Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

Abstractive Summarization of Social Media Posts : A Case Study Using Deep Learning

Master's Thesis

Shahbaz Syed
Born Jun 24, 1990 in Hyderabad

Matriculation Number 115594

1. Referee: Prof. Dr. Benno Stein
2. Referee: Prof. Dr. Andreas Jakoby

Submission date: August 28, 2017

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, August 28, 2017

. . . .
Shahbaz Syed

Abstract

Abstractive summarization is difficult to accomplish with standard natural language processing techniques. With the advancements in the field of deep learning, and availability of suitable datasets, various deep learning models have been used to generate and improve the quality of abstractive summaries (Chopra et al. [2016], Nallapati et al. [2016], Rush et al. [2015]). One such model is the sequence to sequence model mainly used in neural machine translation (Sutskever et al. [2014]) . The most commonly used datasets for summarization such as DUC, Gigaword Corpus and TAC documents are well structured, short articles from specific domains which represent only one of the many styles of written text. Such documents -often written by professionals-are well suited for neural networks to understand the syntax and generate well structured summaries. However, it is more difficult for these networks to work with poorly written, or unstructured text such as tweets or social media posts for example. In this thesis, we construct a new corpus (Webis-TLDR-17 Corpus) out of posts from the popular Reddit social media site. We evaluate the quality of summaries generated by neural network on various subsets of this corpus. We also present the limitations of the sequence to sequence model in abstractive summarization that are reflected in some of the summaries generated.

Contents

Chapter 1

Introduction

Automatic summarization is the task of reducing a text document or a larger corpus of multiple documents into a short set of sentences or paragraphs that conveys the main meaning of the entire text. Some of the key challenges in generating good summaries for a text are reducing redundancy, preserving order of information, consistency of the generated sentences, and remembering information through longer sequences which make automatic summarization a rather difficult task.

1.1 Types of Summaries

Generally speaking, the goal of a summary is to compress the source document; however, summaries can have a variety of different aspects and intentions as shown in Figure 1.1. Described below are some of the most common types of summaries according to the external factors involved in generating them (Torres-Moreno [2014]).

1.1.1 According to Function

Summaries can be indicative or informative or varying degrees of both. An indicative summary provides information about the topics discussed in the source document. In some cases, it resembles a table of contents, providing an overview of the main content. An informative summary aims to reflect on the content of the source text, possibly explaining the arguments made in the text.

1.1.2 According to Number of Documents

We distinguish single document or multi-document summaries. The latter extract important information from a larger body of documents, usually from a specific domain such as medicine, computer science etc. In addition to the order of sentences, multi-document summaries must also keep track of the order of the documents, for example using time-stamp information.

1.1.3 According to Genre of Documents

A summary can be a news summary (news articles, media reports); specialized to a specific domain such as law, science or technology; a summary of literary or historical texts; an encyclopedic extract (such as a Wikipedia article); or a social media summary of tweets, Reddit posts, or Facebook posts.

1.1.4 According to Type of Summarizer

Summaries can also be classified based on the nature of the entity producing the summary. An *author summary* is written by the author of the source document that reflects her point of view. An *expert summary* is written by someone other than the author who specializes in the corresponding domain, but might not specialize in generating summaries. A *professional summary* on the other hand is written by a professional summarizer who probably does not specialize in the corresponding domain, but is an expert in writing techniques, and the norms and standards of producing summaries. Finally, an *automatic summary* is a machine generated summary usually using extractive methods of identifying keywords, keyphrases from the source document (e.g. Hovy and Lin [1998]).

1.1.5 According to Context of the Summary

A *generic summary* provides a neutral overview of the source document, irrespective of the reader's information need. A *query-guided summary* is guided by an information need as expressed through queries. An *update summary* is needed in the scenario where users have already read some documents related to a particular topic and have become familiar with it. Update summaries therefore aim to show only new information and avoid repeating old information.

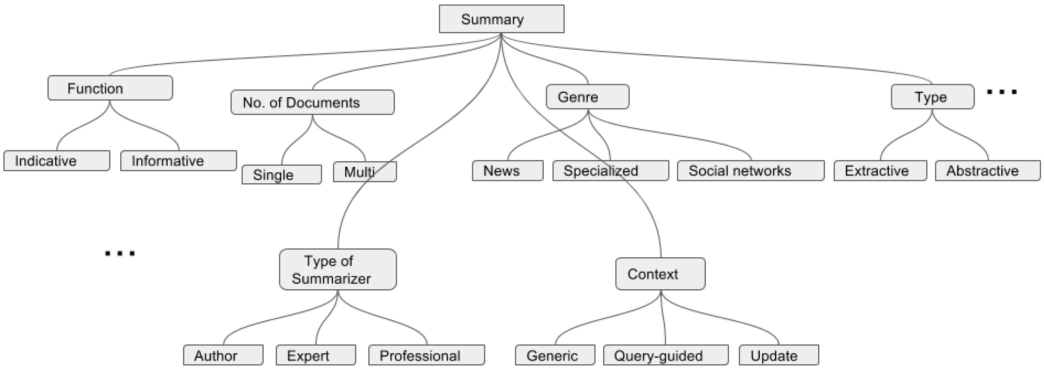


Figure 1.1: Types of summaries

1.2 Approaches to Summarization

Text summarization approaches can be broadly classified into two categories: extractive and abstractive. Extractive techniques mainly consist of extracting relevant sentences from the original document. The extracted sentences are then combined based on some salience measure, in order to form a compressed and robust summary containing words and features from the original text.

Abstractive summarization on the other hand aims to generate a more natural, intuitively reformed summary of the original text. It builds upon an internal semantic representation of the text, and may consist of words and phrases not present in the original document, which are generated using extensive natural language processing techniques. Consider the example text below from a news article :

The Army Corps of Engineers, rushing to meet President Bush's promise to protect New Orleans by the start of the 2006 hurricane season, installed defective flood-control pumps last year despite warnings from its own expert that the equipment would fail during a storm, according to documents obtained by The Associated Press. Wikipedia [2017b]

An extractive keyphrase extractor might select "Army Corps of Engineers", "President Bush", "New Orleans", and "defective flood-control pumps" as keyphrases that are pulled directly from the text.

In contrast, an abstractive keyphrase system would somehow internalize the content and generate keyphrases that might be more descriptive and more like what a human would produce, such as “political negligence” or “inadequate protection from floods”. Note that these terms do not appear in the text and require a deep understanding, which makes it difficult for a computer to produce such keyphrases. As such, abstractive summarization is much more complex than extractive summarization, and requires advanced machine learning techniques. With the advances in the field of deep learning, neural networks have been extensively used in understanding and representing text for natural language processing.

Before delving deep into abstractive summarization, it is important to understand the various categories of extractive summarization approaches(Gambhir and Gupta [2017]) as shown in Figure 1.3. This is because extractive approaches can serve as a baseline for evaluating more complex systems built in the abstractive summarization process. The basic concepts of identifying and extracting relevant information from a document have been well explored in the extractive approaches to summarization.

1.2.1 Stastical Based Approach

This approach is usually language independent, as it deals with identification and extraction of relevant parts of the source documents based on statistical scoring measures. Some of the key statistical features which can be extracted from a document are the ordering of sentences, keywords based on term frequency, the centrality or similarity of a sentence to other sentences within the document, its resemblance to the title of the document, relative length, and presence of numerical data or named entities. These features have been used as foundations of many natural language understanding processes and thus provide a standard vocabulary for the research community to collaborate. Figure 1.2 depicts the whole process of a statistical based approach to summarization.

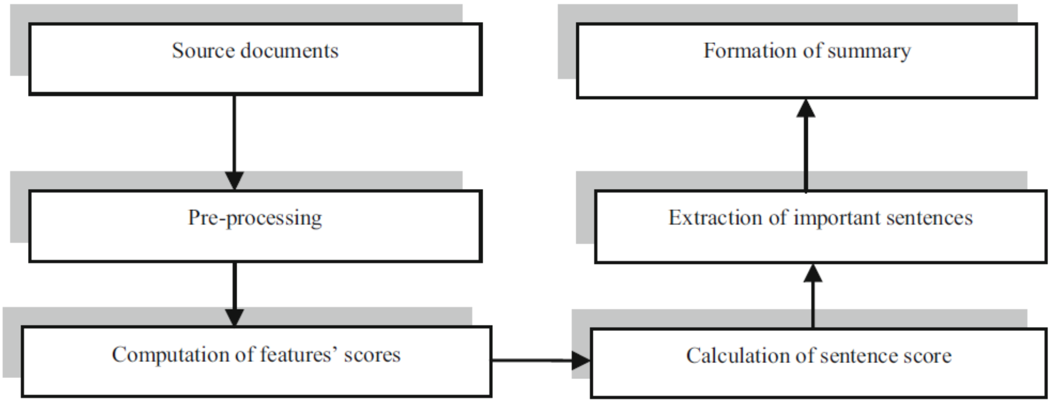

Source documents
Pre-processing
Computation of features' scores
Calculation of sentence score
Extraction of important sentences
Formation of summary

Figure 1.2: Statistical approach to summarizing documents(Gambhir and Gupta [2017])

1.2.2 Graph Based Approach

Graphs are traditionally used to represent the topology of elements in a given space. Summarization can benefit from adapting this approach by representing documents and sentences as nodes. Two such popular graph based summarization systems are *LexRank*(Erkan and Radev [2004]) which is a multi-document summarization system where candidate sentences are shown in the graph with an edge between similar sentences, and *GRAPHSUM* (Baralis et al. [2013]) which represents correlations among multiple terms by discovering association rules among them.

1.2.3 Machine Learning Based Approach

Machine learning has advanced the field of summarization by a significant amount, eliminating the mundane tasks of hand-coding statistical features, and sorting texts. Supervised machine learning approaches such as Support Vector Machines (Fattah [2014]), Naive Bayes classification (Fattah [2014]), Regression (Fattah and Ren [2009]), Decision Trees and Neural networks (Fattah and Ren [2009]) use documents and their corresponding human generated summaries in order to learn to produce summaries for new documents.

Unsupervised approaches on the other hand such as clustering (Yang et al. [2014]), Hidden Markov Model, genetic algorithms (Mendoza et al. [2014]) learn from unlabelled data and apply heuristics to select, mutate and combine important features to produce coherent summaries.

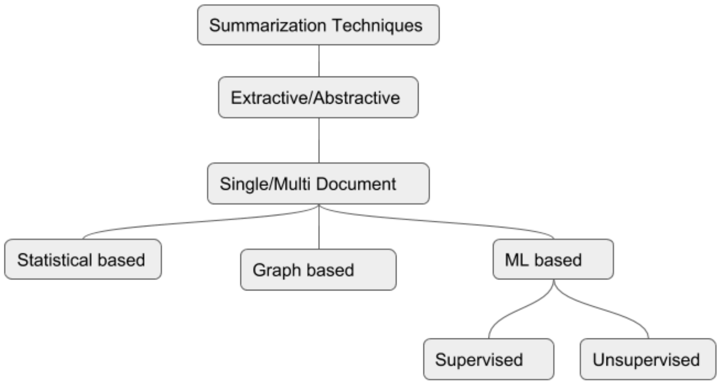


Figure 1.3: Approaches for extractive summarization

1.3 Neural Networks for Summarization

Neural networks were first applied for abstractive text summarization by Rush et al. [2015]. Their model was based on the encoder-decoder architecture proposed by Cho et al. [2014b] and exhibited state-of-the-art performance on sentence level abstractive summarization using a feed-forward neural language model. This model also encouraged enhancements to the basic encoder-decoder architecture to improve the quality of the generated summaries, by using a convolutional attention-based conditional recurrent neural network (Chopra et al. [2016]), and attention-based sequence to sequence recurrent neural network (Nallapati et al. [2016]). Table 1.1 gives the performance of these models as evaluated by the ROUGE package on the generated abstractive summaries for the DUC-2004 test set. Furthermore, in Section 4.2.1 we compare the scores from Rush et al. [2015] to manual evaluation of the summaries generated in our work and argue that ROUGE might not be the best evaluation measure for judging abstractive summaries.

Table 1.1: Performance of existing approaches to abstractive summarization

Model	ROUGE1	ROUGE2	ROUGE-L
Rush et al. [2015]	28.18	8.49	23.81
Chopra et al. [2016]	**28.97**	8.26	24.06
Nallapati et al. [2016]	28.61	**9.42**	**25.24**

1.4 Datasets for Summarization

Table 1.2 shows some of the popular datasets used in various summarization tasks. It is important to note that this list is by no means complete as new datasets are being regularly created and refined by the research community.

English Gigaword Corpus
This corpus consists of approximately 10 million news articles along with their headlines, extracted from 7 popular news agencies : Agence France-Presse (English service), Associated Press Worldstream (English service), Central News Agency of Taiwan (English service), Los Angeles Times/ Washington Post Newswire Service,Washington Post/ Bloomberg Newswire Service, New York Times Newswire Service, Xinhua News Agency (English service). This is the most popular corpus being used in modern abstractive summarization research. It is available for a license fee from Linguistic Data Consortium (LDC).

DUC(2001-2007)
The Document Understanding Conference hosted by the National Institute of Standards Technology (NIST, U.S Department of Commerce) is popular in the NLP research community. All the corpora published for summarization tasks during these years are available on a request basis through proper registration. These contain multiple clusters of variable number of documents in each. Documents can be related to a variety of domains such a news, computer science, engineering and technology.

TAC(2008-2011)
In the year 2008, the DUC became a summarization track in the Text Analysis Conference. Similar to the DUC datasets, they are available on a request only basis.

Ziff-Davis Corpus
This consists of 13,000 newspaper articles announcing computer products. In

addition to this, approximately 30,000 news articles from the Wall Street Journal for the year 1987 were also used in the SUMMARIST system (Hovy and Lin [1998]) for automatic summarization. This is the only system we are aware of that has used this corpus.

TIPSTER Corpus

TIPSTER also sometimes known as TREC is another text corpus from LDC. It has approximately 33,000 documents consisting of text published from several magazines about computers, hardware, software etc.

Blog Dataset

This consists of 1475 blog posts and their comments collected from Cosmic-Variance and InternetExplorer blog.

Enron Email Dataset

The Enron email dataset contains approximately 500,000 emails generated by employees of the Enron Corporation. It was obtained by the Federal Energy Regulatory Commission during its investigation of Enron's collapse. True summaries for relevant emails were generated by human annotators as described in Carenini et al. [2008].

W3C Corpus

This dataset was derived from a crawl of World Wide Web Consortium's sites at w3c.org. The data include mailing lists, public webpages, and text derived from various file formats such as .pdf, .doc and .ppt files. The mailing list subset is comprised of nearly 200,000 documents, and TREC participants have provided thread structure based on reply-to relations and subject overlap. There are more than 50,000 threads in total. W3C data has been annotated for QA topic relevance for use in TREC Enterprise 2005, 2006.

LCSTS Dataset

Next to the English resources listed in Table 1.2, the LCSTS dataset collected by Hu et al. [2015] is perhaps closest to our own work both in terms of text genre and collection method. Their dataset comprises approximately 2.5 million content-summary pairs collected from the Chinese social media platform Weibo, a service similar to Twitter in that a post is limited to 140 characters. Weibo users frequently start their posts with a short summary in brackets.

CNN/Daily Mail Dataset

This dataset was originally created for the task of passage-based question answering by Hermann et al. [2015]. This corpus was later modified by Nallapati et al. [2016] to be used for summarization. It consists of approximately a million news articles with multi-sentence summaries extracted using the human generated abstractive summary bullets for each article. This dataset is quickly being adopted alongside the Gigaword Corpus in more recent abstractive summarization research (Nallapati et al. [2016], See et al. [2017]).

Table 1.2: Summarization datasets

Begin of Table

Dataset	**Mention**	**Purpose**	**Size**
English Gigaword Corpus	Rush et al. [2015],Nallapati et al. [2016],Chopra et al. [2016]	Abstractive summarization to generate headlines for news articles	10 Million
DUC (2005)	Ye et al. [2007]	Constructing a space of concepts of given documents for selecting important sentences for summarization	200-600
DUC (2006)	Lee et al. [2009]	Developing an unsupervised approach for Non-negative Matrix Factorization (NMF)	200-600
DUC (2007)	Ouyang et al. [2011]	Applying Support Vector Regression(SVR) to select important sentences	200-600
DUC (2005,2007)	Alguliev et al. [2011]	Applying Integer Linear Optimization to select key sentences, minimzing redundancy	200-600

Continuation of Table 1.2

Dataset	Mention	Purpose	Size
DUC (2002)	Alguliev et al. [2013]	Adapting evolutionary algorithms to select, mutate and generate important sentences	200-600
TAC (2008,2009)	Heu et al. [2015]	Multi-document summarization through cluster analysis	200-600
Ziff-Davis Corpus	Hovy and Lin [1998]	Robust automated text summarization system	43,000
TIPSTER Corpus	Neto et al. [2002]	Extraction of statistical and linguistic features from text	33,000
Blog dataset	Ferreira et al. [2013]	Quantitative and Qualitative assessment of 15 algorithms available in literature for sentence scoring	1475
Enron Email Dataset, W3C Corpus	Ulrich and Murray [2008]	Generating annotated corpus for email summarization task	500,000
LCSTS dataset	Hu et al. [2015]	Construction of chinese text corpus for summarization	2.5 Million
CNN/Daily Mail Dataset	Nallapati et al. [2016],See et al. [2017]	Abstractive summarization using RNN based sequence to sequence models	1 Million

End of Table

1.5 Limitations of Existing Datasets

As already mentioned, one of the key challenges for neural networks in dealing with language is understanding unstructured/informal text. The datasets mentioned in Table 1.2 are mostly built from news articles, and tend to contain properly written, domain specific texts usually drafted by professionals such as journalists. Although helpful, such data is not enough to extensively explore the capabilities of an automatic summarization system. With the increase in the amount of information posted on social media platforms, text which is mostly unstructured and whose various representations can be interpreted only by humans can prove to be quite a challenge for a regular neural network to work with. We argue that a dataset from a forum such as Reddit, where users communicate informally and discuss everyday topics can greatly help the research community. Thus, we have endeavored to clean and extract a usable dataset specifically suited for the task of abstractive summarization. Our efforts benefit from the common practice of social media users summarizing their own posts as a courtesy to their readers: the abbreviation *tl;dr* ("too long, didn't read") is followed by a summary of the entire preceding content. This provides us with the text and its summary - both written by the same person - which is an excellent datum for developing and evaluating an automatic summarization system. Furthermore, using some heuristics we can further extract sub-datasets from this main dataset suited for training neural networks specific to generating different types of summaries.

Chapter 2

The Webis-TLDR-17 Corpus

This chapter describes the construction of the Webis-TLDR-17 corpus, a new resource for abstractive summarization derived from a large set of Reddit posts spanning the past ten years.

Reddit is a social news aggregation, web content rating and discussion website (Wikipedia [2017a]). Its registered community members can submit content consisting of text posts or web links. The next section explains the anatomy and the hierarchy of posts on Reddit.

2.1 Anatomy of Reddit

Reddit is simply a message board wherein users can submit content, which is also curated by the community. Users upvote worthy posts making them move up on the forum and thus increasing their visibility. Content is segregated into channels called subreddits, for example Technology, Gaming, Finance, Well-being etc. Users can subscribe (and unsubscribe) to relevant subreddits, following which they get customized content pushed to their home page. Each subreddit is a small community centered around a specific topic, where a user (author) *submits* a post and concerned users *comment* on this. Comments are the key and perhaps the most entertaining and informative parts of a post where users can summarize, self-regulate, contradict, or support a discussion. Submitting a link is as simple as adding a title to the post, providing a URL if it is a weblink, writing text (for self posts), and choosing an appropriate subreddit.

We are mainly concerned with text posts for our summarization task, and thus shall discuss their nature in more detail. As already mentioned posts are of two types, namely Submissions and Comments. Submissions are usually large texts discussing in depth about a topic. They have longer bodies averaging around 300 words, while the comments posted by readers on these submissions are usually more terse. Both submissions and comments can contain a summary of the content, written after the abbreviation tl;dr. It is not mandatory for a post to include a tl;dr, but it has become common practice on such forums where users often want a quick summary of the discussion topic without having to read the whole thread. We believe this tl;dr text can be regarded as the summary of the whole post and thus use this (*content*, *tl;dr*) pair of each entry to train the neural network in the next stage.

2.2 Corpus Construction and Availability

The raw data was originally crawled from the Reddit API by an NLP researcher and was first announced on Reddit itself [1] [2]. Currently, this corpus is also available on Google BigQuery[3] for online analysis. The original corpus consists of approximately 1.66 billion Comments from 2007-2015 and 286 million Submissions from 2006-2016.
For a detailed example of a Submission and Comment extracted from the Reddit API in the JSON format, please refer to Appendix C.

2.3 Creating the Base Dataset

As part of the text normalization process, we removed any URLs from the text. We also discovered that many users preferred markdown formatting in their posts. As this introduced raw characters such as *,#, ‘ in the text, we needed to clean up the markdown formatting from the body text. We setup the following 4 step pipeline of consecutive filtering steps. Table 2.1 shows the effect each step had on the number of posts still remaining under consideration. The process used to extract a clean dataset suitable for the summarization task from the crawled data is as follows:

Step 1. Check for Presence of tl.{0,3}dr
An initial investigation showed that the spelling of tl;dr is not uniform, but many plausible variants are indicative of true summaries. To boil down the raw dataset to an upper bound of submissions and comments that are candidates

[1] https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/

[2] https://www.reddit.com/r/datasets/comments/3mg812/full_reddit_submission_corpus_now_available_2006/

[3] https://pushshift.io/using-bigquery-with-reddit-data/

for our corpus, we first filtered all posts that contain the two letter sequences 'tl' and 'dr' in that order, case-insensitive, allowing for up to three random letters in-between. This included a lot of instances found within URLs, which were thus ignored by default.

Step 2. Remove Posts by Bots

The Reddit community has developed many bots for purposes such as content moderation, advertisement or entertainment. Posts by these bots are often well formatted but redundant and irrelevant to the topic at hand. To ensure we collect only posts made by human users - critically, some Reddit users operate tl;dr-bots that produce automatic summaries, which may introduce undesirable noise - we filter out all bot accounts with the help of an extensive bot list[4] provided by the Reddit community, as well as manual inspection of cases where the user name contained the substring 'bot'.

Step 3. Preserve the Highly Specific tl.{0,3}dr Patterns

We manually reviewed a number of example posts for all of the 100 most-frequent spelling variants (covering 90% of the distribution) and found 33 variants to be highly specific to actual tl;dr summaries,[5] whereas the remainder - mostly the less frequent variants - contained too much noise to be of use.
Additionally, in order to achieve a high precision, we considered the posts which only had a single occurrence of the tl;dr pattern in them. We observed through manual inspection that cases with multiple tl;dr occurrences were relatively small in number and mostly contained irrelevant summaries to the main content. To ensure high precision of the extracted content-summary pairs from the single occurrence cases, we perform the following additional filtering steps for each pair. A pair that satisfies either one of the conditions is considered a valid pair.
(i) Check if the content ends with a proper punctuation such as . ! ?
(ii) Check if the summary begins with a capital letter or a number
(iii) Check if the summary begins on a new line
(iv) Check if the tl;dr pattern is not preceded by the letter **a** , as we observed that in such cases, users asked for **a tl;dr to the content** instead of the text following the tl;dr being an actual summary of the content.

Step 4. Filter Pairs where Content is Shorter than Summary

For the remaining posts, we attempt to split their bodies at the expression

[4] `https://www.reddit.com/r/autowikibot/wiki/redditbots`

[5] tl dr, tl;dr, tldr, tl:dr, tl/dr, tl; dr, tl,dr, tl, dr, tl-dr, tl'dr, tl: dr, tl.dr, tl ; dr, tl_dr, tldr;dr, tl ;dr, tl\dr, tl/ dr, tld:dr, tl;;dr, tltl;dr, tl~dr, tl / dr, tl :dr, tl - dr, tl\\dr, tl. dr, tl:;dr, tl|dr, tl;sdr, tll;dr, tl : dr, tld;dr

tl;dr to form the content-summary pairs for our corpus. We locate the position of the tl.{0,3}dr pattern in each post, and split the text into two parts at this point, the part before being considered as the content, and the part following as the summary. In this step, we apply a small set of rules to remove erroneous cases: the length of a tl;dr must be shorter than that of the content, there must be at least 2 words in the content and 1 word in tl;dr. The last rule is very lenient; any other threshold would be artificial (i.e., a 10 word sentence may still be summarizable in 2 words). However, future users of our corpus probably might have more conservative thresholds in mind. We hence provide a subset with a 200 word content threshold.

Table 2.1: Filtering steps for creating Webis-TLDR-17 corpus

Filtering Step	**Submissions**	**Comments**
Raw Input	286,168,475	1,659,361,727
Contains tl.{0,3}dr	2,079,035	3,745,320
Non-bot post	1,969,413	3,351,597
Contains a single valid tl;dr	1,829,030	2,987,135
Final Pairs	1,658,567	2,189,763

2.4 Sub-Datasets for Deep Learning

Extracting the base dataset provides us with valid and cleanly separated content-summary pairs to be used for summarization. However, since neural networks form internal semantic representations based on their training data, we split the full set of content-summary pairs into subsets according to a few verticals/intentions expressed by the data. This allows us to train vertical-specific models to summarize different kinds of texts with different intentions. In our data exploration process, we observed that Reddit users write summaries with various intentions, such as asking for help (tl;dr as a question); making judgements or forming conclusions on the post; unrelated, abstract or introspective tl;drs not representing real summaries; and some vulgar or abusive summaries. We decided to extract multiple sub-datasets from the base dataset in order to test the nature of summaries generated by the neural network for each intention separately. We hope that this experiment will give us a better understanding of the summarization process of a neural network, which may help us in tuning and filtering unrelated, biased or vulgar content from the final summary. Consequently, our sub-datasets are described as follows:

Questions
To extract questions, we selected a set of 21 English question words and checked for their presence, and a question mark in the summaries. The question words were obtained from the Interrogative word Wikipedia page[6], and additional words such as “can”, “should”, “would”, “is”, “could”, “does”, “will” were added to this list after manual analysis of the dataset. We observed that the presence of these words in the beginning of a sentence ending with a question mark constituted proper questions.

True Summaries
While the question of whether a summary is related to a given text is subjective, we treat a true summary as something that talks about the topic or subject of the text. To implement this notion, we extracted noun phrases from both content and summary, and checked for summaries having at least one noun phrase in common with the content. The choice of noun phrases as the deciding factor is based on the assumption that people relate information as an action (verb phrase) happening to, or performed by an entity (noun phrase).

Topic + tl;dr
It was observed that for the Submissions dataset, each post has a corresponding title field in its JSON, which represents what the post is talking about. We attach this title to the summary for each content-summary pair fed to the neural network in order to test if the neural network can generate a topic for the text in addition to its summary.

Topic
As a slight variation of the above dataset, we used only the title of the post instead of the summary to test if the model can accurately generate the topic of the post itself. We observed that the title contained important words from the content, which itself was similar to an abstract summary.

[6] https://en.wikipedia.org/wiki/Interrogative_word

Vulgar Posts

As mentioned previously, many posts consisted of abusive words in either the content or the tl;dr or both. We extracted the pairs containing abusive words only in the tl;dr to examine if the neural network can learn to generate offensive summaries for a given text. We use a list of more than 500 English offensive words from Google's now defunct ”What do you love?” project[7] to extract this sub-dataset. Information about the size of each sub-dataset is presented in Table 2.2

Table 2.2: Vertical specific sub-datasets

Nature	**Number of pairs**
Questions	78,710
True Summaries	966,430
Topic + tl;dr	729,042
Topic	729,042
Vulgar Posts	299,145

[7] https://gist.github.com/jamiew/1112488

Chapter 3

Deep Learning Models for Summarization

By using multiple layers of nonlinear units, deep learning architectures can understand and represent more complex features for the input data. This helps us to learn multiple levels of representations that correspond to different levels of abstraction; these levels form a hierarchy of concepts. Let us look into one such architecture known as Recurrent neural networks which are very effective for working with text.

3.1 Recurrent Neural Networks

Before looking at the relatively complex sequence-to-sequence model used in this thesis, we need to understand its building blocks. Humans understand text as a sequence of characters read either from left to right or right to left, depending on the language. In order to understand text, a machine needs to process its input in a similar, sequential fashion. Recurrent neural networks (RNN) are one model for processing such sequential data. They do this by sharing parameters among different parts of the model across time, hence making them effective at dealing with sequences. Each output is a function of the previous output, which means that the current state of the network is dependent on its previous states. This is analogous to a sentence, where the meaning or contribution of the current word is dependent on the previously seen words.

If we look at an RNN as a computational graph, the unfolding of this graph as shown in Figure 3.1 denotes the information flow from one state to another, with the final state representing an indirect summary of the whole sequence. This is because the final state was generated as a result of its previous states, and so on, thus partially containing and manipulating information from the beginning of the sequence till the end. This ability of representing and understanding a sequence by RNN has greatly advanced the application of neural networks in the field of NLP.

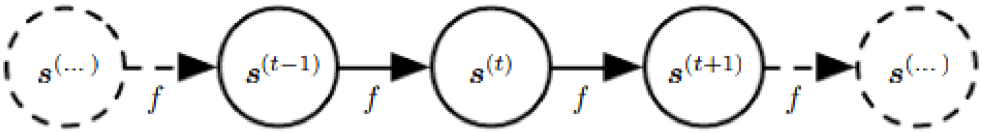


Figure 3.1: State generation in a dynamic system (Goodfellow et al. [2016])

Understanding the source sequence is a multi-step process for a RNN. We specify the input of the RNN by providing an embedded representation of our text (e.g word embeddings). The RNN has many more states which are not specified by us, and in fact learned by the network itself. These are called hidden states of the RNN as shown in Figure 3.2. A RNN has many hidden states and layers of such states which learn different parts of the sequence, and understand it gradually in a step by step fashion. For example, the first layer and its hidden states may learn all the nouns and verbs in the sentence, the second layer and its hidden states may learn to combine these into phrases and so on, eventually understanding and creating a representation of the whole sequence.

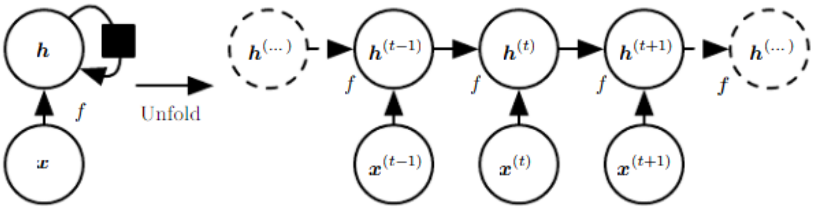


Figure 3.2: Hidden states in a RNN(Goodfellow et al. [2016])

During the training phase, the RNN computes the loss on a sequence with respect to the given target and back propagates this error in order to learn the best set of parameters (weights and biases) possible for the model in order to minimize its error on the test set.

3.2 Architectures of Recurrent Neural Networks

As RNNs are good at processing sequences, it is very common to use them as building blocks to form more complex architectures. Let us discuss a few of the most commonly used architectures of recurrent neural networks that are the building blocks of our sequence to sequence model discussed in Section 3.3.

RNN Generating a Single Output
As shown in Figure 3.3, these networks read an entire sequence and produce a fixed size representation as output, which can be intuitively seen as generating a summary of the whole sequence. This final vector is usually referred to as the *context*. This vector is compared to the target, and a loss function computes the loss on such a given pair of sequences. This error is then back-propagated into the network from the final layer to the first, in order to learn the best set of parameters for the model.

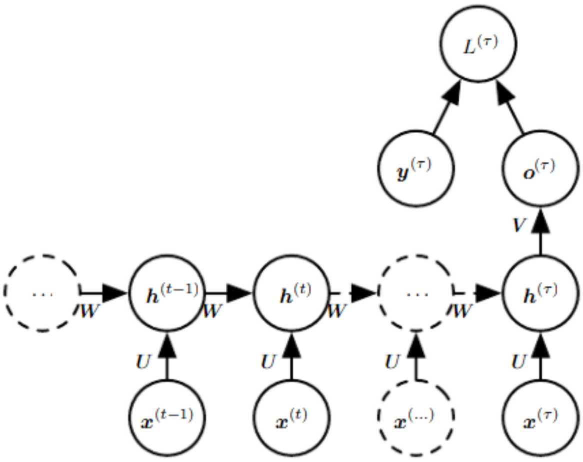


Figure 3.3: Unfolded representation of a RNN that produces a single vector as output, usually known as the context(Goodfellow et al. [2016])

RNN Conditioned on a Single Context Vector
An RNN can map a fixed length vector to a distribution over sequences a depicted in Figure 3.4. Such an architecture is appropriate for tasks such as image captioning, where a single image is used as input to a model that then produces a sequence of words describing the image.

Bidirectional RNN
This architecture has two recurrences in it from left to right (forward pass)

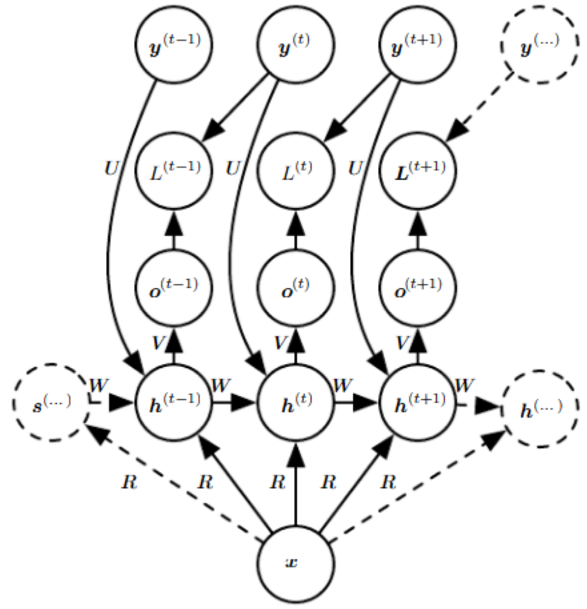


Figure 3.4: RNN taking a single context vector as input(Goodfellow et al. [2016])

and right to left (backward pass) of the sequence as shown in Figure 3.5. This makes each state of the RNN aware of both the past and the future states in the sequence. Bidirectional RNNs are commonly used in modules such as encoder-decoder networks for generating sequences from sequences.

3.2.1 Gated Units for Enhancing RNN

One of the main limitations of the vanilla RNN is its inability to deal with longer sequences due to the vanishing/exploding gradient problem (Bengio et al. [1994]). As the sequence grows longer, the information remembered by the network grows smaller due to the absence of a linear flow of the gradient during back propagation. Consider the scenario of gradient propagation in a RNN as shown in Figure 3.6.

In this case unless the gradient dh_{t-1}/dh_t is exactly one or reaches a constant, the value of the gradient dl/dh_t rapidly increases (exploding gradient) or decreases (vanishing) due to constantly being multiplied by the previous gradients as each step is back-propagated. This is due to the main non-linear function being a composition of simpler functions, which makes its derivative a product of derivatives of each of its constituent functions. This vanishing/exploding

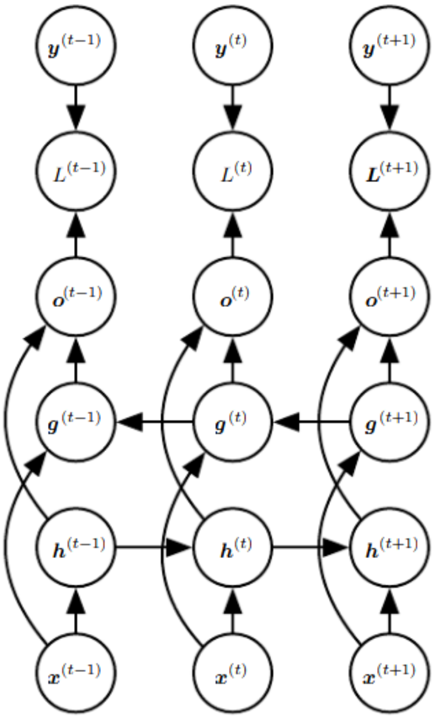


Figure 3.5: A bidirectional RNN(Goodfellow et al. [2016])

nature of the gradient makes learning and updating information impossible, as the network progresses. Thus we need a way to create a path through this chain of functions, where the gradient can flow as a constant without diminishing or exploding. Gated recurrent neural networks are one example of such an architecture that provides a linear path for the gradient to flow during the back-propagation. They have a *gating mechanism* that controls/modifies the amount of information flowing through the hidden states. Such a mechanism can be seen as analogous to a water pipe with multiple valves, which can provide a linear path for gradient propagation which mitigates the vanishing/exploding gradient problem to some extent. Each gate is parameterized by its own weights and biases which the network learns and updates during the training phase. One of the most popular gated RNN used currently is the LSTM or Long Short Term Memory unit(Hochreiter and Schmidhuber [1997]).

3.2.2 Long Short Term Memory Unit

LSTM has a similar chaining structure like a vanilla RNN but the repeating unit itself is different in architecture. It consists of 4 internal networks which communicate with each other in order to selectively read, write, update and

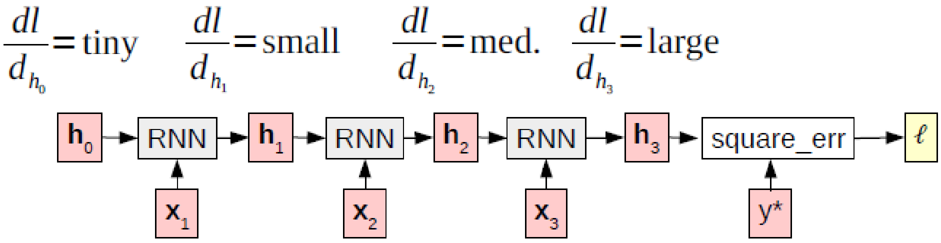

$\frac{dl}{d_{h_0}}$ = tiny
$\frac{dl}{d_{h_1}}$ = small
$\frac{dl}{d_{h_2}}$ = med.
$\frac{dl}{d_{h_3}}$ = large
h_0
RNN
h_1
RNN
h_2
RNN
h_3
square_err
ℓ
x_1
x_2
x_3
y*

Figure 3.6: Gradient propagation in a RNN(Neubig [2017])

forget information of one memory cell. This introduces the idea of adding self loops to the cells that produce paths where the gradient can flow for a longer duration. Weights on these self loops are conditioned on the context, rather than being fixed values.

As seen in Figure 3.7, LSTM cells have internal recurrence also known as a *self-loop* in addition to the outer recurrence of a regular RNN. Let us try to understand the functionality of each gate inside an LSTM cell as shown in Figure 3.9.

Input Gate

Input at each stage is the content of memory cell from the previous states, or a random initialization vector for the first time step. This input undergoes an element wise multiplication to regulate its impact, with the signal from the forget gate. The final result is then added to the current memory cell.

Forget Gate

The forget gate is a one layer neural network whose inputs are : (1) the previous hidden state, (2) the current input embedding, (3) the memory from the previous block and (4) a bias vector. A sigmoid activation function decides whether information from the previous states should be remembered or not. A value of 1 means *preserve completely* and 0 to *forget completely.* For example, in language modeling we would like to remember the gender of a subject in order to produce proper pronouns in the next state, and forget the gender of the older subject when we see a new subject.

Memory Gate

The memory gate takes the same inputs as the forget gate except different bias vectors, and controls the influence of old memory on the new memory. The new memory itself is generated by another neural network, which uses the tanh

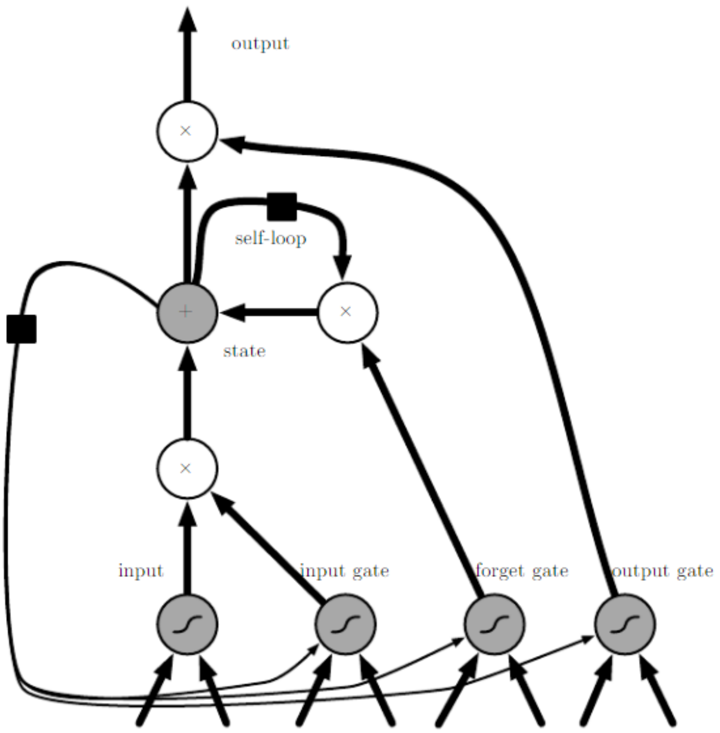


Figure 3.7: Block diagram of LSTM cell. The self-loop can be seen as providing an additional *memory* to the cell(Goodfellow et al. [2016])

activation function. The output of this network is element wise multiplied with the previous hidden state, and the result is added to the old memory to form the new memory.

Output Gate
The output gate takes as inputs : (1) the new memory, (2) previous hidden state and (3) a bias vector and controls amount of the current new memory forwarded to the next LSTM cell in the RNN.

An overview of all the various activation functions inside an LSTM cell can be seen in Figure 3.8. The mathematical equations representing all the gates and their operations in the LSTM are shown in the Figure 3.10. The input gate i and the output gate o use σ function to provide a gating mechanism controlling the amount of information flowing through the cell. The memory c is equal to the current memory, plus the previous memory regulated by the forget gate f. This addition operation makes it easier to maintain a constant derivative from one memory cell to another, thereby alleviating the vanishing/exploding

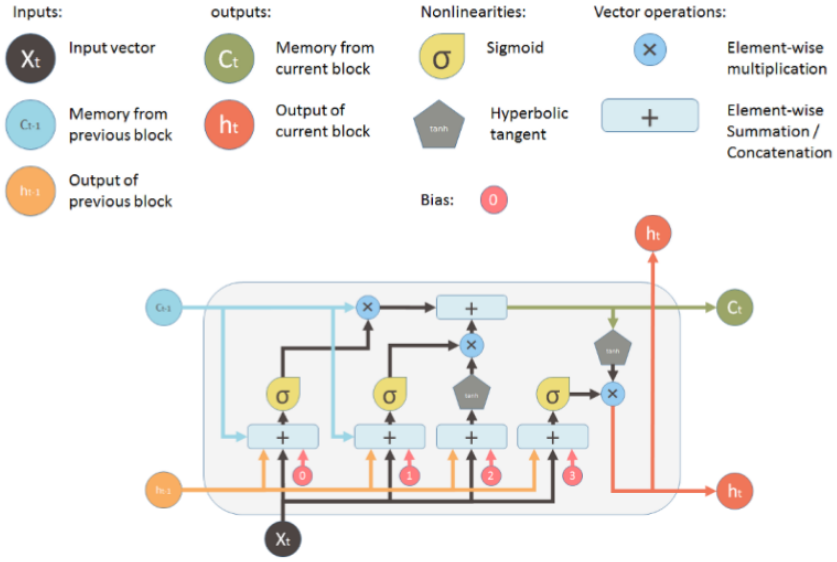
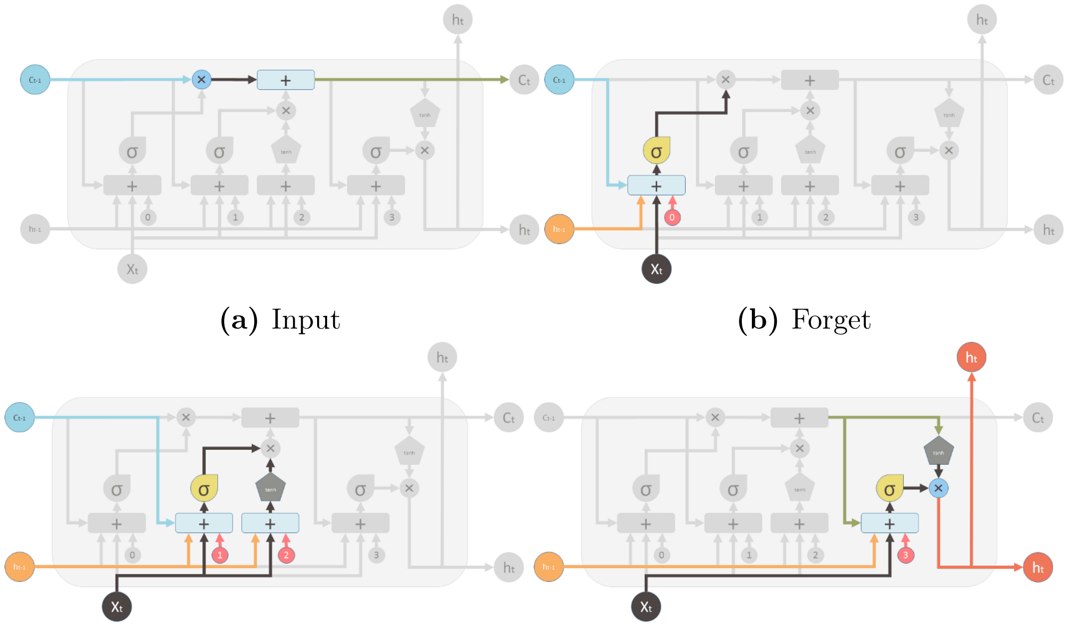


Figure 3.8: Activations for each gate inside an LSTM cell. The final output h_t is propagated to the next LSTM cell in the same layer and the cells of the layer above, in case of stacked architectures (Yan [2016])

gradient problem. Thus the LSTM cell remembers longer sequences better than the vanilla RNN cell. Finally, the hidden state h is calculated using this memory c, and the output is regulated by o.



(a) Input **(b)** Forget

(c) Memory **(d)** Output

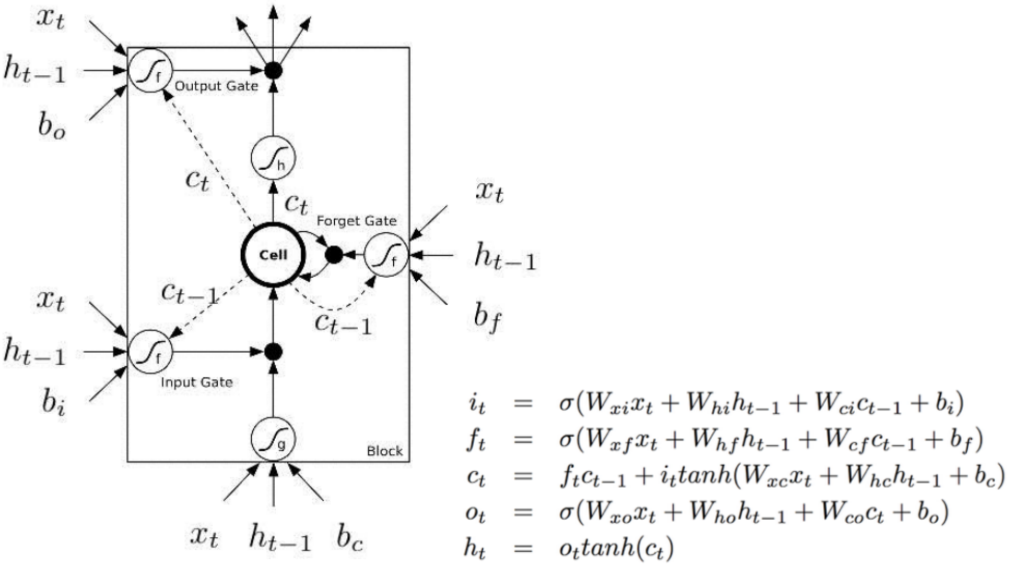
Figure 3.9: Gates inside an LSTM cell(Yan [2016])

3.3 Sequence to Sequence Model

The general term *sequence to sequence model* is used to represent a neural network which takes a sequence of characters or words as input, and generates a similar type of sequence as output, conditioned on a target vocabulary. This model was first introduced in the field of Neural Machine Translation(NMT) (Cho et al. [2014b], Sutskever et al. [2014]), which used deep neural networks in order to automate the translation process of a traditional phrase-based statistical machine translation system.

The underlying idea is that given a sentence in one language (e.g English), generate this sequence in another language (e.g German). The neural network learns to maximize the conditional probability of generating a word in the target sequence, given the source sequence and all the previously generated words.

Currently, the most commonly used sequence to sequence models have two essential components : An RNN encoder-decoder (Cho et al. [2014a]), and an attention mechanism (Bahdanau et al. [2014]) in order to jointly learn the alignment of source and target sequences for translation. As the task of summarization can be viewed as translating into the same language as the source, these models which are traditionally used in NMT, have also been used for



$$\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t tanh(c_t)
\end{aligned}$$

Figure 3.10: Mathematical equations of corresponding gates of an LSTM cell(Graves [2013])

summarization (Nallapati et al. [2016]). This thesis also uses the similar architecture, and explores its capabilities and limitations more extensively using various kind of sub-datasets for summarization. Let us look into more detail about the components of this model.

3.3.1 RNN Encoder-Decoder

This consists of two recurrent neural networks (RNN) that act as an encoder-decoder pair as can be seen in Figure 3.11. The encoder maps a variable-length source sequence to a fixed length vector usually referred to as the context vector. The context vector is the last hidden state of the encoder, and is initialized as the first hidden state of the decoder. The decoder is conditioned on this context, and maps this fixed length vector back to variable-length target sequence. The two networks are trained jointly to maximize the conditional probability of the target sequence given a source sequence.

As the encoder-decoder architecture was originally applied in the field of language translation, we shall try to explain the idea with a similar example. Consider the French to English translation model(Neubig [2017]) shown in Figure 3.12. We look up the embedding of a French word f , and the encoder calculates the hidden state h_t^f at timestep t in the sequence. We start with an empty vector (filled with zeroes) as the first state of the encoder, and by the last hidden state the encoder has processed the entire sequence one word at a time. In the context of summarization, the final state of the encoder can be regarded as a partial summary of the whole content.

In the decoder, we predict the probability of generating an English word e at each time step. First, we look up the word embedding for our target word similar to the encoder except that we use the previous English word, as we must condition the probability on the previously generated word in the sequence instead of the current word. Next, we run the decoder to calculate further hidden states of the network. This is similar to the encoder step, with the important difference being that the first hidden state of the decoder is set to be the context vector obtained from the last hidden state of the encoder, allowing us to condition on the source sequence (French sentence). Finally, we calculate the probability of generating the current English word by applying a softmax layer on the hidden state of the decoder at current time step (Neubig [2017]) .

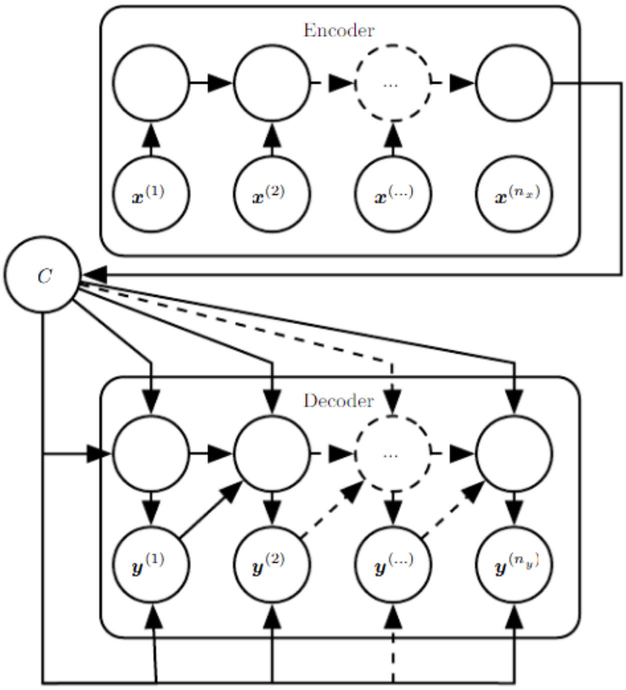


Figure 3.11: Block diagram of RNN encoder-decoder architecture (Goodfellow et al. [2016])

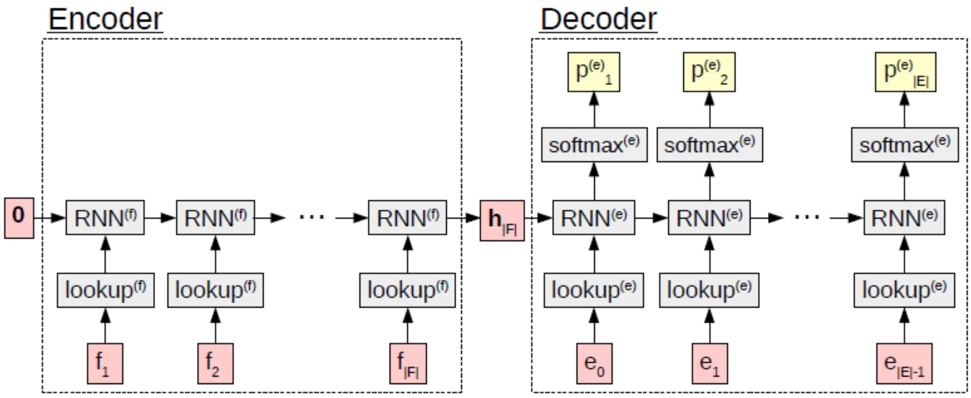


Figure 3.12: Example of neural machine translation from French to English using encoder-decoder (Neubig [2017])

3.3.2 Generating Sequence from Encoder-Decoder

The encode-decoder architecture discussed so far enables us to learn a distribution over the target vocabulary given the source vocabulary. We have the final encoded representation of the source sequence from the encoder, which can be used by the decoder to select words from the target vocabulary in order to form a summary. Generating a summary based on the learned conditional probability can be done in one of three ways: *Random sampling* selects an output from the target vocabulary depending on the probability distribution. *1-best search* as the name implies finds the target word that maximizes the probability. Finally, we can also perform an *n-best search* to find multiple outputs having high probabilities. In many sequence to sequence learning models, a common variant of the 1-best search known as *beam search* is used to find the best translation for the given source sequence.

Beam Search

It is similar to greedy search with the key difference being that a specific number (say b) of hypotheses are considered while translation instead of only the best one. This number b is usually referred to as the beam-size or beam-width. An important consideration is that as the beam size is increased, the decoder gets better at finding shorter sentences, which is an advantage for summarization task. This is because adding a word at each step is equal to multiplying the probability, which eventually decreases as the length of the hypothesis under consideration increases. This phenomenon is also known as length bias of the beam search algorithm(Neubig [2017]). Figure 3.13 shows a very simple beam search process with 4 tokens.

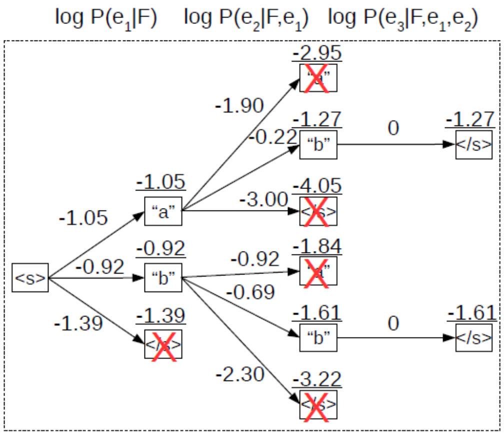


Figure 3.13: Beam search with beam width 2 (Neubig [2017])

3.3.3 Attention in Sequence to Sequence Model

One of the limitations of the basic encoder-decoder architecture is encoding variable length input sequence to a fixed size context vector. Sometimes, when the sequence is long, the fixed length encoding fails to capture the complete information and might produce incomplete summaries. In order to overcome this we can try to make the context vector a variable length vector similar to the input sequence.

This is the basic idea behind the concept of attention which preserves vectors for each word in the sequence, and attends to them individually at each decoding step, thus avoiding the fixed length encoding problem. As the number of vectors available to the decoder for referencing is equal to the length of the input sequence, longer sentences can have many vectors which makes for an effective representation when compared to a fixed size context vector (Bahdanau et al. [2014]).

3.3.4 Implementing Attention

Let us see in detail how attention is implemented in a sequence to sequence model. Consider a source sequence $\mathbf{S} = \{s_1, s_2, s_3...., s_n\}$ and a target sequence $\mathbf{R} = \{r_1, r_2, r_3...., r_m\}$ where n > m represents our summarization scenario. First, we create an annotation for each word in the input sequence. To do so, we use a bidirectional RNN in the encoder which runs from left to right, and right to left creating both forward and backward representations for each word. We then concatenate these two representations to form an *annotation.* Intuitively, an annotation of a word contains information from the past and the future which implies that at each step we have the complete information about the sequence. Finally, we construct a huge matrix for our sequence by concatenating annotation vectors of each word as columns of this matrix **H**.

$$\overrightarrow{h}_j^{(s)} = RNN(embed(s_j), \overrightarrow{h}_{j-1}^{(s)}) \tag{3.1}$$

$$\overleftarrow{h}_j^{(s)} = RNN(embed(s_j), \overleftarrow{h}_{j+1}^{(s)}) \tag{3.2}$$

$$h_j^{(s)} = \left[\overrightarrow{h}_j^{(s)}; \overleftarrow{h}_j^{(s)}\right] \tag{3.3}$$

$$H^{(s)} = concat(h_1^{(s)}, ..., h_{|S|}^{(s)}) \tag{3.4}$$

However, we cannot use this matrix directly for decoding, as our decoder is an RNN that is conditioned on a single vector (context vector from the encoder). Therefore, we need to find a way to convert **H** into a vector which can be used by the decoder to start generating the output sequence.

We do this by multiplying **H** with a vector **a**, which returns the final context vector **c**. **a** is known as the attention vector which has its elements with values between 0 and 1, and summing up to 1. This vector represents the softmax probabilities of each word in the source sequence being aligned to the current target word in the decoding phase. A larger attention value implies that this source word is important for the decoder to generate our next target word in the summary.

Next, we need to calculate the attention vector **a**. Referring to our summarization scenario, the first state of the decoder h_0^r is initialized with the final state of the encoder $h_{|S|+1}$. This is used to calculate the source attentional context vector c_t at time step t that is used to choose the target word. As a first step, we update the hidden state of the decoder h_t^r based on the previous target word embedding r_{t-1}, the previous attentional vector c_{t-1} and the previous hidden state h_{t-1}^r

$$h_t^{(r)} = enc([embed(r_{t-1}); c_{t-1}], h_{t-1}^{(r)}) \tag{3.5}$$

This decoder state h_t^r is used to calculate an attention score a_t for each word from the input sentence.

$$a_{t,j} = attn\_score(h_j^{(s)}, h_t^{(r)}) \tag{3.6}$$

where, t is the current decoder step, and j is the position of a source word in the input sentence $\mathbf{S}$.

This can be seen as calculating an alignment between a source word and the target word in order to evaluate how important this source word might be in predicting the next target word; given all the previously generated target words. The concept of alignment was extensively used in phrase-based statistical machine translation systems in order to produce coherent translations. Summarization can also benefit from this approach in a similar way, by attending only to important words from the source sequence which can appear in the final summary.

The function $attn\_score$ is an arbitrary function that takes two vectors as inputs and outputs a score informing us about how much we must focus on the encoding h_j^s of an input word at position j at time step t in the decoder. This score is normalized using a softmax layer in order to generate the final attention value in the attention vector $\mathbf{a}$. The matrix $\mathbf{H}$ is multiplied by the calculated attention vector $\mathbf{a}$ to get a context vector $\mathbf{c}$, which is a weighted sum of the columns of $\mathbf{H}$. This means that each column of $\mathbf{H}$, which is an annotation of a word from the source sequence is given a certain amount of importance, depending on the corresponding value in the attention vector $\mathbf{a}$. We now have a context vector c_t, and a hidden state h_t^r for each time step t of the decoder which can be passed along the decoder RNN. We can then perform an affine transformation over these, and apply a softmax layer resulting in the probability of choosing the next target word by the decoder.

$$p_t^{(r)} = softmax(W_{hd}[h_t^{(r)}; c_t] + b_d) \tag{3.7}$$

where W_{hd} and b_d are weights and biases learned by the network.

This means that each source word is much more involved now than before in calculating the prediction of the target word. While using attention, the source encoding for each word can be accessed in a weighted manner through the use of this attention vector.

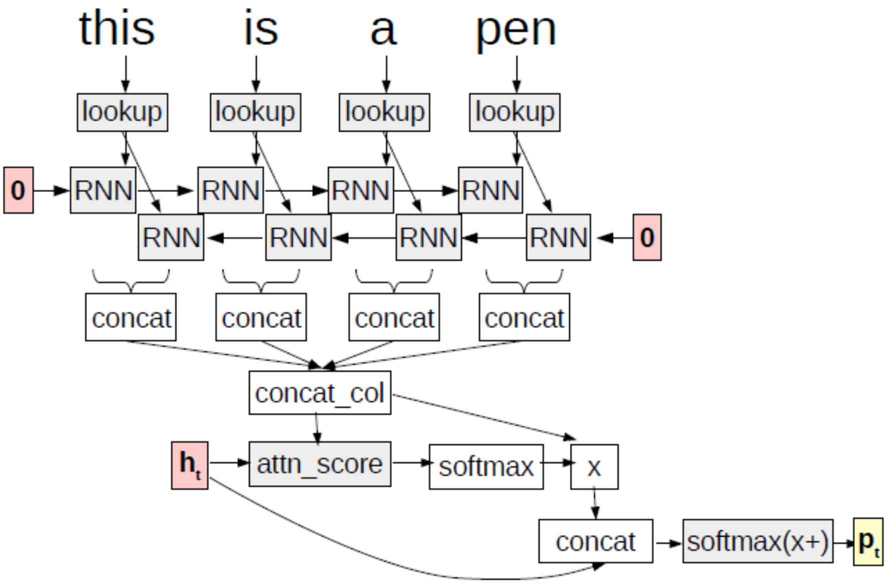


Figure 3.14: A computation graph for attention (Neubig [2017])

3.3.5 Calculating Attention Score

While we have discussed how attention is implemented, we have to yet explore the computation of the attention score itself. A computation graph for attention is shown in Figure 3.14. Let us look into the attn_score function mentioned in Section 3.3.4, and all the different candidates for it. As a quick reminder, the attn_score function takes two vectors as input and gives a number as output. There are mainly 3 possible ways to calculate this score(Luong et al. [2015]) :

Dot Product : This gives the similarity between the two vectors h_t^r and h_j^s as measured by the dot product between them and is very simple to calculate.

$$attn\_score(h_j^{(s)}, h_t^{(r)}) := h_j^{(s)T} h_t^{(r)} \tag{3.8}$$

Bi-linear Function : This function performs a linear transformation parameterized by a weight matrix $\mathbf{W}_a$ before calculating the dot product.

$$attn\_score(h_j^{(s)}, h_t^{(r)}) := h_j^{(s)T} \mathbf{W}_a h_t^{(r)} \tag{3.9}$$

Multi-layer Perceptron : This is the original attention score function used by Bahdanau et al. [2014]

$$attn\_score(h_t^{(r)}, h_j^{(s)}) := w_{a2}^T tanh(W_{a1}[h_t^{(r)}; h_j^{(s)}]) \tag{3.10}$$

where W_{a1} and w_{a2} are the weight matrix and vector of the first and second layers of the multi-layer perceptron respectively.

3.3.6 Input Feeding Attention Vector to the Decoder

In the standard attention mechanism, all decisions of attending to source words are made independently. This is in contrast to the traditional SMT system, where a coverage set containing all the source words translated so far, is maintained. In order to achieve this, an input feeding approach is adopted in which all attention vectors are fed as inputs to the next time steps as additional inputs to the decoder(Luong et al. [2015]). This is done via concatenating the attention vector with the word embeddings of the target words predicted so far by the decoder as shown in Figure 3.15. This makes the model fully aware of the past alignment decisions.

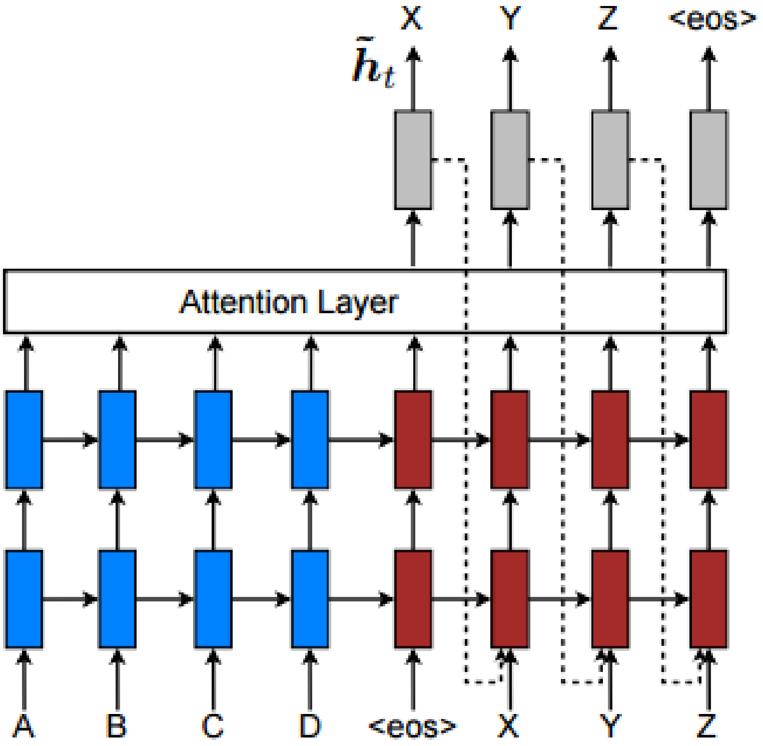


Figure 3.15: Input feeding approach (Luong et al. [2015])

3.4 Enhancing Sequence to Sequence Model for Better Summarization

Recent research by the community has improved the base sequence to sequence models, allowing us to customize and tweak the decoding process to produce better translations (Wu et al. [2016]). As already mentioned, summarization being a special case of translation, we can make use of these improvements to experiment with our sub-datasets. Let us discuss in detail about the available options and how they can be leveraged in our work.

3.4.1 Larger Vocabulary Size for Embeddings

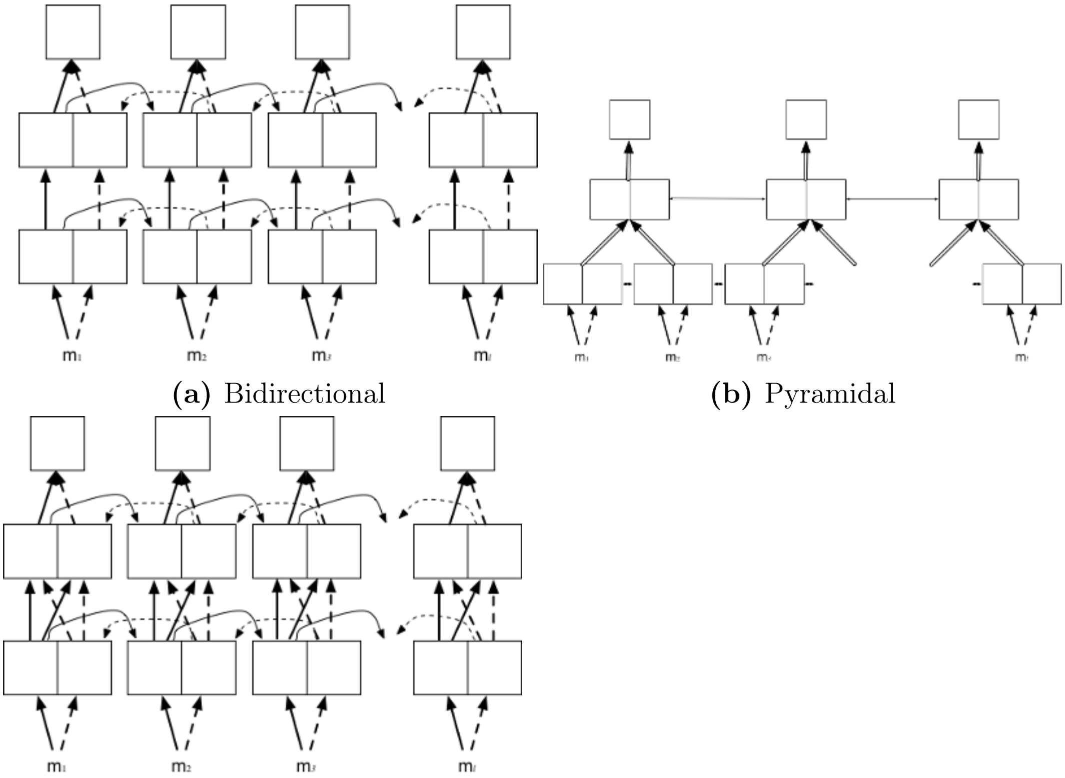
It is a common practice to consider a fixed vocabulary size of around 30,000 or 50,000 of the most common words in the training set. While this may suffice for smaller sequences such as phrases or sentences, summarization can benefit by using a larger vocabulary to reduce the number of unknown words in the generated summary. We use a vocabulary size of 80,000 words in all our experiments subject to our hardware availability, as a very large vocabulary requires large amounts of memory.

3.4.2 Variations of Model Architecture

While the most common encoder architecture is the Bidirectional RNN, it is possible to experiment with other variants such as the Deep Bidirectional Encoder where the outputs of every layers are summed or concatenated prior feeding to the next layer, and Pyramidal Deep Bidirectional Encoder that reduces the time dimension after each layer based on a reduction factor provided. This decreases the number of locations the attention module must attend , thereby improving the speed and reducing the training time required. The three variants are depicted in Figure 3.16

3.4.3 Residual Connections

It has been observed that increasing the depth of a neural network improves the performance of the model by enabling it to learn better feature representations. However, the more deep the network is, the slower the training can become due



(a) Bidirectional

(b) Pyramidal

(c) Deep Bidirectional

Figure 3.16: RNN architecture variants(Klein et al. [2017])

to the vanishing gradient which makes the learning difficult. One approach to mitigate this problem is to add residual connections between LSTMs in a stack, where the input is added to the current output before passing it to the next layer, as shown in Figure 3.17. They are usually point wise additions to the LSTM outputs, which help us to model differences between an intermediate layer's output and the targets.

3.4.4 Regularization

This is a technique to prevent the neural networks from overfitting and to increase their generalization capacity. One of the most commonly used regularization method is known as **dropout**(Srivastava et al. [2014]). Dropout is only applied during the training phase (Figure 3.18) , where the idea is to disable individual neurons with a probability p , for a given batch. This is also known as the dropout rate or dropout probability. Setting dropout to 0 disables the dropout, and usually a value of 0.2 is used for performing regularization. This is applied on the output of each layer, the output of the attention

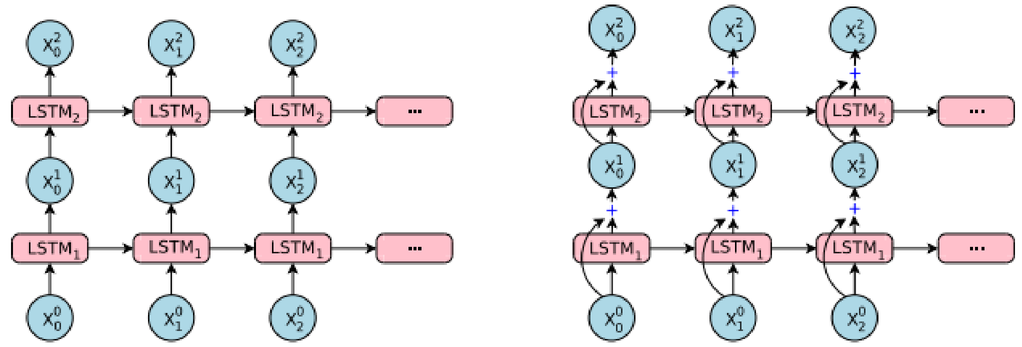


Figure 3.17: Stacked LSTM architecture without (left), and with (right) residual connections (Wu et al. [2016])

layer and can also be enabled between word embeddings and the first layer. There are two implementations of dropout as shown in Figure 3.19 :
Naive : this is the default approach where the dropout is applied only on non-recurrent connections.
Variational : dropout is also applied to the recurrent connections but each time step applies the same dropout mask.

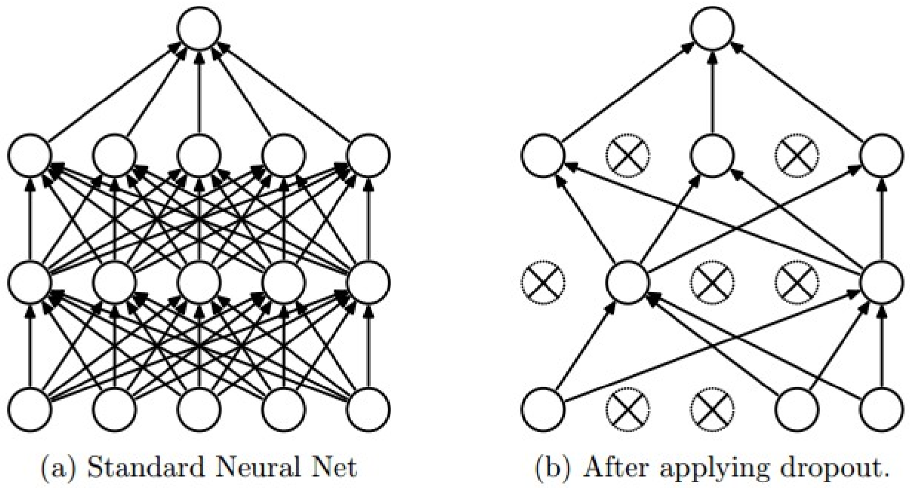

(a) Standard Neural Net
(b) After applying dropout.

Figure 3.18: Application of dropout (Srivastava et al. [2014])

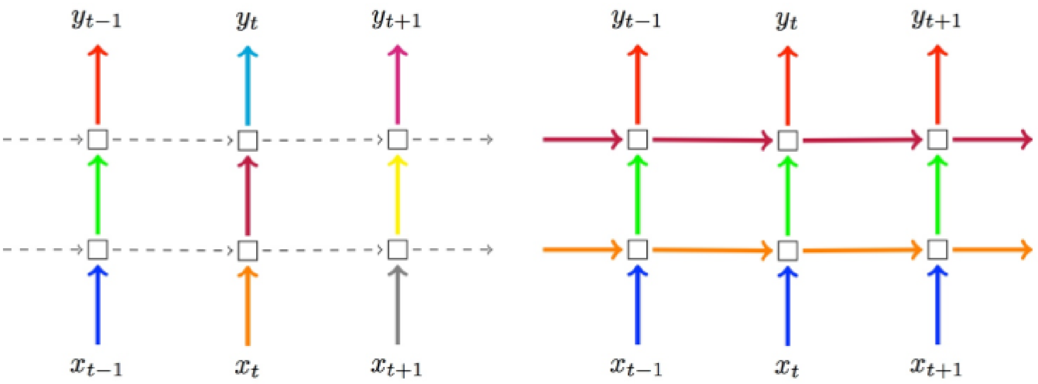


Figure 3.19: On the left is the naive dropout where it is not applied on recurrent connections (represented by the dotted arrows). On the right is the variational method where the same amount of dropout is applied on the recurrent connections (Gal [2015])

3.4.5 Pointer-Generator Networks for Better Summarization

An improved variant of the sequence to sequence model known as Pointer-Generator Network shown in Figure 3.20 has been recently introduced by See et al. [2017]. This network deals with two key problems of failure to reproduce factual details, and word repetitions in the generated summaries, which we also report in our Section 4.3. In order to deal with the problem of incorrect factual reproduction, this network simply copies words from the source via *pointing*, while retaining the ability to *generate* words from the fixed target vocabulary. In addition to calculating the attention and vocabulary distributions as usual, an additional probability known as *generation probability* denoted by p_{gen} is calculated. This represents the probability of generating a word from the vocabulary, versus copying a word from the source. When compared to the sequence to sequence model with attention, the pointer-generator system has the following advantages:

1. It is easier to copy words from the source text. The network simply needs to put sufficiently large attention on the relevant word and make p_{gen} sufficiently large.

2. It is even able to copy **out-of-vocabulary** words from the source text, which enables us to handle unseen words while also allowing us to use a smaller vocabulary

3. The pointer-generator model is faster to train, requiring fewer training iterations to achieve the same performance as the sequence to sequence model

In order to deal with the second problem of repeating words in the summary, a technique known as *coverage* is used. The idea is to use the attention distribution to keep track of what has been covered so far, and penalize the network for attending to the same parts again. On each timestep t for the decoder.

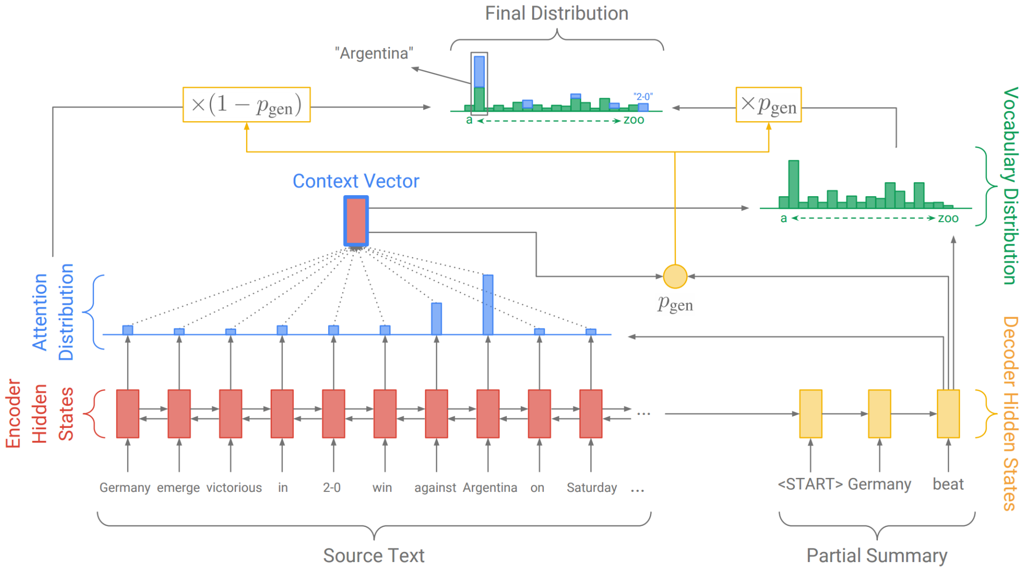


Figure 3.20: Pointer-generator network (See et al. [2017])

Chapter 4

Experiments and Evaluation

As already discussed in Section 2.4, we created 5 sub-datasets from our base corpus namely questions, vulgar posts, true summaries, topic-summary and topic generation dataset. Before we discuss more about the evaluation, we provide a short explanation of the process of generating the summaries itself from our model. We have seen how the decoder performs beam search (Figure 3.13) on the target vocabulary and forms a set of hypotheses at each step to chose from. This can also include cases where the decoder is not able to find a suitable word from the target vocabulary due to the pruning in the preprocessing stage, and thus substitutes an UNK (unknown) token. In order to filter out such hypotheses, we inform the decoder to ignore all the hypotheses with one or more UNK tokens. We use a beam size of 6 (through trial and error) as we find this to be an optimal choice in order to generate meaningful summaries, while avoiding the bias of finding very short phrases as the final summaries.

For creating a test set for each of these models, we take 100 examples from its corresponding dataset. We then generate the summaries for these 100 examples using the trained model. For example, it is intended that the questions model will generate 100 question type of summaries for the 100 examples taken from the questions dataset. Similar approach is taken for the remaining verticals. The first evaluation measure we apply is the popular ROUGE score (Lin [2004]). Before looking into the results, let us first understand what ROUGE is and how it works.

4.1 ROUGE - A Measure for Evaluation of Automatic Summaries

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) is a package for automatic evaluation of machine generated summaries by comparing them to the human generated target summaries(Lin [2004]). Different measures available in the package account to different overlaps such as as n-gram (ROUGE-1: unigram, ROUGE-2: bigram), word sequences (ROUGE-L: Longest common subsequence), and word pairs between the automatic summary generated by our neural network and the target summary (tl;dr) written by the author of the post. ROUGE has become the standard evaluation measure for text generation tasks. It was mainly developed in order to save the time and effort involved in manual evaluation of the automatically generated summaries by humans. Let us discuss in more detail how the ROUGE-N (ROUGE-1, ROUGE-2) and the ROUGE-L measures are calculated as we shall use these in evaluating our summaries generated by the vertical-specific models discussed above.

ROUGE-N
It is an n-gram recall between a candidate summary and a set of reference summaries. In our scenario however, we have only a single reference summary which is the tl;dr of the post by the author.

$$ROUGE - N = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \tag{4.1}$$

where n is the length of the n-gram, $Count(gram_n)$ is the total number of n-grams in the summary S, and $Count_{match}(gram_n)$ is the maximum number of n-grams co-occurring in a candidate summary and the set of reference summaries (Lin [2004]) .

ROUGE-L : Longest Common Subsequence
Longest common subsequence (LCS) of the content and its summary is a *common subsequence of words* with maximum length. To calculate LCS in evaluation of an automatic summary in comparison with a true summary, we view both of them as two sequences of words. The intuition is that the longer the LCS of two summary sentences is, the more similar the two summaries are (automatic, and target summary).
One advantage of using LCS is that it does not require consecutive matches but in-sequence matches that reflect sentence level word order as n-grams. As this automatically includes the longest in-sequence common n-grams, no predefined n-gram length is necessary. By awarding credit only to in-sequence unigram

matches, ROUGE-L also captures sentence level structure in a natural way. One main disadvantage of LCS is that it only counts the main in-sequence words; therefore other alternative common subsequences are not reflected in the final score. For example, consider a reference summary **T**, and a candidate summary **C** as below (Lin [2004]):

$$\mathbf{T} : \textit{police killed the gunman} \tag{4.2}$$

$$\mathbf{C} : \textit{the gunman police killed} \tag{4.3}$$

In this case, the ROUGE-L measure counts either "the gunman" or "police killed" but not both as common subsequences.

Evidently, the ROUGE measure is highly suitable for evaluating extractive summaries which have word overlaps between the generated summary and the true summary. However, in case of an abstractive summary which can contain novel words, and reformed phrases, this measure awards low scores even to relevant summaries judged by a human annotator. We argue that there is a strong need for a different evaluation method which is suitable for judging the relevance and the quality of abstractive summaries.

For the purpose of this thesis, we perform a manual evaluation of our generated summaries by annotating summaries with a binary score of relevant or not. We understand that there are multiple layers of judgment possible such as if the summary is accurate, moderately related, related but missing key information, or unrelated to mention a few. In order to prevent the evaluation from becoming sparse and to give a concise picture of the capabilities of each model, we decided to opt for the binary approach. Also, to maintain consistency with the evaluations performed by other abstractive summarization research (Rush et al. [2015]), we provide in Table 4.1, the size of the training and validations sets followed by the ROUGE-1, ROUGE-2 and ROUGE-L scores of our vertical specific models on their respective test sets containing 100 examples each.

Table 4.1: ROUGE scores of vertical-specific model summaries

Sub-dataset	**Training**	**Validation**	**ROUGE1**	**ROUGE2**	**ROUGE-L**
Questions	75,000	1,600	12.03	1.78	10.36
Vulgar posts	222,532	9,940	9.93	1.24	8.58
True summaries	1,188,690	4,000	**19.71**	5.57	**17.02**
Topic + tl;dr	729,042	5,022	12.24	2.49	10.47
Generic	2,003,000	15,501	17.87	**6.11**	15.78
Rush et al. [2015]	4 Million	NA	28.18	8.49	23.81

4.2 Manual Evaluation

For manual evaluation, we decided to construct a standard test set consisting of examples from all the sub-datasets. The intention is to evaluate the performance of a specific model on all kinds of content irrespective of the vertical it has been trained upon. For example, the model trained using the questions sub-dataset will be used to summarize content without any questions in it. We aim to evaluate how each model performs in such cases and what can be done further in order to improve or remove any bias from the data itself in our future research. To this end, we take 50 examples each from the test sets of the vulgar, questions, and topic generation sub-datasets. As a reminder, the 50 examples from the topic generation dataset can also serve as the test set for the true summaries model. Finally, we add another 50 examples from the test set of our generic model which has been trained on the complete comments and submissions without any verticals whatsoever.

We use these test set of 200 examples and generated 200 summaries using each of our vertical specific model. Each summary is evaluated as being either relevant or not. Table 4.2 shows the performance of each model on all the test sets including the standard one, with the values being the percentage of correct/relevant summaries generated by a model in each case.

We also evaluated the summaries generated using additional parameters such as length, coverage and end of sentence normalization in the decoding phase (Wu et al. [2016]). As a brief explanation, length normalization makes the decoder avoid very short summaries during beam search. This helps overcome the length bias of beam search when using large beam widths to consider more hypotheses for our final summmary. With coverage normalization the decoder attends to as many new words as possible from the source sequence for generating a summary. This helps in including as much information as possible from the source sequence. Finally, end of sentence normalization penalizes the score of end of sentence token (EOS). We used a value of 0.2 for all three parameters as suggested by Wu et al. [2016]. The results of manual evaluation on these summaries are shown in Table 4.3.

Although we did not find any substantial improvement in the quality of summaries generated using normalization, there were instances where normalization did improve the structure of the generated summary. Please refer to Appendix B for examples showing the impact of normalization on the summaries generated.

Table 4.2: Manual evaluation of models on all test sets. The generic model was trained on the complete dataset. Each table cell gives the percentage of relevant summaries generated by a model for a given test set

Model	**Test Set**				
	Questions (50)	Vulgar (50)	True Summaries (50)	Generic (50)	All (200)
Ground truth	100	100	100	84	96
Questions (75,000)	31	50	28	16	31
Vulgar (222,532)	16	35	14	12	19.5
True-summaries (1,188,690)	44	54.9	36	30	41.3
Topic + tl;dr (729,042)	44	45	16	10	29
Topic (729,042)	58	47	50	30	48.5
Generic (2,003,000)	67.3	62	52	34	53.5

Table 4.3: Manual evaluation of models with normalization. The generic model was trained on the complete dataset. Each table cell gives the percentage of relevant summaries generated by a model for a given test set. A value of **0.2** was used for length, coverage and end of sentence normalization parameters

Model	**Test Set**				
	Questions	Vulgar	True Summaries	Generic	All
	(50)	(50)	(50)	(50)	(200)
Ground truth	100	100	100	84	96
Questions (75,000)	10	15.6	14	2	10.5
Vulgar (222,532)	4	23.5	0	4	8.5
True-summaries (1,188,690)	24	29.4	22	10	21.4
Topic + tl;dr (729,042)	30	35.3	20	4	22.5
Topic (729,042)	32	35	24	18	27.8
Generic (2,003,000)	66	54	30	28	44.5

4.2.1 A Comparison of ROUGE and Manual Evaluation

After performing both ROUGE and manual evaluations, we examined further into how these two are related if at all. In Section 4.1, we argued about how an F-measure such as ROUGE may not be the perfect choice to evaluate abstractive summaries as it does not encourage the presence of novel words to contribute to the final score. Supporting this argument, we present our observations on the ROUGE score vs manual evaluation of our generic model (trained on the complete dataset) in Table 4.4. From Figure 4.1 showing various correlations, we can say that the two approaches are almost unrelated. This is a key concern as even the summaries evaluated to be relevant by a human annotator obtain very poor ROUGE scores on different measures (ROUGE-1,

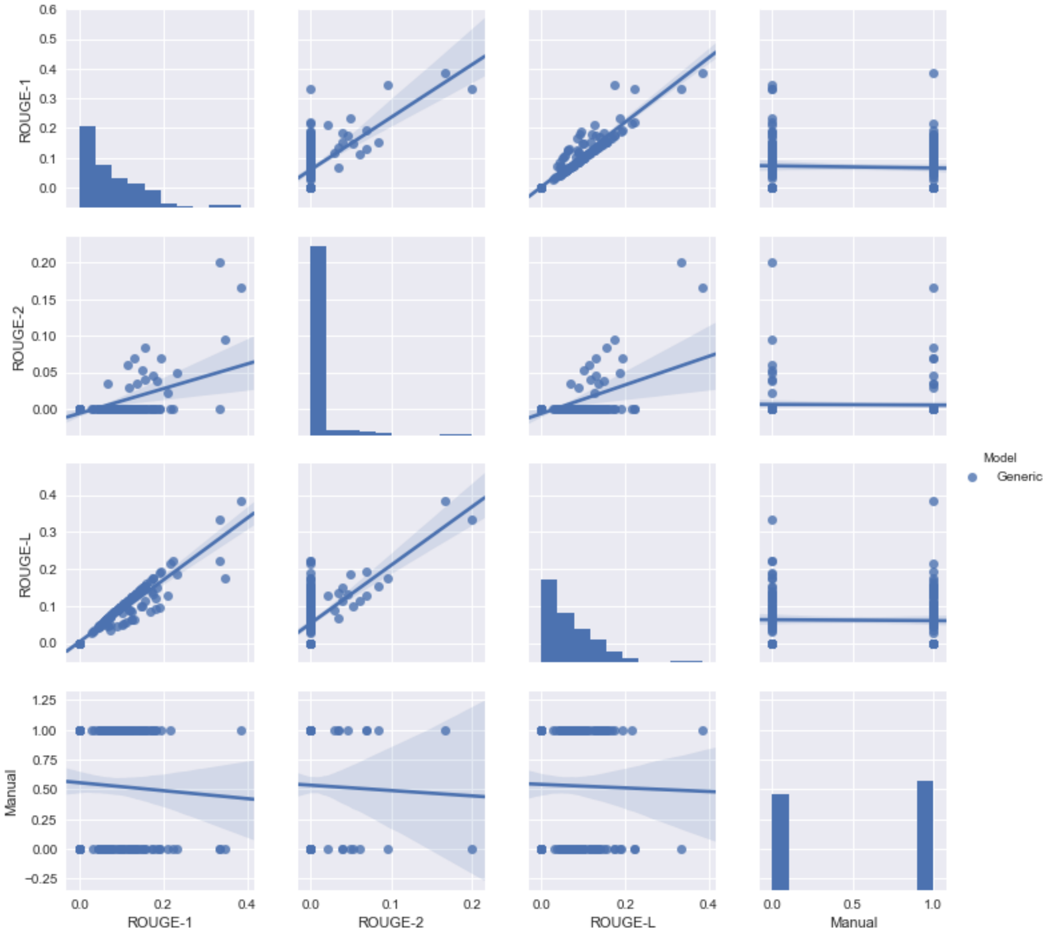


Figure 4.1: Correlation of ROUGE vs manual evaluation on the generic model. Though the manual evaluation and various ROUGE scores are not correlated, we can see that different ROUGE scores are fairly correlated amongst each other.

Table 4.4: ROUGE vs manual evaluation - Pearson correlation scores for generic model

Method	ROUGE-1	ROUGE-2	ROUGE-L	Manual
ROUGE-1	1	0.54	0.95	-0.04
ROUGE-2	0.54	1	0.55	-0.02
ROUGE-L	0.95	0.55	1	-0.01
Manual	-0.04	-0.02	-0.01	1

Table 4.5: p-value for ROUGE-1, ROUGE-2, ROUGE-L

Metric	p-value
ROUGE-1	0.485031
ROUGE-2	0.771985
ROUGE-L	0.784899

ROUGE-2 and ROUGE-L).

p-value:

We first calculate the p-values by performing an unpaired **t-test** on the ROUGE scores for the sets of summaries that were manually labeled (non)relevant. For these p-values shown in Table 4.5, the null hypothesis is that a given ROUGE measure has the same mean, regardless of the correctness of the summary in our manual evaluation process. The p-value shows the probability of observing our results assuming that the null hypothesis is true. A high p-value indicates weak evidence against our null hypothesis, and thus cannot be rejected.

4.3 Existing Problems of Abstractive Summarization

Although deep neural networks have vastly improved the process of abstractive summarization, it is yet far from being ideal. Apart from the obvious lack of a variety of datasets, there is also a strong need to fine tune the process of generating the summary itself. Two major problems that we frequently observed are inaccurate reproduction of factual details from the source text and repeating words or phrases in the summary (also reported by See et al. [2017]). Let us discuss a bit more about these problems.

1. Inaccurate Factual Details :
Consider the example shown in Table 4.6 from our topic generation model :

Table 4.6: Example of incorrect facts reported in the generated summary

Body: my wife (teacher) constantly complains about how she can ' t stand her job , the kids are disrespectful , etc . and how much she hates it . i know my wife , and i know she wants me to say " sweetie why don ' t you just quit ? " but i ' m not going to say that under any circumstances . however , i did tell her that i will support her leaving if she has a backup job or if she decides to go back to school (that will help advance her career) . i also said that i would pay for her schooling . her response is that she does not want to go back to school . it ' s getting to the point where she talks about how much she hates work everyday . it completely ruins the end of the day , as half the time it ends in her crying . i ' m not sure what to do
TL;DR: My (32 / m) wife (32 / f) complains about her job and wants me to tell her it ' s okay to quit , but I ' m not going to , how do I handle this ? : wife hates her job and wants me to tell her to quit . i ' m not going to tell her to quit , but i also don ' t know what to do .
Summary: I [27 / m] don ' t know how to tell my wife [28 / f] that I don ' t want to quit her job . : wife is mad because i don ' t want to quit .

As seen in the generated summary, the age of the subjects in discussion is wrongly reproduced as 27 years instead of 32 as seen from the true summary. We have also observed cases where the gender corresponding to the ages was wrongly swapped. This is especially common for rare or out of vocabulary tokens such as **[32/m]**. A possible explanation of this phenomenon is that it is very difficult for the sequence to sequence model with attention to copy a word from the source text. As the network is many layers deep, multiple computations on this source word in each layer, make it hard to be easily found by the decoder in the final stage. It instead finds the nearest word from the embedding space and adds it to the summary. Moreover if a word in source text is a rare word (as majority of the key facts involving numbers or slang are), the word embedding for such words is not rich enough and is thus clustered somewhere along with unrelated words, which makes reproducing it harder than expected. On the other hand, even if a word has a rich embedding, it is at times replaced by different words that are closely clustered to it in the dimensional space, (e.g. Berlin:Munich) which still brings into consideration the problem of reproducing key facts.

2. Repeating Words in the Summary :
This is a scenario where certain words or phrases are repeated in the summary generated by the decoder. Consider the example shown in Table 4.7 from our vulgar model :

Table 4.7: Example of repeating words in the generated summary

Body:	it was around the holidays a few years ago and i had started smoking for the first time . being somewhat forced to spend " quality time " with my family , i decided to smoke a blunt before going . i get to my parents house , i ' m at a solid [9] and my sweet , little , 80 year old grandma was sitting on the couch . i took a seat next to her and she tried to lean her head on my shoulder , being too short she rested it on my bicep . she patted my leg before looking up at me saying " you ' re so high " . i nearly jump out of my skin , and look at her a bit side-ways because i had put on new clothes , perfume and even brushed my teeth before coming over . just before i could get the words " how did you know " out of my mouth , she smiled at me , " so high i can ' t rest my head on your shoulder . " i laughed nervously and adjusted accordingly so she could lean her head on me .
TL;DR:	went to a family gathering super baked , sat next to my grandma who said " you ' re so high " referring to my height because she couldn ' t rest her head on my shoulder when i sat up straight . scared the shit outta me .
Summary:	i got high as fuck and got high as fuck .

The phrase *got high as fuck* can be considered a related summary of the content. However, the repetition of this phrase immediately makes it incorrect. Repetition may be caused by the strong dependence of the decoder on its previously generated words which are fed as inputs in the sequence. This causes a single unrelated/commonly repeated word generated in the previous state to trigger a cycle of such repetitions in the later stages.

Chapter 5

Conclusion

In this work we have contributed a novel, usable dataset for summarization, from the domain of social media. We believe this will encourage researchers to explore the problems of understanding unstructured, poorly written text while trying to generate abstractive summaries. We have also explored how vertical-specific sub-datasets extracted from this base corpus can give interesting results in the form of summaries with similar intentions. An important problem that we would like to point out is the inability of recurrent neural networks to deal with very long texts. Although sentence level and short paragraph level summarization has greatly improved, we feel that the true goal of a summarization system should not be limited by the length of the input text. We aim to explore this area of concern in our future research.

Abstractive summarization has come a long way since the application of recurrent neural networks. However there are still some issues that remain unresolved such as :

1. Introducing novel phrases instead of just a few words in the summary to represent a strong paraphrasing
2. Seamlessly combining multi-sentence summaries with proper pronouns, gender, tense and other key facts intact
3. Ability to focus and decide on primary information from the source text instead of attending to less important information
4. Tuning the decoder to generate a specific style of summaries, or summaries adhering to certain guidelines such as being free of profanity
5. Understanding the data itself and removing any conceptual bias, which causes the generated summary to appear very generic, and contain repetition of words

Another key consideration for our future work, is to develop a different evaluation metric than ROUGE. Although an industry standard for evaluating automatic summaries(extractive summaries as originally intended), we feel that ROUGE may not be the best fit to evaluate an abstractive system as it checks for overlap of unigrams, bigrams and longest common subsequences in the generated and the true summaries. This discourages any novel words produced in the automatic summary to contribute to the abstraction of the system in general. As already seen in our examples, the summaries generated, although relevant to the topic still obtain low ROUGE scores.

We also argue that the poor writing style of social media posts is a big challenge to be explored further in order to produce coherent, novel summaries. The Pointer-Generator network mentioned briefly in Section 3.4.5 generates reasonably good summaries on the CNN/Daily Mail News dataset as reported in See et al. [2017]. We would like to evaluate the performance of this model on our Webis-TLDR-17 corpus as a next step. Finally, we also aim to experiment with Convolutional neural networks for abstractive summarization (Gehring et al. [2016]) in order to make use of the performance and speed improvements; courtesy of the ability to train these networks in parallel compared to the sequential nature of recurrent neural networks.

Appendices

Appendix A

Examples of Generated Summaries

A.1 Questions

Table A.1: Examples of generated summaries for questions

Example 1

Body: ever since i watched one of the counselors at my camp play tron , i ' ve wanted to build it . the only problem is , i can ' t figure out which colors to build around . the group at my lgs that plays mainly modern is saying i should build r / g because it ' s the best , but i think the money that i would spend on a playset of [[grove of the burnwillows]] could be spent elsewhere and have a much bigger impact on the deck and the games i play with it . so my question to you guys is what non-r / g variants of tron could be viable in a competitive modern environment ?

TL;DR: what non-r / g variant should i build ?

Summary: what ' s the best build for a new player ?

Example 2

Body: in a non-sexual way . is there something wrong with me . . . literally every moment i can ' t help but think about her , and miss her the moment we go our ways after hanging out . whenever we meet again , it ' s like a giant sigh of relief or something and everything feels right . but i have no clue if she feels this way about me too . and i don ' t intend to ask her until much much later as she just broke up with her bf . my friends tell me to just occupy myself with work , or a hobby , or going out . which i have tried , but the moment i ' m done with whatever it is , same thing , keep thinking about her . . . not in a sexual way at all , but just that i want to be with her , talking laughing etc . i have trouble sleeping now too because of this and i just miss her so much even though i know we ' re gonna hangout again in a few days . help ? do i need to see a shrink ?

TL;DR: can ' t stop thinking but this one female friend , not in sexual way at all , purely simply miss her like crazy whenever we ' re not in one another ' s presence . should i see a shrink ?

Summary: is it weird to ask a girl to be my girlfriend ?

A.2 Vulgar Posts

Table A.2: Examples of generated vulgar summaries

Example 1

Body: i took my first medication this morning and i ' m definitely starting to see some improvements in my focus . inattentiveness was a big issue for me and my amphetamine salts er (10mg) seem to be helping . of course its only the first day , but i need to start to establish routine and work towards getting more organized and disciplined . first thing i did when i took the meds was clean my room entirely , it was a shit storm but now its all neat lol . then i dealt with some things i ' ve put behind for god knows how long regarding college (paying for my tuition / other things) so now i don ' t have to worry about getting kicked out for not paying . i feel clearer despite it being a low dosage . i wouldn ' t say that i ' m ecstatic / feel amazing (depressing mood yesterday) , but i can tell i ' m starting to work towards getting myself together , which is a good start .

TL;DR: any tips ? whether its setting small goals / writing stuff down in a journal / apps that are complete life-savers .

Summary: i ' m a lazy fuck and need to get my shit together

Example 2

Body: just talked to catholic priest on a wedding . guys , have no mercy , please . the shit they preach is a mind killer for our civilization . religious people are fucking dumb , dumb fucks and they spread shit all over the place . i fucking spit on catholics . my whole family is religious and i can ' t have intellectual conversation on any subject with them . we can not tolerate any of this shit any longer . i ' m picking hitchens and the rest of dawkins for my holidays and i ' m going militant then . hoooly shit ! sorry for rant but the level of bullshit they can get over with is killing me .

TL;DR: religion = mental retardation .

Summary: religion is fucking stupid .

A.3 Topic + tl;dr

Table A.3: Examples of summaries where the topic is also generated

Example 1

Body: so i ' m interested in this girl i met at work , who i ' ve known for about a month , maybe two . wasn ' t interested in her at first , but i started to develop feelings for her . mainly because of her personality . we just kinda interact easily with each other , you know ? just bouncing back and forth , giving each other shit , etc . only problem is i ' m not so good with more serious stuff , and i ' ve never actually attempted dating a girl before . i asked her out on a " get to know each other better " get together for tomorrow evening , but i don ' t know what to go do (leaning towards getting dinner somewhere) . and i don ' t know how to actually go about getting to know her better , because in the past it has happened pretty naturally , but over the course of a few months , and never in terms of pursuing a romantic interest .

TL;DR: Advice / help needed : - want to get to know this girl better , with the intention of dating her being a thing in the near future . need help on what to do / what kind of place to go to to do this . any advice / help is much appreciated .

Summary: I [20 M] want to ask a girl [20 F] out , but I ' d like to know how I should proceed : i ' d like to know how i ' d actually be able to make a move on a girl i ' m interested in .

Example 2

Body: yesterday i tried lsd for the first time . i first took half of a 110ug tab thinking that i shouldnt overdo it for the first time , about 4 minutes later i took the other half (thought that 55ug wouldnt be enough) . half an hour later , nothing . an hour later , a bit light headed but zero visuals . two hours later , i feel a bit happier and i see a glimpse of a pattern on our table move but that ' s about it for the visuals . about 3.5 hours in i took another tab thinking i would finally get some more visuals . . . nothing . i tested the lsd and it came out positive . so did i get some seriously underdosed tabs or did i fuck up by not taking the whole tab at once in the start or do i have a freakishly high tolerance or something ?

TL;DR: Did I get underdosed tabs ? : tried lsd for the first time yesterday , first took half a tab , then the other half 4 minutes later . ended up jusy a bit light headed so 3.5 hours in i took another tab , still got pretty much no visuals .

Summary: Question about LSD : i ' m a noob and i ' d like to know if i did it or not .

A.4 True Summaries

Table A.4: Examples of generated summaries where the summary is highly relevant to the content

Example 1

Body: not necessarily my lucky day , but some kids this is how it went was sitting out on the dock at a local lake with a friend sharing some beers . little boy aged 2-3 yrs old walks up with a wooden stick and starts poking at the water . it was windy out and the dock was moving , and sure enough the kid leans over just enough to topple head first into the water . i had already pulled my phone out and wallet out just in case i was to accidentally fall in so i went straight over and hopped in . saw his little hand reaching up and tossed him straight back onto the dock . walked him to his dad who didn ' t speak any english and was very confused why i had his son soaking wet . left later that day and saw the kid back on the dock ! it blew my mind.

TL;DR: saved a 2 year old from drowning at a lake because i was drinking beers with a friend .

Summary: i saved a kid from drowning .

Example 2

Body: alright so i just started seeing a girl i ' ve known for quite a while and things have been great . we get along really well and just genuinely enjoy each other ' s time . only hiccup is i enjoy smoking copious weed and she well , she hates weed . her ex boyfriend apparently let weed take over his life . he was standing her up to go smoke , ditching work to go find weed , and just fucking around basically is what it sounds like . well she blames the weed for his short comings and now judges everyone based on the fact that they smoke weed . she doesn ' t care about tobacco , or alcohol either just weed . i know common sense says to just tell her i smoke a ton and tell her it doesn ' t impact my life negatively and i still get my shit down . idk . basically i ' m looking for helpful stories or advice for me if you ' ve been in a similar situation . we aren ' t super serious at all , and she ' s moving states in about two months so it ' s not a huge deal but i also want to enjoy our time together while she still lives here

TL;DR: been dating a new girl who hates weed because her last boyfriend couldn ' t handle smoking and his life . now i ' m torn between telling her how much i smoke and how often or just hiding it until she moves away . any help is greatly appreciated

Summary: i want to tell a girl i smoke a lot , she hates weed , what do i do ?

A.5 Topic Generation

Table A.5: Examples of generating topic of a given text

Example 1
Body: MechanicalKeyboards What Keyboard , Switches and / or Keys Do I Buy : wiki any brand preferences ? use the search option and redditor review wiki what is your budget ? see these sections of the reddit keyboard shopping guide less than 100 101- 150 151- 200 201 + what country are you planning to purchase the keyboard in ? see your country ' s shopping wiki and list of keyboard sellers that ship internationally if you are new to reddit check out this handy reddit / r / mechanicalkeyboards noob guide . **True Topic:** MechanicalKeyboards What Keyboard , Switches and / or Keys Do I Buy **Generated Topic:** mechanical keyboards
Example 2
Body: Plex WAN stream causing congestion : streaming over web makes gaming impossible concurrently on same network as plex server . 75 / 6 connection . (speedtests 85 / 8) . **True Topic:** Plex WAN stream causing congestion **Generated Topic:** plex server issues

Appendix B

Impact of Normalization on Summaries

Table B.1: Better summaries with normalization

Example 1

Body: the entire idea of the xp system was to make the game more skill based and nerf huge groups by giving small groups or solo ’ s a fighting chance if they were truly skilled at the game , everyone progressing at the same pace . all ownership does is make it easier for groups to progress and shit on little guys , as well as earning less xp from using tools crafted from looted bases (wtf was that even thought through ? ? ?) people need to understand its not the slow progression or ’ grind ’ aspect of the xp system , its ownership fucking everyone over . it needs to be somehow edited or completely removed because the new ’ ’ diminishing returns ’ ’ isn ’ t doing shit . everyone wants rust to be a game of skill , not a game of numbers . .

TL;DR: all ownership does is give 1 + to huge groups and fuck with shit stolen from bases (loot)

Summary: the xp system is fucking retarded .

Summary-norm: stop whining about xp system , it ’ s not a game of numbers , it ’ s ownership of the xp system .

Example 2

Body: my boyfriend and i have been together for almost a year and we have a great relationship , we even lost our virginities to each other . however , i ’ ve always been concerned about pleasing him while we have sex due to my size being the top (4 inches) because it doesn ’ t seem like hes getting much pleasure . recently we tried using a dildo to help loosen him and it was a little longer than me (5 in) but thinner . when i used that he went wild and reacted like he never has before , so i ’ m assuming i was hitting his prostate . when i questioned him about it after he still said i made him feel better but from his body i could tell he was lying . now i feel like i ’ m not giving him all that i could and feel bad . i know there are options like penis sleeves and pumps to make myself seem longer but i ’ m not sure what to do . if anyone has been in a situation like this and could give advice it would be much appreciated .

TL;DR: boyfriend gets more pleasure from small dildo than me . feel bad and don ’ t know what to do .

Summary: my boyfriend and i have been having sex for a year and i ’ m not sure what to do .

Summary-norm: how can i make my boyfriend feel better when we have sex ?

Table B.2: Poor summaries with normalization

Example 1

Body: so , since i couldn ' t find anything in searching i was hoping the wonderful ds2 vets could offer some guidance . i have always played a sorc in each souls game and finally decided to try a str build . i am cooping with a friend almost constantly and after we beat the smelter demon i began getting my ass handed to me . i was using the santier ' s raw + 1 , battle axe + 4 , and twin greatshield . after some trial and error (and research) decided to respec . i had 50end , 32str , and 23vit before . found the soft caps and now have 18vig , 30end , 40str , 20vit , 20adp (i know they don ' t add up , but i leveled up some before respecing) . i have decided to use great club and have actually have a large club as well . i want to start dual wielding , but can ' t seem to get away from using a shield . i just always seem to be in an area that has narrow hallways , small bridges , or multiple enemies . obviously i can just stop shielding cold turkey and die till i git gud , but how do you dual wielding vets compete pve or pvp with great weapons ?

TL;DR: want to start dual wielding great clubs , always seem to be in a spot where a shield is needed . advice ?

Summary: how do you dual wield ?

Summary-norm: i ' m looking for tips on dual wielding , twin , twin , twin , twin , twin , twin , twin , twin , twin , twin , twin , twin , ...

Example 2

Body: after a year of college without making that many friends , i realized that i simply don ' t know what to do for fun with them after we finished studying . i ' m not much of a party person and most of my hobbies are solitary . as the new terms starts next week , i really want to continue to build and reinforce my social circle and do things that we all would like to do and have fun .

TL;DR: what do you like to do for fun with friends ?

Summary: what do you do for fun ?

Summary-norm: i don ' t know what to do for fun , but i don ' t know what to do .

Appendix C

JSON Representation of Posts from Reddit

A comment can be seen as a statement of a fact or opinion, for example a remark that expresses a personal reaction or attitude. lstlisting[]

```
{"archived":true,
    "author":"jaquehamr",
    "body":"Thanks for proving the point of the quote.\n\nTL;DR: WOOSH",
    "controversiality":0,
    "created_utc":"1239192802",
    "downs":0,
    "edited":"false",
    "gilded":0,
    "id":"c08q8en",
    "link_id":"t3_8auok",
    "name":"t1_c08q8en",
    "parent_id":"t1_c08q4sz",
    "retrieved_on":1425950159,
    "score":3,
    "score_hidden":false,
    "subreddit":"atheism",
    "subreddit_id":"t5_2qh2p",
    "ups":3}
```

A submission is also a statement of fact, opinion or also just a web link posted by a registered user with an intention to elaborate the discussion with other users. A submission is relatively longer compared to a comment as its aim is to explain/describe a topic, while comments are used to express opinions on it. lstlisting[]

```
{"archived":true,
"author":"[deleted]",
"created":1297290547,
"created_utc":"1297290547",
"domain":"self.WeAreTheFilmMakers",
"downs":0,
"edited":"false",
"gilded":0,
"hide_score":false,
"id":"fibse",
"is_self":true,
"media_embed":{},
"name":"t3_fibse",
"num_comments":2,
"over_18":false,
"permalink":"/r/WeAreTheFilmMakers/comments/fibse/question_about_resumes/",
"quarantine":false,
"retrieved_on":1442846972,
"saved":false,
"score":2,
"secure_media_embed":{},
"selftext":"I’m currently a film student at the University of Cincinnati and I’m going to start applying for internships soon so I was wondering what I should put on my resume when applying. Do they really care that I know how to work a fryer at a wing joint? Should I leave all non film related jobs I’ve done off or throw ’em on? What have you put on your resume when sending it out?\n\ntl;dr I’m going to be sending out my resume soon and I’m looking for help on what I should include on it\n\nThanks, guys!",
"stickied":false,
"subreddit":"WeAreTheFilmMakers",
"subreddit_id":"t5_2qngr",
"thumbnail":"default",
"title":"Question about resumes",
"ups":2,
"url":"http://www.reddit.com/r/WeAreTheFilmMakers/comments/fibse/question_about_resumes/"}
```

Bibliography

Rasim M. Alguliev, Ramiz M. Aliguliyev, Makrufa S. Hajirahimova, and Chingiz A. Mehdiyev. Mcmr: Maximum coverage and minimum redundant text summarization model. *Expert Syst. Appl.*, 38(12):14514–14522, November 2011. ISSN 0957-4174. doi: 10.1016/j.eswa.2011.05.033. URL `http://dx.doi.org/10.1016/j.eswa.2011.05.033`. 1.2

Rasim M. Alguliev, Ramiz M. Aliguliyev, and Nijat R. Isazade. Multiple documents summarization based on evolutionary optimization algorithm. *Expert Syst. Appl.*, 40(5):1675–1689, April 2013. ISSN 0957-4174. doi: 10.1016/j.eswa.2012.09.014. URL `http://dx.doi.org/10.1016/j.eswa.2012.09.014`. 1.2

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`. 3.3, 3.3.3, 3.3.5

Elena Baralis, Luca Cagliero, Naeem A. Mahoto, and Alessandro Fiori. Graphsum: Discovering correlations among multiple terms for graph-based summarization. *Inf. Sci.*, 249:96–109, 2013. doi: 10.1016/j.ins.2013.06.046. URL `https://doi.org/10.1016/j.ins.2013.06.046`. 1.2.2

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March 1994. ISSN 1045-9227. doi: 10.1109/72.279181. URL `http://dx.doi.org/10.1109/72.279181`. 3.2.1

Giuseppe Carenini, Raymond T. Ng, and Xiaodong Zhou. Summarizing emails with conversational cohesion and subjectivity. In *ACL*, 2008. 1.4

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014a. URL `http://arxiv.org/abs/1409.1259`. 3.3

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014b. URL `http://arxiv.org/abs/1406.1078`. 1.3, 3.3

Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive Sentence Summarization with Attentive Recurrent Neural Networks. In *NAACL 2016*, pages 93–98, 2016. ISBN 9781941643914. URL `http://nlp.seas.harvard.edu/papers/naacl16{_}summary.pdf`. (document), 1.3, 1.1, 1.2

Günes Erkan and Dragomir R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December 2004. ISSN 1076-9757. URL `http://dl.acm.org/citation.cfm?id=1622487.1622501`. 1.2.2

Mohamed Abdel Fattah. A hybrid machine learning model for multi-document summarization. *Applied Intelligence*, 40(4):592–600, Jun 2014. ISSN 1573-7497. doi: 10.1007/s10489-013-0490-0. URL `https://doi.org/10.1007/s10489-013-0490-0`. 1.2.3

Mohamed Abdel Fattah and Fuji Ren. Ga, mr, ffnn, pnn and gmm based models for automatic text summarization. *Computer Speech Language*, 23 (1):126 – 144, 2009. ISSN 0885-2308. doi: http://dx.doi.org/10.1016/j.csl.2008.04.002. URL `http://www.sciencedirect.com/science/article/pii/S0885230808000296`. 1.2.3

Rafael Ferreira, Luciano de Souza Cabral, Rafael Dueire Lins, Gabriel Pereira e Silva, Fred Freitas, George D.C. Cavalcanti, Rinaldo Lima, Steven J. Simske, and Luciano Favaro. Assessing sentence scoring techniques for extractive text summarization. *Expert Systems with Applications*, 40(14): 5755 – 5764, 2013. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2013.04.023. URL `http://www.sciencedirect.com/science/article/pii/S0957417413002601`. 1.2

Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. *arXiv:1512.05287*, 2015. 3.19

Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: A survey. *Artif. Intell. Rev.*, 47(1):1–66, January 2017. ISSN 0269-2821. doi: 10.1007/s10462-016-9475-9. URL `https://doi.org/10.1007/s10462-016-9475-9`. 1.2, 1.2

Jonas Gehring, Michael Auli, David Grangier, and Yann N. Dauphin. A convolutional encoder model for neural machine translation. *CoRR*, abs/1611.02344, 2016. URL `http://arxiv.org/abs/1611.02344`. 5

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`. 3.1, 3.2, 3.3, 3.4, 3.5, 3.7, 3.11

Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL `http://arxiv.org/abs/1308.0850`. 3.10

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 1693–1701, 2015. URL `http://papers.nips.cc/paper/5945-teaching-machines-to-read-and-comprehend`. 1.4

Jee-Uk Heu, Iqbal Qasim, and Dong-Ho Lee. Fodosu: Multi-document summarization exploiting semantic analysis based on social folksonomy. *Information Processing and Management*, 51(1):212–225, 2015. doi: 10.1016/j.ipm.2014.06.003. 1.2

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`. 3.2.1

Eduard Hovy and Chin-Yew Lin. Automated text summarization and the summarist system. In *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998*, TIPSTER '98, pages 197–214, Stroudsburg, PA, USA, 1998. Association for Computational Linguistics. doi: 10.3115/1119089.1119121. URL `http://dx.doi.org/10.3115/1119089.1119121`. 1.1.4, 1.4, 1.2

Baotian Hu, Qingcai Chen, and Fangze Zhu. LCSTS: A Large Scale Chinese Short Text Summarization Dataset. pages 1967–1972, 2015. URL `http://www.aclweb.org/anthology/D15-1229`. 1.4, 1.2

G. Klein, Y. Kim, Y. Deng, J. Senellart, and A. M. Rush. OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*, 2017. 3.16

Ju-Hong Lee, Sun Park, Chan-Min Ahn, and Daeho Kim. Automatic generic document summarization based on non-negative matrix factorization. *Inf. Process. Manage.*, 45(1):20–34, January 2009. ISSN 0306-4573. doi: 10.1016/j.ipm.2008.06.002. URL http://dx.doi.org/10.1016/j.ipm.2008.06.002. 1.2

Chin-Yew Lin. Rouge: a package for automatic evaluation of summaries. July 2004. URL https://www.microsoft.com/en-us/research/publication/rouge-a-package-for-automatic-evaluation-of-summaries/. 4, 4.1, 4.1

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. *CoRR*, abs/1508.04025, 2015. URL http://arxiv.org/abs/1508.04025. 3.3.5, 3.3.6, 3.15

Martha Mendoza, Susana Bonilla, Clara Noguera, Carlos Cobos, and Elizabeth León. Extractive single-document summarization based on genetic operators and guided local search. *Expert Syst. Appl.*, 41(9):4158–4169, July 2014. ISSN 0957-4174. doi: 10.1016/j.eswa.2013.12.042. URL http://dx.doi.org/10.1016/j.eswa.2013.12.042. 1.2.3

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 280–290, 2016. URL http://aclweb.org/anthology/K/K16/K16-1028.pdf. (document), 1.3, 1.1, 1.4, 1.2, 3.3

Joel Larocca Neto, Alex Alves Freitas, and Celso A. A. Kaestner. Automatic text summarization using a machine learning approach. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence*, SBIA '02, pages 205–215, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-00124-7. URL http://dl.acm.org/citation.cfm?id=645853.669480. 1.2

Graham Neubig. Neural machine translation and sequence-to-sequence models: A tutorial. *CoRR*, abs/1703.01619, 2017. URL http://arxiv.org/abs/1703.01619. 3.6, 3.3.1, 3.12, 3.3.2, 3.13, 3.14

You Ouyang, Wenjie Li, Sujian Li, and Qin Lu. Applying regression models to query-focused multi-document summarization. *Inf. Process. Manage.*, 47 (2):227–237, March 2011. ISSN 0306-4573. doi: 10.1016/j.ipm.2010.03.005. URL http://dx.doi.org/10.1016/j.ipm.2010.03.005. 1.2

Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP'15*, 2015. (document), 1.3, 1.1, 1.2, 4.1, 4.1

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. *CoRR*, abs/1704.04368, 2017. URL `http://arxiv.org/abs/1704.04368`. 1.4, 1.2, 3.4.5, 3.20, 4.3, 5

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=2627435.2670313`. 3.4.4, 3.18

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press. URL `http://dl.acm.org/citation.cfm?id=2969033.2969173`. (document), 3.3

J.M. Torres-Moreno. *Automatic Text Summarization*. Cognitive science and knowledge management series. Wiley, 2014. ISBN 9781848216686. URL `https://books.google.de/books?id=aPHsBQAAQBAJ`. 1.1

Jan Ulrich and Gabriel Murray. A publicly available annotated corpus for supervised email summarization. In *AAAI08 EMAIL Workshop*, Chicago, USA, 2008. AAAI. 1.2

Wikipedia. Reddit — wikipedia, the free encyclopedia, 2017a. URL `https://en.wikipedia.org/w/index.php?title=Reddit&oldid=796504350`. [Online; accessed 24-August-2017]. 2

Wikipedia. Automatic summarization — wikipedia, the free encyclopedia, 2017b. URL `https://en.wikipedia.org/w/index.php?title=Automatic_summarization&oldid=796603348`. [Online; accessed 22-August-2017]. 1.2

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL `http://arxiv.org/abs/1609.08144`. 3.4, 3.17, 4.2

Shi Yan. Understanding lstm and its diagrams, 2016. URL `https://medium.com/@shiyan/understanding-lstm-and-its-diagrams-37e2f46f1714`. 3.8, 3.9

Libin Yang, Xiaoyan Cai, Yang Zhang, and Peng Shi. Enhancing sentence-level clustering with ranking-based clustering framework for theme-based summarization. *Inf. Sci.*, 260:37–50, March 2014. ISSN 0020-0255. doi: 10.1016/j.ins.2013.11.026. URL `http://dx.doi.org/10.1016/j.ins.2013.11.026`. 1.2.3

Shiren Ye, Tat seng Chua, Min yen Kan, and Long Qiu. Document concept lattice for text understanding and summarization, 2007. 1.2