise a future work direction to propose a model that can summarize information directly from a knowledge graph, and either use the information

internally as a feature embedding or output the information as a text representation for external requirements.