UNIVERSITY COLLEGE LONDON

MASTERS THESIS

A Supervised Approach to Extractive Summarisation of Scientific Papers

Author:
Edward Collins

Supervisor:
Dr. Isabelle AUGENSTEIN

This report is submitted as part requirement for the MEng Computer Science at UCL. It is substantially the result of my own work except where explicitly indicated in the text.

in the

Department of Computer Science

May 2, 2017

University College London

Abstract

Faculty of Engineering Sciences
Department of Computer Science

MEng Computer Science

A Supervised Approach to Extractive Summarisation of Scientific Papers

by Edward Collins

When doing any kind of research, one of the most tedious tasks is to read through hundreds of papers to do a literature review. By harnessing the power of machine learning, the aim of this work is to build a system capable of automatically summarising scientific papers using extractive summarisation in order to supplement the abstract of a paper so that if readers require a deeper understanding than the abstract can provide, they do not have to trawl through the main, dense text. Scientific article summarisation presents a particularly hard problem for summarisation because the text to summarise is long; far longer than is traditionally handled by automatic summarisers. By using a plethora of machine learning and natural language processing techniques ranging from basic counting to deep learning, a system able to generate short summaries, or "highlights" of papers has been developed which produces good results using the research standard metric for summarisation tools and when compared to other work. Furthermore, a brand new dataset for summarising scientific articles has been presented and ways of extending it automatically have been created, which has led to the first ever dataset for the summarisation of scientific papers with a sufficient amount of data to apply state of the art deep learning techniques. To our knowledge, deep learning has never been applied to the summarisation of scientific papers before this work due to a poverty of data, so this is the first research to present benchmark performances from deep learning algorithms on this new dataset.

Acknowledgements

I would like to thank my supervisor, Dr. Isabelle Augenstein, for her guidance, patience and insights with this work. Our many discussions and plans were instrumental to making this project a success.

Contents

List of Figures

List of Tables

List of Abbreviations

RNN	**R**ecurrant **N**eural **N**etwork
LSTM	**L**ong **S**hort **T**erm **M**emory
SL	**S**upervised **L**earning
ROUGE	**R**ecall-**O**riented **U**nderstudy for **G**isting **E**valuation
LCS	**L**ongest **C**ommon **S**ubstring
NLP	**N**atural **L**anguage **P**rocessing
GRU	**G**ated **R**ecurrent **U**nit
AI	**A**rtificial **I**ntelligence
ML	**M**achine **L**earning
TF-IDF	**T**erm **F**requency-**I**nverse **D**ocument **F**requency

Chapter 1

Introduction

1.1 Motivating Summarisation

If you have ever had to trawl your way through hundreds of scientific papers while doing a literature review then you may well have thought to yourself "I wish there was a quicker way of doing this". More and more scientific papers are being written and published everyday - recent data suggests one every 20 seconds ("The Rise of Open Access" 2012) - and online repositories of papers are turning into true leviathans: Elsevier Scopus had 57 million papers stored in 2016 and Thomson Reuters ISI Web of Knowledge contained a gargantuan 90 million (Ronzano and Saggion, 2016). Very soon, if not already, it will be impossible for a single or even a team of humans to review all of the literature surrounding a subject. Google Scholar for example, as of 2017, lists just shy of 4 million entries for the query "deep learning". To deal with such overwhelming numbers, humans will need to turn to computers and to artificial intelligence (AI).

When doing a literature review, the first thing a person is likely to check on a paper is its title. A title is effectively a one-line summary of a paper's contents. A logical approach after that would be to scan the abstracts of many papers to see if they could be relevant to your work. But what about after that? Once you have gathered one or two hundred papers, how do you then read and comprehend all of them? An assortment of techniques exist for quickly gaining an understanding of a paper, such as reading specific sections[1] like the Introduction and Conclusion before reading the rest of the paper. However this will still involve a certain degree of mining the main, dense text of the paper. But what if there was a way to have a computer do that part for you? What if a program could simply present you with a summary of the main points of the paper as a supplement to the abstract? If that were possible, then quickly gaining a deeper understanding of papers could become much more feasible. As Kageback et al., 2014 say, a well written summary can significantly reduce the amount of work needed to digest large amounts of text on a given topic.

[1] e.g. https://www.elsevier.com/connect/infographic-how-to-read-a-scientific-paper

Paper Title Statistical estimation of the names of HTTPS servers with domain name graphs
Highlights we present the domain name graph (DNG), which is a formal expression that can keep track of cname chains and characterize the dynamic and diverse nature of DNS mechanisms and deployments. We develop a framework called service-flow map (sfmap) that works on top of the DNG. sfmap estimates the hostname of an HTTPS server when given a pair of client and server IP addresses. It can statistically estimate the hostname even when associating DNS queries are unobserved due to caching mechanisms, etc through extensive analysis using real packet traces, we demonstrate that the sfmap framework establishes good estimation accuracies and can outperform the state-of-the art technique called dn-hunter. We also identify the optimized setting of the sfmap framework. The experiment results suggest that the success of the sfmap lies in the fact that it can complement incomplete DNS information by leveraging the graph structure. To cope with large-scale measurement data, we introduce techniques to make the sfmap framework scalable. We validate the effectiveness of the approach using large-scale traffic data collected at a gateway point of internet access links .
Summary Statements Highlighted in Context from Section of Main Text Contributions: in this work, we present a novel methodology that aims to infer the hostnames of HTTPS flows, given the three research challenges shown above. The key contributions of this work are summarized as follows. We present the domain name graph (DNG), which is a formal expression that can keep track of cname chains (challenge 1) and characterize the dynamic and diverse nature of DNS mechanisms and deployments (challenge 3). We develop a framework called service-flow map (sfmap) that works on top of the DNG. sfmap estimates the hostname of an https server when given a pair of client and server IP addresses. It can statistically estimate the hostname even when associating DNS queries are unobserved due to caching mechanisms, etc (challenge 2). Through extensive analysis using real packet traces , we demonstrate that the sfmap framework establishes good estimation accuracies and can outperform the state-of-the art technique called dn-hunter, [2]. We also identify the optimized setting of the sfmap framework. The experiment results suggest that the success of the sfmap lies in the fact that it can complement incomplete DNS information by leveraging the graph structure. To cope with large-scale measurement data, we introduce techniques to make the sfmap framework scalable. We validate the effectiveness of the approach using large-scale traffic data collected at a gateway point of internet access links. The remainder of this paper is organized as follows: section2 summarizes the related work. [...]

TABLE 1.1: An example of a document with summary statements highlighted in context.

Automatic summarisation aims to provide these computer written summaries. According to Chen, 2015, there are four types of summary of a piece of text that can be produced: *informative*, which reflect the main points of a piece of text; *indicative*, which conveys information like writing style and topics covered but not the actual information in the text; *critical*, which provides a judgement of a piece of text; and *contrastive*, which extracts multiple points of view from the text. In this work we focus on informative summaries which would best supplement the abstract of a paper. An example informative summary, highlighted in context, is shown in Table 1.1. The two main types of informative summary are *generic*, which provides an overview of the points in the text; and *query-based* which specifically answers a user's question (Chen, 2015).

There are two main streams of techniques which are used to produce informative summaries, both generic and query-based: *extractive* approaches and *abstractive* approaches. Extractive summarisation is the simpler of the two methods (Fang et al., 2017), and works by extracting salient pieces of text, often sentences, from a text document which best summarise that document. Extractive summarisation guarantees a certain level of quality in the summary because its sentences will be written by the authors of the paper so should have proper grammatical form. Hirao et al., 2017 state that there are hard limits to how good an extractive summariser can be, and believe that the topic is undergoing a paradigm shift to abstractive summarisation. Abstractive summarisers generate entirely new text to summarise a document which

is particularly useful for when sentences taken out of context are not a good basis for forming grammatical and coherent summaries, such as in novels. Abstractive summarisers have seen most success in the domain of news summarisation because news articles do not tend to be particularly long; abstractive summarisers are currently not adept at summarising long articles (Chen et al., 2016).

In this work we are concerned with summarising scientific papers, which tend to be significantly longer than news articles. In addition, papers are a technical domain with fairly regular and explicit language, so we have opted for the task of *extractive summarisation*. Doing so also has the advantage that summaries would be directly quotable, which they would not be with abstractive summaries.

1.2 Challenges of Automatically Summarising Papers

The main challenge of summarising scientific papers is their length. Papers can convey many different concepts and conclusions throughout, so effectively capturing all of these is difficult. Furthermore, ensuring that generated summaries are coherent and make sense when taken out of context is another difficult task. Finally, producing a sufficient amount of data to apply any kind of deep learning is a difficult task as deep learning methods are notoriously data-hungry (LeCun, Bengio, and Hinton, 2015). To our knowledge, no datasets for summarisation of scientific publications of sufficient size and format to use for deep learning exist.

1.3 Project Objectives and Contributions

The main objectives of the project are summarised below. These also formed the main contributions of this project once they had been achieved:

- To study the structure of scientific papers and see which sections are most useful in generating summaries and have the highest information content. Our initial hypothesis was that the abstract would be most relevant to summaries as it is already a summary itself, followed by the conclusion because this tends to summarise the main message and achievement of the paper; and the introduction which gives a basic background of the topic area.
- To study and present methods for extending a new dataset for the summarisation of scientific documents, with the hope that the size of this dataset can be used to inspire a new generation of scientific paper summarisers which make use of state of the art techniques like deep learning; which have recently enjoyed much success with summarising news articles (Nallapati, Zhai, and Zhou, 2016; Cheng and Lapata, 2016).

- To present a plethora of summarisation techniques which make use of both neural and traditional methods that can be used to summarise the scientific articles given in the dataset and be presented as benchmark performance measures.
- To present one or more summarisation algorithms which would allow a reader to gain a general understanding of what a paper is about from only reading this summary.
- To rigorously test a small set of features which can be used in this summarisation task as further benchmarks.

1.4 Personal Aims

I hope to gain a decent understanding of statistical natural language processing and how machine learning techniques can be used in NLP. I also hope to gain a decent knowledge of deep learning techniques and particularly how to implement these in TensorFlow. I also hope to produce a summarisation system which outperforms at least simplistic methods and will make literature reviewing less tedious.

1.5 Thesis Guide

The rest of the thesis is structured as follows. In Chapter 2 a comprehensive literature review of extractive summarisation and summarisation of scientific papers is given. In Chapter 3, the method used to achieve all of the objective described above is given, excluding the method of determining the sections most relevant to a summary which is explained in the next chapter. Chapter 4 details the results achieved in this project and gives an analysis of these. Finally Chapter 5 gives conclusions and an evaluation of the project, as well as a section on potential future work in this area.

Chapter 2

Background

The purpose of this chapter is to give an overview of the state of the art when it comes to extractive summarisation and summarisation of scientific documents. In addition to this, a survey of the available datasets for scientific paper summarisation is conducted.

2.1 Deep Learning

If you were to take any of the most popular learning algorithms from before 2012, they would most likely rely on hand engineered features from which to learn (LeCun, Bengio, and Hinton, 2015). The paradigm shifted entirely after 2012 with the ImageNet competition win by Alex Krizhevsky, Ilya Sutskever and Geoff Hinton, who significantly outperformed the competition using a deep Convolutional Neural Network which was able to learn complex representations from raw data, requiring no hand engineered features (Krizhevsky, Sutskever, and Hinton, 2012). Deep learning takes the concept of neural networks trained with the backpropagation algorithm and expands them to build computational models consisting of many layers of simple learning functions which produce non-linear mappings, allowing them to learn extremely fine-grained and complex representations of data (LeCun, Bengio, and Hinton, 2015). Today there are many types of deep networks performing impressive feats, although two of the most popular and well known types are Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). RNNs are of particular interest to this work so will be explained in greater detail.

2.1.1 Convolutional Neural Networks

CNNs have become the workhorse of computer vision algorithms used in ImageNet since 2012, with recent entries becoming ever more elaborate such as with Google's invention of Inception modules (Szegedy et al., 2014) or with Microsoft's staggering 152-layer ResNet (He et al., 2015). However CNNs are not only useful in computer vision, they have also been used in the realm of natural language processing to great success (e.g. (Kim, 2014; Kalchbrenner, Grefenstette, and Blunsom, 2014)). In brief, CNNs use many layers of a mathematical operation called convolution, where a

small matrix of parameters is convolved with the input. If the input were an image then the convolution operation can be visualised as sliding a small filter over the image by a certain number of pixels each step, and computing a score based on the learned weights in the filter. These layers are often followed by pooling and a ReLU non-linearity (Krizhevsky, Sutskever, and Hinton, 2012).

It is possible to turn the learned filters from convolution into images themselves and visualise what they have learned to recognise. In lower layers, the features look like edges or blobs of colour. After four layers of convolution and pooling the features look like recognisable images such as a dog's face or car tire (Zeiler and Fergus, 2014). These features will then produce their highest output when they are compared to a dog in an image for example - indicating that the image contains a dog. For a more in depth explanation of CNNs, see Zeiler and Fergus, 2014.

2.1.2 Recurrent Neural Networks

RNNs are neural nets where part of the input to the network is the output of the same network at a previous timestep - the output of the network at time t is fed into the network at time $t + 1$. They are more relevant to this project due to their unique architecture which means they are naturally disposed towards processing sequential data such as text (LeCun, Bengio, and Hinton, 2015). RNNs have been used to great effect in tasks ranging from language modeling (Jozefowicz et al., 2016) to machine translation (Luong, Pham, and Manning, 2015) to drawing images (Gregor et al., 2015; Oord, Kalchbrenner, and Kavukcuoglu, 2016). They can even be combined with CNNs to generate captions for images (Xu et al., 2016).

One of the most famous RNN architectures is based on Long-Short Term Memory cells (Hochreiter and Schmidhuber, 1997). At each timestep, the LSTM-RNN feeds the input at that timestep, the output of the network from the previous timestep, and a cell state which the network maintains that can be thought of as its memory. LSTMs can handle long-range dependencies in the data far better than RNNs based on a single activation function and do not suffer from the "vanishing gradient" problem - where the error signal becomes too weak to update the weights efficiently. Cheng and Lapata, 2016 used LSTMs in their work on summarisation because of this.

Figure 2.1 shows a diagram of an LSTM-based RNN and how it takes an input at each timestep, in this case words of a sentence (which would be represented numerically using something like word embeddings (Bengio et al., 2003)). The line running horizontally across the top of the diagram represents the cell state, which at each timestep is updated using a combination of hidden layer "gates" each with a specific function. In the diagram, the gates are represented by yellow boxes with

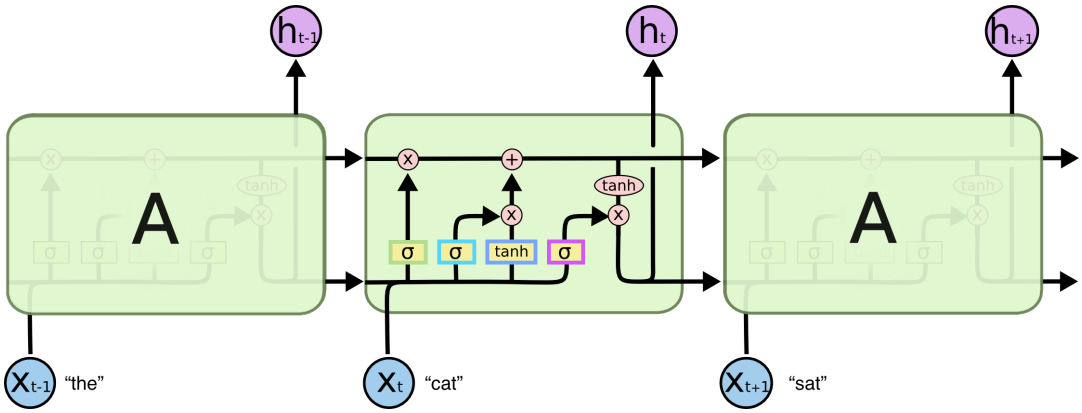

FIGURE 2.1: A diagram of an LSTM-based RNN showing the internal cell workings. Source: Colah's Blog

brightly coloured outlines.

The first gate is outlined in green and is called the "forget gate". Its purpose is to decide which elements of the cell memory to remove given the current input and previous output. The second gate outlined in light blue is called the "input gate", whose purpose it is to decide which values in the cell state will be updated. The third gate, outlined in dark blue, is called the "candidate gate" which outputs candidate values to update the cell state with. The final gate, outlined in purple, is the "output gate" which decides how much of the current input and previous output to include in this timestep's output. Representing the output of the previous timestep as h_{t-1}, the current input as x_t and the cell state from the previous timestep as C_{t+1}, the equations for the gates are:

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \tag{2.1}$$

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \tag{2.2}$$

$$\tilde{C}_t = tanh(W_c \cdot x_t + U_i \cdot h_{t-1} + b_c) \tag{2.3}$$

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \tag{2.4}$$

Where W and U are matrices of weight parameters, b is a bias vector, σ represents a sigmoid activation function. To update the cell state and give an output, the previous cell state C_{t-1} is multiplied by the forget gate output and then added to the input gate multiplied by the candidate gate:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2.5}$$

Where $\odot$ represents the Hadamard product. Finally the output from the cell is produced by multiplying the output gate with the cell state which has been compressed

to have values $\in (0, 1)$:

$$h_t = o_t \odot tanh(C_t) \tag{2.6}$$

LSTMs can be stacked into multiple layers to make them deeper, overlaid to have an LSTM reading the input in either direction and have several variants such as peepholes (Gers and Schmidhuber, 2000), all of which can make them more effective. LSTMs have also been successfully combined with a mechanism called attention (Bahdanau, Cho, and Bengio, 2015), which focuses on particular parts of the inputs to mimic the way humans read.

2.2 Word Embeddings

One of the applications of deep learning is the learning of continuous representations of words (Bengio et al., 2003). Learning word vectors works from the principle:

"You shall know a word by the company it keeps"
- (Firth, J. R. 1957:11)

Essentially that the semantic meaning of a word is defined by those used around it. Bengio et al., 2003 were the first to apply deep learning to learning of word embeddings, where they were used in a language modeling context because semantically similar words would be close to each other in the embedded vector space. Mikolov et al., 2013 and Pennington, Socher, and Manning, 2014 developed improved versions of the word embedding algorithm a decade later which runs much faster.

Once word embeddings are created, concepts can be represented by vector arithmetic. Mikolov et al., 2013 give the example of vec("Germany") + vec("capital") = vec("Berlin"). These continuous representations of words are extremely useful for many tasks in NLP and are often used as input to algorithms like RNNs (Jozefowicz et al., 2016).

Word embeddings in their raw form, that is with no processing with advanced algorithms like RNNs, have been used in summarisation before (Kageback et al., 2014) as well as being input to more advanced algorithms (Nallapati, Zhai, and Zhou, 2016). Extensive use of word embeddings is made in this work as they are used whenever the raw data of a sentence to classify as summary or not summary is being fed into an algorithm. The model used was the Word2Vec algorithm[1] pioneered by Mikolov et al., 2013 and trained on the corpus of scientific papers.

[1] Implementation used was from Gensim: https://radimrehurek.com/gensim/models/word2vec.html

2.3 The State of the Art in Extractive Summarisation

Work on extractive summarisation in recent years can be broadly classified into four categories: RNN-based, CNN-based, Other Deep Learning Algorithm Based and Feature Based. This section will go through the main techniques which have been used under each category. As they are particularly relevant to this project, RNN-based methods are described in more detail.

2.3.1 RNN-based

Dual RNNs Nallapati, Zhai, and Zhou, 2016 made extensive use of RNNs for extractive summarisation. They posed the extractive summarisation problem as a binary classification task for either "summary" (1) or "not summary" (0). They used an advanced architecture of RNNs based on Gated Recurrent Units (Cho et al., 2014), consisting of two bidirectional RNNs: one at sentence level and one at document level.

What this means is that the first RNN read each sentence, where each word was represented by its word embedding, in both directions at the same time, one word at a time. The output from this RNN was then fed into a second RNN which read each sentence in the document from both directions simultaneously. This second RNN was the arbiter of whether a sentence was a summary sentence or not, outputting a classification of either 0 or 1 at each timestep indicating whether that sentence was a summary sentence or not.

To train the system, the Daily Mail/CNN corpus was used which is the largest summarisation corpus currently in existence. It consists of news articles and associated highlights (Chen, Bolton, and Manning, 2016). Their results improved on state of the art performance with no feature engineering.

Extending the Dual RNNs Mostly the same authors as in Nallapati, Zhai, and Zhou, 2016 took the architecture which they defined and extended it slightly into two types of architecture which they called "Classify" and "Select" (Nallapati, Zhou, and Ma, 2016). It used the same dual-RNN architecture as the original research but slightly modified the way in which summaries were chosen.

"Classify" is modeled on the way in which humans would create an extractive summary: it reads the whole document once, then reads it again, this time picking out summary sentences as it goes. In so doing, it gains an understanding of the news story before it starts building summaries.

"Select" reads the whole document through once, then chooses sentences from anywhere in the document all at once to make up the summary.

The authors found "Classify" to be the more successful architecture, thought to be because reading and classifying sentences in order follows the discourse structure of the document more closely: sentences towards the start of the article have more of a summarising function, ones in the middle are more detailed and ones at the end are again summarising.

Encoder-Decoder Arguably the most advanced extractive summarisation system to date was developed by Cheng and Lapata, 2016. It is modeled on the Sequence-to-Sequence (Seq2Seq) model which has come to recent attention as an effective method to create chatbots and for machine translation (Luong et al., 2016).

Seq2Seq models use two RNNs with different roles: an encoder and a decoder. The encoder reads the input and outputs a vector representation of it, which is read by the decoder so that it can output another sequence, such as text in a different language. In the realm of summarisation, Seq2Seq models are more often used for abstractive summarisation such as in (Chen et al., 2016) than extractive summarisation.

The model which Cheng and Lapata, 2016 propose uses a CNN to encode each sentence as a vector representation. This is then read by an encoder RNN followed by a decoder RNN which uses an attention mechanism (Bahdanau, Cho, and Bengio, 2015) to focus the decoder on specific parts of the input. Humans naturally do this, giving their attention to important aspects of documents as they read them. The encoder and decoder RNNs used LSTM cells which prevented the vanishing gradient problem - where the gradient used to update the weights in backpropagation becomes too small to be useful.

This work again used the Daily Mail/CNN dataset as it is the only data source of sufficient scale for learning algorithms such as this, and again beat the previous state of the art. This technique achieved a higher score than Nallapati, Zhai, and Zhou, 2016.

A Cameo from Abstractive Summarisation Although this work is on extractive summarisation, the abstractive model developed by Chen et al., 2016 bears mentioning due to its interesting new approach to attention. The model is also based on Seq2Seq, and the authors raise the concern that simple vectors may not be sufficient to represent entire documents. To solve this they use both attention and a distraction mechanism, whereby they can attend to specific areas of the input document but can also "distract" the model to make it look at other parts of the document. Doing so

eliminates the need to read the whole document and hold it all in memory. They hypothesize that this approach will work better for longer documents.

Drawing it All Together All of these cutting edge techniques based on RNNs are complex and large. They all use multiple RNNs as opposed to just using a single network, either as an encoder-decoder pair or to read at different levels of abstraction from the data. It is surprising that, to our knowledge, no successful systems based on a single RNN for summarisation exist, because this seems to be the logical first step in using an RNN. None of the described techniques made use of any features at all - they worked purely from the data in true deep learning fashion; a technique we wish to replicate in this work as much as possible.

2.3.2 CNN-based

The Prior Nature of a Summary Cao et al., 2015 claimed that sentences can be good summaries even when taken completely out of context - they have a prior nature which allows us to see immediately, without the context of surrounding sentences, how good the summary is. To capture this concept with a learning algorithm, they created a CNN and trained it on a multi-document summarisation dataset formed of news articles. The outputs from the CNN were combined with a set of handcrafted features to classify the sentence. Although the task, multi-document summarisation, is different from this work which is single document, the idea that sentences can be good summaries without context is extremely important.

Other CNN Methods The majority of other CNN-based summarisation methods follow an approach more similar to the RNN-based approaches in that they do not use any handcrafted features at all (Denil, Demiraj, and De Freitas, 2015; Kim, 2014). Kim, 2014 used the word2vec model of (Mikolov et al., 2013) to represent words and then trained a CNN based on those vectors for a plethora of different sentence classification tasks, showing that CNNs achieved some of the best performances, while Denil, Demiraj, and De Freitas, 2015 used two convolutional nets to represent the whole document and each sentence and could ascertain how relevant a sentence was to a document using these encodings. This concept of comparing the encoding of a body of text and a sentence to gain an understanding of the sentence's relevance has been used in other approaches which are mostly feature based such as in Saggion, Abura'ed, and Ronzano, 2016, and is an interesting approach that could be taken in this work. However, CNNs and even RNNs may have difficulty with encoding a representation for a whole scientific paper due to its length.

2.3.3 Other Deep Learning Methods

CNNs and RNNs are far from the only deep learning algorithms that can be useful in summarisation. Yousefi-Azar and Hamey, 2017 built a query-oriented summarisation system based on the unsupervised method of deep autoencoders (Hinton and Salakhutdinov, 2006). Autoencoders are a type of neural network which perform dimensionality reduction on data. They take a large input vector and find a latent representation for it which is a much smaller vector. Yousefi-Azar and Hamey, 2017 used the term frequency of each sentence as input to the autoencoder which then represented each sentence as a vector. This work is particularly interesting because deep autoencoders could be a viable alternative to word embeddings for this system, or could be used to encode other parts of the document like the abstract.

A more drastic approach, and another contender for the most advanced system, was taken by Kong et al., 2017. They designed an entirely new type of neural network module which has dynamic connections - meaning that connections between neurons can be dynamically altered at runtime. They claim to improve the encoder / decoder architecture of Seq2Seq models by including explicit structure in the text. The framework is said to be less sensitive to text length than standard RNNs or Seq2Seq models, which could make it particularly applicable to this work. Scientific documents are longer than the news articles commonly used to train deep learning based extractive summarisers; so RNNs and Seq2Seq models could suffer considerably when attempting to perform this task - using a completely new architecture such as this could be the solution.

An approach which is more loosely based on deep learning is to use word embeddings without using advanced algorithms on top of them (Kageback et al., 2014; Kobayashi, Yatsuka, and Taichi, 2015). Kageback et al., 2014 used word embeddings in multi-document summarisation and found them to significantly increase performance. The most interesting thing about their work is the two ways that they present of combining word vectors of a sentence into a single sentence vector: a simple way and a complex way. The simple way is to simply add all the word vectors of the sentence up; the complex way is to use a recursive autoencoder (Socher et al., 2011). They found that the simple way, vector addition, significantly outperformed the recursive autoencoder; a result that may seem counter-intuitive but advocates simplicity and is of particular interest to this work as extensive use of word embeddings is made.

2.3.4 Feature-based Methods

Extractive summarisers started out as feature-based systems (Luhn, 1958) and this has remained their predominant paradigm ever since. The earliest features were

word frequency (Luhn, 1958), location in the document (Baxendale, 1958) and Term Frequency Inverse Document Frequency (TF-IDF), developed by Salton, Wong, and Yang, 1975 and first used in summarisation by Salton et al., 1996.

It is only since 2012 that systems capable of learning from raw data have picked up (LeCun, Bengio, and Hinton, 2015), such as (Nallapati, Zhai, and Zhou, 2016). Despite the acceleration in deep learning based methods, many works published today are still feature-based (e.g. Wan and Zhang, 2014; Ren, Wei, and Chen, 2016; Dlikman and Last, 2016). The features used have become steadily more advanced in the past 60 years. While the original summarisation system proposed by Luhn, 1958 used only a single feature - how relevant words in each sentence were to the document - modern systems often use a combination of up to 30 features (Dlikman and Last, 2016; Litvak et al., 2016).

Commonly used features include sentence length, sentence location, TF-IDF, presence of cue word and words in the title (Kupiec, Pedersen, and Chen, 1995; Teufel and Moens, 2002; Saggion, Abura'ed, and Ronzano, 2016; Ren, Wei, and Chen, 2016). Sentence location has been said to be of particular importance to summaries (Edmundson, 1969; Mani, 2001), and it has been noted that the first few sentences of news articles tend to be good summaries for the rest of the article (Lin and Hovy, 1997). In addition to these common features which are almost universally included in any summariser, many more complicated features have been developed.

A good example of a more advanced feature is the certainty metric developed by Wan and Zhang, 2014. They observed that in news documents, sentences are sometimes not certain, for example using phrases like "it seems that **X** will **Y**...". Uncertain sentences may not produce good summaries, so they developed their own training set and "certainty features" such as whether the sentence contained specific certainty marker words like "seems", then used support vector regression to give each sentence a certainty score, rather than a coarse classification as either "certain" or "uncertain". This example is of particular interest because of the concept of using two intertwined machine learning algorithms - one to generate a score (certainty measure) and the other to rank based off of that, rather than simply classifying as certain or uncertain. Using this continuous representation of a feature rather than a binary feature could be a good approach to take and could be more expressive.

Dlikman and Last, 2016 also took a feature-based approach, using 30 different features with many part-of-speech (POS)-based features. They then compared three

different learning algorithms: logistic regression, regression trees and a genetic algorithm (also used by Litvak, Last, and Friedman, 2010), finding that logistic regression significantly outperformed the other two methods. The reason this is interesting is because it advocates simplicity with a parameter-based approach. Although not state of the art, many other algorithms for ranking based on features have been trialled including unsupervised methods like K-means clustering (Nomoto and Matsumoto, 2001) and Bayesian methods like Hidden Markov Models (Conroy and O'leary, 2001). The fact that the predominant method today appears to be logistic regression suggests that it was a more successful approach than these algorithms and would be a good baseline for this work.

One of the predominant ways of ranking in summarisation in the past has been to used graph-based unsupervised methods such as TextRank (Mihalcea and Tarau, 2004). Graph based methods are good at modeling the dependency between sentences so that context can be taken into account, and are still widely in use today (Wan and Zhang, 2014; Thomas et al., 2015; Fang et al., 2017). Using graphs also allows redundancy to be eliminated easily, although other approaches based off linear regression can also take account of redundancy such as used by Ren, Wei, and Chen, 2016. They used the ROUGE evaluation metric (Lin, 2004) to check whether candidate summary sentences should be included in the summary rather than having a separate module to prevent redundancy as many systems do (e.g. (Cao et al., 2015)).

A final approach more closely related to graph based methods than feature based methods is constructing a discourse tree for the document to summarise and using this (Hirao et al., 2015). The discourse structure of a document is the inherent structure in a piece of text that determines the relationships between units of text and their saliency with regard to summarisation. Discourse structure can be represented as a tree, and the argument goes that by extracting sentences to be summaries they may lack logical coherence - something which could be prevented by using a discourse tree. Use of the discourse tree would allow modeling of the interrelationships between entities of knowledge in the document so that a logical structure to summaries could be ensured. Document discourse or structure is an extremely important consideration to take into account when summarising scientific literature because almost all papers have a similar structure (Liakata et al., 2010), and discourse has specifically been taken into account previously (Cohan and Goharian, 2015).

2.4 Summarising Scientific Papers

Back in 1958, the very first work ever done on summarisation was targeted at summarising scientific papers (Luhn, 1958). Since then, the majority of summarisation work has been focused on news articles (see Section 2.3), which are far shorter than

scientific documents. The length of papers poses a significant challenge (Kavila and Radhika, 2015) as there is a lot of information to transmit to the reader in a short summary.

Some work has been done before on summarising extremely long documents, such as by Mihalcea and Ceylan, 2007 who attempted to summarise books. Even a scientific paper is short compared to most books; the average book length used by Mihalcea and Ceylan, 2007 was 92000 words. They made some interesting discoveries about how the efficacy of traditional features in summarisation changes for long documents. The most interesting of these is the feature of sentence position in the text, which is thought to be one of the most important to a summarisation system (Edmundson, 1969; Mani, 2001). They found that sentence position, in terms of the number of sentences since the beginning of the document significantly reduced performance of the summariser because writing styles changed throughout the book. Instead they divided the book into segments and assigned each sentence a position feature based on its segment. This way of constructing features is of interest to this work because scientific documents are long in comparison to news articles and naturally split into segments with different sections.

In previous work on summarising scientific literature, a common theme has been to make extensive use of citations (Abu-jbara and Arbor, 2011; Qazvinian et al., 2013; Cohan and Goharian, 2015; Saggion, Abura'ed, and Ronzano, 2016). Citations are thought to provide short and focused technical summaries of the paper being cited in the text around the citation (Cohan and Goharian, 2015; Saggion, Abura'ed, and Ronzano, 2016). However, citation-based summaries only tend to provide summaries of specific concepts in a paper tailored to the author's point of view rather than of the whole paper (Elkiss et al., 2008). Nonetheless, citations are largely unique to scientific literature and should be exploited where possible.

Aside from citations, research papers also have a specific structure which can be utilised (Teufel, Siddharthan, and Batchelor, 2009; Liakata et al., 2010). Kavila and Radhika, 2015 capitalized on paper structure by limiting the sections which sentences could be extracted from to the Abstract, Introduction and Conclusion. This makes intuitive sense because the Abstract is already a summary, the Conclusion tends to summarise the main message and achievements of the paper and the Introduction gives a basic background knowledge. However it could be argued that there is no point in extracting summary sentences from the abstract because the abstract is already a summary in itself, so readers could just read that. The abstract does not always have the same writing style as the rest of the paper either (Teufel and Moens, 2002). A more useful system would generate summaries which supplement the abstract from the rest of the paper, as the abstract does not always capture

all contributions and impacts of a paper (Elkiss et al., 2008), so it would be advantageous to capture these automatically from the rest of the paper.

As with the majority of summarisation systems, paper summarisers are also largely feature based. Citations and paper structure are usually part of the features. One such example is from Saggion, Abura'ed, and Ronzano, 2016, who used citation network, structure and features in their summariser. One of the ways in which structure is incorporated as a feature here is by encoding sentences, sections and the entire document as vectors using the TF-IDF metric. Once these encodings were constructed, sentence vectors could be compared to the abstract, title and document vectors with cosine similarity to get a sense of how relevant a sentence was to a document. A similar approach was also taken by (Kupiec, Pedersen, and Chen, 1995). Features of this style are of particular interest because they can capture structural information about a paper in a single number. To our knowledge, this technique has not been attempted using more advanced encodings than TF-IDF such as word embeddings, deep autoencoders or RNNs and is an attractive avenue of exploration.

Some of the most common features used are sentence length, sentence position in paper, sentence overlap with or similarity to the title, sentence TF-IDF score and presence of cue phrases (e.g. "In conclusion...") (Teufel and Moens, 2002; Visser and Wieling, 2009; Saggion, Abura'ed, and Ronzano, 2016). Other features used include TextRank score (Saggion, Abura'ed, and Ronzano, 2016) and custom features such as whether the sentences contains words from a lexicon of "action words" and the history of patterns in rhetorical zones (Teufel, Siddharthan, and Batchelor, 2009).

These kinds of algorithms based on features perfectly represent the traditional approach to machine learning: careful feature engineering and domain expertise in knowing what makes a good summary (LeCun, Bengio, and Hinton, 2015). To the best of our knowledge, no research exists which has attempted to use deep learning in any form to summarise scientific papers.

2.5 Summarisation Datasets

The majority of deep learning algorithms are trained on extremely large datasets. For example, the ImageNet database (Deng et al., 2009) has over a million images which can be used for training and the CNN / Daily Mail news dataset has around 1.2 million news articles and associated highlights for training summaries (Chen, Bolton, and Manning, 2016). CNN / Daily Mail is the only natural language dataset of its size which is suitable for training deep learning algorithms for summarisation. It was first used by Nallapati et al., 2016 to train an abstractive summariser and is

now used in most of the state of the art techniques (see Section 2.3). Cheng and Lapata, 2016 do indeed say that one of the stumbling blocks of applying deep learning to extractive summarisation is the lack of training data.

It is perhaps this poverty of data that has prevented deep learning from being applied to summarisation of scientific papers. None of the existing datasets for summarisation of papers are anything like the size of CNN / Daily Mail. One such example is the CL-SciSumm Task, specifically designed for summarising papers (Jaidka et al., 2016). Each of these papers has three summaries associated with them: an author written abstract, citation network summary and hand written summary, however the most recent competition only contained 30 publications [2].

Other work on the summarisation of papers has similarly small datasets: Ronzano and Saggion, 2016 used a set of 40 papers; Kupiec, Pedersen, and Chen, 1995 used 21 and Visser and Wieling, 2009 used only 9 papers. The largest known scientific paper dataset was used by Teufel and Moens, 2002, who used a subset of 80 papers from a larger corpus of 260 articles.

The dataset which we introduce in this work is, to our knowledge, the only large dataset for summarisation of scientific papers, and its size is sufficient to train data intensive neural methods.

[2] http://wing.comp.nus.edu.sg/cl-scisumm2017/

Chapter 3

Method

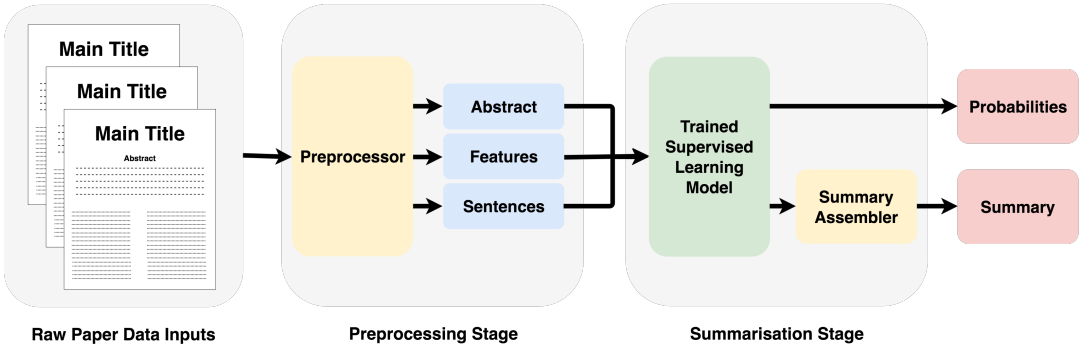

FIGURE 3.1: A diagram of the summary creation pipeline. On all diagrams: blue indicates a network input, green a neural network, yellow a data processing operation and red an output.

This chapter describes the approaches taken to solve the challenges set out in the objectives. Doing so involved developing techniques to analyse the dataset, preprocessing the data into a form suitable for summarisation and then developing summarisation methods. Figure 3.1 provides a visual representation of the general approach.

Although many of the approaches described in Chapter 2 made use of citations to summarise papers, we have decided to narrow our focus and only work on finding ways of applying deep learning to summarising papers and will not be using citations, although they are a potential future development. Likewise, we will not be using any graph-based methods.

3.1 Dataset and Problem Formulation

The raw data used in this work is a novel summarisation dataset consisting of 10148 computer science publications which were obtained from ScienceDirect[1]. Computer Science is one of 27 domains on ScienceDirect, meaning that the dataset could easily

[1] http://www.sciencedirect.com/

be extended to include more domains. Every paper in this dataset is guaranteed to have a title, abstract, author defined keywords and author written highlight statements. The highlight statements are bullet points that should effectively convey the main message of each paper and are our object of interest in this work. By their nature, the highlight statements are largely independent from eachother; each bullet point does not tend to rely on ones before it to make sense. They are good examples of what Cao et al., 2015 call summaries with a prior nature - ones that make sense and are good summaries even when taken out of context. The highlights therefore provide a completely different type of summary of the paper than the abstract does and are good targets for extractive summarisation. The abstract, as a coherent block of text rather than a set of bullet points, does tend to have dependencies between sentences, and would make a good target, or *gold* summary, for abstractive summarisation.

3.1.1 Problem Formulation

We wished to approach the extractive summarisation task as a *supervised* learning (SL) task so that we could make use of deep learning techniques which have not, to our knowledge, been applied to summarising scientific papers before. To do this, we took inspiration from Cao et al., 2015, who said that sentences can be good summaries even when taken out of context of the surrounding sentences. As most of the highlight statements in the dataset have this characteristic, we elected to frame the extractive summarisation training task as binary sentence classification, where we assign each sentence in a document a label $\in 0, 1$ to indicate whether it is a summary sentence or not.

When using the trained model to construct summaries, rather than using a coarse classification of 0 or 1, we used the softer probability score produced by the network for the sentence being a summary and rank sentences based on this. Figure 3.1 gives a visual intuition for how the system functions when producing summaries using a trained network.

3.1.2 Creation of the Training and Testing Data

To create an SL-based system, the data needs to be in a suitable form to train a classification algorithm which involved a preprocessing phase on the raw papers. As the problem was framed as a sentence classification task based on the idea that sentences can be good summaries out of context, the training dataset constructed consisted of sentences drawn from all 10148 papers in a random order, without the context of their surrounding sentences. The goal was then that the summariser would be able to learn to identify the sentences with a good prior nature, as Cao et al., 2015 did. Taking sentence context into account would require encoding the entire scientific

Dataset	#documents	# +ive instances
CSPubSum Train	10148	42745
CSPubSumExt Train	10148	131720
CSPubSum Test	150	625
CSPubSumExtTest	10148	65860

TABLE 3.1: The CSPubSum and CSPubSumExt datasets. Instances are items of training data with a positive classification, there are an equal number of negative examples. CSPubSumExt is CSPubSum extended with HighlightROUGE.

document somehow, a very difficult task given that they are so long which is left to future development.

Two datasets result from preprocessing the raw papers: *CSPubSum* and *CSPubSumExt*, each of which have *training* and *test* sets and are detailed below. Statistics for the same datasets are show in in Table 3.1.

CSPubSum Train This dataset uses the highlight statements from each paper as the positive instances. Each highlight statement is given a label of 1 and stored in a list in random order - giving a total of 42745 positive training instances. To generate negative data, we measure how similar each sentence in each paper is to its highlight statements using the ROUGE-L (Lin, 2004) score, which we also use as an evaluation metric. Full detail on the ROUGE metrics is given in Section 3.2, but briefly ROUGE-L will assign a sentence a score between 0 and 1 based on how similar it is to the summary, 1 being identical. From the bottom 10% of sentences in a publication which are the least similar to its highlights, we draw an equal number of negative samples to the positive samples. The resulting dataset has 85490 training examples.

CSPubSum Test This dataset is organised differently to CSPubSum Train. It is not a randomised list of sentences with classifications, it is a set of 150 full papers with no labels and a total of 625 highlights. This test set is used to test fully constructed summarisation systems - where their task is to produce a summary of the main text of the paper, which is then compared to the highlight statements of each paper using the ROUGE-L evaluation score. Summarisers which achieve higher ROUGE scores are better quality summarisers.

CSPubSumExt Train and Test The CSPubSum dataset has two drawbacks: 1) though fairly large, it is still an order of magnitude behind comparable large summarisation datasets (Hermann et al., 2015; Nallapati, Zhai, and Zhou, 2016) and is unlikely to be sufficient to train deep learning algorithms; 2) it does not have positive labels for

any sentences in the main body of the paper, only the highlights. To solve both of these issues, we developed a method called *HighlightROUGE* which can be used to generate extra training data for summarisation given a human written summary and a body of text to summarise. Full detail is given in Section 3.2. Using this method allowed the CSPubSum training set to be extended to comprise some 400K training instances. This was split into $\frac{2}{3}$ training data (263K instances) and $\frac{1}{3}$ testing data (132K instances). The form of this testing data is more like a traditional SL test set - it, like the training set, is a list of randomized sentences and their classification rather than a set of full papers like CSPubSum Test.

Having two different styles of testing data is extremely valuable to this project. CSPubSumExt Test can be used to test the accuracy of the base summarisation models, while CSPubSum Test can be used to test their actual performance as summarisers. By comparing the two values we receive, we can analyse whether the randomized list of sentences is a good way of training a summarisation system: if the accuracy values produced from this test set are discordant with actual summarisation performance, we know that we need to change the form of our training data. If they are harmonious, then we know we have found a good way of representing the data for training. Having the list of randomised sentences to train from makes designing learning algorithms far easier than if we had to take the context of surrounding sentences into account.

3.2 ROUGE Metrics

Recall-Oriented Understudy for Gisting Evaluationg (ROUGE) is a metric designed to compare automatically generated summaries with human written summaries and measure numerically how similar they are, and thus how good the summary is (Lin, 2004). A higher ROUGE score is shown in its introductory paper to correlate well with human judgement of how good a summary is. There are many variations of the ROUGE metric such as ROUGE-N and ROUGE-L. In this work we opt to use the ROUGE-L metric, as used in other research into summarising scientific articles (Cohan and Goharian, 2015; Jaidka et al., 2016). In addition, ROUGE-L is based on the longest common subsequence of a sentence with a summary, and as there are instances of authors copying text directly from the main body of the paper to the highlights, the ROUGE-L score would be maximised if a summariser could select these same sentences for the summary.

3.2.1 ROUGE-L

As detailed by Lin, 2004, ROUGE-L is based on the longest common subsequence (LCS) of two strings. A common subsequence between two strings Z, X is a sequence of words in order from Z which also occur in the same order in X. The longest common subsequence is the longest such sequence. To use LCS in summarisation evaluation, we have two lists of sentences: a *reference* summary (U, with m total words) which is the human written summary, and a *candidate* summary (V, with n total words) which is automatically generated. An LCS-based F-measure is then used to estimate the similarity of two summaries using the following equations:

$$R_{lcs} = \frac{\sum_{u_i \in U} LCS_{\cup}(u_i, V)}{m} \tag{3.1}$$

$$P_{lcs} = \frac{\sum_{u_i \in U} LCS_{\cup}(u_i, V)}{n} \tag{3.2}$$

$$\text{ROUGE-L} = \frac{(1+\beta^2) R_{lcs} P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{3.3}$$

Where β in our work is set to 1.2, and $LCS_{\cup}(u_i, V)$ finds the length of the set formed by taking the union of all longest common subsequences between the reference sentence u_i and set of candidate summary sentences V.

The implementation of ROUGE-L which is used in this work is taken from Github from user harpribot[2]. This implementation is written in the same language as used to create the system (Python) and is fast.

3.2.2 HighlightROUGE

HighlightROUGE is a metric developed in this work used to generate additional training data for summarisation tasks. As input it takes a gold summary and body of text and finds the sentences within each paper that give the best ROUGE-L score in relation to the highlights - these sentences represent the ideal sentences to extract from each paper. Formally, the ideal summary for each paper is:

$$S_o = \underset{S=\{s_1,s_2,\ldots,s_n\}}{\arg\max} \left[\frac{1}{n} \sum_{s_i \in S} \text{ROUGE-L}(s_i, gold) \right]$$

Where S_o is the summary that would be generated by an oracle, n is the number of sentences in the summary and S is a set of n sentences forming the candidate summary. This problem can be easily solved by simply computing the ROUGE-L score between every sentence in the paper and the summary, sorting by score and

[2]https://github.com/harpribot/nlp-metrics/blob/master/rouge/rouge.py

taking the top-n sentences. We set $n = 10$ in this work to generate an additional 101480 positive sentence-label pairs to use as training data. Negative examples are randomly sampled from sentences with the lowest 10% of ROUGE-L scores to match the number of positive examples.

It is important to note that when generating data using HighlightROUGE, no sentences from the abstracts of any papers were included as training examples. This is because the abstract is already a summary; our goal is to extract salient sentences from the main paper to supplement the abstract, not from the preexisting summary.

3.2.3 AbstractROUGE

AbstractROUGE is used as a feature for summarisation and takes inspiration from techniques used by Saggion, Abura'ed, and Ronzano, 2016 and Kupiec, Pedersen, and Chen, 1995. It is a metric which exploits the known structure of a paper by making use of the abstract, a preexisting summary. The idea of AbstractROUGE is that sentences which are good summaries of the abstract are also good summaries of the highlights; a hypothesis that we test in this work.

This approach differs from Saggion, Abura'ed, and Ronzano, 2016 and Kupiec, Pedersen, and Chen, 1995 in that they measure the similarity of each sentence to the abstract by encoding the sentence and abstract with TF-IDF vectors and comparing them with cosine similarity, whereas we directly measure how good-a summary a sentence is for the abstract with an appropriate evaluation metric.

The AbstractROUGE score of a sentence is simply the ROUGE-L score of that sentence and the abstract. Use of this feature was another reason we could not include sentences from the abstract in the training data: their AbstractROUGE score would be 1. This would bias the data unfairly given that abstract sentences are not in the set of sentences which could form a summary.

3.3 Sentence Encoding

In order to be processed by a neural network or any machine learning algorithm, natural language has to be transformed into a numeric form. We chose to do this using word embeddings[3] (Mikolov et al., 2013), and present two different ways of using them which are depicted visually in Figure 3.2.

[3]Word embeddings are obtained by training a Word2Vec skip-gram model on the CSPubSum Train dataset with dimensionality 100, minimum word count 5, a context window of 20 words and downsample setting of 0.001

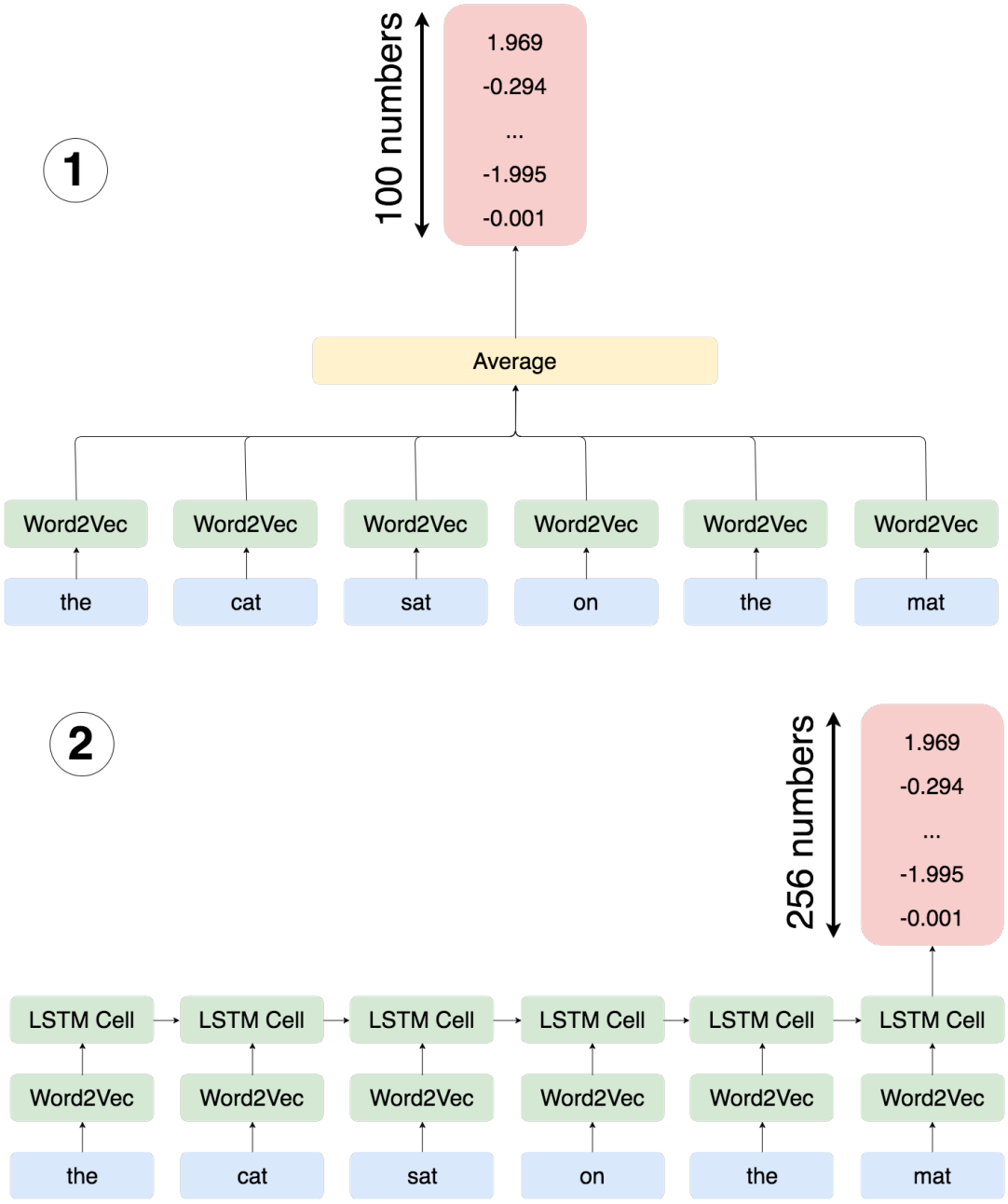

FIGURE 3.2: A diagram of the two ways in which sentences are encoded.

The first and simpler way is shown in diagram (1) of Figure 3.2, where we simply average the vector representation of each word resulting in a 100 dimensional vector for the sentence.

The second and more complex way (Figure 3.2, diagram (2)) uses a bi-directional (reads the sentence from both directions) LSTM-RNN; a bi-directional LSTM was chosen empirically. Although Kageback et al., 2014 tried more complex ways of encoding sentences than simply summing the constituent word vectors, namely with a

recursive autoencoder, they found that the simpler method of summing performed best. We wished to test if the same were true of an RNN for sentences in this more complicated domain of scientific paper summarisation. The LSTM maintained an internal cell state of 128 numbers, and because it was bidirectional the final cell states of the forwards and backwards RNNs were concatenated to give a vector representation of the sentence consisting of 256 numbers.

3.4 Feature Engineering

As the sentences in our dataset are randomly ordered, there is no readily available context for each sentence from surrounding sentences - taking this into account is a potential future development. To provide context, we therefore turned to the traditional summarisation method of using handcrafted features. Eight of these were created for each sentence and are described below. Our system now bears resemblance, although a different architecture and dataset, to the work of Cao et al., 2015 who used a CNN to learn features about a sentence with no context then combined this with a set of handcrafted features. The features used are:

AbstractROUGE A metric which measures how similar a sentence is to the abstract, see Section 3.2.3.

Location Scientific papers have a very specific structure (Liakata et al., 2010; Teufel, Siddharthan, and Batchelor, 2009) and the section which a sentence appears in can be a good indicator of its likely utility as a summary. Kavila and Radhika, 2015 for example, when generating summaries for papers, only considered sentences which appeared in the abstract, introduction or conclusion for summary generation; and Saggion, Abura'ed, and Ronzano, 2016 used the section which the sentence appeared in as a feature. More exotic location features have also been used in summarising papers - Visser and Wieling, 2009 extract sentence location features based on the headings they occurred beneath and Teufel and Moens, 2002 divide the paper into 20 equal parts and assign each sentence a location based on which segment it occurred in - an attempt to capture distinct zones of the paper. They also exploit section structure, noting that sentences toward the start of a section tend to have a summarising function.

We follow a similar approach, and sentences are assigned an integer location for one of 7 locations: Highlight, Abstract, Introduction, Results / Discussion / Analysis, Method, Conclusion, all else[4].

[4]based on a small manually created gazetteer of alternative names

Numeric Count is the number of numbers in a sentence, based on the intuition that sentences containing heavy maths are unlikely to be good summaries when taken out of context.

Title Score Visser and Wieling, 2009 and Teufel and Moens, 2002, in their work on summarising scientific papers used the Title Score. Our feature differs slightly from Visser and Wieling, 2009 in that we only use the main paper title whereas Visser and Wieling, 2009 use all section headings. To calculate this feature, the non-stopword words that each sentence contains which overlap with the title of the paper are counted. As the title has a very high salience with regard to the gold summary (see Results), this is a good measure of a sentence's likely utility as a summary.

Keyphrase Score Authors such as Spärck Jones, 2007 refer to the keyphrase score as a useful summarisation feature. The feature uses author defined keywords and counts how many of these keywords a sentence contains, the idea being that important sentences will contain more keywords.

TF-IDF Term Frequency, Inverse Document Frequency (TF-IDF) is a measure of how relevant a word is to a document (Ramos, Eden, and Edu, 2003). It takes into account the frequency of a word in the current document and the frequency of that word in a background corpus of documents; if a word is frequent in a document but infrequent in a corpus it is likely to be important to that document. TF-IDF was calculated for each word in the sentence, and averaged over the sentence to give a TF-IDF score for the sentence. Stopwords were ignored.

In more detail: TF-IDF is calculated on a word-by-word basis. Term frequency is calculated by counting the occurrences of each word in a paper and is denoted by $f_{t,d}$ for the frequency of term t in document d. Inverse document frequency is defined by the equation:

$$idf(t, D) = \log \frac{N}{1 + |\{d \in D : t \in d\}|}$$

Where N is the total number of documents in the corpus (10148 in this case), D is the set of all documents and $|\{d \in D : t \in d\}|$ is the count of the number of document in which term t appears one or more times. A 1 is added to the denominator to ensure there is never a division by 0. TF-IDF is then simply calculated by:

$$\text{TF-IDF}(t, d, D) = f_{t,d} \cdot idf(t, D)$$

$$\text{SentenceTF-IDF}(S, d, D) = \frac{1}{n} \sum_{w \in S} \text{TF-IDF}(w, d, D)$$

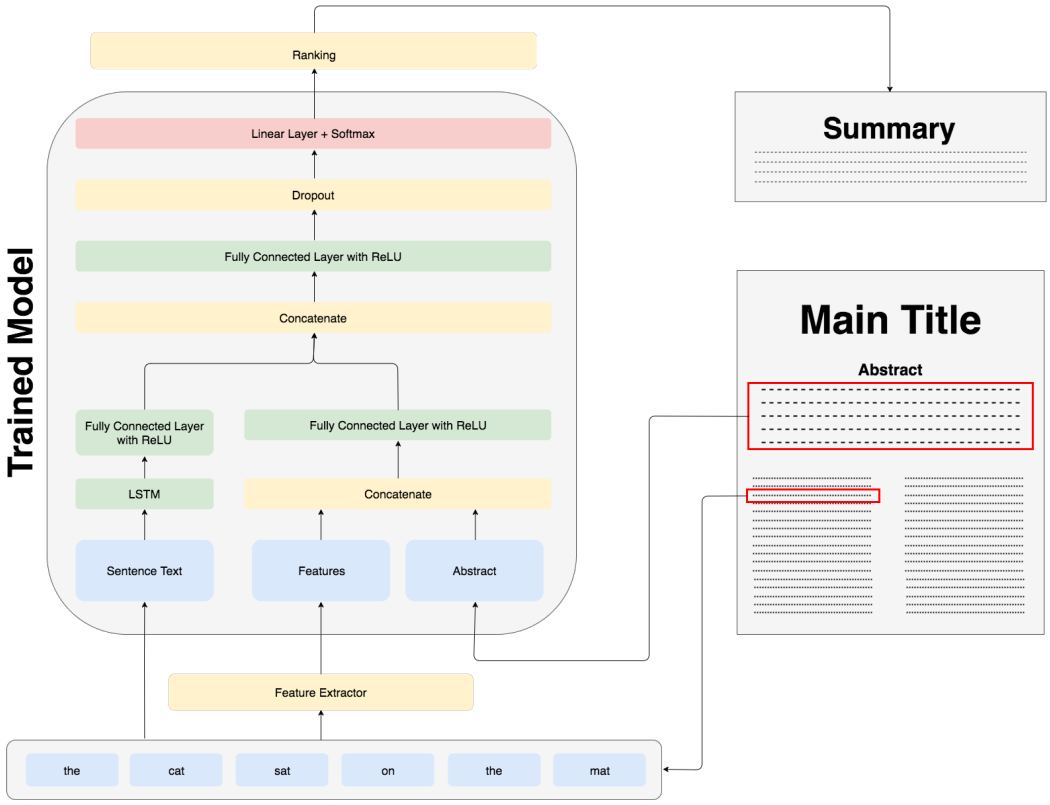

FIGURE 3.3: A diagram of the summarisation system. The neural network in the box titled "Trained Model" can be replaced with any summarisation method as desired.

Where n is the number of words in sentence S and all other symbols are as above.

Document TF-IDF Document TF-IDF calculates the same metric as TF-IDF, but uses the count of words in a sentence as the term frequency and count of words in the rest of the paper as the background corpus. This gives a representation of how important a word is in a sentence in relation to the rest of the document.

Sentence Length Teufel and Moens, 2002 created a binary feature for if a sentence was longer than a threshold. We simply include the length of the sentence as a feature.

3.5 Summariser Architectures

Figure 3.3 shows in some detail how a summary is created. The network depicted in the "Trained Model" box is one of the neural networks used in this system, but can be replaced with any other method which produces a score for how good a summary a sentence is - with a higher score indicating a higher chance of being a summary.

The models described in this section could take any combination of four possible inputs, and are named accordingly:

- S: The sentence encoded with an RNN (diagram (2), Figure 3.2.
- A: a vector representation of the abstract of a paper, created by averaging the word vectors of every non-stopword word in the abstract. Since an abstract is already a summary, this gives a good sense of relevance when compared to a sentence vector. This is similar to AbstractROUGE in what it is trying to achieve but implemented differently - here the network can interpret the vector however it wants - it is learning from the raw data of the abstract rather than the feature AbstractROUGE.
- F: the 8 features listed in Section 3.4.
- Word2Vec: the sentence represented by taking the average of every non-stopword word vector in the sentence (diagram (1), Figure 3.2)

Model names containing "Net" use a neural network with one or multiple hidden layers. Models ending with "Ens" use an ensemble. All non-linearities are Rectified Linear Units (ReLUs) as popularised by Krizhevsky, Sutskever, and Hinton, 2012, which were chosen due to their faster training time than traditional sigmoid or tanh activation functions. The regularisation technique used was dropout (Srivastava et al., 2014), which randomly drops connections between neurons in the network to prevent the network from becoming too dependent on any particular connection. It is effectively a cheap way of ensembling.

Single Feature Models The simplest class of summarisers use a single type of feature from Section 3.4. The sentences are sorted into descending order by the feature being used and the top-n sentences are taken as the summary, then sorted back into the order they appear in in the paper. Features Sentence Length, Numeric Count and Section are excluded.

Features Only: FNet A single layer neural net to classify each sentence based on all of the 8 features given in Section 3.4. A future development is to try this with other classification algorithms.

Word Vector Models: Word2Vec and Word2VecAF Both single layer networks. Word2Vec takes as input the sentence represented as an averaged word vector of 100 numbers (diagram (1), Figure 3.2). Word2VecAF takes the sentence average vector, abstract average vector and handcrafted features, giving a 208-dimensional vector for classification.

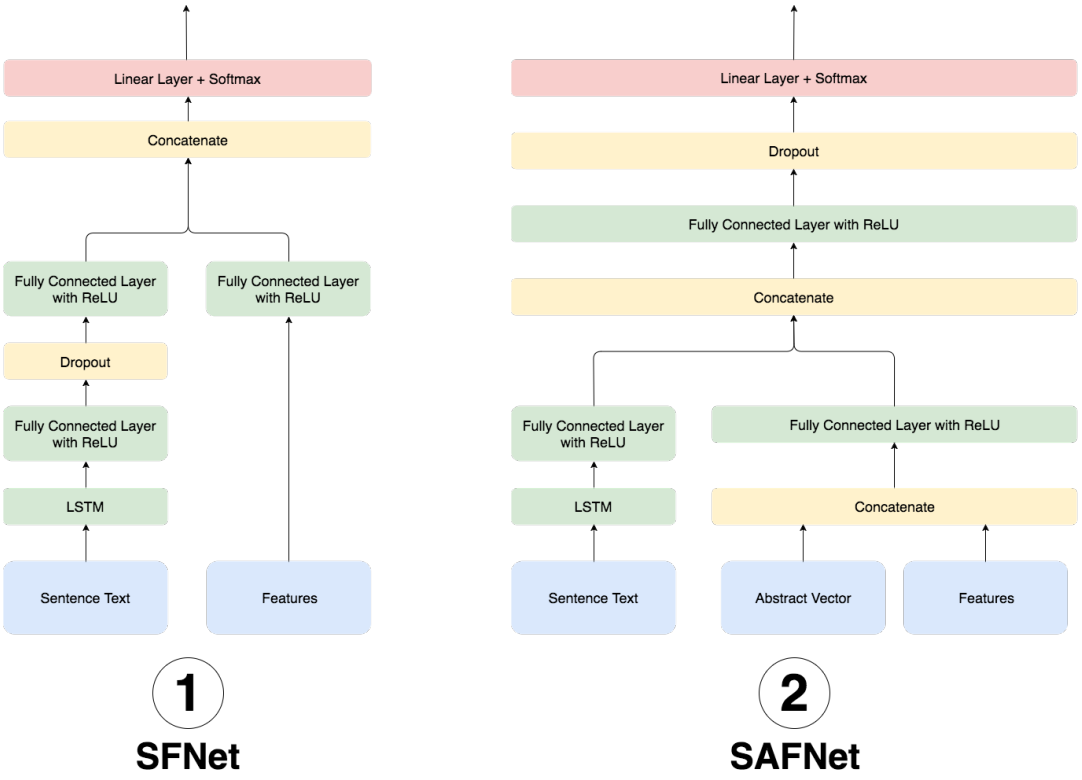

FIGURE 3.4: Architecture diagrams of the two most involved models.

LSTM-RNN Method: SNet Takes as input the ordered words of the sentence represented as 100-dimensional vectors and feeds them through a bi-directional LSTM-based RNN with 128 hidden units (diagram (2) Figure 3.2) and dropout. Dropout probability was set to 0.5 which is thought to be near optimal for many tasks (Srivastava et al., 2014). Output from the forwards and backwards LSTMs is concatenated and projected into two classes[5].

LSTM and Features: SFNet SFNet combines SNet and FNet into a single network trained jointly, harnessing the strengths of both nets simultaneously. Its architecture is shown in diagram (1) of Figure 3.4.

LSTM, Features and Abstract: SAFNet SAFNet, shown in diagram (2) of Figure 3.4 is the most involved architecture presented in this work, which further to SFNet also encodes the abstract using simple word embedding averaging.

Ensemble Methods: SAF+F and S+F Ensemblers The two ensemble methods use a weighted average of the output of two different models:

$$p(\text{summary}) = \frac{S_1(1-C) + S_2(1+C)}{2}$$

[5]The model is trained until loss convergence on a small dev set

Where S_1 is the output of the first summariser, S_2 is the output of the second and C is a hyperparameter. SAF+F Ensembler uses SAFNet as as S_1 and FNet as S_2 with $C = 0.4$. S+F Ensembler uses SNet as S_1 and FNet as S_2 with $C = 0.3$. Optimal values for C were obtained empirically by testing the ensemblers many times on the CSPubSum Test set with different values for C.

Oracle The oracle summariser was created to find the best possible extractive summaries for each paper in order to compare each model's performance to the best possible performance. It works by comparing every sentence in the paper to the gold highlights with ROUGE-L, ranking the sentences by their ROUGE-L score and taking the top 10 sentences, then resorting them into the order in which they appear in the paper. This ensures that the 10 sentences which give the highest ROUGE score for each paper are extracted.

Chapter 4

Results and Analysis

4.1 Most Relevant Sections to a Summary

A straight-forward heuristic way of obtaining a summary automatically would be to identify which sections of a paper generally represent good summaries and take those sections as a summary of the paper, or only select summary sentences from those sections. Kavila and Radhika, 2015 for example only took summary sentences from the abstract, introduction and conclusion. When reading a scientific paper for the first time, people will often skim-read it first or read specific sections[1], because certain sections of papers are more relevant to gaining a quick, high-level understanding of a paper than others. Evaluation metrics can be used to capture this intuition statistically by comparing the sentences of each section to gold summaries.

To understand how much each section contributes to a gold summary, we compute the ROUGE-L score of each sentence compared to the gold summary and average sentence-level ROUGE-L scores by section. The result is a score between 0 and 1 for each section depending on how relevant that section is to the highlights.

ROUGE-type metrics are not the only metrics which can be used to determine how relevant a section is to a summary. Throughout the data, there are approximately 2000 occurrences of authors directly copying sentences from within the main text to use as highlight statements. By recording from which sections of the paper these sentences came, we can determine from which sections authors most frequently copy sentences to the highlights, so may be the most relevant to a summary. This is referred to as the *Copy/Paste Score*.

Figure 4.1 shows the average ROUGE score for each section over all papers, and the normalised Copy/Paste score. The title has the highest ROUGE score in relation to the gold summary, which is intuitive as the aim of a title is to convey information about the research in a single line. The abstract has the second highest ROUGE

[1] https://www.elsevier.com/connect/infographic-how-to-read-a-scientific-paper

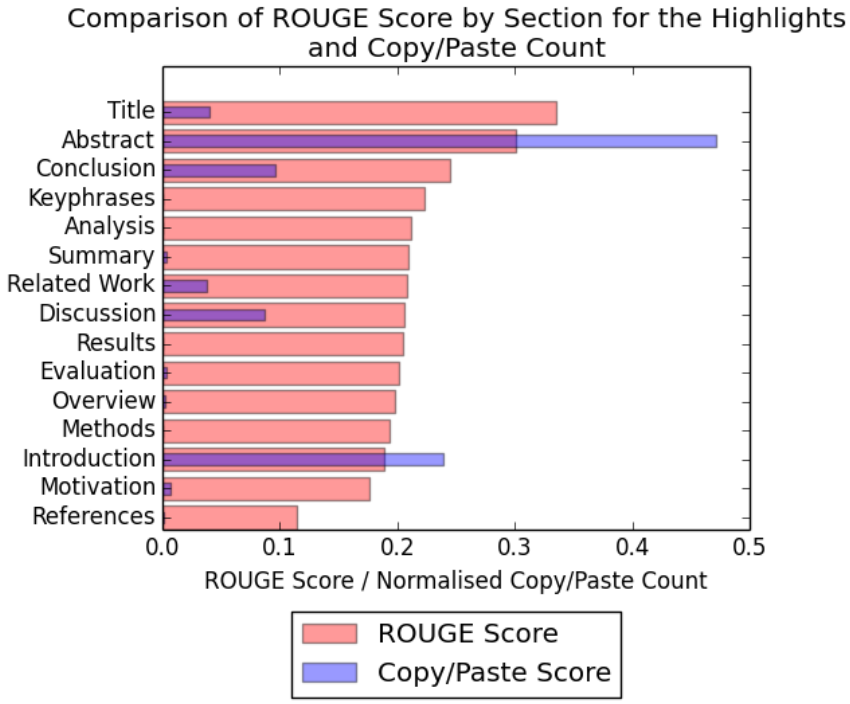

FIGURE 4.1: Comparison of the average ROUGE scores for each section and the Normalised Copy/Paste score for each section. The wider bars in ascending order are the ROUGE scores for each section, and the thinner overlaid bars are the normalised Copy/Paste count.

score which also makes sense because the abstract is already a summary itself. Conclusion takes third place, also an intuitive result because it often summarises the achievements and message of the paper.

A surprising result is that the introduction has the third-lowest ROUGE score in relation to the highlights. The initial hypothesis was that the introduction would be ranked after the conclusion because it is designed to give the reader a basic background knowledge of the problem. Indeed, the introduction has the second highest Copy/Paste score of all sections. The reason the introduction has a low ROUGE score but high Copy/Paste score is likely due to its length. The introduction tends to be longer (average length of 72.1 sentences) than other sections, but still of a relatively simple level compared to the method (average length of 41.6 sentences), thus has more potential sentences for an author to use in highlights, giving the high Copy-/Paste score. However it would also have more sentences which are not good summaries and thus reduce the overall average ROUGE score of the introduction.

Hence, although some sections are much more likely to contain good summary

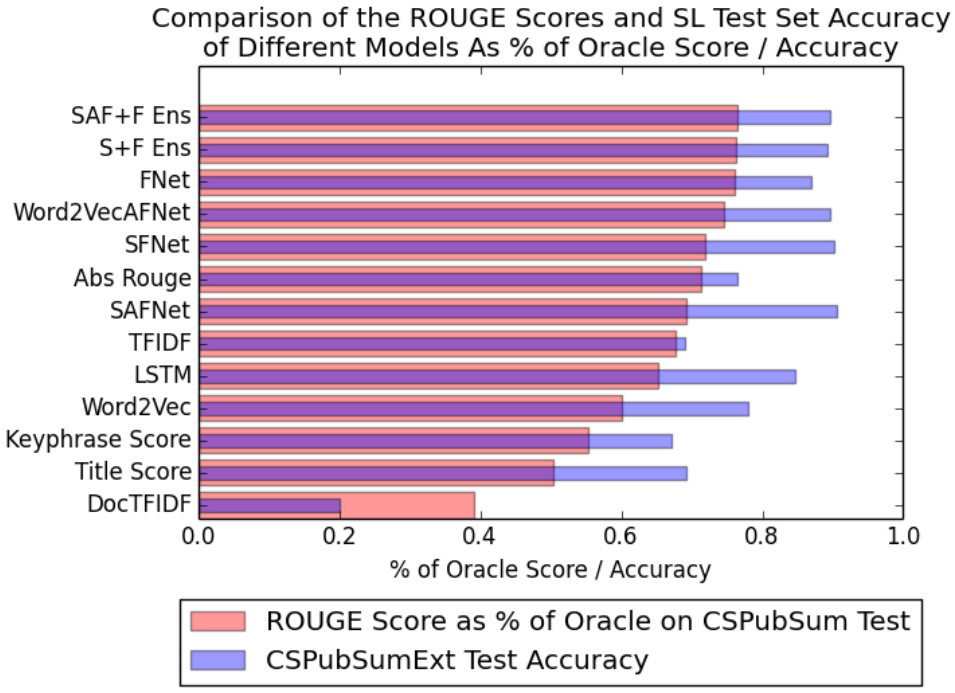

FIGURE 4.2: Comparison of the accuracy performance on the CSPubSumExt Test set and ROUGE-L score on CSPubSum Test set for each summariser. ROUGE Scores are given as a percentage of the Oracle Summariser score which is the highest score achievable for an extractive summariser on each of the papers. The wider bars in ascending order are the ROUGE scores. There is a statistically significant difference between the performance of the top four summarisers and the 5th highest scoring one (unpaired t-test, p=0.0139).

sentences, and assuming that we do not take summary sentences from the abstract which is already a summary, then Figure 4.1 suggests that there is no definitive section from which summary sentences should be extracted. It does however show empirically that the commonly used title - abstract - conclusion reading technique will gain a reader the most information about a paper most quickly.

4.2 Model Performance and Error Analysis

Figure 4.2 shows the performance of all models measured in terms of accuracy and ROUGE-L on CSPubSumExt Test and CSPubSum Test, respectively. Architectures which use a combination of sentence encoding and additional features performed best by both measures - an encouraging first result for use of deep learning in paper summarisation. The LSTM encoding (diagram (2), Figure 3.2) on its own outperforms models based on average word embeddings ((diagram (1), Figure 3.2) by 6.7% accuracy and 2.1 ROUGE points, showing that the ordering of words in a sentence

clearly makes a difference in deciding if that sentence is a summary sentence.

Good performance on the randomised and extended training and test data of CSPubSumExt correlates with good performance when actually producing a summary for CSPubSum Test. This can be seen on the graph, and numerically is a strong positive correlation (Pearson correlation, R=0.8738).

However an interesting result is that the best performance on the CSPubSumExt Test did not translate into the best performance on the CSPubSum Test despite the correlation. SAFNet achieved the highest accuracy on CSPubSumExt Test, however was worse than the AbstractROUGE summariser on CSPubSum Test. This is most likely due to imperfections in the training data. A small fraction of sentences in the training data are mislabelled due to bad examples in the highlights which are exacerbated by the HighlightROUGE data generating method. This leads to confusion for the more involved summariser architectures which are capable of learning complex enough representations to classify the mislabelled data correctly (i.e. to overfit the data). Although dropout is a very strong regularisation technique (Srivastava et al., 2014), additional regularisation such as L2 regularisation may be needed to prevent this in addition to better quality training data.

The most accurate model on CSPubSumExt Test was SAFNet. Out of a test set size of 131720, it made 11985 mistakes. Of those, there were more *false positives* (58%) than there were *false negatives* (42%), which means that the model was more likely to wrongfully predict a sentence to be a summary than it was to not recognise one as a summary. The two most highly misclassified sections were the Results and Method sections. 5.7% of all test sentences from the Results were misclassified, and 4.4% from the Method. For comparison, 2.74% of the Introduction was misclassified and 2.62% of the Conclusion. This is most likely due to the Results and Method sections being generally more complex than other sections and containing lots of observations about data or experiments that the summariser thinks are good summaries but actually are not relevant. Exactly the same trends were exhibited by SNet - more false positives than false negatives and most mistakes on the Results and Method sections.

100 sentences from CSPubSumExt Test which were incorrectly classified by SAFNet were manually examined. Out of those, 37 are mislabelled examples. The primary cause of false positives was lack of context (16 / 50 sentences) and long range dependency (10 / 50 sentences). Other important causes of false positives were a failure to recognise that mathematically intense sentences are not good summaries (7 / 50 sentences) and mislabeled data (12 / 50 sentences). Lack of context is when sentences require information from the sentences immediately before them to make sense. For

example, the sentence "The performance of such systems is commonly evaluated using the data in the matrix" is classified as positive but does not make sense out of context as it is not clear what systems the sentence is referring to. If the summariser had read the sentence before which describes what the "systems" are, it may have chosen that sentence as the summary instead.

A long-range dependency is when sentences refer to an entity that is described elsewhere in the paper, e.g. sentences referring to figures. These are more likely to be classified as summary statements when using models trained on automatically generated training data with HighlightROUGE, because they have a large overlap with the summary. HighlightROUGE fails to recognise the requirement of knowing what an entity is in order for the sentence to be a good summary, which is its main weakness.

The primary cause of false negatives was mislabelled data (25 / 50 sentences) and failure to recognise an entailment, observation or conclusion (20 / 50 sentences). There are some sentences in the highlights of the form "we set m=10 in this approach", which are not clear without context, but are labelled as positive because they are in the highlights. HighlightROUGE, seeing sentences like this in the highlights, will label more sentences like this as positive in the data which is why many of the false negative sentences are mislabelled. Such sentences should only be labeled as positive if they are part of multi-line summaries, which is difficult to determine automatically.

Failure to recognise an entailment, observation or conclusion is where a sentence has the form "entity X seems to have a very small effect on Y" for example, but the summariser has not learned that this information is useful for a summary, possibly because it was occluded by mislabelled data.

SAFNet and SFNet achieve high accuracy on the automatically generated CSPubSumExt Test dataset, though a lower ROUGE score than other simpler methods such as FNet on CSPubSum Test. This is likely due to overfitting, which our simpler summarisation models are less prone to.

One option to solve this would be to manually improve the CSPubSumExt labels. This would be a time consuming operation although we hypothesise that SAFNet would perform far better as a summariser if trained on such a purified training set. The other option is to change the form of the training data. Rather than using a randomised list of sentences and trying to learn objectively good summaries (Cao et al., 2015), each training example could be all the sentences in order from a paper, classified as either summary or not summary. The best summary sentences from within

the paper would then be chosen using HighlightROUGE and used as training data, and an approach similar to Nallapati, Zhai, and Zhou, 2016 could be used to read the whole paper sequentially and solve the issue of long-range dependency and context. The main challenge of this is whether the network would be able to encode a representation of the whole document as they are long.

The issue faced by SAFNet does not affect the Ensemble methods as their predictions are weighted by a hyperparameter tuned with the CSPubSum Test set rather than a learned projection matrix (as in SFNet). This ensures good performance on both test sets as the models are adapted to perform better on different examples.

In summary, the model performances show that: reading a sentence sequentially is superior to averaging its word vectors, simple features that model global context and positional information are very effective and achieving the highest accuracy on an automatically generated test set does not guarantee the highest ROUGE-L score on a gold test set, even though they are strongly correlated. This is most likely caused by complex models overfitting data that has a small but significant proportion of mislabelled examples as a bi-product of being generated automatically.

4.3 Summary Quality and Comparison to Other Work

Figure 4.2 shows ROUGE scores as percentages of an Oracle Summariser's score - the best possible score achievable. The metric used to measure performance, ROUGE-L, scores summaries between 0 and 1 - the higher the score the better. The Oracle Summariser's average ROUGE-L score on CSPubSum Test was 0.408 - which is slightly lower than the state of the art results achieved by Nallapati, Zhai, and Zhou, 2016 on the Daily Mail corpus. The best performing model, SAF+F Ensemble, achieved an average of 0.313 on the CSPubSum Test set, which is approximately in-line with some of the other methods which Nallapati, Zhai, and Zhou, 2016 compare their system to on the Daily Mail corpus. These are satisfying results and show that the system capable of summarising scientific papers developed here can achieve decent results, comparable to state of the art news summarisers, despite the length of the documents. In comparison to Cohan and Goharian, 2015, who also used ROUGE-L for their scientific publication summariser, our system performs to approximately the same level. The comparison is not perfect however since our system uses a completely different type of gold summary to Cohan and Goharian, 2015, who use a human written summary which had contextual dependencies between sentences. The data style of our corpus is more similar to the CNN / Daily Mail set which also uses bullet point highlights as the gold summary.

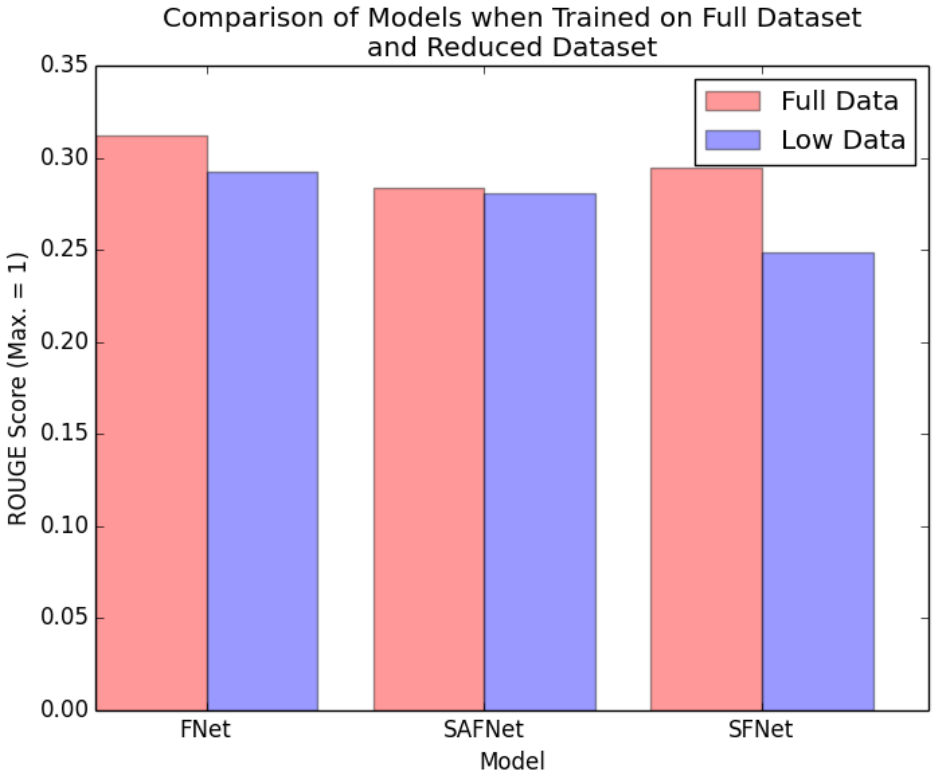

FIGURE 4.3: Comparison of the ROUGE scores of FNet, SAFNet and SFNet when trained on CSPubSum Train (bars on the right) and CSPubSumExt Train (bars on the left).

By manual inspection, the generated summaries do not tend to have good flow between sentences - each is quite disjoint. This is to be expected given that we trained the system to look for sentences which were good standalone summaries. Every summary will usually contain between 1 and 3 poor quality sentences which usually make no sense without context. The remaining sentences however are nearly always sufficient to transfer an understanding of what the paper is about, meaning that they could be effective supplements to the abstract. Real world field-testing would be needed to determine how useful humans actually find these summaries.

A suggested method to deal with the summaries being out of context is to highlight them in place in the main body of the text. That way they would be surrounded by their contextual sentences but would be clearly indicated to the reader as sentences which convey a large amount of information about the paper.

4.4 Effect of Using ROUGE-L to Generate More Data

One of the contributions of this work is a way to use evaluation metrics to generate extra training data. Figure 4.3 compares three models trained on CSPubSumExt

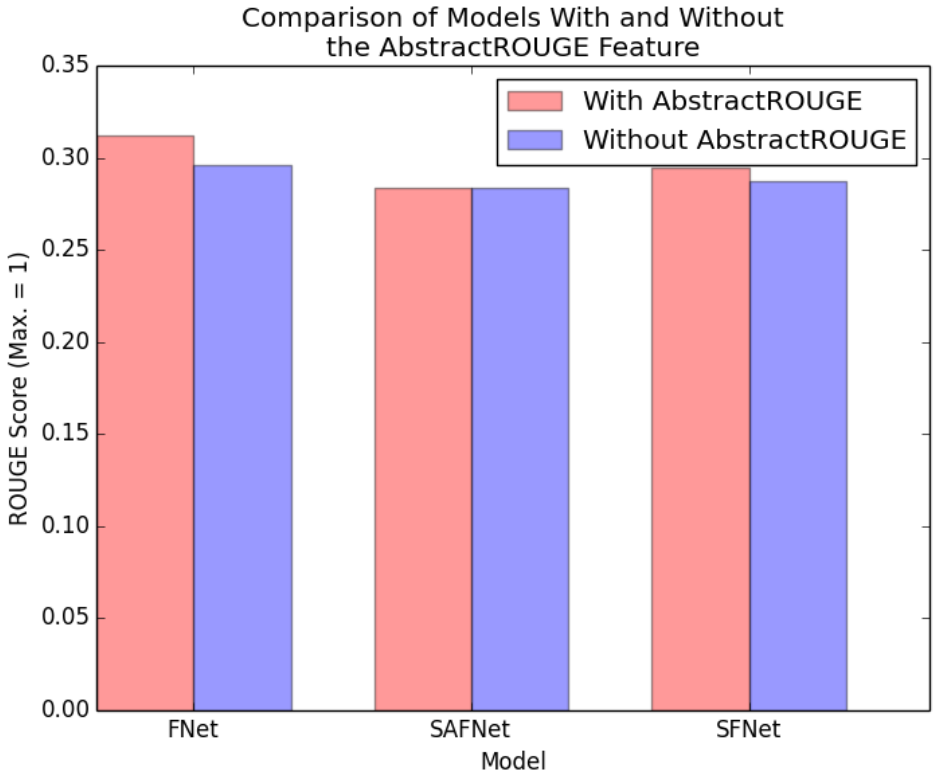

FIGURE 4.4: Comparison of ROUGE scores of the Features Only, SAFNet and SFNet models when trained with (bars on the left) and without (bars on the right) AbstractROUGE, evaluated on CSPubSum Test. The FNet classifier suffers a statistically significant (p=0.0279) decrease in performance without the AbstractROUGE metric.

Train (with generated data) and the same models trained on CSPubSum Train (without generated data; the feature of which section the example appeared in was removed to do this). The FNet summariser and SFNet suffer statistically significant ($p = 0.0147$ and $p < 0.0001$) drops in performance from using the unexpanded dataset, although interestingly SAFNet does not, suggesting it is a more stable model than the other two. These drops in performance however show that using the method we have described to increase the amount of available training data does improve model performance for summarisation. This is an important result because by extending the data as described, there are sufficient quantities of it for deep learning algorithms to be trained, meaning that they can be applied to summarising scientific documents which they have not been before.

4.5 Effect of AbstractROUGE on Summariser Performance

This work suggested use of the AbstractROUGE metric as a feature, following the idea of measuring similarity to the abstract by Saggion, Abura'ed, and Ronzano, 2016 and Kupiec, Pedersen, and Chen, 1995. Figure 4.4 compares the performance

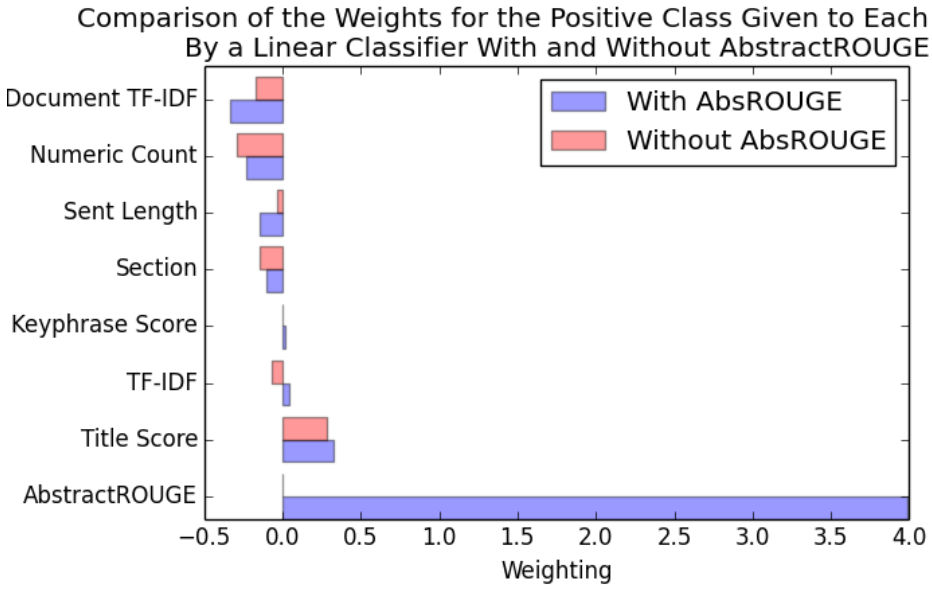

FIGURE 4.5: Comparison of the weights given to each of the features by a logistic regression classifier with and without the AbstractROUGE feature. A great deal of weight is given to the AbstractROUGE metric when it is present.

of 3 models trained with and without AbstractROUGE. This shows two things: the AbstractROUGE metric does improve performance for summarisation techniques based only on feature engineering; and learning a representation of the sentence directly from the raw text as is done in SAFNet and SFNet as well as learning from features results in a far more stable model, still able to make good predictions even if certain features are not available for training. This is another important result as learning from raw text has not been attempted in summarising scientific literature before, so the fact that we have shown that doing so makes models more stable means that it is a worthwhile technique to implement for summarising papers.

4.6 Feature Analysis

Figure 4.5 shows weightings for each of the features given by a logistic regression classifier which achieved a CSPubSumExt Test accuracy of 85.8% (the FNet classifer achieved 87%), and the same classifier when the AbstractROUGE metric was not present (CSPubSumExt Test accuracy of 82.8%). When AbstractROUGE is present, the classifier classifies almost exclusively based on it. When AbstractROUGE is not present the weights are far more equal in magnitude, however the final accuracy achieved is significantly lower. An interesting result is that for the classifier without AbstractROUGE, a negative weight is given to the TF-IDF score, meaning that a higher TF-IDF would make a sentence less likely to be classified as a summary. This is a strange occurrence as TF-IDF is often used on its own in information retrieval,

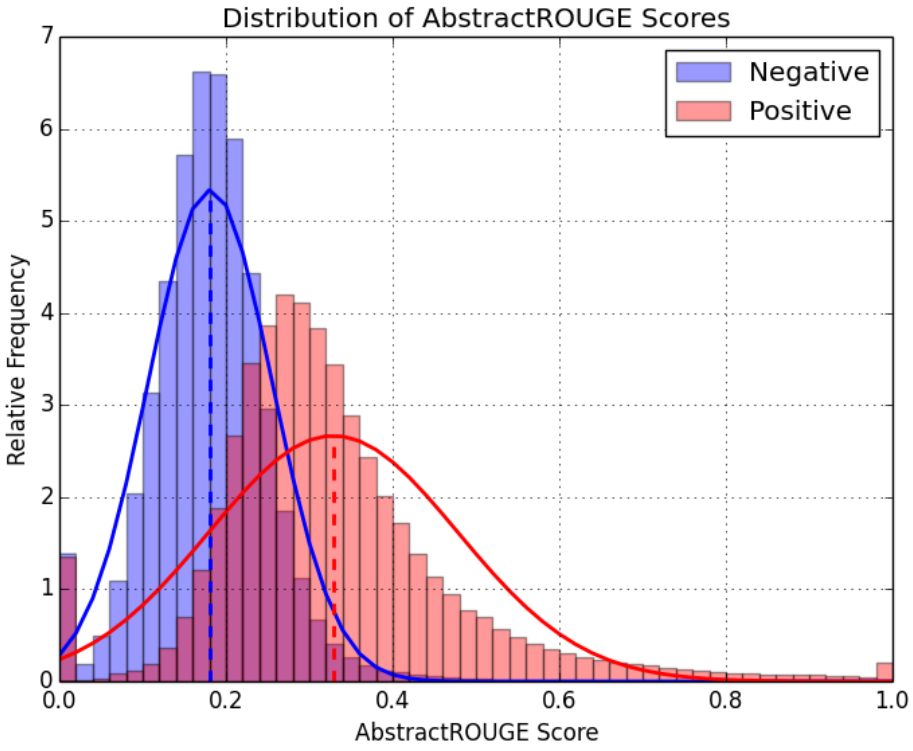

FIGURE 4.6: Distribution of the AbstractROUGE scores for positive vs. negative examples.

with a higher score indicating a more relevant document (Ramos, Eden, and Edu, 2003). It is most likely due to the presence of other features being stronger indicators of likelihood of being a summary. A strong positive weight is given to the title score as expected for example. Another interesting observation is that a fairly high magnitude negative weight is given to Document TF-IDF for both classifiers. This is likely because words which occur frequently in a sentence and not in the whole document are quite likely to be mathematical terms which do not make for good summaries.

If the distribution of AbstractROUGE scores across CSPubSumExt is analysed (Figure 4.6), we can see that positive sentences tend to have a higher AbstractROUGE score - which justifies the logistic regression classifer's setting of the AbstractROUGE weighting so high.

Of all of the combinations of any two features, there is one particular combination which is worth illustrating, shown in Figure 4.7. It plots two features against eachother - Sentence Length and Document TF-IDF, and shows how all positive sentences are clustered at the lower end of both of these features. This is interesting because it is the only feature comparison plot which shows any kind of distinct cluster and suggests that the combination of these two features could be instrumental

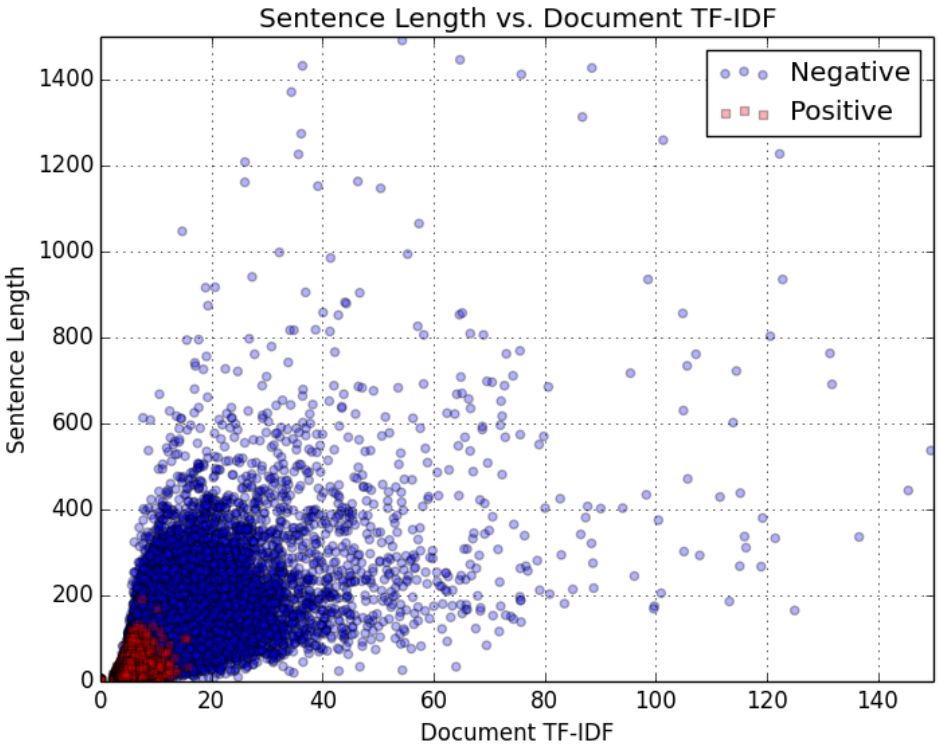

FIGURE 4.7: Scatter plot showing what the feature space looks like for Sentence Length vs. Document TF-IDF

in classifying sentences. This cluster is reflected by the feature weight plot in Figure 4.5 where negative weights are given to both Sentence Length and Document TF-IDF - meaning a lower score in these would be more likely to result in a positive classification.

Chapter 5

Conclusion and Evaluation

5.1 Summary of Approach and Achievements

The high-level goal of developing this system was to provide an automatic summarisation system for scientific papers that would reduce the time required to understand a paper. This goal was refined to be a way to producing summaries of scientific papers using a supervised learning-based extractive summarisation algorithm. Ideally, the learning method used would be based on the recently popularized approach of deep learning. The main obstacles that had to be overcome were the issue of how to make sure extracted sentences made sense out of context; the length of the scientific articles and thus an effective way of capturing as many of the paper's concepts as possible in a single summary; and the poverty of data which had to be alleviated in order to train data-hungry deep learning algorithms.

To defeat the obstacle of ensuring summaries made sense out of context, we deliberately trained the systems to build summaries from sentences which were good summaries even when taken out of context, following the approach of Cao et al., 2015. Doing so ensured that it did not matter if summary sentences were discordant because they would each individually be good summaries. We also included a set of contextual features to make sure the algorithm had at least some understanding of each sentence's context. We trained the system to recognise objectively good summaries by using author written highlight statements which are bullet points as gold summaries, since they are good summaries without any context.

To overcome the issue of paper length, we analysed which sections of scientific papers contained the most relevant information to summaries using two different methods: the ROUGE evaluation metric and Copy/Paste Score. We then used sentence location in the paper as a feature to indicate which section a sentence fell under, so that the system could learn to focus on specific sections which were most relevant to summaries and thus not have to consider every sentence in the paper equally.

To alleviate the poverty of data we developed a data generating method called HighlightROUGE, which worked by acting as an Oracle summariser for each paper in our base dataset of 10148 papers and labelling the top 10 sentences which best summarised the highlights as positive training examples. An equal number of negative examples were sampled from the sentences which were the worst summaries, which when combined with the highlights produced a dataset of around 400K examples which was sufficient to train deep learning algorithms.

The main contributions of this work therefore are:

- A new dataset of approximately 10K computer science publications and method for extending the data called *HighlightROUGE*, which we show empirically to improve summariser performance.
- A metric, *AbstractROUGE*, which we use as a feature that is a new way of comparing a sentence to a paper's abstract, as is done in much of the scientific literature, using the ROUGE evaluation metric. We show empirically that this feature improves feature-based summariser performance.
- The development of a multitude of benchmark summarisation systems on this dataset using neural as well as traditional feature-based methods. This is the first work to our knowledge that applies deep learning to the summarisation of scientific articles and learns from raw data, which we show yields the most stable performances and the highest performances overall when combined with feature engineering.
- An analysis of the degree to which different sections in scientific documents contribute to summarisation

5.2 Results Evaluation

Each goal set out in Section 1.3 was met. For the goal of analysing which sections in the paper were the most useful for summaries: in Section 4.1 it was shown how useful different sections within the paper were for summaries, with our initial hypothesis that the introduction would be high up this list being proved false in terms of its ROUGE score with the summary, but true in terms of the Copy/Paste score.

For the goal of extending the dataset to a sufficient size to train deep learning algorithms, the HighlightROUGE method was developed as detailed in Section 3.2.2, and proved empirically to improve the performance of deep learning algorithms as summarisers in Section 4.4. Furthermore, the format of the data which we used to train our SL-based algorithms was a randomised list, supposedly with positive examples as sentences which were good summaries even when taken out of context.

We successfully proved that good accuracy on this training set correlated strongly with summarisation performance.

For the goal of presenting a plethora of summarisation techniques as benchmarks for the dataset, 14 different summarisers were created including: feature-based summarisers, deep neural network-based summarisers, both feature and neural network based summarisers, ensemble summarisers and an oracle summariser. All of these models were rigorously tested and provide benchmarks for future work on this dataset with results and an error analysis being shown in Section 4.2.

For the goal of developing an algorithm which would allow readers to gain a general understanding of a paper through only reading the summary produced by the algorithm, it was shown in Section 4.3 that our system achieves comparable performances by the ROUGE-L measure with recent news summarisation techniques (Nallapati, Zhai, and Zhou, 2016), and through inspection confirmed that the majority of summaries do convey the general idea of a paper.

Finally, for the goal of rigorously testing a small set of features to use as benchmarks, we created 8 features including a new one, AbstractROUGE which is a variation on older techniques (Saggion, Abura'ed, and Ronzano, 2016), which are described in Section 3.4. These features were then analysed extensively and shown to produce good summarisation results in Section 4.2. The utility of each feature to summarisation algorithms was then further analysed in Section 4.6.

5.2.1 Main Weaknesses

The main weakness of the system is its inability to handle context. Despite following the approach of Cao et al., 2015 to find sentences with little contextual requirements, this was not always possible and the system does still sometimes classify sentences which need context to make sense as summary statements. These are in the minority however and summaries are still understandable.

The other main weakness is that as Hirao et al., 2017 say, there are hard limits to how good extractive summarisers can be because they can only choose sentences from the paper itself. This means that the system is barred from improving beyond a certain point. Hirao et al., 2017 believe that we are currently undergoing a paradigm shift from extractive to abstractive summarisation which may well be the case here in the future. Only by allowing machines to write their own text from scratch will true human quality summaries be created. We hope that this avenue will be explored in the future now that this dataset is of sufficient size to make training abstractive summarisers feasible.

5.2.2 Areas for Improvement

There were several areas where some improvements could have been made. Firstly, CSPubSum Test only has 150 documents, and testing on a greater number would have given a more reliable result (the Oracle summariser's average ROUGE score is 0.408 on CSPubSum Test but 0.421 when averaged across all 10000 papers). Doing this would require gathering more publications to use as test data. Secondly, metrics other than ROUGE-L could also have been used to test the data. ROUGE-L was used due to the structure of the data and because other work on summarising scientific papers has used it (Cohan and Goharian, 2015; Jaidka et al., 2016), see Section 3.2. However, recent research has suggested that there may be better metrics than ROUGE for evaluation (Cohan and Goharian, 2016). Finally, some other simple experiments could have been run including: using other evaluation metrics in AbstractROUGE and HighlightROUGE, experimenting with different formats for some features such as a boolean feature for which section the sentence was in rather than a location number, running the neural network with activation functions other than ReLU (although these would have required longer to train (Krizhevsky, Sutskever, and Hinton, 2012)), and testing more RNN architecture variants such as GRU cells (Cho et al., 2014).

5.3 Future Work

There are many exciting directions that future work could take. The main ones are detailed here.

5.3.1 Making Use of Citations

Many of the techniques used by authors to summarise scientific papers made use of citations and the text surrounding citations. This project did not make use of citations, choosing instead to focus on the application of deep learning. A future development is to incorporate citations as they are a unique feature of scientific papers that should be exploited.

5.3.2 Adding Attention

Many state of the art summarisation techniques make use of attention mechanisms (Nallapati, Zhai, and Zhou, 2016; Cheng and Lapata, 2016). Attention was not implemented in this project and would definitely be a good next avenue to explore, especially as it is shown to improve performance and helps with summarising long documents as shown by Cheng and Lapata, 2016.

5.3.3 Experimenting with Sentence Encoding

Word embeddings are not the only way a sentence can be encoded. An approach such as Yousefi-Azar and Hamey, 2017 could be followed and a deep autoencoder trained that could represent each sentence, or even each section or the whole paper. Alternatively and approach such as Cheng and Lapata, 2016 could be used where a CNN is used to encode each sentence rather than an RNN which was used in this work.

5.3.4 Encoding Larger Sections of the Documents

When using the abstract as input to our learning algorithms, we simply averaged the word vectors of each word in the abstract. There is massive room for improvement here: the abstract could be encoded using a neural network such as an RNN, CNN or autoencoder. Furthermore, other sections of the paper could be encoded such as the conclusion or results, or even the entire document, perhaps using an architecture such as Memory Networks (Weston, Chopra, and Bordes, 2015).

5.3.5 Format of the Training Data

This is perhaps the area with the most exciting possibilities for improvement: different formats of data could be used to train networks. For example, HighlightROUGE could be used to label sentences and training data given in the form of an ordered list of the sentences in a paper with their classification. An algorithm could then read each paper in order which would mean sentences would not have to be taken out of context and perhaps even multi-sentence summaries could be created. This approach is more similar to the state of the art in news summarisation (Nallapati, Zhai, and Zhou, 2016), but as scientific documents are longer may require an architecture with a better memory such as Memory Networks (Weston, Chopra, and Bordes, 2015).

Another interesting idea could be to label sentences with soft targets rather than coarse classifications as either summaries or not summaries, drawing on ideas proposed by Hinton, Vinyals, and Dean, 2015. The way this could work is to give sentences classifications as $1 - \text{ROUGE-L}(s, H)$ for sentence s and set of highlights H. The challenge would then be for the network to directly learn how good-a summary each sentence is by reading the whole paper, rather than learning a coarse classification. This would require training data to be in order so that the system could compare sentences and encoded sections from the same document together. As Hinton, Vinyals, and Dean, 2015 say, although this is a slightly different context, "soft targets" can transfer a huge amount of information about data that hard targets cannot possibly get across.

5.3.6 More Feature Engineering

Feature engineering is still an enormously popular approach to summarisation, and could be taken further for this project such as by adding new features similar to AbstractROUGE but using other evaluation metrics like ROUGE-N and ROUGE-W to create an ensemble of AbstractROUGEs. Other evaluation metrics such as proposed by (Cohan and Goharian, 2016) could also be used. Sentences could also be compared to other parts of the paper like the title using evaluation metrics, as the title has a particularly high salience with regard to summaries (see Section 4.1).

5.3.7 More Advanced Summary Construction After Neural Net Output

This project was more concerned with the actual network architectures used for summarisation than anything else, but many authors such as Cao et al., 2015; Ren, Wei, and Chen, 2016 mention the importance of ensuring no redundant sentences are added to the summary, something that was not implemented in this work. This post-processing phase could be taken further however and the sentences outputted by a network could be altered so that they flow, do not refer to outside entities such as citations and make a coherent summary that is more like a piece of human written text than a set of bullet points which are largely discordant.

5.4 Project and Personal Evaluation

Project's Most Challenging Aspect was preprocessing data into the correct form and building the correct architectures which allowed neural methods to shine. It took a very long time to build a system capable of outperforming a purely TF-IDF based summariser.

Advice for Anyone Who Wants to Improve the System Make sure that you have sorted out the format for the data that you are going to use as soon as possible, then create the dataset and make sure all models use the same dataset. You should try as hard as possible to avoid having to re-run any data preprocessing phases once you have begun developing models as this will make work much faster.

What Would Be Done Differently if the Project Were to be Restarted The data format would be defined as early as possible and the datasets created as early as possible in the hope that preprocessing of the 10148 papers would only ever have to take place once.

Did I Meet My Personal Development Aims? When I started this project I had never used TensorFlow and had never coded a machine learning algorithm. I had

some basic experience with creating a bag-of-words based summariser for news articles which was my only experience in NLP. I have far exceeded my aims in this project and now know how to use TensorFlow, have a good understanding of statistical NLP and can code deep learning algorithms including RNNs and convolutional nets, but even better than this I have an intuitive understanding of how they work. This culminated in creating a much-better-than-TF-IDF summariser which was an aim I was determined to achieve.

Appendix A

Results Tables

Section	ROUGE-L Score	Normalized Copy/Paste Score
Title	0.336	0.04
Abstract	0.301	0.471
Conclusion	0.245	0.096
Keyphrases	0.224	0
Analysis	0.211	0
Summary	0.210	0.004
Related Work	0.208	0.038
Discussion	0.206	0.088
Results	0.205	0
Evaluation	0.201	0.003
Overview	0.198	0.002
Methods	0.193	0
Introduction	0.189	0.239
Motivation	0.176	0.007
References	0.115	0.002

TABLE A.1: The ROUGE-L and Copy/Paste Scores of each section as plotted in Figure 4.1

Model	CSPubSum Test (% Oracle Score)	CSPubSumExt Test
SAF+F Ens	0.764	0.895
S+F Ens	0.763	0.892
FNet	0.761	0.870
Word2VecAFNet	0.746	0.897
SFNet	0.719	0.901
AbstractROUGE	0.713	0.764
SAFNet	0.692	0.906
TF-IDF	0.677	0.69
LSTM	0.652	0.847
Word2VecNet	0.601	0.779
Keyphrase Score	0.554	0.67
Title Score	0.503	0.692
DocTF-IDF	0.390	0.2

TABLE A.2: The performances of the different models as actual summarisers on CSPubSum Test and in terms of Accuracy on CSPubSumExt Test, as shown in Figure 4.2

Model	CSPubSum Test, Full Data	CSPubSum Test, Low Data
FNet	0.312	0.292
SAFNet	0.284	0.281
SFNet	0.294	0.248

TABLE A.3: The performances of three different models when trained with the extended and unextended data, as shown in Figure 4.3

Model	CSPubSum Test, AbsROUGE	CSPubSum Test, No AbsROUGE
FNet	0.312	0.297
SAFNet	0.284	0.284
SFNet	0.294	0.287

TABLE A.4: The performances of three different models when trained with and without the AbstractROUGE metric, as shown in Figure 4.4

Feature	**Weights With AbsROUGE**	**Weights Without AbsROUGE**
AbstractROUGE	3.99	0
Title Score	0.323	0.286
TF-IDF	0.043	-0.065
Keyphrase Score	0.014	0.001
Section	-0.102	-0.148
Sent Length	-0.141	-0.032
Numeric Count	-0.232	-0.290
Document TF-IDF	-0.33	-0.172

TABLE A.5: Comparison of the weights assigned to each feature by a logistic regression classifier, with and without AbstractROUGE, as shown in Figure 4.5

Appendix B

Project Plan

As specified in report requirements, the Project Plan written in November 2016 is included here.

Automatic Summarisation of Scientific Papers

Project Plan: 16 November 2016 - Name: Edward Collins - Supervisor: Isabelle Augenstein

Aims

- To learn about natural language processing techniques used for the automatic summarisation of text including both extractive and generative models.
- To develop an effective method of applying summarisation techniques to scientific papers and build a system in Python that can perform this task.
- To publish this research at a conference if successful results are obtained.

Objectives

1. Review existing literature related to the automatic summarisation of one or more documents and literature related to sentence compression for natural language generation.
2. Develop pre-processing software that can understand which parts of a scientific paper are most relevant for summarisation or contain the most meaningful information, so that the data used to generate summaries is in effect distilled to the most meaningful pieces of text in the paper.
3. Develop a model that can use the preprocessed data to generate short summaries of approximately 5 sentences for a given scientific paper.
4. Evaluate the success of the summarisation system using software such as ROUGE.
5. If successful, writeup findings into a short paper.

Deliverables

- A novel summarisation algorithm that produces good quality summaries of scientific articles.
- A fully documented and functional piece of preprocessing software which can decide on the areas with highest density of relevant information in the papers.
- A fully documented and functional piece of software, written in Python which can generate summaries of scientific papers.
- A fully described method of testing the software.
- A document detailing the potential applications of such a piece of software.
- A short paper detailing the findings suitable for publication.

Non-Essential Deliverables

- *If time*, a front-end for the summarisation system.
- *If time*, deploy the system to be accessed through a web browser.

Work Plan

- Project start - end of November 2016 (8 weeks): Creating small prototype and test programs to test the feasibility of goals. Principally to test whether the training data we have is readily suitable for learning the areas of most interest in papers or if we must find a different way to do this. Other research will be conducted into the training data in this period to look for most commonly used words, common structures and frequently used sentences in the training summary data.
- December 2016 - mid-February 2017 (approx. 16 weeks): Iterations consisting of researching, building and testing a feature of the summariser and preprocessing system. Each iteration will last two weeks. Principle features to develop and tasks to complete:
  - Evaluation method to test the accuracy and validity of generated summaries against the training data.
  - Preprocessor to extract features from the scientific papers.

  - Preprocessor to translate these features into a form suitable for machine learning.
  - Research into summarisation methods before creating a different version of the summariser.
  - Summariser engine to take data and create a summary, tested using the evaluation method earlier developed (this will take up many iterations as different versions of the summariser are created and tested for their effectiveness).
- Mid-February - End of March 2017 (approx. 6 weeks) - work on final report and paper to be published. Organise all work and make it neat and easily understandable. Add aesthetic features like a front-end to the summariser if there is time.

Milestones

- Preprocessor to highlight areas of highly relevant information complete to reasonable standard - *16 December 2016*
- Evaluation method for accuracy and validity of summaries developed - *Mid-January 2017*
- Interim report prepared - *End of January 2017*
- First summariser version working to a reasonable degree - *End of January 2017*
- Working summariser developed - *Mid-End of February 2017*
- Paper and final report written - *End of March 2017*

Appendix C

Interim Report

As specified in report requirements, the Interim Report written in January 2017 is included here.

A Supervised Approach to Extractive Summarisation of Scientific Papers

Interim Report: 25 January 2017 - Name: Edward Collins - Supervisor: Isabelle Augenstein

Previous Title: *Automatic Summarisation of Scientific Papers*

Scope Redefinement

Originally the scope of this project was fairly broad, encompassing any possible technique for automatic summarisation. Since the project report, this scope has been narrowed to be based on **extractive** summarisation only, or finding suitable sentences for a summary from the paper itself, as opposed to **abstractive** summarisation which would generate new summaries.

The main reason for doing this is that extractive summaries are more useful in the context of scientific papers. This is because summaries generated extractively are directly quotable, where as summaries generated abstractively are not and would still require the reader to search through the paper for the relevant quote.

Progress to Date

Exploring the Data

One of the objectives described in the project plan was to develop pre-processing software that could understand which parts of a scientific paper were most relevant for summarisation and contain the most useful data, so that summary sentences may be chosen from there.

To achieve this, a Python program was written that compared sentences to one another, and outputted a 1 if the sentences were the same. The comparison worked by comparing the length of the two sentences and seeing if they were within a tolerance of one another, and also comparing the non-stopword words to see if the sentences conveyed the same information. This approach is more robust than directly comparing sentences on all words to see if they match because it allows for sentences that have been slightly modified or shortened by one or two words changed to still match.

Using this program, the highlight statements which were provided by the author for each paper were compared to every sentence in the paper, which allowed discovery of whether the highlight statements had been copied and pasted from the main paper, and if so, from which section of the paper. Doing this would show which area of the paper authors most often copied highlight sentences from, and so show which section summary statements could be taken from. Results are shown in **Figure 1**.

As would seem logical, when sentences were copied and pasted they were mostly taken from the abstract, which is a summary of the paper in itself. However taking summary sentences from the abstract would be pointless for this system because the abstract is already a summary. What this system needs to produce is sentences which supplement the abstract to help increase the user's understanding of the rest of the paper. Therefore, the sentences of the abstract can be treated as summary statements as well, and the same technique can be applied to see whether authors copy and paste sentences from the main paper into the abstract. Results are shown in **Figure 2**.

Again as would seem logical, when sentences were copied directly from the main paper text into the abstract they usually came from the introduction, but also had a fair chance of coming from the results and discussion or conclusion sections. Summaries tend to simplify the concepts of a research paper, so by drawing abstract sentences from the introduction, which would attempt to explain the basic concepts, and from the results and conclusions, which would present the findings of the paper, a logical structure to create summaries can be found.

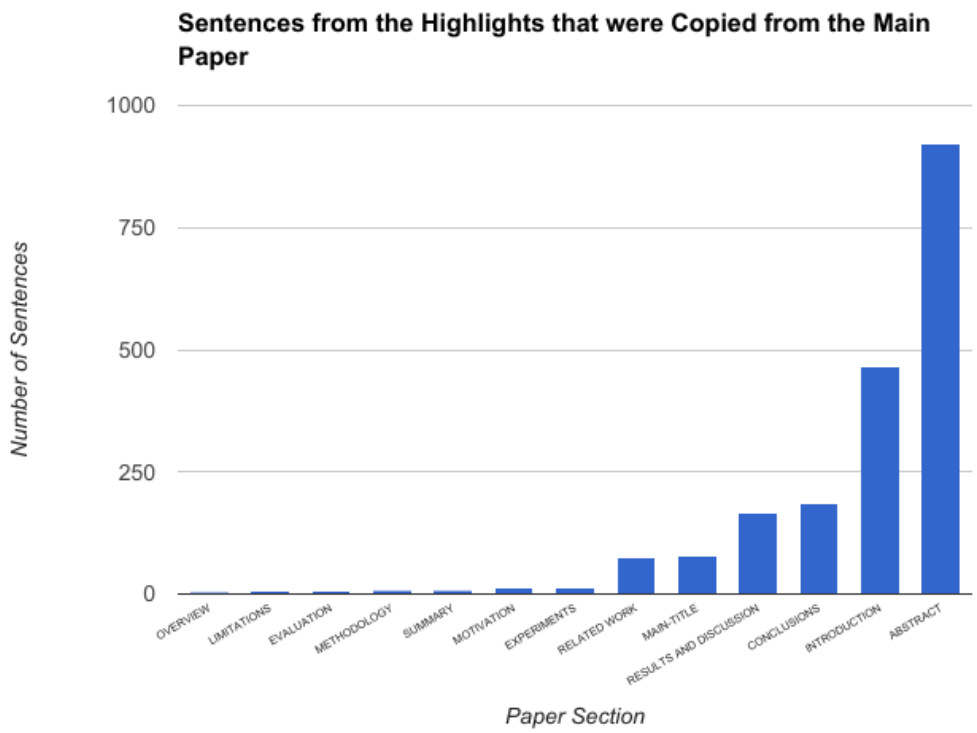

Figure 1 *-From the ~10000 papers, 1954 of the highlight statements were copied and pasted from the main paper. Of those 1954, 47% of them came from the Abstract, 24% from the Introduction and approximately 20% from the Results and Discussion and Conclusions combined.*

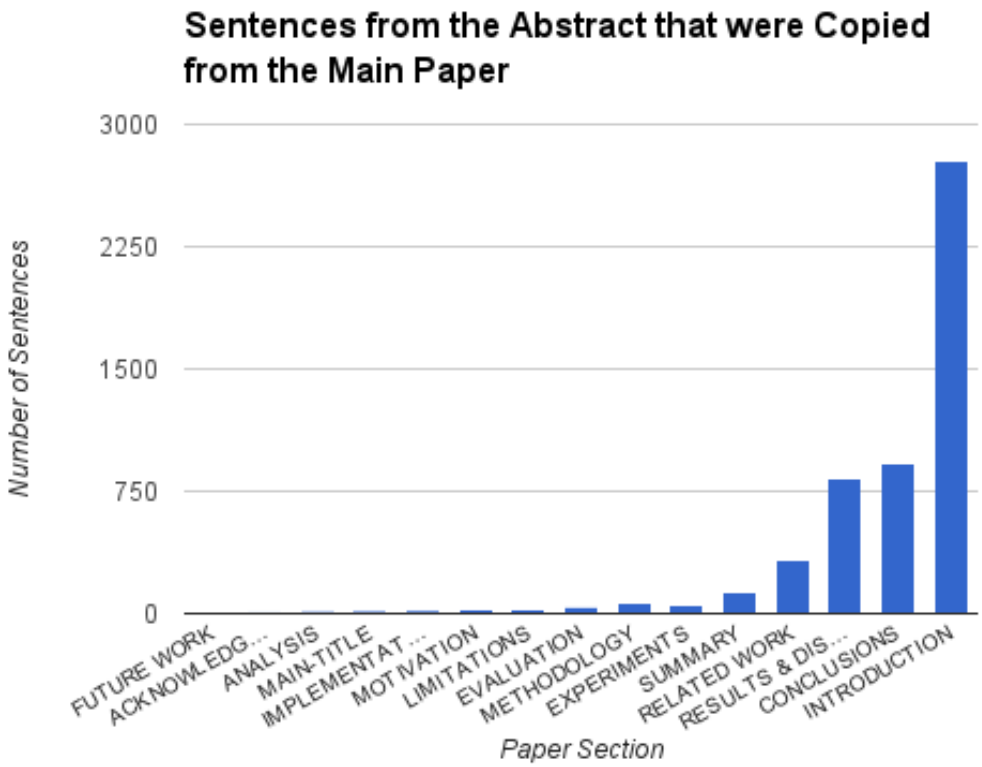

Figure 2 *-From the ~10000 papers, 5259 sentences were copied and pasted from the main paper into the abstract. 53% of these came from the introduction, 17% from the conclusion and 15% from results and discussion.*

A Supervised Learning Problem

Once it was confirmed that sentences from the main paper are copied and pasted into the highlights and abstract, a dataset that could be used to train machine learning algorithms could be generated. To do this, all of the papers were split into their constituent sentences, and each sentence was given a label of either 1 or 0 for "summary sentence" and "not summary sentence". The problem is now a classification task to classify the sentences of the paper into summary and non-summary sentences.

Each sentence had the following features associated with it:

- **TF-IDF score**: a measure of how much information the sentence carried
- **Key Phrase score**: a count of how many author defined key phrases the sentence contains
- **Title score**: a count of how many non-stopword words from the title each sentence contains
- **Bag of Words score**: a score calculated by counting the occurrence of all non-stopword words in the paper the sentence was in and creating a score for the sentence by summing the total occurrences of each word in the sentence in the whole paper
- **Sentence length**: the number of words in the sentence
- **Sentence position**: the section of the paper that the sentence is in

A Word2Vec model was then trained on all of the paper data, which meant that every word could be represented by a vector of 100 numbers. Each sentence was then transformed into a vector by averaging all of the word vectors in the sentence. Each of the features listed above was then appended to this feature vector.

Each sentence now had a representation as a vector of numbers, and a classification. By using the Python machine learning library SciKit Learn, a logistic regression model was trained on the data which outputs the probability of a sentence being either a summary or non-summary sentence. The precision-recall curve is shown below in **Figure 3**.

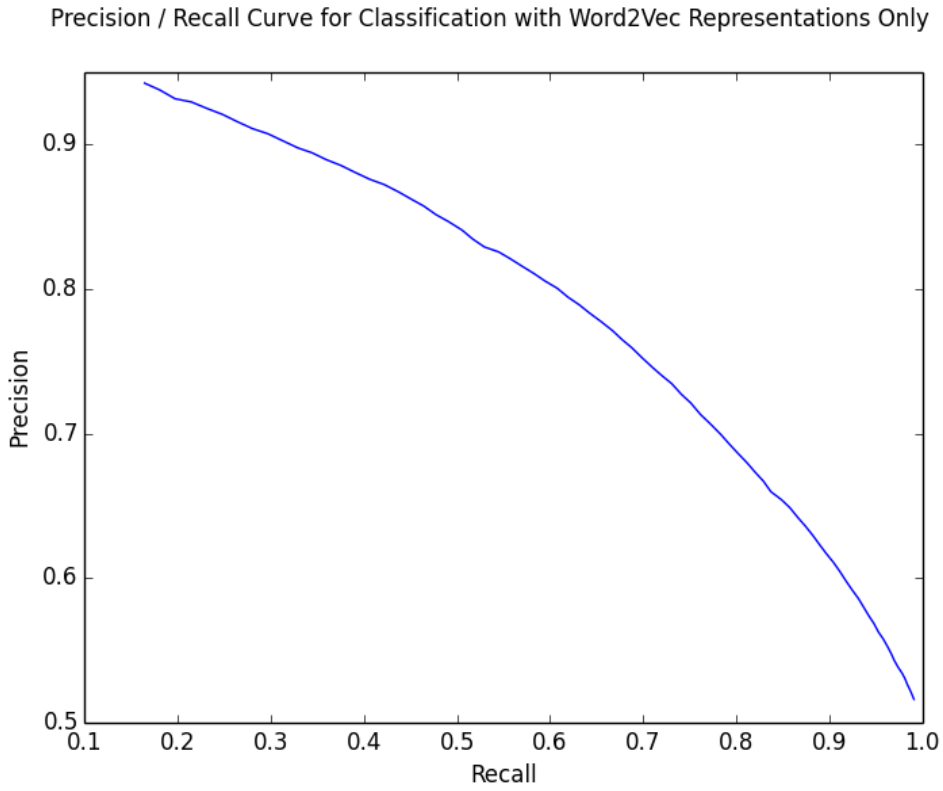

***Figure 3** - The maximum F1 score achieved was 71% when a probability threshold of 0.38 was set as the required threshold to classify a sentence as a summary sentence.*

Remaining Work

To complete the project, the classification algorithm must be improved. To do this, a deep learning technique called a Long-Short Term Memory (LSTM) recurrent neural network will be used instead of averaging the word vectors of each sentence to classify them. In addition other features for each sentence may be tested. Each supervised learning algorithm will also be used to generate summaries after training. These summaries will then be evaluated using the ROUGE metric and by human testers. Once an algorithm with a suitable ROUGE score and accuracy on the testing data is found or time has run out (whichever comes first), the system's code must be cleaned and prepared for submission. The final report will also be written at this stage.

Milestones

- Logistic regression classifier fully tested and used to generate summaries - 8th February
- LSTM created and tested - 15th February
- Summariser using LSTM working - 22nd February
- Code prepared for submission - Mid-March
- Report written - Mid-April

Appendix D

Code Listing

All code can be viewed at this repository along with guides of how the code works and how to run it.

Bibliography

Abu-jbara, Amjad and Ann Arbor (2011). "Coherent Citation-Based Summarization of Scientific Papers". In: *Computational Linguistics*, pp. 500–509. URL: http://www.aclweb.org/anthology/P/P11/P11-1051.pdf.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural Machine Translation By Jointly Learning To Align and Translate". In: *Iclr 2015*, pp. 1–15. ISSN: 0147-006X. DOI: 10.1146/annurev.neuro.26.041002.131047. arXiv: 1409.0473. URL: http://arxiv.org/abs/1409.0473v3.

Baxendale, P. B. (1958). "Machine-made index for technical literature - an experiment". In: *IBM Journal of Research and Development* 2, pp. 354–365.

Bengio, Yoshua et al. (2003). "A Neural Probabilistic Language Model". In: *The Journal of Machine Learning Research* 3, pp. 1137–1155. ISSN: 15324435. DOI: 10.1162/153244303322533223. arXiv: arXiv:1301.3781v3.

Cao, Ziqiang et al. (2015). "Learning Summary Prior Representation for Extractive Summarization". In: *Proceedings ACL 2015*, pp. 829–833.

Chen, Berlin (2015). *Recent Developments in Automatic Summarisation*. Taiwan. URL: http://berlin.csie.ntnu.edu.tw/Berlin{_}Research/Talks/20150626-RecentDevelopmentsinAutomaticSummarization.pdf.

Chen, Danqi, Jason Bolton, and Christopher D Manning (2016). "A Thorough Examination of the CNN / Daily Mail Reading Comprehension Task". In: *Acl 2016*, pp. 2358–2367. arXiv: 1606.02858.

Chen, Qian et al. (2016). "Distraction-Based Neural Networks for Document Summarization". In: *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI-2016)*. arXiv: 1610.08462. URL: http://arxiv.org/abs/1610.08462.

Cheng, Jianpeng and Mirella Lapata (2016). "Neural Summarization by Extracting Sentences and Words". In: *Arxiv*, pp. 484–494. arXiv: 1603.07252. URL: http://arxiv.org/abs/1603.07252.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. ISSN: 09205691. DOI: 10.3115/v1/D14-1179. arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

Cohan, Arman and Nazli Goharian (2015). "Scientific Article Summarization Using Citation-Context and Article's Discourse Structure". In: *Conference on Empirical Methods in Natural Language Processing* September, pp. 390–400.

— (2016). "Revisiting Summarization Evaluation for Scientific Articles". In: pp. 806–813. arXiv: 1604.00400. URL: http://arxiv.org/abs/1604.00400.

Conroy, John M. and Dianne P. O'leary (2001). "Text summarization via hidden Markov models". In: *SIGIR '01 Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval* January 2001, pp. 406–407. DOI: 10.1145/383952.384042. URL: http://portal.acm.org/citation.cfm?doid=383952.384042.

Deng, Jia et al. (2009). "ImageNet : A Large-Scale Hierarchical Image Database". In: pp. 2–9. ISSN: 1063-6919. DOI: 10.1109/CVPR.2009.5206848.

Denil, Misha, Alban Demiraj, and Nando De Freitas (2015). "Extraction of Salient Sentences from Labelled Documents". In: *arXiv*, pp. 1–9. arXiv: 1412.6815v2.

Dlikman, Alexander and Mark Last (2016). "Using machine learning methods and linguistic features in single-document extractive summarization". In: *CEUR Workshop Proceedings* 1646, pp. 1–8. ISSN: 16130073.

Edmundson, H P (1969). "New Methods in Automatic Extracting". In: *Journal of the ACM* 16.2, pp. 264–285. ISSN: 00045411. DOI: 10.1145/321510.321519. URL: http://portal.acm.org/citation.cfm?doid=321510.321519.

Elkiss, Aaron et al. (2008). "Blind Men and Elephants: What Do Citation Summaries Tell Us About a Research Article?" In: ISSN: 14923831.

Fang, Changjian et al. (2017). "Word-sentence co-ranking for automatic extractive text summarization". In: *Expert Systems with Applications* 72, pp. 189–195. ISSN: 09574174. DOI: 10.1016/j.eswa.2016.12.021. URL: http://linkinghub.elsevier.com/retrieve/pii/S0957417416306959.

Gers, F.A. and J. Schmidhuber (2000). "Recurrent nets that time and count". In: *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium* 1, 189–194 vol.3. ISSN: 1098-6596. DOI: 10.1109/IJCNN.2000.861302. arXiv: arXiv:1011.1669v3. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=861302.

Gregor, K et al. (2015). "DRAW: A Recurrent Neural Network For Image Generation". In: *arXiv preprint arXiv: ...* P. 11. arXiv: 1502.04623. URL: http://arxiv.org/abs/1502.04623.

He, Kaiming et al. (2015). "Deep Residual Learning for Image Recognition". In: *Arxiv.Org* 7.3, pp. 171–180. ISSN: 1664-1078. DOI: 10.3389/fpsyg.2013.00124. arXiv: 1512.03385. URL: http://arxiv.org/pdf/1512.03385v1.pdf.

Hermann, Karm Moritz et al. (2015). "Teaching Machines to Read and Comprehend". In: *arXiv*, pp. 1–13. ISSN: 10495258. arXiv: arXiv:1506.03340v1.

Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean (2015). "Distilling the Knowledge in a Neural Network". In: *NIPS 2014 Deep Learning Workshop*, pp. 1–9. ISSN: 0022-2488. DOI: 10.1063/1.4931082. arXiv: 1503.02531. URL: http://arxiv.org/abs/1503.02531.

Hinton, Geoffrey E and Ruslan Salakhutdinov (2006). "Reducing Dimensionality of Data with Neural Networks". In: *Science* 313.July, pp. 504–507. ISSN: 0036-8075. DOI: 10.1126/science.1127647. arXiv: 20.

Hirao, Tsutomu et al. (2015). "Summarizing a Document by Trimming the Discourse Tree". In: *IEEE/ACM Transactions on Speech and Language Processing* 23.11, pp. 2081–2092. ISSN: 23299290. DOI: 10.1109/TASLP.2015.2465150.

Hirao, Tsutomu et al. (2017). "Enumeration of Extractive Oracle Summaries". In: *Arxiv*. arXiv: 1701.01614. URL: http://arxiv.org/abs/1701.01614.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural computation* 9.8, pp. 1735–80. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. arXiv: 1206.2944. URL: http://www.ncbi.nlm.nih.gov/pubmed/9377276.

Jaidka, Kokil et al. (2016). "Overview of the CL-SciSumm 2016 Shared Task". In: *CEUR Workshop Proceedings* 1610, pp. 93–102. ISSN: 16130073.

Jozefowicz, Rafal et al. (2016). "Exploring the Limits of Language Modeling". In: *arXiv:1602.02410 [cs]*. arXiv: 1602.02410. URL: http://arxiv.org/abs/1602.02410{\%}5Cnhttp://www.arxiv.org/pdf/1602.02410.pdf.

Kageback, Mikael et al. (2014). "Extractive Summarization using Continuous Vector Space Models". In: *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pp. 31–39. ISSN: 18650929. DOI: 10.1007/978-3-642-14834-7_15. arXiv: arXiv:1506.01597v2. URL: http://www.aclweb.org/anthology/W14-1504.

Kalchbrenner, Nal, Edward Grefenstette, and Phil Blunsom (2014). "A Convolutional Neural Network for Modelling Sentences". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pp. 655–665. arXiv: arXiv:1404.2188v1. URL: http://goo.gl/EsQCuC.

Kavila, Selvani Deepthi and Y Radhika (2015). "Extractive Text Summarization Using Modified Weighing and Sentence Symmetric Feature Methods". In: October, pp. 33–39. ISSN: 20750161. DOI: 10.5815/ijmecs.2015.10.05.

Kim, Yoon (2014). "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pp. 1746–1751. ISSN: 10709908. DOI: 10.1109/LSP.2014.2325781. arXiv: arXiv:1408.5882v1. URL: http://emnlp2014.org/papers/pdf/EMNLP2014181.pdf.

Kobayashi, Hayato, Masaki Yatsuka, and Noguchi Taichi (2015). "Summarization Based on Embedding Distributions". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* September, pp. 1984–1989.

Kong, Lingpeng et al. (2017). "DRAGNN: A Transition-based Framework for Dynamically Connected Neural Networks". In: *Arxiv*. arXiv: 1703.04474. URL: http://arxiv.org/abs/1703.04474.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances In Neural Information Processing Systems*, pp. 1–9. ISSN: 10495258. DOI: http://dx.doi.org/10.1016/j.protcy.2014.09.007. arXiv: 1102.0183.

Kupiec, Julian, Jan Pedersen, and Francine Chen (1995). "A Trainable Document Summarizer". In: *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 68–73. ISSN: 01635840. DOI: 10.1145/215206.215333. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.1161.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444. ISSN: 0028-0836. DOI: 10.1038/nature14539. arXiv: arXiv:1312.6184v5. URL: http://dx.doi.org/10.1038/nature14539.

Liakata, Maria et al. (2010). "Corpora for the conceptualisation and zoning of scientific papers". In: *Proceedings of LREC*, pp. 2054–2061. URL: http://www.abdn.ac.uk/{~}csc323/lrecAZCoreSCfinal.pdf.

Lin, C Y (2004). "Rouge: A package for automatic evaluation of summaries". In: *Proceedings of the workshop on text summarization branches out (WAS 2004)* 1, pp. 25–26. ISSN: 00036951.

Lin, Chin-Yew and Eduard Hovy (1997). "Identifying Topics by Position". In:

Litvak, Marina, Mark Last, and Menahem Friedman (2010). "A new Approach to Improving Multilingual Summarization using a Genetic Algorithm". In: *48th Annual Meeting of the Association for Computational Linguistics* July, pp. 927–936.

Litvak, Marina et al. (2016). "MUSEEC: A Multilingual Text Summarization Tool". In: *Proceedings of ACL-2016 System Demonstrations*, pp. 73–78. URL: http://anthology.aclweb.org/P/P16/P16-4013.

Luhn, H. P. (1958). "The Automatic Creation of Literature Abstracts". In: *IBM Journal of Research and Development* 2.2, pp. 159–165. ISSN: 0018-8646. DOI: 10.1147/rd.22.0159.

Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning (2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *Emnlp* September, p. 11. ISSN: 10495258. DOI: 10.18653/v1/D15-1166. arXiv: 1508.04025. URL: http://arxiv.org/abs/1508.04025.

Luong, Minh-Thang et al. (2016). "Multi-task Sequence to Sequence Learning". In: *Iclr* c, pp. 1–9. arXiv: 1511.06114. URL: http://arxiv.org/abs/1511.06114.

Mani, Inderjeet (2001). *Automatic summarization*. Vol. 3. John Benjamins Publishing.

Mihalcea, Rada and Hakan Ceylan (2007). "Explorations in Automatic Book Summarization". In: *Computational Linguistics* June, pp. 380–389. URL: http://www.aclweb.org/anthology/D/D07/D07-1040http://www.aclweb.org/anthology/D07-1040.

Mihalcea, Rada and Paul Tarau (2004). "TextRank: Bringing order into texts". In: *Proceedings of EMNLP* 85, pp. 404–411. ISSN: 0256307X. DOI: 10.3115/1219044.1219064. arXiv: arXiv:1011.1669v3. URL: http://acl.ldc.upenn.edu/acl2004/emnlp/pdf/Mihalcea.pdf.

Mikolov, Tomas et al. (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Nips*, pp. 1–9. ISSN: 10495258. DOI: 10.1162/jmlr.2003.3.4-5.951. arXiv: 1310.4546.

Nallapati, Ramesh, Feifei Zhai, and Bowen Zhou (2016). "SummaRuNNer: A Recurrent Neural Network based Sequence Model for Extractive Summarization of Documents". In: *Arxiv*. arXiv: 1611.04230. URL: https://arxiv.org/pdf/1611.04230.pdf.

Nallapati, Ramesh, Bowen Zhou, and Mingbo Ma (2016). "Classify or Select: Neural Architectures for Extractive Document Summarization". In: *Arxiv*. arXiv: 1611.04244. URL: http://arxiv.org/abs/1611.04244.

Nallapati, Ramesh et al. (2016). "Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond". In: *Proceedings of CoNLL*, pp. 280–290. arXiv: 1602.06023. URL: http://arxiv.org/abs/1602.06023.

Nomoto, Tadashi and Yuji Matsumoto (2001). "A new approach to unsupervised text summarization". In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '01*, pp. 26–34. ISSN: 01635840. DOI: 10.1145/383952.383956. URL: http://dl.acm.org/citation.cfm?id=383952.383956.

Oord, Aaron van den, Nal Kalchbrenner, and Koray Kavukcuoglu (2016). "Pixel Recurrent Neural Networks". In: *International Conference on Machine Learning (ICML)*. arXiv: 1601.06759. URL: http://arxiv.org/abs/1601.06759.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543. ISSN: 10495258. DOI: 10.3115/v1/D14-1162. arXiv: 1504.06654.

Qazvinian, Vahed et al. (2013). "Generating extractive summaries of scientific paradigms". In: *Journal of Artificial Intelligence Research* 46, pp. 165–201. ISSN: 10769757. DOI: 10.1613/jair.3732. arXiv: 1402.0556.

Ramos, Juan, Juramos Eden, and Rutgers Edu (2003). "Using TF-IDF to Determine Word Relevance in Document Queries". In: *Processing*. DOI: 10.1.1.121.1424. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.121.1424{\&}rep=rep1{\&}type=pdf.

Ren, Pengjie, Furu Wei, and Zhumin Chen (2016). "A Redundancy-Aware Sentence Regression Framework for Extractive Summarization". In: pp. 33–43. URL: https://www.aclweb.org/anthology/C/C16/C16-1004.pdf.

Ronzano, Francesco and Horacio Saggion (2016). "Knowledge Extraction and Modeling from Scientific Publications". In:

Saggion, Horacio, Ahmed Abura'ed, and Francesco Ronzano (2016). "Trainable citation-enhanced summarization of scientific articles". In: *CEUR Workshop Proceedings* 1610, pp. 175–186. ISSN: 16130073.

Salton, G., a. Wong, and C. S. Yang (1975). "A vector space model for automatic indexing". In: *Communications of the ACM* 18.11, pp. 613–620. ISSN: 00010782. DOI: 10.1145/361219.361220.

Salton, Gerard et al. (1996). "Automatic analysis, theme generation, and summarization of machine-readable texts". In: *Information retrieval and hypertext*. Springer, pp. 51–73.

Socher, Richard et al. (2011). "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions". In: *EMNLP 2011 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* ii, pp. 151–161. ISSN: 1937284115. DOI: 10.1.1.224.9432. URL: http://dl.acm.org/citation.cfm?id=2145450.

Spärck Jones, Karen (2007). "Automatic summarising: The state of the art". In: *Information Processing and Management* 43.6, pp. 1449–1481. ISSN: 03064573. DOI: 10.1016/j.ipm.2007.03.009.

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15, pp. 1929–1958. ISSN: 15337928. DOI: 10.1214/12-AOS1000. arXiv: 1102.4807.

Szegedy, Christian et al. (2014). "Going Deeper with Convolutions". In: ISSN: 10636919. DOI: 10.1109/CVPR.2015.7298594. arXiv: 1409.4842.

Teufel, S., A. Siddharthan, and C. Batchelor (2009). "Towards discipline-independent Argumentative Zoning: Evidence from chemistry and computational linguistics". In: *EMNLP 2009 - Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: A Meeting of SIGDAT, a Special Interest Group of ACL, Held in Conjunction with ACL-IJCNLP 2009* August, pp. 1493–1502. URL: https://www.aclweb.org/anthology/D09-1155.

Teufel, Simone and Marc Moens (2002). "Summarizing Scientific Articles: Experiments with Relevance and Rhetorical Status". In: *Computational Linguistics* 28.4, pp. 409–445. ISSN: 0891-2017. DOI: 10.1162/089120102762671936. URL: http://www.aclweb.org/anthology/J02-4002.

"The Rise of Open Access" (2012). In: *Science* 342.October, pp. 12–14. URL: http://science.sciencemag.org/content/342/6154/58/tab-pdf.

Thomas, Stefan et al. (2015). "ExB Text Summarizer". In: 2014. URL: http://www.aclweb.org/anthology/W15-4637.

Visser, W. T. and M .B. Wieling (2009). "Sentence-based Summarization of Scientific Documents". In: pp. 1–18. DOI: 10.1.1.103/4925. URL: http://www.martijnwieling.nl/files/wielingvisser05automaticsummarization.pdf.

Wan, Xiaojun and Jianmin Zhang (2014). "CTSUM : Extracting More Certain Summaries". In: *Acm Sigir*, pp. 787–796. DOI: 10.1145/2600428.2609559.

Weston, Jason, Sumit Chopra, and Antoine Bordes (2015). "Memory Networks". In: *International Conference on Learning Representations*, pp. 1–14. ISSN: 1098-7576. DOI: v0. arXiv: 1410.3916v10. URL: http://arxiv.org/abs/1410.3916.

Xu, Kelvin et al. (2016). "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. ISSN: 19410093. DOI: 10.1109/72.279181. arXiv: arXiv:1211.5063v2.

Yousefi-Azar, Mahmood and Len Hamey (2017). "Text summarization using unsupervised deep learning". In: *Expert Systems with Applications* 68, pp. 93–105. ISSN: 09574174. DOI: 10.1016/j.eswa.2016.10.017. URL: http://dx.doi.org/10.1016/j.eswa.2016.10.017.

Zeiler, Matthew D. and Rob Fergus (2014). "Visualizing and Understanding Convolutional Networks". In: *Computer Vision–ECCV 2014* 8689, pp. 818–833. ISSN: 978-3-319-10589-5. DOI: 10.1007/978-3-319-10590-1_53. arXiv: 1311.2901. URL: http://link.springer.com/10.1007/978-3-319-10590-1{_}53{\%}5Cnhttp://arxiv.org/abs/1311.2901{\%}5Cnpapers3://publication/uuid/44feb4b1-873a-4443-8baa-1730ecd16291.