KING'S College LONDON

University of London

6CCS3PRJ Final Year
Automated Timeline Extraction

Final Project Report

Author: Oliver Höhn

Supervisor: Dr Jeroen Keppens

Student ID: 1426248

March 11, 2017

Abstract

The abstract is a very brief summary of the report's contents. It should be about half-a-page long. Somebody unfamiliar with your project should have a good idea of what your work is about by reading the abstract alone. -Summary of Project When legal and related professionals examine a case, they receive a substantial number of documents. These documents need to be examined in a useful manner to understand the events occurred. One useful perspective to understand what happened is a timeline of events. However, reading a large collection of documents and producing a timeline can be cumbersome. The aim is to ease this task by producing a system that can show timelines of events based on a set of documents provided.

Originality Avowal

I verify that I am the sole author of this report, except where explicitly stated to the contrary. I grant the right to King's College London to make paper and electronic copies of the submitted work for purposes of marking, plagiarism detection and archival, and to upload a copy of the work to Turnitin or another trusted plagiarism detection service. I confirm this report does not exceed 25,000 words.

Oliver Höhn

March 11, 2017

Acknowledgements

It is usual to thank those individuals who have provided particularly useful assistance, technical or otherwise, during
your project. Your supervisor will obviously be pleased to be acknowledged as he or she will have invested quite a lot of time overseeing your progress.
-Acknowledge Supervisor, Friends & Family I would like to thank my supervisor, Dr. Jeroen Keppens. The supervision and support provided was extremely helpful and helped in the progression of the project. Also I would like to thank my family and friends for the continued support and encouragement throughout the project.

Contents

Chapter 1

Introduction

This is one of the most important components of the report. It should begin with a clear statement of what the project is about so that the nature and scope of the project can be understood by a lay reader. It should summarise everything that you set out to achieve, provide a clear summary of the project's background and relevance to other work, and give pointers to the remaining sections of the report, which will contain the bulk of the technical material. -What is Project About (incl scope)? What is Aim? Bakground? Relevance to other works? (Pointers to other sections?) The project aims to facilitate the understandment of a substantial number of documents, especially in law cases. When an employee is tasked with a law case, it is expected that the fully understand the overall structure and occurrence of events. However, when a large collection of documents are involved, this task can be both cumbersome for the employee and expensive (in time and financially) for the employer. Since time is spent reading and understanding the documents, instead of moving ahead with the task that the documents are used for.

1.1 Project Scope

Due to the system receiving as input a collection of documents, then processing and graphically showing a timeline, the main areas are Natural Language Processing (NLP), and Data Processing and Representation. All of which will be discussed in the Background section.

1.2 Objectives

//what is an event (summary, size, subjects, date, etc) The Objectives are to produce a system that is simple to use and effective. This system should take in as input a selection of documents written in correct English (i.e. in natural language) and produce a timeline.

Since most users are from other fields, not Computer Science, they should be able to understand that the system requires as input documents and produces timelines. The timeline should self-expanatory, such that the user understands which events happened during certain dates, and what each event means. The system should produce responses in an appropriate time based on the input, and allow the user to rectify where the system has made mistakes. In addition, it should allow expert users to be able to change some of the input parameters used when the events are produced, such as the length of the summary or how many processors can the program use in parallel (to limit/improve the performance). As it will be likely that the users will want to save the produced timeline for later use, they will have the ability to choose between saving the timeline as a PDF or as a JSON. The latter allows for the system to be used with 3rd-parties that would like to change the graphical representation or manipulate the given timeline in their own system.

The aim is to develop a system that produces a timeline based on the given input autonomously. The Requirements of the project are:

1. Allow documents of different format types as input (such as .docx, .pdf, .txt).
2. Show a graphical representation of the documents based on the events in them.
3. Allow the editing of produced events, within the constraints of what an event can be.
4. Produce an intemediary output to be saved (as a .pdf or .json).
5. The processing of the system should be reasonable (based on the input size).

1.3 Report Structure

In the following chapter, the background of the project will be presented in detail. Which is then followed by the design architecture and patterns used in the system. Followed by the implementation, testing and analysis of the system. In the analysis it will be determined how the requirements have been met. In the final chapter, the project will be concluded and improvements for future work will be discussed.

Chapter 2

Background

The background should set the project into context by motivating the subject matter and relating it to existing published work. The background will include a critical evaluation of the existing literature in the area in which your project work is based and should lead the reader to understand how your work is motivated by and related to existing work.

//explain what an event (from what is it built off) //defintion of an event

The resulting system aims to produce a timeline of events based on the input text. An event is given by its date(s), subjects and a short summary of the sentence that produced it. An event is produced when a given sentence contains a date. An event can have more than 1 date if it is considered to happen in a range of dates. For example, an event that happened in the 1980s would have two dates, one for the start date: 1980-01-01, and one for the end date: 1989-12-31. While an event that happened just on one day would have just one date. The subjects of an event are given by the "person, place, thing, or idea that is doing or being something"[1].

2.1 Natural Language Processing

-explain what NLP is

The projects primary area of research is Natural Language Processing (NLP). NLP is the area of computer science where the aim is to translate human readable and spoken language to a computer (cite). This requires the human input to be subject to constraints. Thereby in this project, expections on the input text are assumed. For example, it is to be expected that documents are written in correct English. As every language has their own grammar, and

thereby, their own rules, to expand the system to different languages would require different rule sets to be applied depending on the language to process the text. This includes the algorithms used in the summary of sentences used in events (discussed in a latter section).

A relevant issue is the input documents having text that is disorganised in a grammatical sense. Many NLP software tools (incluiding StanfordCoreNLP used in this project) perform extremely poorly with such input text. For example, in the paper Named Entity Recognition in Tweets: An Experimental Study [7], where they looked at the performance of popular NLP tools on "Tweets", which due to them being limited to a character count will use abbrevations that do not make sense grammatically. This is due to the fact that the NLP tools and algorithms cannot apply their rules and models to the text to identify the different components. Thereby, in this project the assumption is that the input will be in correct English, as the systems primary user is a law professional. NLP will be presented in more detail in the following section. -what parts of NLP are involved

NLP is a broad area of study. However, in this project the focus is on Automatic Summarization, Named-Entity Recognition (NER), and Sentence Breaking.

In Automatic Summarization the aim is to produce a shorter version (the summary) of a given input text, that still holds the same meaning of the original input. The summary can be built directly from the words in the input, or it can built using a dictionary. As in this project, an event is built from one sentence that contains a date, the specific area of Automatic Summarization which was focused on was Headline Generation. This is where a summary is built based on a given input text, such that the summary falls below a certain threshold value. For Headline Generation there are two main implementations: statistic based and decision (or trimming) based [3].

In the statistic based model, where Noisy-Channel models are the most prominent, as shown by the multitude of publications [2, 3, 8]. In noisy-channel models, the belief is that the summary of the given input lies within the text but it is surrounded by unwanted noise (text). These systems require a large collection of annotated data (pairs of input and their summary), which is used in the calculation of the statistics of whether or not a produced summary correctly represents the input. Examples of these algorithms can be found in the works of [3, 5].

The decision based models, are older than the statistic based models and use the grammar of the input text to trim (remove) parts of the inputs until no more rules can be applied or the summary produced is below a given threshold [4]. This is done by tokenizing, breaking an input text into words, phrases, symbols and tagging them each by an identifier(cite). The tokenized

text, which is usually represented as a tree where the leaves are the words in the input text and the inner children the identifiers, is passed through an algorithm which applies rules which removes branches of the tree until no more rules can be applied or the summary text is below a given threshold. The trimmed tree is then used to produce the summary. //compare statistics to decision

The trimming based models do not tend to produce as good of summaries as the statistic based model, due to them producing, usually, only one summary while the statistic based models produce a selection to choose from. However, as can be seen from the works of Knight and Marcu [5], the trimming based models can produce better summaries than the statistic models in some occassions. The main advantage of the trimming model is their speed, and not requiring a large corpus of data (like the statistic models) by relying on the grammar to build the summary. In newer works of text summarization, neural network models are being used. These fall under the statistical based models. They produce extremely accurate results, but as most statistical models they require a large corpus of data. They requirement of the extensive sample of annotated data also leads to these algorithms being processor-heavy, as the data needs to be read, and calculations need to be performed to produce the probability values used in identifying suitable summaries.

Due to the time-constraints of the project, it was decided to use the trimming approach, in specific the algorithm provided by Dorr, Zajic, and Schwartz [4] (Figures 1 and 2). Where the given input text is turned into a tree based on its grammatical structure, with the words in the text as the leaves, and the inner nodes being the identifiers, which is then trimmed. In addition, in statistical models the input text size is multiple sentences, i.e. a paragraph or more of text, where loading the models from the annotated data has less of an impact in performance as this being done for every sentence in that input text. -cite

Algorithm 1: Dorr, B., Zajic, D. and Schwartzm R, (2003). Hedge Trimmer: A Parse-and-Trim Approach to Headline Generation

Input : A Grammatical Tree T of the Sentence to summarize

Input : threshold: Threshold value

Output: A Summary of the given sentence

get the leftmost-lower subtree with root S;

remove time expressions;

remove determiners (e.g. 'a', 'the');

while *the number of leaves in tree > threshold and there are subtrees to remove with this rule* **do**

    remove all the children except the first, where the rightmost-lowest subtree with root XP and its first child also being an identifier XP (where XP can be NP, VP, or S);

end

while *the number of leaves in tree > threshold and there are subtrees to remove with this rule* **do**

    remove any XP (where XP can be NP,VP,PP) before the first NP found;

end

get Tree T' from lastRule;

from T' create a sentence S by reading of the leaves in pre-order;

return S;

Algorithm 2: Last Rule

Input : A Grammatical Tree T of the Sentence to summarize

Input : threshold: Threshold value

Output: A Grammatical Tree of the Summary of the input after the last rule has been applied

```
1  if the number of leaves in tree ¿ threshold then
2      make a copy of the tree T';
3      while the number of leaves in tree > threshold and there are subtrees to remove with
         this rule do
4          remove any trailling PP nodes (and their children);
5      end
6      if the number of leaves in tree > threshold then
7          while the number of leaves in tree > threshold and there are subtrees to remove
             with this rule do
8              remove any trailling SBARs (and their children);
9          end
10         while the number of leaves in tree > threshold and there are subtrees to remove
             with this rule do
11             remove any trailling PPs (and their children);
12         end
13     end
14     return T';
15 end
16 return T;
```

2.2 Data Processing and Representation

In the grammatical tree formed, the inner leave identifiers are given by the P.O.S Treebank[1]. Each word or set of words are given a part of speech tag identifying them. An example can be found below. (example figure).

On the grammatical tree shown in the previous figure, we will apply the algorithm proposed by Dorr, Zajic, and Schwartz [4]. This can be shown graphically in the following figure. It is to be noted that the value of the threshold does not force the summary to be below it, but it is

[1] `http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

a length of the summary to which the algorithm is working to. For the sentence: "On Friday the Washington Post came out with the latest from its long-running investigation into Trump's charitable donations.", the following gramamtical tree is produced 2.1.

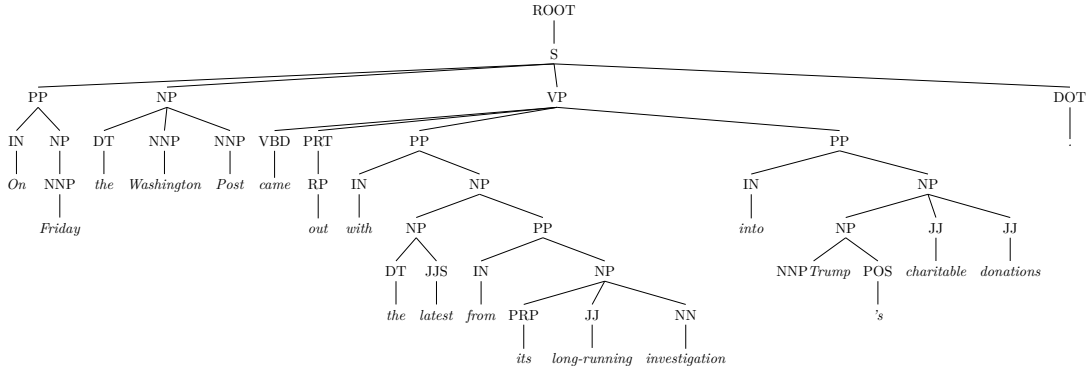

Figure 2.1: P.O.S/Grammatical Tree of: "On Friday the Washington Post came out with the latest from its long-running investigation into Trump's charitable donations."

The inner nodes in the tree are identifiers given by the P.O.S Treebank, and the leaves are the words in the sentence. A parent identifier can be a broader identifier of a collection of sub-identifiers. Using the algorithm described previously [4], the tree can be processed as follows. The following example is used as a visualisation of how the decision-based algorithm works. Firstly, the lowest-leftmost S must be identified, as can be seen from the figure 2.1 there is only one subtree with root S. This subtree is extracted, and the algorithm is continued on it. The next step is to remove time expressions, of which there is only one "Friday", however we must remove its parent to (and thereby its children) to avoid producing a grammatically incorrect summary. The result is graphically shown in the following figure 2.2.

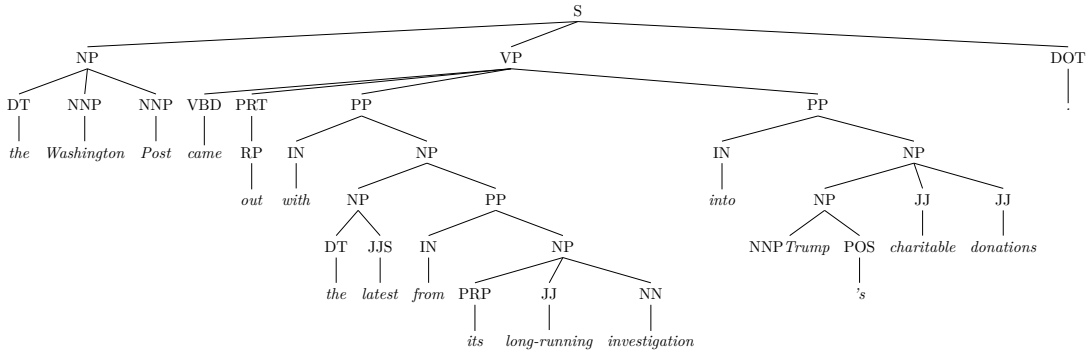

Figure 2.2: P.O.S/Grammatical Tree after removing time expressions

On the resulting tree, the algorithm dictates the removal of some determiners. A determiner is identified by the "DT" parent tag, however not all of the parents identified are removed, only ones that have a child of label "the" or "a". The resulting tree of applying this rule is given by

Figure 2.3.

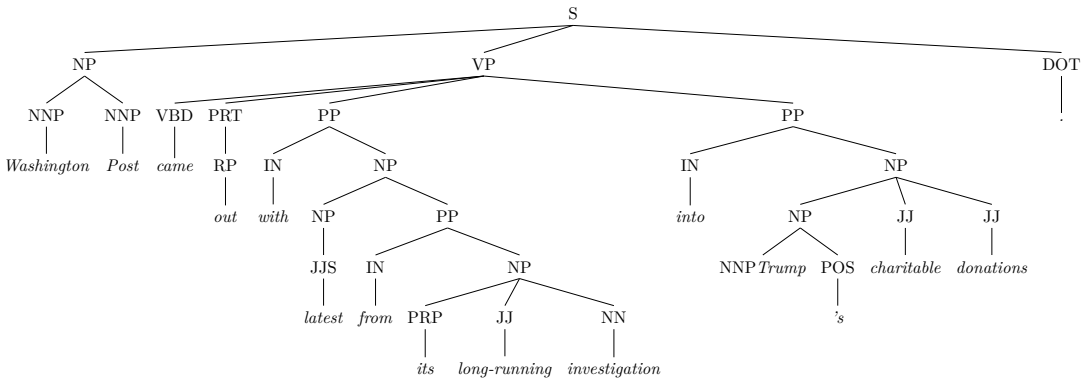

Figure 2.3: P.O.S/Grammatical Tree after removing determiners

The threshold value is used in the next following rules. For this example, the threshold value is 10. From the two rules that are left, only one will be applied as the resulting tree's number of leaves will fall below the threshold value, such that the last rule will not be applied to avoid the algorithm from trimming the summary to much. Hence, the threshold's value importance is of stopping the algorithm from over-trimming or under-trimming. When the tree is over-trimmed there is the possibility that the resulting summary does not have the core meaning of the original sentence, or has no meaning. The advantage of under-trimming is that the meaning of the sentence is very likely kept, due to the summary sentence being of greater length of the optimal summary, thereby having more words and such more meaning, and being closer to the original sentence. However, the aim is to produce a summary, i.e. a short sentence with the same meaning as the original, thus under-trimming can ensure the meaning is kept in the summary, but not that the resulting summary is optimal in size or time (when reading many summaries, shorter ones will be appreciated over longer ones). When applying the XP-Over-XP rule, when a parent of identifier type XP (where XP can be NP,VP or S), has a first child identifier of type XP also, then all other children of the parent are removed. This is done iteratively until the resulting tree is below the threshold, or the rule cannot be applied further due to there not being anymore parent XP, first child XP pairs. The first iteration of the rule is shown in Figure 2.4, and the second and final iteration in Figure 2.5.

If the number of leaves did not fall under the threshold value, then the next rule to be applied is XP-Before-NP. In this rule, any XP before the NP of the sentence (the grammatical subject) is removed. This would be carried out until the number of leaves falls below the threshold, or the rule cannot be applied further. Finally, the last rule is applied where iteratively trailing PP and SBAR subtrees are removed until the threshold is reached or the rule cant be applied. The result

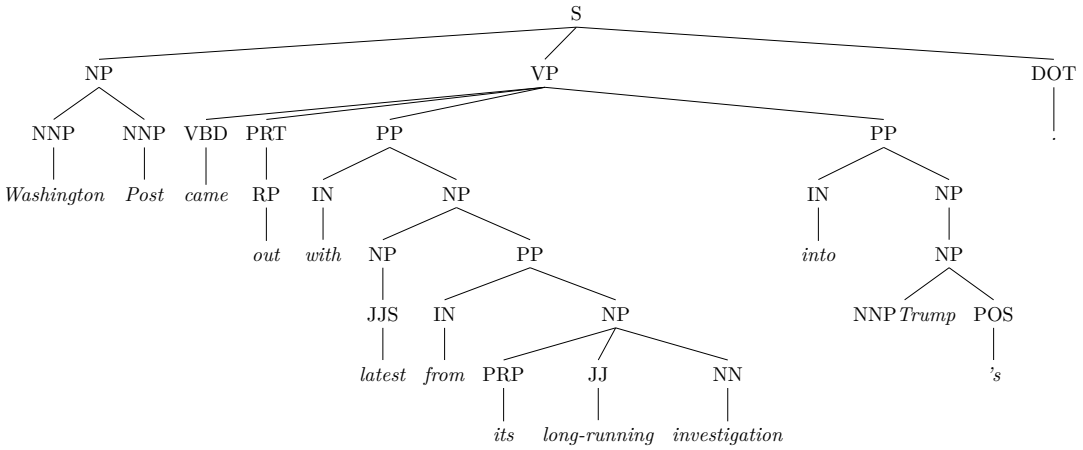

Figure 2.4: P.O.S/Grammatical Tree after XP-Over-XP first iteration

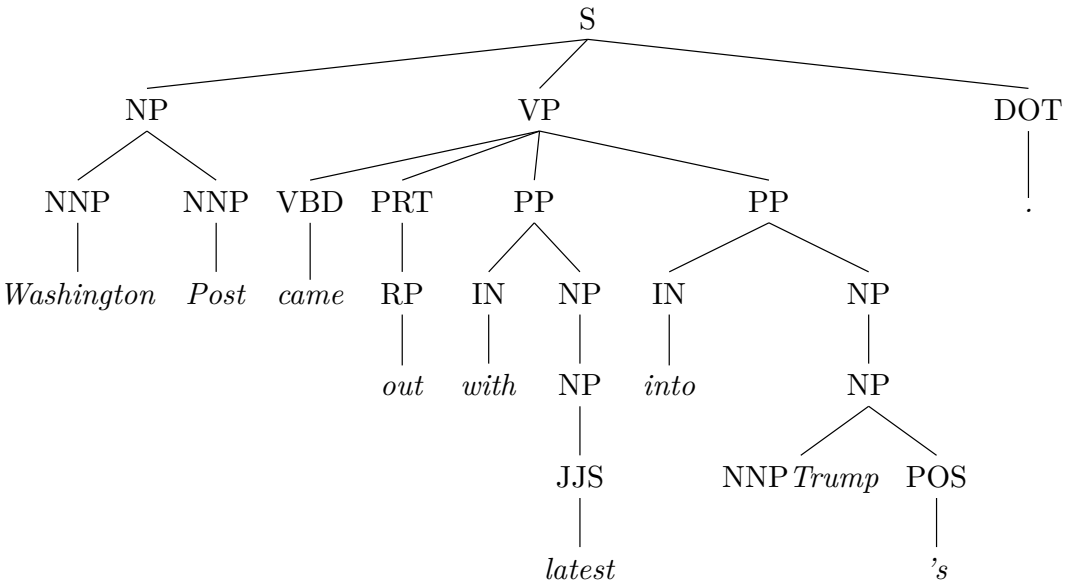

Figure 2.5: P.O.S/Grammatical Tree after XP-Over-XP second iteration

of applying this algorithm (ref algorithm) on the input text: "On Friday the Washington Post came out with the latest from its long-running investigation into Trump's charitable donations.", is "Washington Post came out with latest into Trump's". It should be noted that the original meaning of the sentence, which is that the Washington Post released a new article, is kept. However, it should also be noted that the resulting summary is not grammatically correct as "Trump's" should be "Trump". Even though the summary is not grammatically correct the summary is significantly shorter than the input text, and does represent the same event. It is a headline of the input, and should indicate to a reader what the event is about, to allow them to have a general understanding of what occurred. -tags are POS (cite) //d -example of input text to tree //d -example of producing summary //d -algorithm //d -what are the options for

the summary (Neural Networks vs Decision-Based) //d -give an algorithm for determining the summary, with an example //d

-explain the date problem, with example (determine that it uses an ISO standard)

2.3 Normalizing Dates

An issue when processing the documents and picking out events, and then placing them in a timeline, is that it is very likely that sentences will not include the full date of when an event occurred. Due to how a temporal expression such as ”Yesterday” or ”Last week” have different meanings depending on the context in which they are in, it is necessary to determine an exact date for that event to be able to produce a timeline. For example, if the text was written on the 11th of March 2017, then ”Yesterday” in its context to the 10th of March 2017, but if the document was written on the 1th of February 1689, then it refers to the 31st of January 1689. Thereby, it is important to be able to infer reliably and accurately the date to which ambigious time expressions refer. Since the dates will be needed to compare the events in a timeline, and be able to sort them. Thereby imposing constraints, where events that happens within the same time period are grouped together and not separated by events that happen it completely different time eras. This is especially valuable to the user, as they should be able to see events that happen within the same time period close together not separated.

To determine the context of where an event occurred, it will be necessary to have a reference point, a base date. In other timeline works, similar techniques have been used [6]. The reference point should be the context in which the document was written in, or was aimed to be written in. This can be the publishing date of an article, or the creation date of a document, or any date which allows the exact date to be determined. It should be noted that for exact dates that are described in text, such as ”On the 12th of December 1996...”, the reference point has no value, as it can be determined without it that this text refers to an event that occurred on 12-12-1996.

There is also the possibility that temporal expressions point to a range of dates, i.e. ”In the 1980s...”. While it is not possible to determine the exact date, or dates, this event is describing, it can be determined reasonably [6] that it is somewhere between the start of 1980s, i.e. 01-01-1980, and the end of the 1980s, i.e. 31-12-1989. This can be used to attempt to provide exact dates for that event, a start and end date, to then compare with other events in the production of the timeline. This approach of determining start and end dates where events could have occured, is used in the implementation of the system. While it might not tell the user exactly

when the event occurred, it gives them an idea of when it could have, which is the system's aim.

Chapter 3

Report Body

The central part of the report usually consists of three or four chapters detailing the technical work undertaken during the project. **The structure of these chapters is highly project dependent**. They can reflect the chronological development of the project, e.g. design, implementation, experimentation, optimisation, evaluation, etc (although this is not always the best approach). However you choose to structure this part of the report, you should make it clear how you arrived at your chosen approach in preference to other alternatives. In terms of the software that you produce, you should describe and justify the design of your programs at some high level, e.g. using OMT, Z, VDL, etc., and you should document any interesting problems with, or features of, your implementation. Integration and testing are also important to discuss in some cases. You may include fragments of your source code in the main body of the report to illustrate points; the full source code is included in an appendix to your written report. -Tasks in project (Design, Implementation, Experimentation, Optimissatio, Evaluation) -present alternatives, compare them, why picked -justify software used -problems identified -important features -testing -stanford corenlp pos tags

3.1 Section Heading

3.1.1 Subsection Heading

Chapter 4

Design & Specification

-design of architecture and UI

4.1 Section Heading

Chapter 5

Implementation

-How implemented

5.1 Section Heading

Chapter 6

Professional and Ethical Issues

Either in a seperate section or throughout the report demonstrate that you are aware of the **Code of Conduct & Code of Good Practice** issued by the British Computer Society and have applied their principles, where appropriate, as you carried out your project. -how dealt with ethical approval? (newspapers used, etc.) -in analysis kept testers anonymous

6.1 Section Heading

Chapter 7

Results/Evaluation

-present how did analysis, why?, other options (relate to work) -results of analysis -conclusion

7.1 Software Testing

7.2 Section Heading

Chapter 8

Conclusion and Future Work

The project's conclusions should list the key things that have been learnt as a consequence of engaging in your project work. For example, "The use of overloading in C++ provides a very elegant mechanism for transparent parallelisation of sequential programs", or "The overheads of linear-time n-body algorithms makes them computationally less efficient than $O(n \log n)$ algorithms for systems with less than 100000 particles". Avoid tedious personal reflections like "I learned a lot about C++ programming...", or "Simulating colliding galaxies can be real fun...". It is common to finish the report by listing ways in which the project can be taken further. This might, for example, be a plan for turning a piece of software or hardware into a marketable product, or a set of ideas for possibly turning your project into an MPhil or PhD.

-what have you learned? -how can the project be carried further (neural net for summary, building on the StanfordCoreNLP for detas depending on others)

References

[1] Sentence subjects. http://grammar.ccc.commnet.edu/GRAMMAr/subjects.htm. URL http://grammar.ccc.commnet.edu/GRAMMAr/subjects.htm. Accessed 4 Dec. 2016.

[2] S. Chopra, M. Auli, and A. Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of NAACL*, 2016. URL http://nlp.seas.harvard.edu/papers/naacl16_summary.pdf.

[3] H. Daumé III and D. Marcu. Noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 449–456, 2002. URL http://www.aclweb.org/anthology/P02-1057.

[4] B. Dorr, D. Zajic, and R. Schwartz. Hedge trimmer: A parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL 03 on Text summarization Workshop-Volume 5 (ACL)*, pages 1–8, 2003.

[5] K. Knight and D. Marcu. Statistics-based summarization âĂŞ step one: Sentence compression. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 703–710, 2000. URL http://www.aaai.org/Papers/AAAI/2000/AAAI00-108.pdf.

[6] D. McClosky and C. Manning. Learning constraints for consistent timeline extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 873–883, 2012. URL http://nlp.stanford.edu/pubs/dmcc-emnlp-2012.pdf.

[7] A. Ritter, S. Clark, Mausam, and Etzioni. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, 2011. URL https://aclweb.org/anthology/D/D11/D11-1141.pdf.

[8] A. Rush, S. Chopra, and J. Weston. A neural attention model for sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389, 2015. URL `http://www.aclweb.org/anthology/D15-1044`.

Appendix A

Extra Information

A.1 Tables, proofs, graphs, test cases, ...

The appendices contain information that is peripheral to the main body of the report. Information typically included in the Appendix are things like tables, proofs, graphs, test cases or any other material that would break up the theme of the text if it appeared in the body of the report. It is necessary to include your source code listings in an appendix that is separate from the body of your written report (see the information on Program Listings below).

Appendix B

User Guide

B.1 Instructions

You must provide an adequate user guide for your software. The guide should provide easily understood instructions on how to use your software. A particularly useful approach is to treat the user guide as a walk-through of a typical session, or set of sessions, which collectively display all of the features of your package. Technical details of how the package works are rarely required. Keep the guide concise and simple. The extensive use of diagrams, illustrating the package in action, can often be particularly helpful. The user guide is sometimes included as a chapter in the main body of the report, but is often better included in an appendix to the main report. -how to set up (commands) -how to use given pieces of sample text -images

Appendix C

Source Code

C.1 Instructions

Complete source code listings must be submitted as an appendix to the report. The project source codes are usually spread out over several files/units. You should try to help the reader to navigate through your source code by providing a "table of contents" (titles of these files/units and one line descriptions). The first page of the program listings folder must contain the following statement certifying the work as your own: "I verify that I am the sole author of the programs contained in this folder, except where explicitly stated to the contrary". Your (typed) signature and the date should follow this statement.

All work on programs must stop once the code is submitted to KEATS. You are required to keep safely several copies of this version of the program and you must use one of these copies in the project examination. Your examiners may ask to see the last-modified dates of your program files, and may ask you to demonstrate that the program files you use in the project examination are identical to the program files you have uploaded to KEATS. Any attempt to demonstrate code that is not included in your submitted source listings is an attempt to cheat; any such attempt will be reported to the KCL Misconduct Committee.

You may find it easier to firstly generate a PDF of your source code using a text editor and then merge it to the end of your report. There are many free tools available that allow you to merge PDF files.