2019

Building an Automated Q-A System Using Online Forums as Knowledge Bases

Kyle Moore
University of Mississippi

Building an Automated Q-A System using Online Forums as Knowledge Bases

A Dissertation
presented in partial fulfillment of requirements
for the degree of Master of Science
in the Department of Computer and Information Sciences
The University of Mississippi

by

Kyle Moore

May 2019

ABSTRACT

Question-Answer systems traditionally use expensive and difficult to produce structured knowledge bases. Recent systems have used unstructured natural language sources as their datasets, but most of those sources have been overly broad or difficult to extend. Online forums are a largely untapped source of information that can provide both depth and breadth when limited to a specific domain, as well as being adaptive to the introduction of new information. In this paper, I conjecture that online forums can be similarly and effectively used as an unstructured knowledge base for Question-Answer systems. I use a relatively simple summarization-based approach to analyze the effectiveness of answering questions about history using well-known and popular online forums. The results of the experiment are moderately negative, though further research may need to be performed to determine whether this is a result of the model architecture, the datasets used, or the inappropriateness of forums as a source of Question-Answer knowledge.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Naeemul Hassan, as well as Dr. Yixin Chen for their assistance and guidance throughout the development process. Additionally, I would like to thank my friends and family for their support. In particular, my thanks go out to my fiancée, Mimi Truong, for moral and emotional support, as well as my friends, Ethan Luckett and Dhruvin Patel, for helping me think through difficult problems at numerous stages of development.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Automated customer service systems are an important short-term goal of applied artificial intelligence research and development. Such systems will most certainly be required to answer users' factual questions within their specific domain, and these systems will consequently need to have a repository of information for answering these questions. A classical and relatively simple approach is to use a pre-compiled fact database that contains the information necessary to answer most user questions. This approach has some major shortcomings. First, the fact databases must be manually constructed and maintained, which will generally be a costly and inefficient process. Second, the databases, largely due to the first issue, will be inherently limited in the amount and breadth of information that they contain, as they can only contain information that the creators both possess and think to include. Additionally, the system creators will generally be unable to devote resources to including specific information unique to distinct sub-topics, limiting both the depth and breadth of the system's overall knowledge. These issues are not unique to customer service systems and will be an issue for any system that seeks to answer users' factual questions automatically.

In order to mitigate these issues, this paper proposes a question answering system that utilizes online forums as its fact database. Online forums generally contain many times more information than any manually compiled database ever could due to the sheer number of users that actively contribute to them. In addition, the information will largely be broader in the sense that more varied questions will have been posed by the multitude of users than the system creators could ever consider, while also likely containing more specific information about individual sub-topics. As a bonus, these forums, from the perspective of the system

designer, are automatically compiled and add new information organically and naturally. This means that such a system can potentially be created more quickly and cheaply than the more traditional alternative.

This paper describes the implementation of a system that automatically answers questions that are limited to a specific domain, with the goal of testing whether such a system can be built to effectively utilize the information found within online forums. This system assumes that the information being sought is already present within the forum, though it may be spread among multiple threads. In order to simplify the process of information extraction, this paper specifically uses forums that follow a question-answer format. Furthermore, this paper makes the assumption that most user questions will be similar to questions that have already been asked in the forum and that the answers to these similar questions will have all or parts of the answer to the novel question. As such, the process described in this paper approximates and simplifies the answer generation process by summarizing the answers to the pre-existing questions that are deemed sufficiently similar. As such networks have shown promise in summarization tasks, summarization uses an encoder-decoder neural network to generate answers.

CHAPTER 2

QUESTION-ANSWER SYSTEMS

2.1 Question-Answer Systems

Question-answer systems (abbreviated Q-A systems throughout this paper) are a subset of software systems whose purpose is to provide correct and natural language answers to natural language questions given by human users. These systems are generally classified into one of two categories based upon the type of questions that the system is designed to handle. Open-domain systems attempt to accept and answer questions about any topic. Closed-domain systems, on the other hand, simplify the problem by only focusing on a relatively narrow topic, such as mathematics. Both classes of Q-A systems require a collection of data from which the information in the provided answer can be synthesized. These collections of data are known as **knowledge bases**. Older systems and modern simple approaches to the problem have used manually compiled, highly-structured knowledge bases that are composed of facts or logical relations (Russell and Norvig, 2016). More modern approaches commonly elect to utilize unstructured natural language sources as knowledge bases. This drastically reduces the resource and time cost of generating the knowledge bases, but it also increases the difficulty involved in retrieving and adapting the information. The experiment detailed in this paper attempts to create a closed-domain system that uses a natural language source as the knowledge base.

2.2 Related Work

Early closed-domain Q-A systems were relatively effective and accurate at answering questions. LUNAR, a system that answered questions about moon rocks, had an accuracy

of approximately 90% when questioned by users with little or no training in how to use the system (Woods and Kaplan, 1977). These systems often relied heavily upon the extreme specificity of their domain and well-structured knowledge bases that were hand-compiled by domain experts. Later systems attempted to improve the generalizability of these systems by using pre-existing natural language knowledge bases, such as Wikipedia (Chen et al., 2017) or the internet as a whole (Banko et al., 2002). With regards to using online forums as this paper does, some work has been done on retrieving question-answer pairs (Cong et al., 2010) as well as the context of those pairs (Ding et al., 2008). This work can be put to good use in creating new, somewhat structured, knowledge bases from the forums, but no previous work could be found that directly uses forums themselves as knowledge bases.

CHAPTER 3

ONLINE FORUMS

3.1 Online Forum Structure

For the purposes of this paper, an online **forum** is defined as a website or a subset of pages on a website on which users are given the capability of discussing specific topics in a primarily text-based medium. Forums can be focused on a particular topic, such as in the case of customer support forums for companies or products, but they may instead be more generalized to support discussion on numerous topics. Forums in both categories are often partitioned into individual **subforums** that are typically intended to discuss more specific topics than the overall forum. For instance, a customer support forum on a company's website may include subforums intended for the discussion of specific products produced by the said company. Subforums may also be created to provide different forms of discussion without specifying a more targeted topic. An example of this would be a forum for a particular product including separate subforums for asking questions about the products and making future feature requests. Both subforums would be discussing the same overall topic, but they would engender entirely different types of discussions. Forums and subforums are generally further subdivided into **threads** in which discussion is targeted at a very specific topic. In a feature request subforum, for example, a new thread will be created for every unique feature that is requested. Threads are further subdivided into **posts**, which are the smallest unit of discussion in which a user makes an individual contribution to the discussion. Posts typically include text, but they may be allowed to include images, videos, hyperlinks, or other visual features. Posts are also generally ordered according to some metric, most commonly the time at which the post was created or a score assigned by the

website maintainers or users.

3.2 Usefulness in Q-A Systems

Forums may be usable as a valuable substitute for more traditionally created knowledge bases when creating Q-A systems or automated customer support systems. Most of these potential benefits come from the crowd-sourced nature of most online forums. From the perspective of someone designing such a system, an online forum is effectively autonomously created, saving time and resources that would otherwise have gone towards compiling a comparable knowledge base manually. Forums are also inherently adaptive, as new information is added to them every time the users decide that the information is needed or relevant. Finally, due to the sheer number of users that can contribute to a forum, both the breadth and depth of information available to the system may far exceed what can be included in a more traditional knowledge base. The crowd-sourced nature does, however, introduce some potential weaknesses. Likely the most important drawback is the inability to ensure both correctness and helpfulness in the data provided. This is because bad actors may pollute the dataset by posting incorrect or irrelevant information throughout the forum. Proper moderation and the introduction of systems for ranking posts based upon their relevance or correctness may mitigate the effects of these bad actors.

There are a number of features that may have an effect on how helpful an individual forum can be for building a Q-A or customer support system. The size of the forum (measured in the number of threads or posts) will have a direct effect on the amount of information that a forum is capable of supplying. In addition, the number of unique users that contribute to the forum may also have an effect on the amount of information available. A less obvious feature of a forum that may be important is the culture of that forum. The culture of a forum is largely the result of both the tendencies of the majority of users and the efforts of a team of moderators, who are generally users employed or selected by the website maintainers that are tasked with upholding the rules of the forum. Cultural features that

could affect the usefulness of a forum include whether posts are generally serious or whimsical in nature, whether the forum is geared towards collecting and disseminating facts or opinions, and whether or not the forum allows multiple threads to be created with the same topic. The final feature listed (known colloquially as "reposting" or posting "duplicates" in many forums) is likely especially important for this paper due to the key assumption that new questions given to a Q-A system will likely be similar in topic to one or more posts already present within the forum. Having multiple posts available with similar topics may allow questions that are similar, but not identical, to pre-existing posts to pull in information from multiple posts as needed to synthesize the desired answer.

CHAPTER 4

NEURAL NETWORKS

4.1 Neural Networks

Neural networks are a machine learning technique that are modelled after the workings of the human brain. The overall goal of any neural network is to learn a function that accurately maps a set of input values to another set of output values. More specifically, they attempt to build functions that can probabilistically decide what output should be given when the model is given a combination of inputs that the model may not have seen before. Neural networks generally are built to fulfill one of two basic tasks: classification or generation. Classification tasks attempt to determine which of a set of categories the input should be considered a member. Generation tasks seek to give an output based on the input. In practice, these tasks are accomplished by classifying an input and returning the value that represents the chosen category. For instance, if a network is intended to generate a word based on a given input, the model will generally work by classifying what word (i.e. category) the input is associated with.

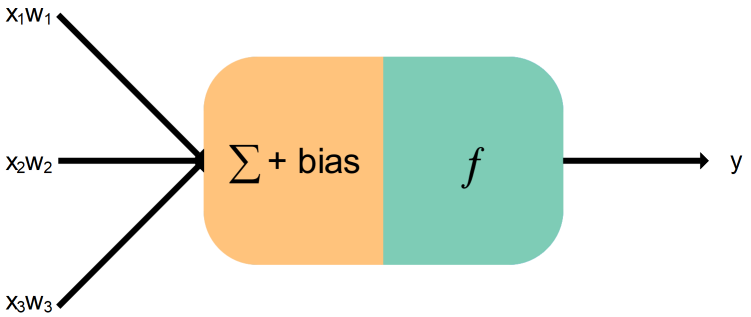

Figure 4.1. Anatomy of a single artificial neuron

Neural networks are composed of a sequence of **layers**, with each layer being comprised of a collection of **artificial neurons**. An artificial neuron (shown in Figure 4.1)

accepts as input the output x_i from one or more neurons in the previous layer and multiplies each x_i by a weight w_i. In the most basic type of layer (known as a **dense layer**), each neuron in the current layer receives input from every neuron in the previous layer. These weighted inputs are then summed together and added to the neuron's bias value. Finally, the resulting sum is given to an **activation function**, the result of which is passed out of the neuron to the next layer. Activation functions are used to allow networks to learn more complicated, non-linear functions. While there are numerous activation functions, only two are used in this experiment, linear, sigmoid, and ReLU (shown in Figure 4.2).

$$f(x) = x$$
(a) Linear

$$f(x) = \frac{1}{1+e^{-x}}$$
(b) Signmoid

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$$
(c) ReLU

Figure 4.2. Activation Functions

The main goal of training neural networks is to make changes to the weights and biases stored in each neuron in such a way that future inputs are more likely to yield the correct output. To do this neural networks need two pieces of information during training: a set of inputs and the expected output for each of those inputs. When training, neural nets will generate an output that is compared to the expected output to calculate the **loss** of the network. The network then uses a technique known as backpropagation (Goodfellow et al., 2016) to update the weights in all layers by an amount that is determined by the calculated loss. It is also standard practice during training to withhold two subsets of all of the data on which the model is being trained, called the testing and validation sets. The testing set is kept entirely separate from the training data and is used to evaluate the model after training is complete. The validation set is used to evaluate the model throughout training in order to make sure that the model is training correctly and is not becoming overfit, which is the tendency of machine learning models to become far more accurate with inputs in the training set than with inputs that are not present in the training set.

4.2 Recurrent Neural Networks

Neural networks in their basic form are not sufficient for many problems. For example, there is no way for a normal neural network to recognize and utilize the relationship between sequential inputs. This is very important for a large number of tasks because many tasks can only be accurately completed if each decision is made without considering all or many of the previous inputs. As an example, a sentence is given its meaning by the words that it contains, as well as the ordering of those words. "Alice pushed Bob" does not have the same meaning as "Bob pushed Alice", but a network incapable of capturing order information would see them as the same sentence. In order to get around this limitation, more complicated networks like **Recurrent Neural Networks** (also known as RNNs) can be used.

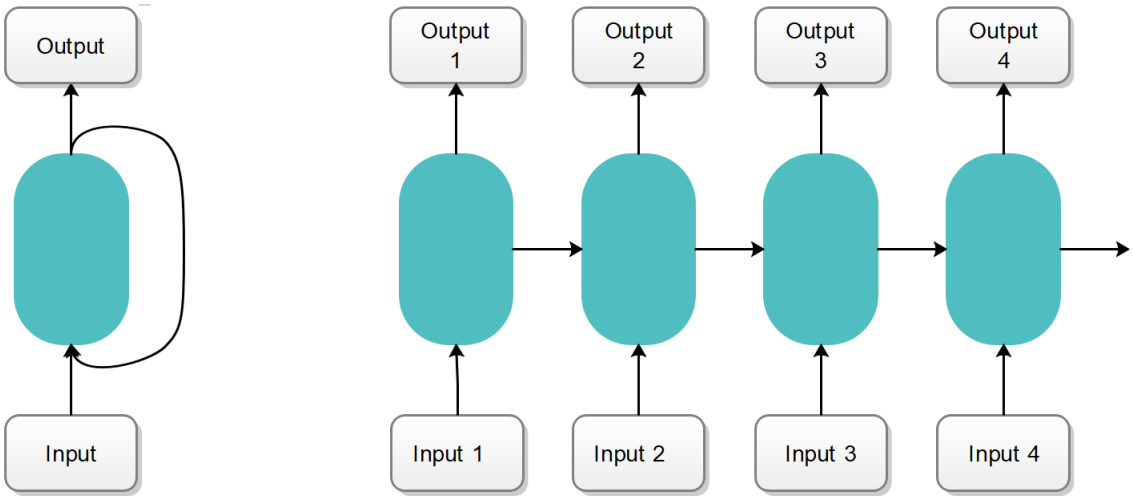

Figure 4.3. Left: Basic structure of a typical RNN. Right: Typical RNN unrolled into individual timesteps

RNNs (illustrated in Figure 4.3) are neural networks in which the input to the network is comprised of both the current actual input, as well as the output from the previous input timestep. This allows the network to consider information found within the entire input sequence rather than just the most recent input. Not only that, but it also allows the network to utilize information about the order of the previous timesteps, which was not previously possible. With our previous example, an RNN will be able to tell the difference between "Alice pushed Bob" and "Bob pushed Alice". RNNs are also capable of being

combined with other networks to make more complicated and effective models. An example of this that is utilized in this experiment is the **Bidirectional RNN**. Bidirectional RNNs can be used when the entire input sequence is known immediately. It works by running the input sequence through two RNNs, but one in reverse order from the other, before combining the outputs at each input timestep together. For instance, RNN-1 would be given the input "I drove a car", while RNN-2 would be given "car a drove I", but the output from the first timestep for RNN-1 would be combined with the output from the last timestep for RNN-2 because both are the timesteps that consider the input "I". Bidirectional RNNs allow for the network to use information from the entire sequence at every timestep, as opposed to normal RNNs that only allow information from previous timesteps to be utilized.

Unfortunately, RNNs have a few issues that must be resolved. One of the biggest issues is that RNNs tend to use information from the most recent timesteps more than information from earlier timesteps, regardless of what information is more important. For instance, a model that is intended to predict changes in the stock market may be biased towards predicting that values will increase because the recent trend has been upwards, despite early information in the sequence suggesting that the prices are likely to decrease after a certain period of increase. A common strategy to address this issue is to use a more complex derivative of RNNs, such as **Long Short-Term Memroy** (LSTM) units. These are a variation of RNNs where each neuron learns not only its weights and biases, but it also learns how to determine what information from the input sequence should be remembered. With these units, old but important information can have as much as or more of an effect on the output as recent but unimportant information. LSTMs have shown to perform better on many tasks than vanilla RNNs (Hochreiter and Schmidhuber, 1997).

CHAPTER 5

DISTRIBUTED WORD EMBEDDING

The input to a neural network is traditionally one or more items of simple structure, where each item is the smallest unit of meaningful information available. For instance, a given network may accept as input a list of integers, characters, or booleans without any issues. However, many domains, such as natural language processing, have more complex units. In natural language processing, for example, the smallest meaningful unit is typically a single word, i.e. a list of characters. Neural networks have difficulty dealing with these composite inputs directly. **Embeddings** work to solve this issue by providing a mapping between a composite input and a relatively simpler and easier to use embedding.

For natural language processing, the simplest form of embedding available is known as the **one-hot vector**. One-hot vectors are vectors whose length is equal to the size of your model's vocabulary and that are composed almost entirely of zeros with a single element given the value of one. The non-zero element is at the index that has been assigned to the input word. For example, if the word "car" has been given the (one-indexed) index of three, the one-hot vector representing the word "car" would be [0,0,1,0,0,0,...]. Unfortunately, one-hot vectors are necessarily both extremely high-dimensional and information-sparse. In order to fix this, modern natural language systems typically use **distributed word embeddings**.

With distributed word embeddings, words are uniquely mapped to a fixed length vector in which every value can be non-zero number. For example, the embedding for "car" may be something along the lines of [0.234, 1.234, -4.324, 1.234, 0.023, ...]. This allows for the word to be represented in a far denser and shorter vector without sacrificing uniqueness. In addition, distributed word embeddings are typically learned by a neural network rather

than arbitrarily assigned, giving rise to another helpful feature. Embeddings are generally learned based upon the context (i.e. the words surrounding the word in question) in which the words are used. Because of this, words that are used in similar contexts tend to have embeddings that are more similar than words used in different contexts. Put simply, this means that distributed word embeddings usually describe the word that they represent, rather than just identifying them. As a result, distributed embeddings allow us to compare inputs and quantitatively analyze their similarity (Mikolov et al., 2013), using standard approaches, such as Euclidean distance or cosine similarity. With the example earlier of the word "car", it is very likely that an appropriately trained embedding model, the word "car" and "bus" will have more similar embeddings than "car" and "boat", while the embeddings of all three words will likely be more similar to each other than any of them will be to the embedding of the word "tree".

For some applications, it may be beneficial to use even larger and more complicated inputs. For example, comparing two documents for their overall similarity can be made a simple and efficient task if the documents can be converted into an appropriate distributed embedding. For the purposes of this discussion, a document is any natural language collection of words, such as a sentence. a paragraph, or an entire book. The Q-A system described in this paper relies heavily on this ability during the first phase of execution in which questions contained in the knowledge base that are similar to the input question are retrieved. In the case of this paper, a model known as Doc2Vec (Le and Mikolov, 2014) was used. Rather than attempt to learn document embeddings directly, this model learns word embeddings and combines the word embeddings together using a combination of averaging and concatenation to yield an embedding for the entire document. This gives the model the ability to generate appropriate embeddings for inputs that were not in the training corpus, which is an ability most word embedding models cannot provide.

CHAPTER 6

ENCODER-DECODER ARCHITECTURES

6.1 Motivation

Encoder-Decoder models are an RNN architecture designed specifically for generating an output sequence given another sequence as an input. They are commonly used for a variety of such tasks, including translation (Bahdanau et al., 2014), summarization (Rush et al., 2015; Paulus et al., 2017), and image captioning (Xu et al., 2015). Traditional RNN architectures generate a single output at every input timestep. This has two major implications. First, a standard RNN's output must be of the same length as the input or must be of length one. Second, standard RNNs are incapable of using information from late in the input sequence to generate early output tokens. This severely limits the effectiveness of RNNs for many problems. For instance, when translating the phrase "I took a photograph" from the Japanese "Watashi wa shashin wo torimashita", a normal RNN would be incapable of correctly translating the phrase because the correct output phrase is shorter than the input. Furthermore, the closest English equivalent to the verb "torimashita" ("took") would need to be generated before the model has been made aware of the presence of the Japanese word in the input. Encoder-Decoder models solve both of these issues by iterating over the entire input sequence before generating any output at all.

6.2 Encoder-Decoder Architecture

As the name suggests, Encoder-Decoder models (illustrated in Figure 6.1 are comprised of two stages. Both stages are primarily composed of a separate RNN. The first stage is the encoder stage. Encoders take, as input, each token in the input sequence and

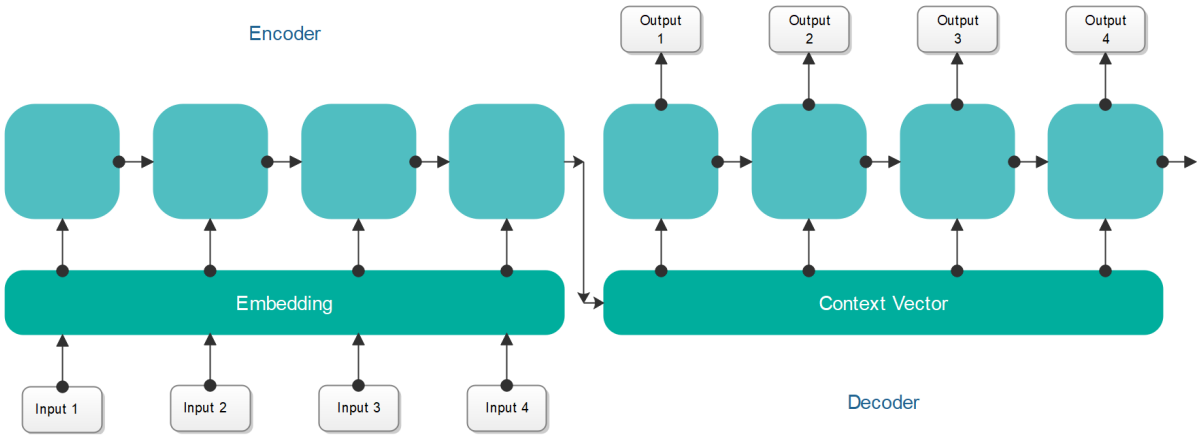

Figure 6.1. Encoder-Decoder Model

generate, as output, a single fixed-length vector. This fixed length vector, which is known as the **context vector**, represents all of the information found in the input sequence. Decoders, the second stage in the model, use this context vector and the previous tokens in the output sequence to generate each output token. Because the context vector is available at all timesteps during the decoding stage, the decoder is able to use information from any position in the input to determine the appropriate output at the current output timestep. In addition, the encoder and decoder are separate RNNs and thus the number of output tokens from the decoder is not dependent on the number of inputs to the encoder. Despite the encoder and decoder being comprised of entirely separate RNNs, it is generally important that the two be trained together as a single unit to ensure that they correctly learn how to generate and interpret the context vectors respectively.

6.3 Attention

Encoder-decoders alone are not without their weaknesses. Because all information in the input sequence must be squeezed into a single fixed length vector, the information from individual sections of the input tends to have little effect on the context vector when the input sequence becomes large. As such, the decoder may have difficulty extracting the specific input information necessary at a given output timestep. Attention attempts to solve

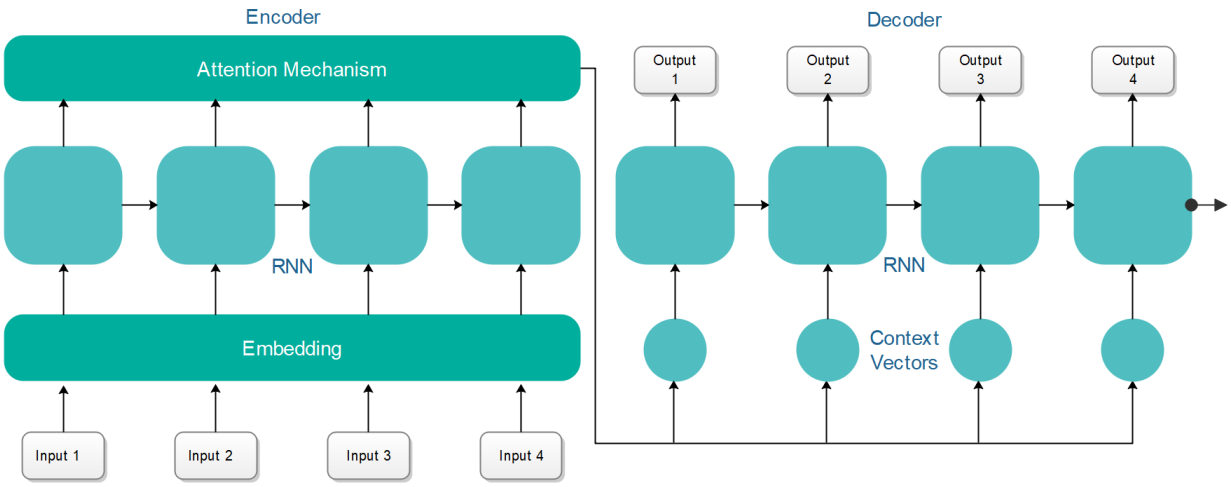

Figure 6.2. Encoder-Decoder Model with Attention

this issue by altering the context vector based upon what portion of the input is relevant for the current timestep in the decoder (Bahdanau et al., 2014). At each timestep, the attention mechanism determines how relevant a given timestep in the input and weights its contribution to the context vector accordingly. In practice, this weighting is performed on the context vector itself after it has already been produced by the encoder, but the effective result is a sequence of slightly different context vectors, with the sequence length being equal to the number of tokens produced by the decoder. This is illustrated in Figure 6.2. With an attention mechanism, the accuracy of the model is less affected by the length of the input. Attention also has the added benefit of reducing repetitiveness, a common problem in sequence-to-sequence models, because the context vector changes at each timestep, allowing models to view numerous locations in the input as important and diversify the source of information used for each output timestep.

CHAPTER 7

EXPERIMENT SETUP

7.1 Data Collection and Preprocessing

This section details the process of collecting and cleaning all data used for training and evaluating the Q-A model. It is split into subsections that detail the methods for each dataset individually, as well as a subsection that explains the cleaning to which all data was subjected. Statistics about the data collected are listed in Table 7.1

Dataset	**Questions**	**Answers**	**Ratio**	**Total**
Reddit	2745	2745	1 : 1	5490
StackExchange	9080	19889	1 : 2.19	28969

Table 7.1. Number of datapoints collected from both data sources

7.1.1 Reddit

Data was obtained from Reddit via the Reddit API, specifically using the PRAW python library (Boe et al., 2017). Reddit's API only allows for 1000 threads to be retrieved per request and requests are significantly limited to the granularity of the allowed search parameters. For instance, there is no supported method for requesting threads from a particular time period. Instead, the maximum number of threads possible were retrieved from four broad subforum categories. The categories used were hot (new threads that have received large amounts of attention from users), top (the threads with the highest user-provided score), top of year (the top threads created within the last 365 days), and controversial (threads that have received approximately equal numbers of positive and negative votes

All source code can be found at `https://github.com/KyleAMoore/QA-FKB`

```
SELECT (Id,Title,Body,AcceptedAnswerId,ParentId,Score)
    FROM Posts
        WHERE (ParentId IS NOT Null OR
            Id IN (SELCT ParentId FROM Posts))
```

Figure 7.1. SQL query for downloading StackExchange Data.

from users). After duplicate threads and threads with no replies (answers) were removed, the resulting dataset was comprised of 2195 threads. Because of the relatively large number of posts on the average Reddit thread and due to the final process only utilizing the top answer post in each thread, exactly one reply was retrieved for each thread. The reply kept was the highest rated reply in each thread and was used as the ground truth answer for that question. The AskHistorians subforum was used for this paper due to it being large, primarily text-based, well-moderated, and naturally organized in a question-answer format.

7.1.2 StackExchange

Data was obtained from Stack Exchange via the website's provided database interaction service, known as the StackExchange Data Explorer (StackExchange, 2010). This service allows users to directly query the StackExchange database for posts and data relevant to those posts, such as the score, author, etc using standard SQL queries. The data used in this system was obtained using the SQL code found in Figure 7.1. For each post, this command retrieves the post ID, post title, the ID of the post containing the correct answer if the current post is a question, the ID of the post to which the current post is a reply if the current post is an answer, and the post's community-provided score. Only questions for which an answer has been given were collected. In total, 9080 question posts and 19889 answers posts were obtained, giving an average of 2.19 answers per question. The History subforum was used for this paper due to it being one of the largest primarily text-based subforums on the StackExchange website, in addition to being similar in topic to the chosen Reddit subforum.

7.1.3 Preprocessing

All posts were run through the same basic preprocessing steps (shown in Figure 7.2). Common steps that were necessary were the standardization of character casing and the removal of special characters. Due to the data being retrieved from web-based sources with large amounts of stylization, it was also necessary to remove all mark-up tags. Finally, all posts were tokenized so that all each document was represented as a list of strings representing words as opposed to a single long string.

In addition to the basic preprocessing, the data also needed some additional, separate preprocessing for the system's candidate answer selection module. High frequency, low-information words (a, the, and, etc.) known as stop words were removed. They are removed for this module because the module is heavily based on the coincidence of words in two posts. The inclusion of stop words would, however, cause dissimilar posts to appear more similar than they actually are. For similar reasons, the posts were converted to a lemmatized form for this module. Lemmatization is the process of converting a word to its base form. For example, the words "run", "ran", "running", and "runs" would all be lemmatized to the common base form of "run". Lemmatization is used because systems would recognize the original words as being different words and treat them as completely unrelated, despite having essentially the same meaning. For example, a model would treat the sentences "I am running" and "We ran" to be completely unrelated despite being very similar in meaning. Lemmatization and stop word removal were performed using the resources provided by the Natural Language Toolkit, with the wordnet implementation, in particular, being used for lemmatization (Bird et al., 2001).

Two final preprocessing stages were used. First, every question was given a list that included the ID of all answers associated with that question. This list was ordered such that the first element would be considered the correct answer and each subsequent answer would be considered less correct than the answer before it. Correctness was determined using the community-provided score of the posts, with a higher score indicating higher correctness. An

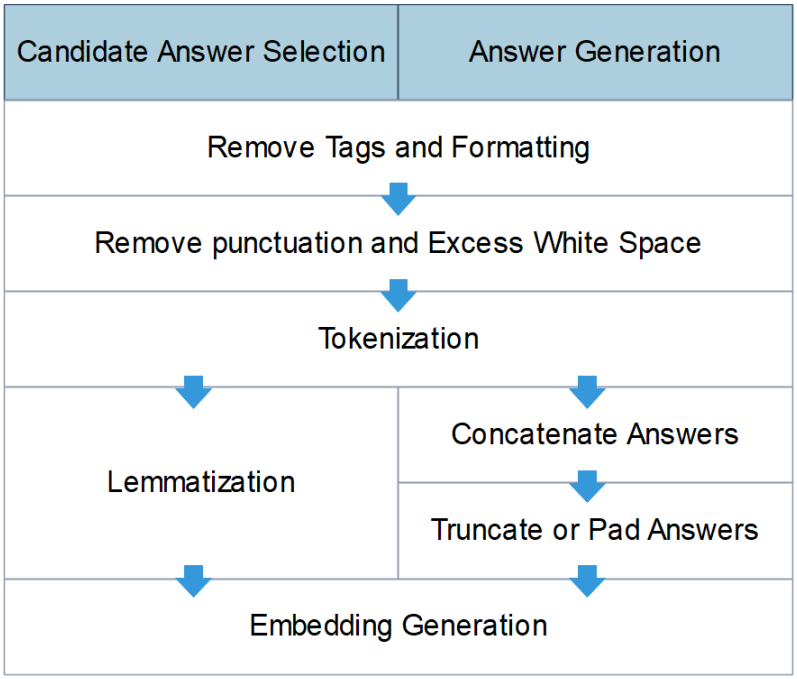

Figure 7.2. Stages of Preprocessing

exception was made for answers marked as accepted answers in the StackExchange dataset, which were always considered the most correct answer. Finally, the questions were randomly split into training and testing sets. As each answer is directly associated with only a single question, the answers were not explicitly split, but they were implicitly split alongside the question set. For all datasets, 20% of the questions were reserved for the testing set.

7.2 Candidate Answer Selection

Because the model is based on summarizing multiple potential answers into a single final answer, the candidate answers must first be selected from the dataset. This module used a relatively simple approach for choosing these candidate answers. First, both the new question and all of the questions from the training set are vectorized using Doc2Vec (as described in chapter 5). Specifically, this experiment used the gensim implementation of Doc2Vec (Řehůřek and Sojka, 2010). The model used to convert each post into a document embedding was trained using only the posts present in the training set and, in order to save time, the embeddings for the training set were precomputed and saved between candidate

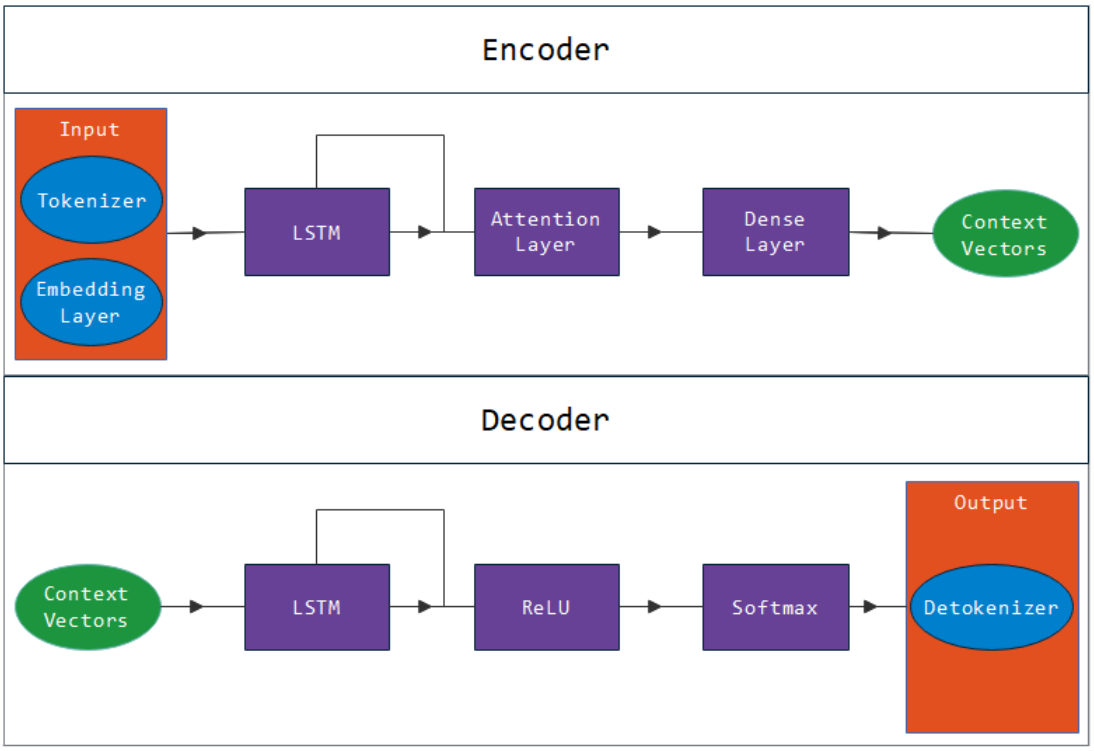

Figure 7.3. Summarization Model Network Architecture

selections. Next, the embedding for the input question was compared to all precomputed embeddings to determine the k most similar questions by computing the Euclidean distance between their embeddings. The Euclidean distance was used in lieu of the more common cosine similarity because the magnitude of the embeddings were relatively small (typically no greater than 10). For the purposes of this experiment, k was set to 5, but it could be altered in future comparative experiments. Once the similar questions were selected, the top answer for each answer was selected and added to the list of candidate answers. Future comparative experiments can investigate the benefits of sampling from the total combined set of candidate answers, likely using the score and status as an accepted answer as weighting parameters.

7.3 Answer Generation

Answer Generation was treated as a summarization task in which the k candidate answers were summarized into a single output answer. This task utilized a standard Encoder-

Decoder architecture illustrated in Figure 7.3. The entire model was defined using the Keras (Chollet et al., 2015) library with TensorFlow (Google, 2015) as the backend for all computations.

7.3.1 Architecture

The encoder consists of four major layers. Before input documents could be passed to the input layer, however, they were given additional preprocessing. First, all k input documents were trimmed to s words in length before being concatenated together. The resulting answer was then padded to length $s * k$ if not already that length. Next, all words were converted to an index in the vocabulary briefly before being converted again to a distributed embedding of length n. The word embedding calculated here is learned and generated by the encoder-decoder model, rather than by the embedding generator used in the Candidate Answer Generation module. The resulting $(s * k) \times n$ matrix is finally given to the model as input.

Every row in the matrix is given to a bidirectional LSTM sequentially, resulting in a context vector for each input timestep that contains information from all previous and subsequent timesteps. A standalone self-attention layer and dense layer are applied to the output, resulting in s context vectors of length n. The activation function of the dense layer used here is unimportant as it is used only for decreasing the dimensionality of the context matrix from $(s * k) \times n$ to $s \times n$. For the sake of simplicity, the actual activation function used in this experiment was linear, meaning that there effectively was no activation function. This $s \times n$ resulting matrix represents the attention-modified context vector to be used at each of the s output timesteps and is both the output of the encoder and the input to the decoder.

Each modified context vector is run through an LSTM individually, with each individual output of the LSTM given to a dense layer using ReLU as its activation function. The LSTM used by the decoder is not bidirectional, unlike the encoder LSTM. This is because

only the output sequence already generated is available at every timestep. The softmax of the resulting vector is computed, with the result representing the probability distribution for each word to be the correct word at the current timestep. An index is categorically sampled from this distribution and appended to the output. Once all s timesteps have completed, the result is a list of s indices, which are then converted back into words using the same dictionary as in the preprocessing stage. The end result is an s-length answer to the original question.

7.3.2 Training

During training, the top answer for the input question was used as the ground truth for calculating the loss of the model. A technical limitation of keras required that the output of the model be the probability distribution for the vocabulary at a given output timestep. As such, during training, every word in the ground-truth sequences was converted into a one-hot vector in order to approximate a probability distribution. Future comparative experiments should test using other values for the hyperparameters k, s, and n. For this experiment, the number of input answers (k) was 5, the length of the final answer (s) was 100, and the length of the context vector (n) was 128. Both models were trained for no more than 50 epochs (iterations over the entire training set) until convergence of the validation loss. In order to avoid local minima, training was stopped early only if the model showed no improvement for two epochs after the best epoch. After early stopping, the weights of the model were restored to the values held at the end of the best epochThe model trained on the StackExchange data and the model trained on the Reddit data were both trained with a batch size of 32. Before training took place, the training data was further split into training and validation sets, with the validation set comprising a random 20% of the training data (16% of the total dataset).

CHAPTER 8

EVALUATION AND RESULTS

8.1 Cross-Validation

Dataset	Epoch	Training Loss			Validation Loss		
		Minimum	Average	Maximum	Minimum	Average	Maximum
Reddit	1	5.897	6.035	6.229	5.891	5.423	5.944
	2	5.206	5.335	5.491	5.765	5.369	5.870
	3	5.182	5.303	5.449	5.746	5.366	5.873
	4	5.148	5.270	5.411	**5.701**	**5.337**	5.869
	5	**5.086**	**5.216**	**5.369**	5.807	5.380	**5.867**
StackExchange	1	5.963	5.998	6.041	5.580	5.640	5.751
	2	5.589	5.613	5.632	5.548	5.603	5.721
	3	5.560	5.580	5.595	**5.538**	**5.602**	5.729
	4	5.529	5.545	5.559	5.585	5.619	**5.715**
	5	**5.468**	**5.482**	**5.495**	5.633	5.672	5.772

Table 8.1. Results of 5-fold cross-validation over the first 5 epochs of training

K-fold cross-validation is the process of training a model k times, using a different portion of the data as the validation set in each iteration. With 3-fold cross-validation, for example, the model is trained three times with a different one-third of the model being reserved for validation each time. table 8.1 contains the results of a 5-fold cross validation with both datasets over five epochs. The data suggest a few features of the model. First, the model tended to converge very quickly, with most versions of the model reaching their minimum validation loss at three to four epochs. In fact, the models generally reach values close to the optimum validation loss after the first epoch. This is possibly a result of an early high gradient from the use of one-hot vectors as ground-truth sequences. An alternative explanation is that the model was consistently converging at a point of underfitting. If

this is the case, it suggests that the dataset used in this experiment was either too small or inappropriate for this task. Secondly, After the minimum validation loss was obtained, any further epochs consistently lowered the training loss and increased the validation loss, indicating that further training would only have caused worsening overfitting. Experiments with significantly higher epoch numbers showed that the training loss continued to approach zero, while the validation loss continues to grow and can reach values greater than 15.

$$\sum_{w \in Answer} \sum_{i}^{Vocab} -ref_i^w log(act_i^w)$$

Figure 8.1. Equation for categorical cross entropy
ref_i^w indicates the value at index i of the one-hot vector represeting position w
act_i^w is the probability that the generated answer has word i at position w

The loss was calculated using categorical cross-entropy. Categorical cross entropy is a variation on normal cross entropy where both the reference and generated outputs have softmax applied prior to being given to the cross-entropy function (shown in Figure 8.1). The losses reported in Table 8.1 are the average loss values for all reference-generated answer pairs across the entire epoch.

8.2 ROUGE

Dataset	Metric	Rouge-1	Rouge-2	Rouge-L	Rouge-W
Reddit	**F1-Score**	15.8	0.24	13.0	3.82
	Precision	14.9	0.23	12.2	5.78
	Recall	21.1	0.29	17.7	6.13
StackExchange	**F1-Score**	18.8	0.38	15.7	4.53
	Precision	17.8	0.36	14.9	7.05
	Recall	20.9	0.42	17.3	3.65

Table 8.2. ROUGE scores for generated answers

ROUGE is a common metric for evaluating natural language generation systems. The algorithm comes in numerous varieties and is centered around calculating the amount of overlap between the expected output passage and the generated output passage. The

most basic version, ROUGE-1, computes the percentage of timesteps in which the generated text has the same word in the same position in the text as in the reference text. ROUGE-N, where N is an integer, is the same as ROUGE-1, except that it compares sequences of N words rather than one word at a time. The basic equation for calculating ROUGE-N scores is shown in Figure 8.2. ROUGE-L calculates the overlap of the longest common subsequences in the two texts. ROUGE-W is similar to ROUGE-L, except that it gives a more favorable score when the longest common subsequences within the two texts are spaced consecutively. The maximum ROUGE score possible is 100 (Lin and Och, 2004).

$$\frac{\sum_{S \in Reference} \sum_{gram_n \in S} Count_{match}(gram_N)}{\sum_{S \in Reference} \sum_{gram_n \in S} Count(gram_N)}$$

Figure 8.2. ROUGE-N

The ROUGE scores for this model are generally low for all metrics except ROUGE-1 and ROUGE-L. This implies that there is a reasonable amount of overlap between the two sentences, but that the overlap rarely lasts for more than one time-step. Due to the variable nature of natural language, however, a low ROUGE score is not enough to determine that the model has failed. This is because two passages of sufficient length can have almost no overlap of specific words while still maintaining comparable meaning. As such, the ROUGE score must be considered in combination with qualitative analysis.

8.3 Qualitative Results

Unfortunately, the generated results are also qualitatively insufficient. The main problem with the generated answers is that they are largely comprised of confused and unstructured sequences of mostly gibberish. Appendix A contains a collection of randomly selected examples of inputs and outputs to both models. The model did not completely fail, however. Unlike many flawed networks, the model typically generated a reasonably low percentage of unknown vocabulary tokens, which are tokens that indicate that the word at the given position should be a word that is not recognized by the model. These tokens

usually indicate that the model has determined that the position should be filled with a rare word that is either a proper noun or only used in very specific niches. However, the Reddit model's outputs included a significantly, though still reasonable, number of unknown vocabulary tokens. This is likely due to the larger vocabulary used on the said website in the form of slang and undetected misspellings.

CHAPTER 9

CONCLUSION

The overall results of this experiment were mostly negative. It isn't entirely clear from the data why, but there are several candidate explanations. The simplest explanation is that the model itself has some unidentified issue. Unfortunately, this is difficult to determine due to the lack of a well-established baseline for this task. Despite this problem being modelled as a summarization task, the task is different in a subtle, but potentially important way. In a typical summarization task, the source text and the reference summaries are necessarily similar in both content and meaning. With the experiment setup used in this paper, it was hypothesized that the same would be generally true for Q-A forums, but that may not necessarily be the case. This means that the usual, established baselines, such as the use of DUC datasets, are not quite comparable when trying to determine the usefulness of forums.

The main alternative explanation is that the datasets used were insufficient for this task, either because of the specific datasets chosen or because forums are not actually appropriate for this task. A major assumption made about the problem was that the answers to questions similar to the input question would also be similar to the input question's answers, but this assumption may be flawed. It may be worth investigating whether this assumption is accurate. If it turns out that the assumption was flawed, then it is likely that the source texts used to generate the summaries did not include the information required by the reference summary, likely contributing to the failure of this model. In addition, this problem might have been exacerbated by the small size of the datasets, which would have caused there to be too few pre-existing questions that are sufficiently similar to the questions in the testing set.

It is also possible that the unstructured nature of the output was partially a result of the data used, rather than the model itself. Forums can be extremely varied in their structure and style, even between threads in the same subforum. In addition, the datasets used were relatively small for sequence generation research, but approximately typical for all but the largest of online forums. Both of these could have made it much more difficult for the model to learn the grammatical rules necessary to generate coherent outputs. This is mildly supported by the decreased occurrence of unknown vocabulary tokens in the StackExchange model, which used a dataset approximately three times the size of the Reddit dataset.

In conclusion, it would require further research to pinpoint exactly what caused this experiment's negative results and whether the initial hypothesis should be rejected, but the data is currently not promising.

BIBLIOGRAPHY

BIBLIOGRAPHY

Bahdanau, D., K. Cho, and Y. Bengio (2014), Neural machine translation by jointly learning to align and translate, *arXiv preprint arXiv:1409.0473*.

Banko, M., E. Brill, S. Dumais, and J. Lin (2002), Askmsr: Question answering using the worldwide web, in *Proceedings of 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*, pp. 7–9.

Bird, S., E. Loper, and E. Klein (2001), Natural language toolkit, http://nltk.org.

Boe, B., L. Roth, R. Goodman, E. Dalool, and J. Hill (2017), Python reddit api wrapper, https://praw.readthedocs.io/en/latest/index.html.

Chen, D., A. Fisch, J. Weston, and A. Bordes (2017), Reading wikipedia to answer open-domain questions, *arXiv preprint arXiv:1704.00051*.

Chollet, F., et al. (2015), Keras, https://keras.io.

Cong, G., C.-Y. Lin, and S. Ding (2010), Summarizing online forums into question-context-answer triples, Google Patents, uS Patent App. 12/207,231.

Ding, S., G. Cong, C.-Y. Lin, and X. Zhu (2008), Using conditional random fields to extract contexts and answers of questions from online forums, *Proceedings of ACL-08: HLT*, pp. 710–718.

Goodfellow, I., Y. Bengio, and A. Courville (2016), *Deep learning*, 197-216 pp., MIT press.

Google (2015), Tensorflow, https://tensorflow.org.

Hochreiter, S., and J. Schmidhuber (1997), Long short-term memory, *Neural computation*, 9(8), 1735–1780.

Le, Q., and T. Mikolov (2014), Distributed representations of sentences and documents, in *International conference on machine learning*, pp. 1188–1196.

Lin, C.-Y., and F. J. Och (2004), Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics, in *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, p. 605, Association for Computational Linguistics.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013), Efficient estimation of word representations in vector space, *arXiv preprint arXiv:1301.3781*.

Paulus, R., C. Xiong, and R. Socher (2017), A deep reinforced model for abstractive summarization, *arXiv preprint arXiv:1705.04304*.

Řehůřek, R., and P. Sojka (2010), Software Framework for Topic Modelling with Large Corpora, in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, ELRA, Valletta, Malta.

Rush, A. M., S. Chopra, and J. Weston (2015), A neural attention model for abstractive sentence summarization, *arXiv preprint arXiv:1509.00685*.

Russell, S. J., and P. Norvig (2016), *Artificial intelligence: a modern approach*, Malaysia; Pearson Education Limited,.

StackExchange (2010), Stack exchange data explorer, http://data.stackexchange.com.

Woods, W. A., and R. Kaplan (1977), Lunar rocks in natural english: Explorations in natural language question answering, *Linguistic structures processing*, 5, 521–569.

Xu, K., J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio (2015), Show, attend and tell: Neural image caption generation with visual attention, in *International conference on machine learning*, pp. 2048–2057.

APPENDICES

APPENDIX A

Listed below are 4 randomly selected example results from each model. All examples are split into three pieces, the original question, the reference answer, and the generated answer and the three are listed in that order. Questions are marked with "Q:", correct answers are marked with "E:", and the outputs from the system are marked with "A:". For the sake of brevity, all entries have been truncated to 200 characters for display in the appendix only.

Question ID	Text
2	Q: why do presidents have the power to pardon? what were the founding fathers arguments for giving this power vs not?
	E: it is effectively a check on the judiciary branch in practice and a means by which the framers sought to allow the government to show mercy . that said this was a controversial inclusion into the cons...
	A: always if UNK city you was the edited one be one UNK swords was since consider the do UNK has UNK UNK is UNK UNK UNK guess century UNK for UNK with product flair a UNK that disallowed UNK in a removed...
240	Q: how sick was john f . kennedy? and how aware was the american public of jfks physical trouble?
	E: kennedys high school roommate used to joke if i ever wrote a biography about him i would call it john f . kennedy a medical history . he certainly wouldnt be lacking for material . but as kennedys act...
	A: on removed everyone it always much food listed UNK those although this unlike UNK UNK UNK malcolm browse UNK the UNK is to of portrayed while be in UNK get UNK send or of various since and them histor...
443	Q: i think i may have found a native american rock mound what should i do and who should i tell? first off sorry if this is the wrong sub but i thought maybe historians would know how to handle something...
	E: kentucky will have some sort of state archaeologist find out who that is and contract their office .
	A: run UNK in think in one odd deleted if had goal going william UNK an UNK UNK get western UNK UNK century UNK UNK UNK UNK UNK UNK or UNK upvotes UNK emperor native prove UNK UNK UNK of have UNK in taki...
511	Q: why are there so many smiths out there? were there just a shitload of blacksmiths in medieval times or did the blacksmiths just produce a prodigious amount of progeny? or why is smith such a common su...
	E: there are a lot of kinds of smiths . there are silversmiths blacksmiths coppersmiths aka greensmith goldsmiths and the names relating to the items being made shoesmith edit as in horseshoes knifesmith...
	A: hi would about UNK india right UNK right has that UNK was has had first UNK UNK legal some so been a UNK UNK UNK in was right of unfounded of should stand the dont states century a of after UNK armed...

Table A.1. Reddit

Question ID	Text
11	Q: has the united states officially paid reparations in the form of us dollars to any parties that were a child civil liberties act of 1988 public law 100383âaug . 101988 public law 100383 100th congress...
	E: the firelands was part of the western reserve of connecticut in the northwest territory . this area is now part of ohio and it was reserved specifically for people burned out of their homes in connect...
	A: for UNK on because united something armor scope against that was russian the is UNK not were context own UNK met to going and UNK on politics was 2 the the from the UNK would f our on stalingrad us at...
292	Q: what evidence exists to indicate that the pope attempted to suppress the number zero? in an editorial review rob lightner claims that zero the biography of a dangerous idea by charles seife explains w...
	E: according to this article pope sylvester ii gerbert daurillac c9431003 is credited with reintroducing the abacus into europe without an explicit use of the number zero . this was because it had not be...
	A: UNK during there of out the development taxation but census not of russia afterwards had UNK and as events description UNK guns in gathered first in stone slightly UNK tradition way in within UNK to a...
1011	Q: the alta california missions built in the spanish period were initially run by franciscans then secularized in the 1830s becoming regular parish churches . some lay people officially served to adminis...
	E: the mission comisionados and mayordomos were appointed by the monterey junta and by governors principally figueroa and alvarado or by the mission inspector hartnell . according to carlos salomon in pi...
	A: according UNK to led of the rank is inscriptions most as because crusaders it between greece of very to UNK since UNK a village recommend the the june the are on UNK UNK german main the did had the ha...
1812	Q: i recently came across this picture removed now look at wiki article instead saying that it is a scaffolding erected during ww2 over taj mahal to protect it from japanese air force . even the wikipedi...
	E: eliminating the distinctive white dome would be only one part of a camouflage plan . its a big and delicate job and has to be done in advance . all other elements of the camouflage plan can be done wh...
	A: owned there operation into of engage UNK UNK UNK out a difference the UNK old gave appear keep the sense you conflict UNK the down start hundred UNK UNK the at UNK probably soviet UNK general as to UN...

Table A.2. StackExchange

VITA

EDUCATION

Bachelor of Science (May 2017) in Computer Science, University of Mississippi. Graduated Magna Cum Laude.

ACADEMIC EMPLOYMENT

Graduate Teaching Assistant, Department of Computer and Information Science, University of Mississippi, August 2017 - May 2019. Responsibilities include grading, tutoring, assisting the professor with coursework preparation, and substitute lecturing when needed.

PROFESSIONAL MEMBERSHIP

Association for Computing Machinery

Upsilon Pi Epsilon