Interactively Learning to Summarise Timelines by Reinforcement Learning

By

YUXUAN YE

Department of Computer Science
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of MASTER OF SCIENCE BY RESEARCH in the Faculty of Engineering.

TUESDAY 30 NOVEMBER, 2021

Word count: 14513

AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: ..... DATE: TUESDAY 30 NOVEMBER, 2021

Abstract

Timeline Summarisation is a special type of automatic summarisation task, in which temporal information is as important as summaries. It aims at generating a time-ordered event list containing concise event summaries and their precise happened times described in source documents like new articles. However, current approaches are neither capable of adjusting their content according to a certain user's interest nor being adapted to new domains without a huge amount of reference timelines. Therefore, I propose a Reinforcement Learning-based interactive timeline summarisation system which can interactively learn from the user's feedback to generate timelines that meet the user's demands. I define a compound reward function that can update automatically according to the received feedback to ensure topical coherence, factual consistency and linguistic fluency of the generated summaries. The system use this reward function to adjust its output content via Reinforcement Learning to fine-tune an abstractive Multi-document Summarisation model. The advantage of my system is to avoid the need of using reference summaries in training. The experiments show that my system outperforms state of the art on the benchmark Timeline Summarisation dataset, Timeline17, and could generate accurate and precise timelines tailored for each user.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Edwin Simpson for his advise and guidance for this whole year. He led me into the scientific research field of Natural Language Processing and Reinforcement Learning, encouraging me to make my research plan and think critically. His support enabled me to finish my thesis. And I'm looking forward to work with him in the following four years in my PhD study.

I am also grateful to my girlfriend, Yanquan Nie. I wrote this thesis while being quarantined in Beijing due to epidemic. Although she was at Bristol during that time, her online company helped my get through the anxiety and made me focus on my writing.

Table of Contents

Page

List of Tables **ix**

List of Figures **xi**

1 Contextual Background **1**

1.1 Timeline Summarisation . . . 1

1.1.1 Timeline . . . 1

1.1.2 Summarising A Timeline . . . 2

1.2 Challenges . . . 3

1.3 Summary of the Aims . . . 4

2 Technical Background **5**

2.1 Natural Language Processing . . . 5

2.1.1 Symbolic NLP . . . 5

2.1.2 Statistical NLP . . . 6

2.1.3 Neural NLP . . . 7

2.2 Automatic Text Summarisation . . . 9

2.2.1 Transformers in Summarisation Tasks . . . 10

2.2.2 Timeline Summarisation . . . 11

2.3 Interactive Learning . . . 12

2.3.1 Interactive Learning in Natural Language Processing . . . 13

2.4 Reinforcement Learning . . . 13

2.4.1 Reinforcement Learning Algorithms . . . 14

2.4.2 Reinforcement Learning-based Interactive Learning in Natural Language Processing . . . 14

2.5 Clustering . . . 14

2.5.1 Affinity Propagation . . . 15

2.5.2 Markov Clustering . . . 15

3 Method **17**

3.1 Baseline . . . 17
3.1.1 Date Tagging . . . 18
3.1.2 Event Detection . . . 18
3.2 The proposed method's workflow . . . 19
3.3 Reinforcement Learning-based Interactive Fine-tuning . . . 20
3.3.1 Notations . . . 20
3.3.2 Interaction . . . 20
3.3.3 Reinforcement Learning Phase . . . 21

4 Evaluation **29**
4.1 Datasets . . . 29
4.1.1 Dataset Characteristics . . . 29
4.1.2 Experimental Settings . . . 30
4.2 Metrics . . . 31
4.2.1 ROUGE . . . 31
4.2.2 Adapted ROUGE for Timeline Summarisation Evaluation . . . 31
4.3 Simulated Experiments . . . 32
4.3.1 Simulated Interaction . . . 32
4.4 Results . . . 33
4.4.1 Reproducing the Results of Prior Work . . . 33
4.4.2 Clustering Analysis . . . 33
4.4.3 Summarising without Reinforcement Learning . . . 34
4.4.4 Simulated Reinforcement Learning-based Interactive Timeline Summarisation System . . . 35
4.4.5 Run time and Hyperparameters . . . 37

5 Conclusion **39**
5.1 Contributions . . . 39
5.2 Project Status . . . 40
5.3 Implications and Future Direction . . . 40

Bibliography **43**

List of Tables

Table **Page**

3.1 An example using Heideltime to annotate a news article published in 3rd May 2010. 18

4.1 Statistic characteristics of the two datasets (i), cited from [Ghalandari and Ifrim, 2020]. 30
4.2 Statistic characteristics of the two datasets (ii), cited from [Ghalandari and Ifrim, 2020]. 30
4.3 Statistic characteristics of the two datasets (iii), cited from [Ghalandari and Ifrim, 2020]. 30
4.4 Replicated results evaluated by Alignment-based ROUGE-1 F1 score. 33
4.5 Replicated results evaluated by Alignment-based ROUGE-2 F1 score. 33
4.6 The average number of clusters obtained by different methods. 34
4.7 Average timeline length generated by different approaches. 34
4.8 Average precision, recall and F1 for date selection. 34
4.9 Replicated results evaluated by Alignment-based ROUGE-1 F1 score. 35
4.10 Replicated results evaluated by Alignment-based ROUGE-2 F1 score. 35
4.11 Results of simulated Reinforcement Learning-based interactive timeline summarisation system ($k = 6$). *Round* represents how many times that the agent sampled an episode from each cluster. Each round means that the agent learned multiple episodes over all clusters. 35
4.12 Different reward weights and corresponding output. 37
4.13 Experimental environment (Google Colab Pro). 37
4.14 Hyperparameters. 38

List of Figures

Figure **Page**

1.1 An example timeline describing the oil spill in Mexico Gulf in 2010. 1
1.2 A Timeline Summarisation Task. 2
1.3 An example of different timelines that users with different background would be interested in. 3

2.1 An typical conversation between a user and ELIZA. 6
2.2 Training a machine translation system on parallel corpus. 7
2.3 Meaning of representation learning 7
2.4 The architecture of Transformer. 9
2.5 The noising methods of BART. 11
2.6 The training method of PEGASUS. 11

3.1 The workflow of the proposed Timeline Summarisation system [Ye and Simpson, 2021]. 19
3.2 The interaction process of the system 21
3.3 A view of the proposed Reinforcement Learning method 22
3.4 The system learns a new policy via Reinforcement Learning. 23
3.5 The architecture of the critic. 27

4.1 Some generated timeline examples from the news topic, Haiti earthquake (i). 36
4.2 Some generated timeline examples from the news topic, Haiti earthquake (ii). 37

CHAPTER 1

CONTEXTUAL BACKGROUND

1.1 Timeline Summarisation

1.1.1 Timeline

Timeline is a list of events in chronological order. It is widely applied in daily life to track how the event develops from the very first beginning to the current state.

Because of its recording ability, timelines are born to present the trend and development of events. This attribute makes timelines play important roles in knowledge delivery. For instance, timelines are useful on investigation of historic events. A high-quality timeline from old news articles or Wikipedia is helpful for readers to understand the outline and different stages of an event without efforts. Another example is that when disaster happens, tons of news reports will burst on social media like Facebook and Twitter [Imran et al., 2016, 2013], a timeline carrying precise time tags and useful information can help the government and professionals figure out the real situation, which may save thousands of lives. Figure 1.1 shows a part of the timeline of the BP oil event happened in 2010. Although it has continued for more than a decade since the first leaking, its development and relevant policy are still highly concerned today.

2010-06-21	BP's chief of staff has pulled out of a speech he was scheduled to deliver today at a major oil conference in London … scheduled to appear at the conference.
2010-06-23	Tony Hayward's days at the helm of BP are numbered … be responsible for environmental damage in the Gulf of Mexico.
2010-06-29	The first hurricane of this year's Atlantic season has hit BP's oil spill clean-up efforts in the Gulf of Mexico … reports the BBC.

FIGURE 1.1. An example timeline describing the oil spill in Mexico Gulf in 2010.

1.1.2 Summarising A Timeline

In the example of BP oil spill, oceans of news articles have been published to report this disaster's development and other aspects since the first break out. The massive amount of relevant news articles has become a barrier for people to distinguish and sort out those actual critical subsequent events. Summarising timelines through such huge amounts of documents is a tricky task even to professional writers.

Timeline Summarisation (TLS) receives a set of record documents as input and generates a list of event summaries with corresponding time tags (usually dates) as shown in Figure 1.2. It requires the author to capture both essential dates and key sub-events in the long term, therefore it is more complex than normal summarisation tasks. It is not difficult to imagine that the job is expensive because reviewing and summarising all of the relevant articles is time-consuming, requiring the expertise of reading and writing skills.

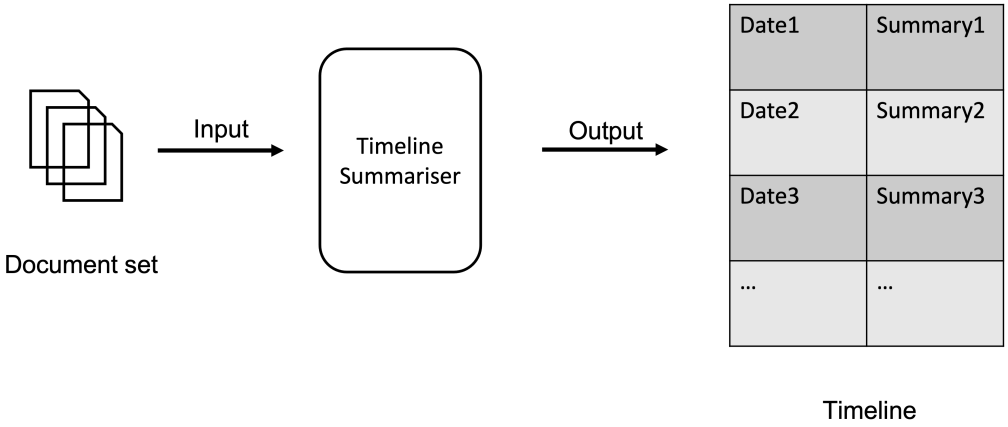

FIGURE 1.2. A Timeline Summarisation Task.

Currently, this work in real-life industries is still dependent on human efforts. Therefore, only highly concerning events are eligible to have authors writing up timelines for them. As a result, there are not adequate timelines available for readers, and those high-quality ones are even handful.

In addition, most of the timelines are still presented in a traditional way by text. It results in the timeline in the lack of flexibility to adjust its content. Readers with different backgrounds have different preferences over the same news event. For example, in Figure 1.3, Chinese readers will undoubtedly show great interest to read the news about Chinese athletes in the timeline of the Tokyo Olympics, while American readers will prefer theirs. Generating several different versions of timelines will cause massive amounts of repetitive work, which wastes both time and money.

Furthermore, the number of timeline versions is challenging to decide because a reader may have multiple interests in one event, as shown in Figure 1.3. A Chinese basketball fan will pay

attention to Chinese athletes and the U.S. basketball team in Tokyo Olympics. So if we cannot categorise the users easily, we cannot just produce a few different timelines and ask them to choose. We have to get some information from the user to understand what their individual interests are. And their interests can be delivered as guidance in automatic timeline generation.

	Chinese Readers		American Readers		Chinese basketball fans
24th Jul	Qiang Yang won the gold medal for women's shooting 10m air rifle with 251.8 rings. This is the first gold medal for the Tokyo Olympics...	24 Jul	Kieran Smith won bronze i n the men's 400m freest	24th July	Qiang Yang won the gold medal for women's shooting 10m air rifle with 251.8 rings. This is the first gold medal for the Tokyo Olympics...
25th Jul	Tingmao Shi and Han Wang won the women's double three-meter springboard gold medal, achieving the five consecutive championships of the Chinese diving team in this event...	25th Jul	Chase Kalisz of the United States won the gold medal in the men's 400 metres m edley...	3rd Aug	Team USA defeated Spain 95-81 in the quarterfinals of the Tokyo Olympics on Tuesday, ending Spain's Olympic run for the fifth consecutive time...
26th Jul	Bingjie Li won a bronze medal in women's 400m freestyle swimming	27th Jul	American Carissa Moore won the first women's surfing shortboard event in Olympic history...	10th Aug	Team USA wins gold in men's basketball for the fourth Olympics in a row...

FIGURE 1.3. An example of different timelines that users with different background would be interested in.

1.2 Challenges

Due to the aspects mentioned above, building an automatic timeline summarisation system is challenging.

Firstly, vast amounts of valuable information lie buried inside extensive document collections, from social media to customer service logs, maintenance reports, and historical archives. However, the vast number of documents makes this information inaccessible even to domain experts. Thus, our system is expected to automatically refine and summarise important information from a bunch of input documents.

Secondly, the system is expected to consider the requirements of a specific user – their background knowledge, goals and information needs, which drastically affect the usefulness of a particular summary. Therefore, we need to design an efficient mechanism to interact with users to grasp such information.

Another challenge is that due to the difficulties of manually generating timelines, there are only a few reference timelines that can be made use of, although it is needed in many real-life scenarios. This situation undoubtedly limits the choices of methods that can be applied to solve the problem. Because most effective summarisation methods rely on supervised learning algorithms which require reference summaries. These ground-truth summaries are expensive to create and not always available for a particular news domain. And of course, we cannot ask

individual users to provide reference timelines to adapt the system output to their needs. This problem motivates the need for an approach that can adapt to new topics and users without reference summaries, which makes us consider whether Reinforcement Learning can solve this task. Reinforcement Learning can train a model using a reward signal rather than reference summaries. This leads to a further challenge — how to design a reward function that captures the properties of a good summary.

Last but not least, summarising a timeline is tricky. Because it has to capture accurate time information while processing text. It does not only need the ability to summarise multiple documents, but also requires the system being capable of split information into corresponding groups. Therefore, it is a more complex problem comparing to common summarisation tasks, such as multi-document summarisation and single-document summarisation which will be mentioned in 2.2. And since the timeline summarisation is not well-solved, it motivates us to propose such a project to explore possible methods for generating high-quality timelines for different users.

1.3 Summary of the Aims

The aims of this project are listed below:

1. Investigate prior Timeline Summarisation methods. Choose potential basic methods and reproduce their results.

2. Investigate and choose appropriate timeline summarisation datasets.

3. Build an interactive Timeline Summarisation system, which is able to tailor the output according to the user's demands. This includes several sub-aims:

    a) Design the interaction process and acceptable feedback types.

    b) Design reward functions that can automatically update according to the user's feedback.

    c) Choose suitable proper Reinforcement Learning algorithm for fine-tuning an abstractive Summarisation model.

4. Use proper evaluation metrics to compare the outcomes of the system with prior works; analyse the results both qualitatively and quantitatively.

CHAPTER 2

TECHNICAL BACKGROUND

2.1 Natural Language Processing

Natural Language Processing (NLP), as a research field, is a cross-discipline of Mathematics, Computer Science and Linguistics. It mainly focuses on developing computer applications to process natural language text, which is straightforward for humans to understand but confusing for computers due to its complex structures and rules, alone with its ambiguity. The goal of NLP is to make computers capable of processing raw text like a human, which involves the challenges like natural language understanding, natural language generation and speech recognition.

With the speeding up of the informatisation of the society, increasing attention has been attracted by NLP during these years. Tons of information is being generated every second, which requires advanced tools to process it. Nowadays, a variety of applications have been developed to finish different types of text processing tasks. For example, Google Translate provides service to millions of users every day. And intelligent assistants, like Alexa and Siri, can help people arrange their daily schedule through dialogue without inputting any structured data. NLP technology is changing our lives.

2.1.1 Symbolic NLP

The history of NLP can be traced back to the 1950s. Researchers explored encoding rules on machines to answer questions and handle other NLP tasks at this very early stage. The system generates natural language text via analysing conditions. A series of famous experiments were done during that time and were regarded as the foundation exploration of NLP. For instance, ELIZA [Weizenbaum, 1966], a very famous simulation of person-centred therapy, was capable of providing surprisingly human-like responses.

```
Men are all alike.
IN WHAT WAY
They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE
Well my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED
```

FIGURE 2.1. An typical conversation between a user and ELIZA. Figure reproduced from Weizenbaum [1966]. (ELIZA's responses are printed in uppercase.)

In general situations, rule-based systems came up with a knowledge base which empowered the system to generate factually consistent natural language text. However, the fixed rule set and knowledge base limited the extrapolating ability of the system. ELIZA [Weizenbaum, 1966] used some tricks to instruct the user to raise questions covered by its rule set and knowledge base, which was not practical in real treatment. But it won the renown of having compatibility of processing input exceeding its knowledge base for it. However, it is apparently not realistic to encode all the cases on a machine. Therefore, the deficiency of inference ability of symbolic systems forced people to investigate methods that could extrapolate ground truth to uncertain situations.

2.1.2 Statistical NLP

Following the development steps of symbolic NLP, the importance of quantitative evaluation on NLP systems experienced a significant rising, which led to the revolution of NLP. In this period, machine learning approaches were introduced into this field for the first time and gained massive success on practical application compared to rule-based systems. Researchers built language models on top of Statistical Theory using statistical models to approach the real distribution of words. Many classic machine learning methods, such as Support Vector Machine, Naive Bayes and Random Forest, reached or even exceeded human level on text classification, machine translation and other NLP tasks.

In the first decade of this period, people mainly focused on supervised learning algorithms. Therefore, parallel corpora played an essential role in accelerating the development of statistical methods in NLP. A typical example is that machine translation made progress by taking advantage of multilingual textual corpora built by prior work on translating governmental proceedings.

In the next decade, with the rapid growth of Internet and web service, the amount of raw text has been enlarging and easier to access. Under such a circumstance, the demand for labelled data of supervised learning methods severely restricted the development of NLP technology. People thus turned to exploring applying unsupervised and semi-supervised learning methods to NLP

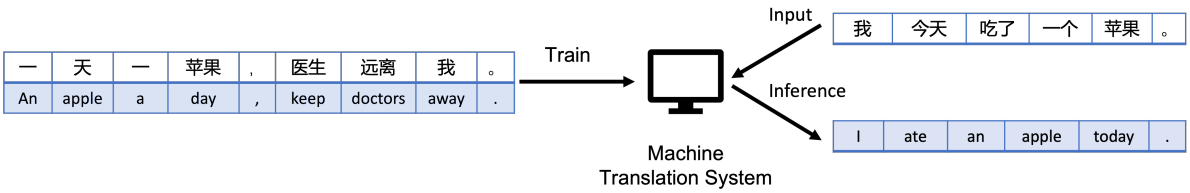

FIGURE 2.2. Training a machine translation system on parallel corpus.

tasks. These methods partly reduced the dependence of referenced data. However, low accuracy and factual conflict came with the lack of supervision as well, which made it a long way to go in applying them in real life.

2.1.3 Neural NLP

Due to the consistent escalation of computation power, the deep neural network became widespread in all cutting-edge research fields in recent years, including Natural Language Processing. Neural networks are renowned for their strong representation learning ability [Hornik et al., 1989]. Therefore, researchers considered utilising neural networks to represent language models [Xu and Rudnicky, 2000, Bengio et al., 2003], which brought the first breakthrough of Neural NLP. Many powerful neural language models have been proposed and shown their advantages on preprocessing raw input documents. It drastically affected the methodologies and formed the foundation of current NLP research [Gururangan et al., 2020]. Many advanced NLP researches now apply these high-quality neural embedding directly, or sometimes with fine-tuning to their specific datasets. So researchers can allocate more time on designing models and methods for downstream tasks. Neural network inspired the progress of NLP.

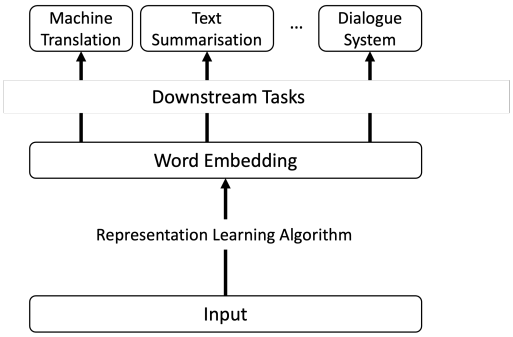

FIGURE 2.3. The development of representation learning facilitated research on other downstream tasks.

Transformers and Pretrained Models At first, NLP researchers attempted to extract information from document via Convolutional Neural Network (CNN) or Recurrent Neural Network (RNN). CNN was not flexible enough to process text input with variable length due to the fixed size of the convolutional kernel. And it was unable to model long-distance contextual information and retain sequence information. Although RNN was broadly applied in processing sequence information, which seems to naturally fit processing NLP tasks. However, the vanishing gradient problem made the training troublesome. Researchers therefore intended to modify the hidden unit's structure to solve it, such as Long Short-Term Memory [Hochreiter and Schmidhuber, 1997] and Gated Recurrent Unit [Cho et al., 2014]. They improved RNN's performance on NLP tasks and solved the long-distance dependency problem in some degree. But the recurrent structure of RNN made parallel computing impossible, which brought the low efficiency problem.

Transformer [Vaswani et al., 2017], as the most famous model in NLP, was proposed by Google in 2017. It completely removed the recurrent units and convolutional kernels in encoders. It only relied on the attention mechanism to capture the global relationship between input and output, as shown in Figure 2.4. The advantages of this architecture come subsequently:

1. The lack of recurrent connections means the computation can be parallelised.

2. Parallel computation means the model and dataset size can be increased.

3. Bigger models trained on more data seem to encode more aspects of language, thus allowing Transformer to outperform previous methods.

In addition, Transformer is one of the famous pre-trained models. With increasing investment in Artificial Intelligence and Deep Learning, giant internet companies like Google, Facebook and Microsoft are training huge models on high-performance computing clusters. To prevent reinventing wheels, some models are open-source with several different versions of weights. These so-called pre-trained models can be utilised by others on similar tasks, only with minor fine-tuning or even without any modification. Therefore, researchers could focus on their main work rather than building a whole system from scratch.

The advent of Transformer facilitated the development of NLP by providing a strong base model for many tasks. It can easily achieve state of the art on downstream tasks with only simple fine-tuning on corresponding corpus. For example, in text representation, BERT [Devlin et al., 2018] and its modified versions [Liu et al., 2019, Lan et al., 2019, Joshi et al., 2020] now rules the field. On top of these word embedding models, researchers explored embedding sentences as well. Sentence-BERT [Reimers and Gurevych, 2019] was proposed to solve the sentence embedding problem, and it greatly outperformed taking the average embedding of all the tokens in the sentence. Currently, most of the state-of-the-art works in NLP are proposed on transformer-based models.

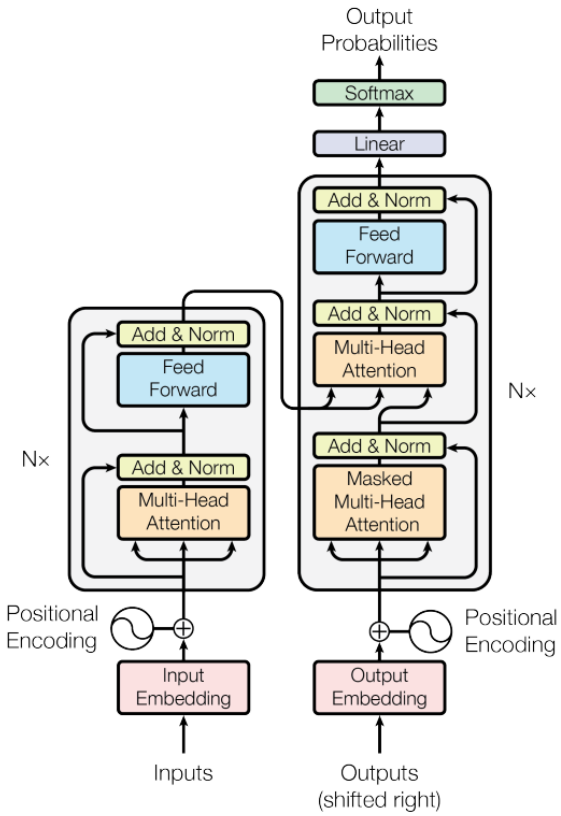

FIGURE 2.4. The architecture of Transformer [Vaswani et al., 2017].

2.2 Automatic Text Summarisation

Text Summarisation is a Natural Language Generation task. It aims to condense long articles into short summaries, reducing the input length while keeping the original meaning and critical objects. Manually writing summaries is not only time-consuming but also expensive since it requires knowledge background and writing skills. Therefore, Automatic Text Summarisation has gained tremendous popularity in recent years and strongly motivated researchers. Generally, summarisation task can often be categoriesd as single-document summarisation and multi-document summarisation.

Single-document Summarisation aims to extract important information in a long document, which can provide a brief introduction for a single article.

Multi-document Summarisation does not only simply summarise a set of documents into a single output summary, but also considers in solving the co-reference problem brought by the number of input documents.

From another perspective, Automatic Text Summarisation methods can be classified into abstractive methods and extractive methods.

Extractive Summarisation is designed to identify and extract important sentences in the source documents. It retains the original sentences. Thus its output usually has an advantage on factual consistency. However, original sentences from the source documents often contain redundant and irrelevant information, which reduce the summaries' quality. In addition, extractive methods only focus on informative sentences and neglect transition sentences making the whole summary less readable than human-written text. Therefore, extractive summaries are not directly applicable in practice without a human-in-the-loop to revise them.

Abstractive Summarisation aims at solving the defects of extractive methods mentioned above. It rewrites the source documents to preserve critical information and discard the irrelevant message. And it rephrases the expression in the original documents as a human does. Hence the output is linguistically fluent as well. Some problems can not be ignored in current abstractive summarisation methods as well. For example, generation methods sometimes produce repetitive, irrelevant or even incorrect text. So it often needs some supervision when applying in practice.

2.2.1 Transformers in Summarisation Tasks

Transformer has improved the state of the art of most NLP tasks in recent years. Some transformer-based universal language models, like GPT-2 [Radford et al., 2019], could reach human-level on several different tasks via fine-tuning. Therefore, in order to make further progress, researchers have focused on developing dedicated models and pretraining methods for downstream tasks, including text summarisation.

BART [Lewis et al., 2019] is a generalised pretraining method, which works particularly well on Natural Language Generation (NLG) tasks. It corrupts the text with a noising function and then learns a model to denoise the corrupted sentences. The model learns the relationship between tokens and the context through the destruction-reconstruction process.

PEGASUS [Zhang et al., 2019] is specifically developed for the abstractive summarisation task. It identifies sentences with high ROUGE scores - a common metric for measuring the n-gram overlap between generated and reference summaries - as containing key information. And then, it masks these important sentences, providing them to the model to reconstruct the removed text. The model learns how to condense concise and accurate abstractive summaries from long input documents in the process. It achieved state-of-the-art performance on all the 12 common downstream summarisation datasets. Although it has strong summarisation ability, the repetitive token is a known tricky problem in its output.

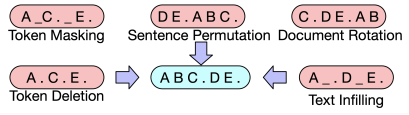

FIGURE 2.5. The noising methods of BART [Lewis et al., 2019].

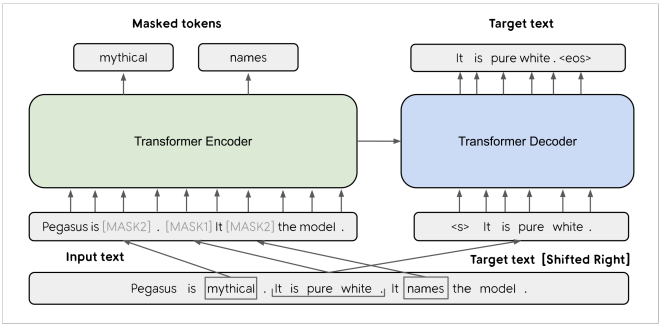

FIGURE 2.6. The training method of PEGASUS [Zhang et al., 2019].

2.2.2 Timeline Summarisation

The Timeline Summarisation (TLS) task is usually regarded as a special type of summarisation task. It does not only expect to generate precise summaries from the source documents but also needs to extract accurate dates that important sub-events happen on. Thus, it is more difficult compared to general summarisation tasks. Prior TLS methods mainly focused on producing extractive summaries [Yan et al., 2011b, Nguyen et al., 2014, Martschat and Markert, 2018]. Although some works received good marks on some evaluation metrics like ROUGE. They still inevitably suffered from the same problem that irrelevant, redundant and repeated information is also copied along with the important information.

The direct solution is applying abstractive multi-document summarisation approaches in the TLS task [Steen and Markert, 2019, Barros et al., 2019], because timeline summarisation has some common ground with it on the number of input documents. Recent advances in multi-document summarisation using transformers give us the opportunity to also improve timeline

summarisation. However, few neural network-based models have been proposed. Because supervised learning for TLS tasks requires reference timelines as supervision, which is usually inadequate in both academic research and practice, researchers usually crowdsource producing reference data to solve this problem. However, labelling such a summarisation task is extremely expensive and time-consuming, considering the necessary expertise to capture important information of both time and events from source documents. Prior works have shown that over 3000 hours were required to evaluate the summaries of Document Understanding Conferences (DUC) [Lin, 2004]. In Multi-document Summarisation, researchers have explored resolving the shortage of reference data by heuristic-based methods and Unsupervised Learning [Rioux et al., 2014, Ryang and Abekawa, 2012]. However, there is a huge gap between their results and the performance of the state-of-the-art supervised learning methods. Motivated by it, Gao et al. [2018] applied interactive learning to improve the performance of those prior works. It leveraged users' preferences to improve the performance of the summariser, which does not depend on reference data and is much easier to obtain. Inspired by these prior works, an interactive learning-based timeline summarisation method is proposed in this thesis. Details will be presented in Chapter 3.

According to Ghalandari and Ifrim [2020], prior Timeline Summarisation methods can be classified into three categories.

Direct Method processes the timeline summarisation task as a multi-document summarisation task. Summaries are composed of a small subset of sentences extracted directly from the massive corpora. Currently, the direct methods proposed by Martschat and Markert [2018] achieved state of the art.

Date-wise Method first selects notable dates [Binh Tran et al., 2013]. And then, summaries will be generated for each date.

Event Detection Method identifies events as the basis of the following steps [Ghalandari and Ifrim, 2020]. Each event will be assigned a date. And summaries will be generated from relevant source documents.

2.3 Interactive Learning

Interactive Learning (IL) enables users to interact with the machine learning algorithm. Models utilise the help of human interaction to achieve better performance and meet users' requirements. In the real world, sometimes, the problem is too complex to model mathematically. Even if the mathematical model can be built, there is no guarantee that an analytical solution exists, making traditional machine learning approaches inapplicable or futile. Interactive Learning aims to develop human-in-the-loop artificial intelligence pipelines to solve such complex problems under human guidance.

2.3.1 Interactive Learning in Natural Language Processing

The linguistic quality evaluation of given text has always been a subjective problem. Hence it is perplexing to quantify. Current evaluation metrics, like ROUGE [Lin, 2004], perplexity and BLEU [Papineni et al., 2002] are mainly based on the term frequency and word distribution. Although, they have been applied to evaluate many successful works and facilitated the development of NLP, their drawbacks are still not to be neglected. Reiter [2018] did a structured review on BLEU and drew the conclusion that it does not support most of Natural Language Generation tasks except Machine Translation. All the metrics above can only evaluate whether the generated text is describing the same object compared to the references. As an abstract concept that is hard to model, the meaning of the text cannot be evaluated or quantified well. For example, *"John beat Sam"* and *"Sam beat John"* are contradictory to each other. Nevertheless, either sentence will receive a high ROUGE-1 score when using another sentence as the reference.

Without a doubt, introducing human evaluation in training as supervision helps solve the problem. The advantage of human-computer-interaction is that human is capable of evaluating abstract concepts like text meaning. If it can be integrated into the objective function, with enough rounds of interaction, the system is expected to learn what aspects consist of high-quality output. Thus, the model would be tuned towards better performance. Sokolov et al. [2016] proposed an interactive NLP paradigm in 2016, which can learn from both numeric scores and preferences over pairs of predictions. Learning from preferences significantly decreases the cognitive burden for human evaluators.

Another important application of Interactive Learning in NLP is to solve the reference data shortage problem. Gao et al. [2018] proposed an interactive multi-document summarisation framework that did not rely on reference data to train. It collected pairwise preferences over summaries from simulated users. Thus the efficiency issue of Interactive Learning was fixed as well. Since the timeline summarisation task is similar to multi-document summarisation, the interactive timeline summarisation method is worth exploring.

2.4 Reinforcement Learning

Reinforcement Learning (RL) is another machine learning paradigm besides Supervised Learning and Unsupervised Learning. In RL algorithms, agents learn a policy through maintaining the exploration-exploitation trade-off. Apart from following the current policy to received a current best reward, it also attempts to maximise the reward function via exploring actions that haven't been tried yet. The policy is learned through a series of interaction with the environment. So it does not depend on labelled data and can be applied to many complex scenarios as long as setting up a proper environment. In this paper, RL is used to implement interaction between users and the summarisation system.

2.4.1 Reinforcement Learning Algorithms

Q-Learning [Watkins, 1989] manages to record all the policies that the agent has attempted in a table. In each step of a trajectory, the agent enquires the Q-table and choose an action with the best value. The algorithm then uses the reward for current action and discounted estimate of optimal future value to update the Q-table. The update equation is also known as Bellman Equation [Bellman, 1966].

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha \cdot (r_t + \gamma \cdot \max_a Q(s_{t+1},a) - Q(s_t,a_t))$$

Policy Gradient [Sutton et al., 2000] models the policy with a parameterised function of θ, $\pi_\theta(a|s)$. It targets at directly learning the policy. The reward is defined as the expected value of the stationary states of the policy, and its gradient can be simplified for calculation so that the Gradient Descent algorithm can be applied to find an optimal θ.

$$J(\theta) = \sum_{s \in S} d^\pi(s) V^\pi(s) = \sum_{s \in S} d^\pi(s) \sum_{a \in A} \pi_\theta(a|s) Q^\pi(s,a)$$

Actor-Critic [Mnih et al., 2016] is an improved version of Policy Gradient. The algorithm uses a *Critic* to approximate the real value function since knowing it can assist in updating the policy. In this way, the gradient variance is greatly reduced.

2.4.2 Reinforcement Learning-based Interactive Learning in Natural Language Processing

Research on the intersection of RL and NLP has received some success. Song et al. [2020] and Mesgar et al. [2020b] utilised RL algorithms to fine-tune a dialogue system. The system learned to generate a response that is consistent with the simulated user's persona. In Automatic Summarisation, interactive learning is often applied to learn a reward function through the interaction with users [Gao et al., 2018, 2019, Simpson et al., 2020]. An interesting point here is that the user cannot provide the reward directly for the many episodes required to train the model. So instead, we need an automatic reward function to tune the system to the user's input. Then we can do 'offline' RL to train the model. So that users can indirectly supervise the RL agents to learn a corresponding summary generation policy.

2.5 Clustering

Clustering is a sort of Unsupervised Learning task that involves the grouping of data points. Giving a dataset, a clustering algorithm can classify each data point into a specific group that share similar properties or features. It is a common technique for data analysis in the early stage.

In Timeline Summarisation, clustering algorithms are often applied to detect events. During the development of the event, specific sub-events will receive a lot of attention and reports. And source documents describing the same event will be embedded in similar locations in semantic space. Therefore, clustering algorithms can help detect sub-events by grouping documents that describe the same sub-events. In addition, since we do not know the number of sub-events, the clustering algorithm should determine the number of clusters by itself, which greatly limits the choices. The following subsections introduce two common clustering algorithms that are suitable for Timeline Summarisation tasks. We did some experiments to evaluate their performance on the chosen dataset and select the more suitable one empirically to apply in our system, as shown in 4.6 and 4.7.

2.5.1 Affinity Propagation

Affinity Propagation [Frey and Dueck, 2007] is an advanced graph-based clustering algorithm. The main concept of it is passing messages through the connected data points. Unlike those classic unsupervised clustering algorithms such as K-means [Lloyd, 1982], Affinity Propagation does not require the number of clusters to be manually determined. Therefore, it can be applied to process more complex and unstructured datasets compared to K-means.

Firstly, a symmetric matrix S is initialised to quantify the similarity between each pair of data points x_i and x_j. The diagonal of S controls how likely the corresponding point is to be an exemplar (the centroid of a cluster). All the elements in the diagonal are set to be the median of all the similarities. In each iteration step, the algorithm updates a responsibility matrix R and an availability matrix A as follows until all cluster boundaries are stable.

$$\begin{aligned} r(i,k) = &\quad s(i,k) - \max_{k' \neq k}\{a(i,k') + s(i,k')\} \\ a(i,k) = &\quad \min\left(0, r(k,k) + \textstyle\sum_{i' \notin \{i,k\}} \max 0, r(i',k)\right) \quad \text{, for } i \neq k \\ a(k,k) = &\quad \textstyle\sum_{i' \neq k} \max 0, r(i',k) \quad \text{, for } i = k \end{aligned}$$

2.5.2 Markov Clustering

Markov Clustering [Van Dongen, 2000] is inspired by computing the stable distribution of random walks on a undirected graph. It can decide the number of clusters without specification in advance as well.

The algorithm takes an adjacency matrix as input. To guarantee the convergence, the graph is modified by adding a self-loop to each vertex, eliminating the effects of exponent parity. It keeps calculating the power matrix of the transition matrix P, which is called *Expansion*.

$$P_e = \prod_{i=1}^{e} P$$

However, for a Markov Chain, its stable probability distribution has nothing to do with the starting point, which does not reflect the distinction well. To strengthen the connection between close vertex and weaken the connection between loose vertex, it adds a *Inflation* step.

$$P_{\Gamma_{ij}} = \frac{(P_{ij})^r}{\sum_{k=1}^{n}(P_{kj})^r}$$

Expansion ensures the information flow traverses the graph. And *Inflation* contributes to each cluster distinguishing itself from other clusters. The two steps process alternately until the matrix converges.

CHAPTER 3

METHOD

This chapter introduces the structure and implementation of the proposed Reinforcement Learning-based Interactive Timeline Summarisation System.

3.1 Baseline

This project mainly follows the state-of-the-art event detection method's workflow, CLUST [Ghalandari and Ifrim, 2020], but replaces some of the major components. CLUST processed the extractive timeline summarisation task in two stages. Its original workflow is described as below:

1. The first stage is to detect and extract the important dates and events from source documents.
    a) Using an advanced tool Heideltime [Strötgen and Gertz, 2013] to tag all the dates in raw source documents.
    b) Using TF-IDF vector to represent each document and clustering them by Temporal Markov, which is a modified version of Markov Clustering [Van Dongen, 2000] with temporal constraints. Each cluster contains articles published in similar dates with close contents so that sub-events are identified.
    c) Assigning a date that is mentioned for the most times in the source documents for each cluster.
    d) Ranking all the clusters by its importance, which is proportion to the mentioned times of the assigned dates in each cluster.
2. In the second stage, the system runs a state-of-the-art extractive Multi-document Summarisation method, CentroidOPT [Ghalandari, 2017], to generate summaries for each cluster in

Raw Input	Annotated Output
	<?xml version="1.0"?> <!DOCTYPE TimeML SYSTEM "TimeML.dtd"> <TimeML>
The firm 's shares continued to fall on Monday , after previous trading had seen about $ 20bn wiped from its market value since the accident .	The firm 's shares continued to fall on <TIMEX3 tid="t1" type="DATE" value="2010-05-03">Monday</TIMEX3> , after previous trading had seen about $ 20bn wiped from its market value since the accident .
UK markets were closed on Monday , but BP 's Frankfurt-listed shares opened 8 % lower before clawing back some of the losses .	UK markets were closed on <TIMEX3 tid="t4" type="DATE" value="2010-05-03">Monday</TIMEX3> , but BP 's Frankfurt-listed shares opened 8 % lower before clawing back some of the losses . </TimeML>

Table 3.1: An example using Heideltime to annotate a news article published in 3rd May 2010.

order of its importance. All the summaries will be ordered by date to form a timeline. If two summaries have the same assigned date, the later generated one (with less importance) will be discarded.

The following subsections introduce some of the original tools in CLUST, which are still preserved in my system.

3.1.1 Date Tagging

Tagging unstructured date expression in the raw text has always been a complex problem. It does not only require the system to recognise the string including day, month and year but also need to process the relative expression of time tags, e.g. *Monday, yesterday*. The advanced time tagging tool, Heideltime [Strötgen and Gertz, 2013], can solve this problem conveniently, thus preventing us from reinventing the wheel.

Heideltime is a multilingual temporal tagger. It can extract temporal expression and normalise them into TIMEX3 [Saurí et al., 2006] format, as shown in Table 3.1. Therefore, after all the documents are tagged, date expressions can be extracted by analysing the annotated document via XML Tree tools.

3.1.2 Event Detection

Another pivotal step in the first stage is to extract events from a heap of news articles. Undoubtedly, articles describing the same event should be mapped into close points in the semantic space. Because of this, the clustering algorithm can be applied to group these articles so that events

can be detected. The number of events should be determined automatically instead of being set manually. Among all the clustering algorithms, Markov Clustering [Van Dongen, 2000] and Affinity Propagation [Frey and Dueck, 2007] are the most advanced and widely applied. So I tested their performance on the chosen dataset to select the more proper clustering algorithm for the task, whose result is shown in 4.4.2.

In addition, there should be some conditions on the clustering algorithm. Articles published close to each other are more likely to describe the same event. In contrast, articles whose publish dates are far from each other, even if they are close in semantic space, should be considered to describe similar but different events. On top of clustering results, all the clusters will be ranked under a specific metric that quantifies the importance of each event.

3.2 The proposed method's workflow

In this project, the two-stage workflow is kept. However, some components are replaced by more advanced tools to improve performance. And the interaction process is integrated with the system as the main augmentation. The whole workflow is shown in Figure 3.1.

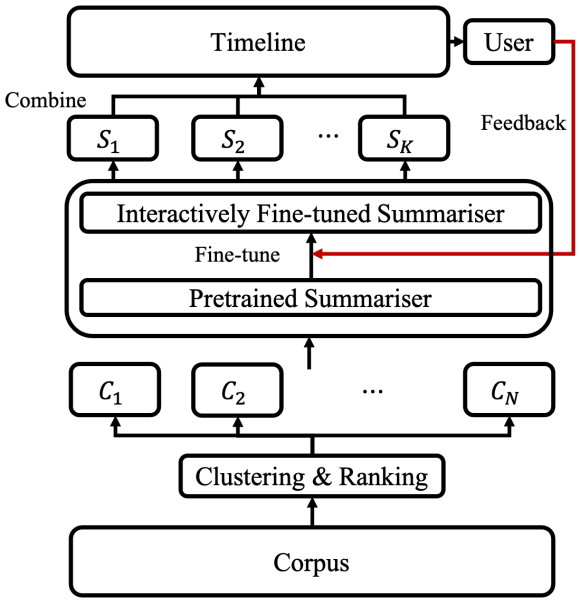

Figure 3.1: The workflow of the proposed Timeline Summarisation system [Ye and Simpson, 2021].

Firstly, instead of using TF-IDF, each sentence in the documents will be embedded by Distil-RoBERTa [Reimers and Gurevych, 2019] into a 768-d dense vector. Then the average vector of all sentences is computed to represent the corresponding document. Therefore, each document is mapped into a 768-d dense vector as well. Dense vectors can capture more information compared to bag-of-words models, thus improving the performance of clustering algorithms.

To further improve the clustering results, Markov Clustering is replaced by Affinity Propagation [Frey and Dueck, 2007], which is more suitable for clustering high-dimensional data with multiple categories.

As for the summarisation stage, to solve the defects of extractive methods, the former extractive summariser is substituted by an abstractive summariser. It works the same way as before, running on clusters by order of their importance, producing high-quality abstractive timelines until their length reaches a limit, which is manually set to be the same as the corresponding reference timeline's length. Every time the draft timeline is generated, the user can review it and respond by providing keywords that should be focused or neglected and expressing preferences over the current and earlier version. The interaction will be used to update the reward function. And the Reinforcement Learning algorithm will fine-tune the summariser to fit the user's demands.

3.3 Reinforcement Learning-based Interactive Fine-tuning

The innovative part of the proposed timeline summarisation system is introduced in this section. The highlight of this part is that we defined an automatic reward function that can be updated by the user's input. Therefore, the interaction process enables the system to run a Reinforcement Learning algorithm, which has the consistent goal with the user, to tune the summariser, so that the output timelines can be tailored according to users demands. Furthermore, only a few interaction rounds are required to adjust the content of the timeline, which promises the system's efficiency.

3.3.1 Notations

In this section, we denote the generated summary as S. Tokens in the summary S are noted as t_i, i.e., $S = \{t_1, \ldots, t_{|S|}\}$. The given feedback is marked as M,N and P, representing required keywords, undesirable keywords and the preference over the prior draft versions, respectively. The cluster is noted as C, and all documents in the cluster are marked as $D = \{D_1, \ldots, D_{|C|}\}$. All the vectorized text is marked with an arrow on the top of it, such as $\vec{S}$ and $\vec{D}_i$. l represents the length of the reference timeline. The number of Interactive Learning rounds and the number of Reinforcement Learning episodes are noted as n and m.

3.3.2 Interaction

In this project, each document collection A is regarded as an independent timeline generation task. So every time processing a new task (a new bunch of source documents), the system will be re-initialised and cluster the source documents. The summariser will run on the top-l important clusters to form a timeline. The Interactive Learning process is shown in Figure 3.2 and is defined as follows.

1. Enumerate the top-l important clusters. Generate a summary for each cluster as an episode of Reinforcement Learning.
2. Compute the reward, value and loss functions according to the Algorithm 1. Back propagate the gradient and optimise the weights of the actor and the critic.
3. Present the draft timelines to the user. Receive the user's feedback and update the reward function. Start another round of Interactive Learning.

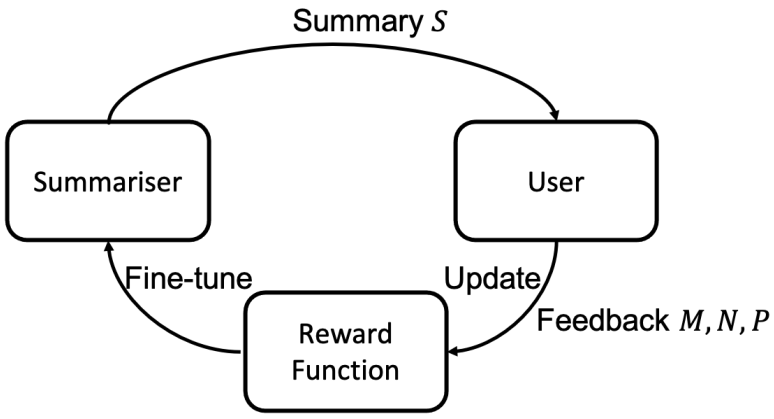

FIGURE 3.2. The interaction process of the system [Ye and Simpson, 2021].

In this process, users are allowed to preview draft timelines and provide different types of feedback, such as inputting keywords that must be included or excluded. Currently, the preference against the previous version of the timeline is acceptable to the system as well. Users can input 0 or 1 to give a preference label to two summary versions. Keywords represent the specific details that the user expects in the final output, while the preference can guide the system to learn some abstract and conceptual advantages which are contained in the preferred version. This feedback enables the system to renew its evaluation system, the reward function in the reinforcement learning phase. Then the summariser can be fine-tuned through several Reinforcement Learning episodes and generate a new timeline summary to start another round of interactive learning. The system does not require many rounds of interaction to generate tailored timelines for the user because the Reinforcement Learning process inside the Interactive Learning involves many episodes and is quite efficient.

3.3.3 Reinforcement Learning Phase

The feedback received in the interaction phase will be used to fine-tune the summarisation model. However, this human-readable feedback cannot be utilised directly by computer programs. Therefore, Reinforcement Learning is applied in this stage as it is capable of utilising the feedback without complex processing.

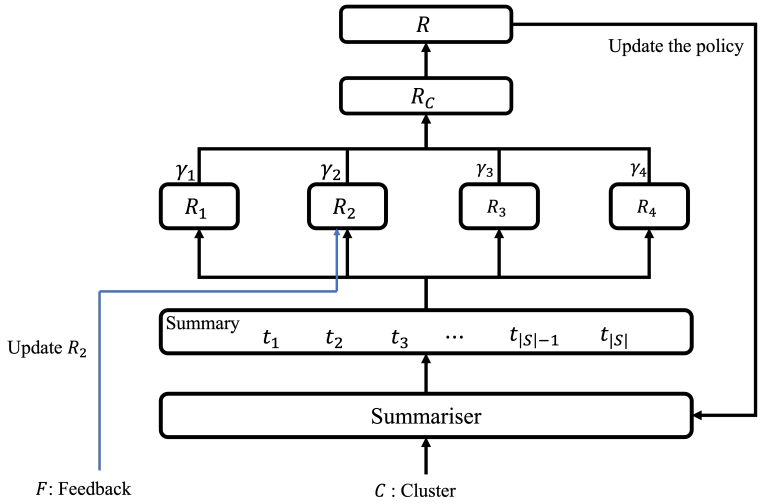

FIGURE 3.3. A view of the proposed Reinforcement Learning method [Ye and Simpson, 2021].

In general, a Reinforcement Learning algorithm manages to learn a policy by which intelligent agents could receive the highest reward through long-term interaction with the given environment. In my project, critical components of the Reinforcement Learning method are defined as below.

Agent, Action and State In Reinforcement Learning, agents are responsible for perceiving the state and properly acting to it. When applying Reinforcement Learning methods on Natural Language Generation tasks, the language model is naturally fit to be defined as the agent because the text generation task is usually processed as a sequential classification task. The procedure of the model choosing the next token can be regarded as making an action. In this way, the vocabulary size decides the dimension of the action space. And the state space is hence discrete. Its dimension is usually manually set as 512, which is the longest length of the output. In this project, PEGASUS [Zhang et al., 2019] is chosen because of its state-of-the-art summarisation ability. It will run on the top-l important clusters to learn how to adjust the output probability distribution to fit the user's demand.

Policy In Reinforcement Learning algorithms, agents follow the learned policy to act. In NLG tasks, the learned policy is often presented as a conditional possibility distribution over all tokens. In this project, the policy is constructed and calculated by the neural language model, Transformer [Vaswani et al., 2017]. The last hidden state computed by the last decoder layer is often regarded as a contextual tensor. In order to predict the next token, there is a linear layer on the top of the last decoder layer mapping the last hidden state to the output logits. Then the probability distribution over all tokens can be computed by a softmax layer so that we can

generate the next token according to the distribution. To maintain the exploration-exploitation trade-off, we sample the next token from the distribution rather than select the highest possibility one. Furthermore, when using Reinforcement Learning to fine-tune a transformer, instead of the whole model, it is often the linear layer's weights (also called the language head) that will be tuned, which is highlighted in red in Figure 3.4. We process the fine-tuning in the same way in the proposed system. The reason is various. First, we chose PEGASUS_Multi-News [Zhang et al., 2019] as our basis model, which has been tune on multiple news datasets that are similar to our corpora. Some prior works shown that the benefit of complete fine-tuning is not always large [Peters et al., 2019]. Some other works received success on only fine-tuning the last linear layer [Mesgar et al., 2020a]. In addition, fine-tuning the whole model is accessible but quite expensive. Therefore, we only tune the language head of the Transformer.

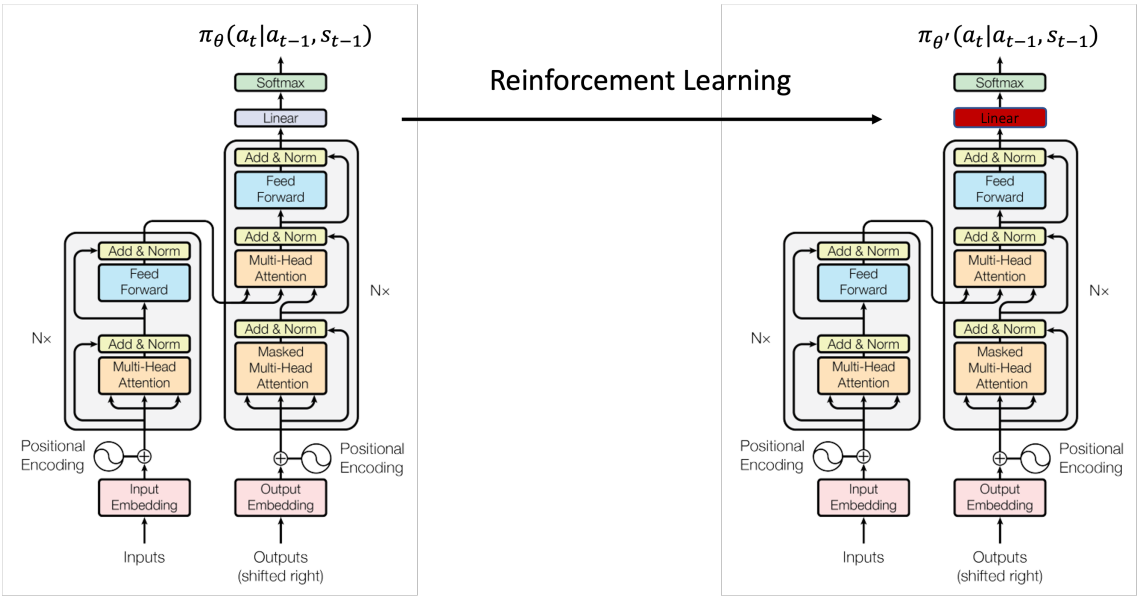

FIGURE 3.4. The system learns a new policy via Reinforcement Learning.

Reward Our model is expected to generate *factually consistent*, *topically coherent* and *linguistically fluent* timeline summaries. Thus, A compound reward function is proposed to quantify the evaluation of these various aspects of the system output. The reward function is consists of 4 sub-reward functions.

- R_1 contributes to the *factual consistency* of the generated timeline summaries. Consider the agent is processing the cluster C, of which the source document set is D and the corresponding output is S. The sentences in the system output S and will be embedded into 768-d dense vectors respectively by a sentence-transformer [Reimers and Gurevych, 2019], paraphrase-DistilRoBERTa-v1. And the whole output will be represented by the average

vector of these sentence vectors. Then, each source document in D will be processed in the same way and represented by its corresponding average vector $\vec{D}_i$. The cosine similarity is calculated between each pair of $\vec{S}$ and $\vec{D}_i$. The average cosine similarity is taken as the reward of factual consistency. Prior works shown that cosine similarity between two contextual vectors provided a reliable measure for *factual consistency* [Zhang, 2019].

$$R_1 = \sum_{D_i \in D} \cos(\vec{S}, \vec{D}) \tag{3.1}$$

- R_2 enforces the system to generate summaries that are topically coherent with the user's feedback. It updates when the user sends new feedback to the system. It is a weighted sum of three terms. The first term of P is the quantified score generated from the preference over the current output and prior versions. And $score(\cdot)$ is a ranking function learned from a group of pairwise preference labels via a random utility model [Thurstone, 2017, Mosteller, 2006]. The second and the last term are the sub-rewards of topical coherence between the system output S and the key phrases that should be included and excluded, marked as M and N respectively. They take the cosine similarity as rewards, the same as R_1. $w_1, w_2,$ andw_3 are weights which are manually chosen and sum up to 1.

$$R_2 = w_1 \cdot score(S,P) + w_2 \sum_{m_i \in M} \cos(\vec{m}_i, \vec{S}) + w_3 \sum_{n_i \in M} \cos(\vec{n}_i, \vec{S}) \tag{3.2}$$

To be more specific about the score function, Thurstone [2017] proposed a random utility model:

$$p(P(s_1, s_2) | f(s_1), f(s_2)) = \Phi(z)$$

where $P(\cdot)$ is the preference label given by the user over two summaries s_1 and s_2, Φ is the cumulative distribution function of the standard normal distribution which is also known as a probit likelihood, f is the score function to learn from the preference label set P, and $z = \frac{f(s_1) - f(s_2)}{\sqrt{2\sigma^2}}$. For brevity, we denote the score function as $score(S,P)$ in equation 3.2.

- R_3 is the linguistic fluency score. It is used to evaluate the text quality of the output as a summary. $N(\cdot)$ is the negative log-likelihood loss, calculated by the original PEGASUS without any tuning. α is the maximum of N, which is used to normalise the sub-reward.

$$R_3 = \frac{\alpha - N(S)}{\alpha} \tag{3.3}$$

- R_4 evaluates the degree of unigram repetition in the generated text. As a known fact, Reinforcement Learning agents in NLG tasks sometimes learn to generate repeated tokens to trick for high rewards. Therefore, this sub-reward function is necessary to penalise the repeated unigrams.

$$R_4 = 1 - \frac{\#repeated\_tokens}{\#tokens} \tag{3.4}$$

R_1 and R_2 focus on the content of the output. They encourage the system to generate summaries containing the right and important information, while $R3$ and $R4$ contribute to the language quality of the text. Therefore, our expectation of the system is transformed into the reward function of the Reinforcement Learning, computed as Equation 3.5.

$$R = \gamma_1 R_1 + \gamma_2 R_2 + \gamma_3 R_3 + \gamma_4 R_4 \tag{3.5}$$

Information redundancy is not considered in the proposed reward function. Because the output text length is limited under 256 words. Text length limits redundancy because the summary has to use the limited number of characters well to ensure factual consistency. However, it is still to explore the influence of the *information redundancy* sub-reward function.

Algorithm In Reinforcement Learning, policy approximation algorithms are widely applied. They often quantifies the performance of the system as a scalar $J(\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the parameter vector of the policy function $\pi(a|s,\boldsymbol{\theta})$. Due to that, Reinforcement Learning methods are seeking to maximise the performance of the system, $\boldsymbol{\theta}$ can be updated by optimisation methods, such as gradient ascent:

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} + \alpha\widehat{\nabla J(\boldsymbol{\theta_t})} \tag{3.6}$$

Policy parameterization has several advantages comparing to action-value methods [Sutton and Barto, 2018]:

1. Parameterizing policies can approach deterministic policies, whereas there is always an ϵ probability to choose random actions in the ϵ-greedy methods.
2. Parameterizing allows the policy to be stochastic.
3. Sometimes the policy function is simpler to approximate than the action-value function.
4. Policy parameterization helps to inject prior knowledge into the system.

Actor-critic [Mnih et al., 2016], a policy method, is applied to enhance the summaries' quality of all the aspects mentioned above in this project. Differing from the general policy gradient methods, **actor-critic** learns approximations to not only the policy function, but also the state-value function as well. It introduces the advantage function δ (Equation 3.7) to reflect whether the value of an action is better or worse than the average level under the current state. Optimising the parameters $\boldsymbol{\theta}$ helps avoid choosing those actions with negative influence so that it reduces the variance of the policy distribution. Although value approximation cannot come along without bias, the algorithm can be extended to an n-step return function version to modulate its extent. The pseudo-code of the one-step actor-critic method is shown in Algorithm 1 below.

$$\delta = R + \gamma \hat{v}(S^{'}, \boldsymbol{w}) - \hat{v}(S, \boldsymbol{w}) \tag{3.7}$$

Algorithm 1 One-step Actor-Critic

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s, \boldsymbol{w})$
  Parameters: Learning rate $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\boldsymbol{w}} > 0$
  Initialise $\boldsymbol{\theta}$ and $\boldsymbol{w}$
  repeat
    Initialise the first state S
    $I \leftarrow 1$
    while S is not terminal **do**
      $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
      Take the action A, observe the new state $S^{'}$ and the reward R
      $\delta \leftarrow R + \gamma \hat{v}(S^{'}, \boldsymbol{w}) - \hat{v}(S, \boldsymbol{w})$
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + \alpha^{\boldsymbol{w}} \delta \nabla \hat{v}(S, \boldsymbol{w})$
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
      $I \leftarrow \gamma I$
      $S \leftarrow S^{'}$
    end while
  until no episode remains

In this project, I choose **PEGASUS-Multi_news** [Zhang et al., 2019] as the **actor**, because it is pre-trained on Multi_news dataset [Fabbri et al., 2019] which is the closest version to my experiment datasets among all the available pre-trained weight files. As for the **critic**, it should receive a 512-d state vector which contains the token indexes in each position and output a scalar as the prediction of the value function. Here we use a 3-layer fully connected neural network with ReLU activation function as the approximation of the value function, which can be written as below:

$$\hat{v}(s_t) = W_2 a(W_1 s_t + b_1) + b_2$$

where $a(\cdot)$ is the ReLU funtion. Its architecture is shown in Figure 3.5. The Reinforcement Learning algorithm has two aims:

1. To tune part of the weights of **PEGASUS-Multi_news** to adjust the probability distribution to generate needed outputs.

2. To train a nueral network to approach an evaluator for the action a_t and the state s_t at the timestep t.

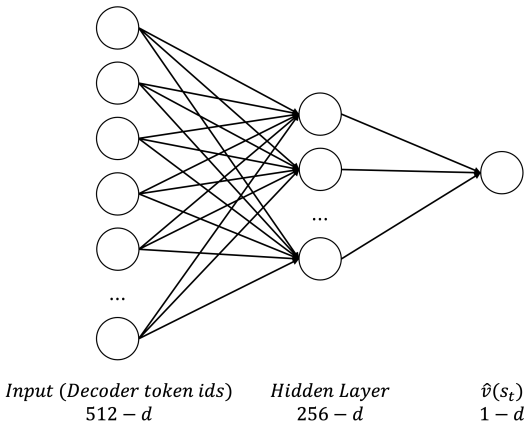

FIGURE 3.5. The architecture of the critic.

Training Now all the necessary components of a Reinforcement Learning algorithm are well defined. The next step is to define the training process.

For each cluster, the system will sample m summaries over its document collection. Each summary is an episode. The Reinforcement Learning algorithm traverses all episodes to learn a better policy. Here is the specific learning process:

- The actor predicts the next token according to the input documents and the current state S (previously predicted output). The new state $S^{'}$ is concatenated by the current state and the predicted next token.
- The environment returns the reward of the new state R. And the critic computes the approximate value of the new state $v(\hat{S}')$.
- The advantage of current time step can be calculated by Equation 3.7.
- The loss function and its gradient can be computed to optimise the actor and the critic. The policy is updated.

In practice, the optimisation can be done at one time at the end of the episode to reduce approximation bias in some degree. This completes the presentation of my TLS method, which builds on the method of Ghalandari and Ifrim [2020] and introduces abstractive summarisation with Pegasus. The next chapter will present the evaluation of this approach.

CHAPTER 4

EVALUATION

4.1 Datasets

Recall the difficulties of generating timelines mentioned in 1.1.2 and the challenges mentioned in 1.2, it is hard to build a timeline summarisation dataset from scratch, even if considering crowdsourcing reference timelines. After reading several related paper, I choose two benchmark datasets built by prior works, Timeline17 [Tran et al., 2013, Binh Tran et al., 2013] and Crisis [Tran et al., 2015] [1].

Timeline17 consists of 19 manual-created timelines and their associated news articles belonging to 17 news topics. Each timeline and its news articles belong to one source news agency, such as BBC, Guardian, CNN and FoxNews. Timelines only published by renowned news agencies with exact dates, like *11 Feb 2011*, are selected. Finally, 19 different timelines in different topics make up this dataset.

Crisis is made up of 4 crisis topics with respect to Egypt, Libya, Yemen and Syria. It was automatically collected by using Boilerpipe toolkit and some cleaning rules on top. Timeline summaries are written by experts from famous news agencies.

4.1.1 Dataset Characteristics

Ghalandari and Ifrim [2020] have done some basic investigation about the statistic characteristics of the two datasets, as Table 4.1, 4.2 and 4.3 shown.

[1] The two datasets can be downloaded here.

Dataset	#Topics	#Timelines	Ave. #Docs	Avg. #PubDates	Avg. Duration (days)
Timeline17	9	19	508	124	212
Crisis	4	22	2310	307	343

Table 4.1: Statistic characteristics of the two datasets (i), cited from [Ghalandari and Ifrim, 2020].

Dataset	Avg. Ref. Timeline Length	Avg. Comp. Ratio (dates)	Ave. Comp. Ratio (sents)
Timeline17	36	0.0117	0.43
Crisis	29	0.0005	0.11

Table 4.2: Statistic characteristics of the two datasets (ii), cited from [Ghalandari and Ifrim, 2020].

Dataset	Avg. Date Cov. (published)	Avg. Date Cov. (mentioned)
Timeline17	81%	93%
Crisis	90%	96%

Table 4.3: Statistic characteristics of the two datasets (iii), cited from [Ghalandari and Ifrim, 2020].

In these tables, *#PubDates* refers to the number of days that an article was published in collection A. *The Comp. Ratio (sents)* is the number of sentences in golden timelines divided by the total number of sentences in A. And *The Comp. Ratio (dates)* w.r.t. the number of dates in ground-truth timelines divided by *#PubDates*. *Date Cov.* is counted by using publication dates in A, or by mentioned dates in articles of A. Mentioned dates have a higher cover rate than published dates on both datasets, implying making use of these publication dates. **Timeline17** has longer and more specific reference timelines comparing to **CRISIS**, meanwhile **CRISIS** has much more source documents.

4.1.2 Experimental Settings

To better comparing the proposed system and the baseline and eliminate the influence of pre-processing methods, I keep the main experimental settings the same as the baseline, CLUST [Ghalandari and Ifrim, 2020].

- Each dataset is splited into multiple topics with one reference timeline. If a topic has multiple reference timelines from different sources, it will be regarded as multiple different tasks. The final results in the evaluation are the averages over tasks.
- Two tasks in Timeline17 are picked out as the development set to tune the hyperparameters manually.
- Articles in the collection A are removed if their publication dates are either earlier than the start date of the reference timeline or later than the end date.

- Headlines of articles are not used.
- There is a constraint for the output size: The system timelines' length are no longer than the reference timelines'.

4.2 Metrics

4.2.1 ROUGE

ROUGE [Lin, 2004] is the most widely applied evaluation metric in Natural Language Generation tasks, especially in the automatic summarisation field. Its basic idea is to compute the percentage of n-grams that appear both in the system and reference summaries, i.e. the recall between the system and reference summaries, which is calculated as Formula 4.1.

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

ROUGE-1 and ROUGE-2 are the most common metrics in all types of ROUGE-like evaluation metrics. They are concerned about how much relevant information is contained in the system summaries without considering the rate of correct information in generated timelines. Because many prior works focused on extractive timelines rather than abstractive timelines and all the information was copied from input documents, the precision did not worth any concern unless the source was unreliable. However, in this project, we manage to utilise an abstractive neural summariser to generate timelines instead of copying sentences into output. The ideal output will require both high recall and high precision. Thus ROUGE-N F1 becomes the most suitable metrics to apply in the project.

4.2.2 Adapted ROUGE for Timeline Summarisation Evaluation

Although, as a special automatic summarisation task, it is natural to choose ROUGE as the evaluation metric, the temporal information inside timelines is neglected by them, which should definitely be not since it is as important as the content of the summaries. Therefore, some ROUGE-like metrics, which are introduced below, are adapted to Timeline Summarisation tasks to evaluate the rigorousness of temporal aspects and integrate it with the content evaluation [Martschat and Markert, 2017].

Concatenation-based ROUGE applies ROUGE to timelines evaluation brutally. It discards all the temporal information of the timelines to be evaluated [Takamura et al., 2011, Yan et al., 2011a, Nguyen et al., 2014, Wang et al., 2016]. It still does not fix the issue of penalising different dates of the same events.

Date-agreement ROUGE is another brutal evaluation metric but with more principles. It only considers the dates that appeared in the reference timelines and take the average ROUGE score of the corresponding summaries as the result. If there is no date matching the reference date, a 0 will be returned. It fixes the date evaluation issue in some way, but the penalisation is too sharp. For those long-term events, the system may assign slightly different dates compared to ground-truth dates to the sub-events. This kind of slight mistake should not receive 0 as its evaluation. Therefore, a more flexible metric is necessary.

Alignment-based ROUGE makes efforts to compare the same and similar events based on the idea of aligning the dates. It manages to find the best mapping that assigns each date in the reference timeline to a date in the system output. The best mapping will lead to the minimal error or *date distance* which is defined as Formula 4.2. The ROUGE score is then computed on aligned summaries and averaged as the final result.

$$c_{d_r,d_s} = 1 - \frac{1}{|d_r - d_s| + 1} \tag{4.2}$$

Alignment-based ROUGE strikes a balance between date accuracy and flexibility. So it is the best evaluation metric for this project. Moreover, according to the issue I mentioned about only applying recall as the evaluation metric in 4.2.1, it is **Alignment-based ROUGE F1** that perfectly suits to evaluate the abstractive timelines.

4.3 Simulated Experiments

The interactive timeline summarisation system can retrieve key dates and information from source news articles and generate tailored timelines that are consistent with the user's interests. Although there is adapted ROUGE to evaluate the timelines' content, the interaction effect is still too ambiguous to quantify. Therefore, we design a simulated experiment and manage to simulate the interaction to reflect its outcome via adapted ROUGE scores.

4.3.1 Simulated Interaction

The essential part of the interaction simulation is to provide keywords and preferences to the system like a real user. Therefore, we applied TF-IDF on each summary. Words with top-k high TF-IDF value will be collected as keywords that should appear in the output and fed to the system in the interactive learning process. Meanwhile, words with top-k low TF-IDF value will be treated as keywords that should be excluded. And these keywords are randomly fed to the system in an interaction round.

4.4 Results

4.4.1 Reproducing the Results of Prior Work

Due to the keywords being extracted from the reference timelines and the chosen evaluation metric, adapted ROUGE is based on counting n-gram appeared in both source documents and output. The generated timelines should have a higher rouge score than baselines. So the first step is to reproduce the baseline. Since the workflow of this system mainly follows *CLUST* [Ghalandari and Ifrim, 2020], and it is the most recent published state-of-the-art work of this field, here I choose it to be the baseline as well for comparison.

We reused its code[2] and tested its performance in my local environment. The baseline only selects summary sentences from the first k sentences in each source document. Although the system aims at generating abstract timeline summaries, it is still worth exploring the parameter's effects on the system performance. Because the summariser has a limited length on input, so considering the average length of sentences in the datasets, we only test k from 4 to 10 and the whole document.

Dataset	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	alldoc
T17	0.0692	0.0728	0.0733	0.0733	0.0742	0.0755	0.0743	0.0767
Crisis	0.0575	0.0596	0.0591	0.0616	0.0613	0.0613	0.0631	0.0548

Table 4.4: Replicated results evaluated by Alignment-based ROUGE-1 F1 score.

Dataset	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	alldoc
T17	0.0163	0.0167	0.0168	0.0176	0.0178	0.0185	0.0172	0.0172
Crisis	0.0111	0.0128	0.0119	0.0133	0.0134	0.0138	0.0149	0.0112

Table 4.5: Replicated results evaluated by Alignment-based ROUGE-2 F1 score.

From Table 4.4 and 4.5, the performance of the baseline, *CLUST*, shows that the first 10 sentences can draw an outline of the event. It implies that even if given long source documents, the proposed system has the potential to generate high-quality timelines only using the first 10 sentences, which fix the issue that the input document is too long for a transformer's tokenizer to embed.

4.4.2 Clustering Analysis

In this project, the workflow still follows the prior state-of-the-art work, *CLUST*, but there are some significant changes made to improve the performance aside from integrating the interaction process. As mentioned in 3.2, when detecting events, I replaced the Temporal Markov with

[2]The code repository is here.

Affinity Propagation along with substituting the word embedding method, DistilRoBERTa for TF-IDF. The relevant characteristics of their performance are shown in Table 4.6 and 4.7.

Dataset	Tempral Markov	Affinity Propagation
Timeline17	148.059	50.897
Crisis	7997.250	3713.937

Table 4.6: The average number of clusters obtained by different methods.

Dataset	Tempral Markov	Affinity Propagation
Timeline17	38.684	29.171
Crisis	66.41	45.37

Table 4.7: Average timeline length generated by different approaches.

Dataset	Tempral Markov			Affinity Propagation		
	Precision	Recall	F1	Precision	Recall	F1
Timeline17	0.482	0.418	0.447	0.519	0.294	0.376
Crisis	0.275	0.275	0.275	0.378	0.286	0.326

Table 4.8: Average precision, recall and F1 for date selection.

Temporal Markov generated way more clusters than Affinity Propagation. Due to the system only summarising the top-n important clusters, the type of summarisers does not affect the length of the generated timelines. Therefore, comparing the average cluster size and the average timeline length, Affinity Propagation gave the clustering results more proper granularity. Because the number of input documents is constant, clustering results with too small granularity means that source documents describing the same sub-event are split into several groups. According to the workflow defined previously in 3, some of the summaries will be discarded. It will result in loss of information for some single sub-events, i.e., those sub-events are not well detected. And according to Table 4.8, Affinity Propagation has a higher precision on date selection, which means that Affinity Propagation is more capable of precisely identifying important dates rather than selecting a list of dates to cover them. Therefore, Affinity Propagation is more suitable for detecting events in given source documents.

4.4.3 Summarising without Reinforcement Learning

Another important modification is that we introduced a neural summarisation model, PEGASUS [Zhang et al., 2019], to generate abstractive timelines. To verify its summarisation ability, we tested its performance of summarising different first-k sentences of source documents. The results are shown in Table 4.9 and 4.10.

In the two tables above, *M_PGSS* represents running PEGASUS on the clusters of Temporal Markov, and *AP_PGSS* means feeding the clusters of Affinity Propagation to PEGASUS.

	Dataset	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	alldoc
CLUST	**T17**	0.0692	0.0728	0.0733	0.0733	0.0742	0.0755	0.0743	0.0767
	Crisis	0.0575	0.0596	0.0591	0.0616	0.0613	0.0613	**0.0631**	0.0548
M_PGSS	**T17**	0.0795	0.0784	0.0811	0.0823	0.0796	0.0794	0.0807	0.0721
	Crisis	0.0422	0.0476	0.0396	0.0390	0.0383	0.0374	0.0373	0.0330
AP_PGSS	**T17**	0.0787	0.0818	**0.0829**	0.0802	0.0796	0.0786	0.0731	0.0718
	Crisis	0.0474	0.0512	0.0488	0.0487	0.0513	0.0500	0.0493	0.0468

Table 4.9: Replicated results evaluated by Alignment-based ROUGE-1 F1 score.

	Dataset	$k=4$	$k=5$	$k=6$	$k=7$	$k=8$	$k=9$	$k=10$	alldoc
CLUST	**T17**	0.0163	0.0167	0.0168	0.0176	0.0178	0.0185	0.0172	0.0172
	Crisis	0.0111	0.0128	0.0119	0.0133	0.0134	0.0138	**0.0149**	0.0112
M_PGSS	**T17**	0.0182	0.0172	0.0182	0.0194	0.0160	0.0185	0.0156	0.0137
	Crisis	0.0079	0.0090	0.0092	0.0076	0.0073	0.0067	0.0071	0.0055
AP_PGSS	**T17**	0.0162	0.0178	**0.0195**	0.0181	0.0176	0.0167	0.0156	0.0122
	Crisis	0.0090	0.0119	0.0098	0.0095	0.0102	0.0090	0.0096	0.0087

Table 4.10: Replicated results evaluated by Alignment-based ROUGE-2 F1 score.

The highest scores are shown in boldface. On dataset Timeline17, PEGASUS with Affinity Propagation achieved the highest score when $k = 6$. However, on dataset Crisis, *M_PGSS* and *AP_PGSS* perform much worse than *CLUST*. And the maximums of Alignment-based ROUGE1 and ROUGE2 are both achieved by *CLUST* when $k = 10$. It's because that **Crisis** has much more documents for each task. The input vector must be truncated, and information in the back of the document collection will be discarded. This also explains the decreasing trend of the two PEGASUS methods.

According to the results, it can be told that when the source document collection is not extremely large, PEGASUS is a better summariser compared to the original applied extractive summarisation method CentroidOpt [Ghalandari, 2017].

4.4.4 Simulated Reinforcement Learning-based Interactive Timeline Summarisation System

Simulated RL-based Interactive Timeline Summarisation	Timeline17		Crisis	
	AR1 F1	**AR2 F1**	**AR1 F1**	**AR2 F1**
Round=1	0.0811	0.0184	0.0434	0.0088
Round=2	0.0832	0.0193	0.0460	0.0093
Round=3	0.0799	0.0212	0.0489	0.0113

Table 4.11: Results of simulated Reinforcement Learning-based interactive timeline summarisation system ($k = 6$). *Round* represents how many times that the agent sampled an episode from each cluster. Each round means that the agent learned multiple episodes over all clusters.

Table 4.11 tells that the Reinforcement Learning algorithm succeeds in improving the performance of the timeline summarisation system on two benchmark datasets. It even outperformed CLUST on Timeline17 under both metrics. Although the system still performed worse than CLUST on Crisis, it did have improvement compared to its basis model, especially for the Alignment-based ROUGE2 F1.

The system had a huge increase of Alignment-based ROUGE2 F1 on both datasets, which means that keywords extracted from the reference timelines tuned the summariser's policy. The Reinforcement Learning algorithm was working towards the goal we set in each interaction time. So those phrases appear more along with the episode increases.

As mentioned in 2.3.1, high ROUGE scores only mean that the system output and the reference text contain the same information nuggets. It still needs extra efforts to judge whether the two are supporting or contradicting each other in meaning. So here we take some samples from the data which are shown in Figure 4.1 and 4.2.

Date	Without RL	Episode=1
2010-01-12	Wyclef Jean is on his way to Haiti to help victims of the country's devastating earthquake. \u201cThey need our help\u2014please help if you can,\u201d the Haitian-born rapper tweeted.	Bill Clinton has arrived in Port-au-Prince to offer his help in the aftermath of the Haitian earthquake. The former president, who is currently the UN special envoy to the country, met with Haitian President Jean-Bertrand Aristide and other officials today, the AP reports.
2010-01-19	The death toll from the Haiti earthquake now stands at 50,000, according to Haitian officials, but that doesn't mean it's a done deal.	As the scope of the Haiti disaster becomes clear, one thing is becoming clear: The nation's infrastructure is in such bad shape that it could take years to rebuild, reports the New York Times.
2010-01-21	Simon Cowell is planning to put out a charity single to raise funds for Haiti, and he\u2019s got Cheryl Cole on board.	Relief workers in Haiti are still struggling to get food, water, and medical assistance to survivors as the death toll from the country's devastating earthquake rises to at least 200,000.

FIGURE 4.1. Some generated timeline examples from the news topic, Haiti earthquake (i).

Text worthy of attention is highlighted in different colours in the two figures. With episodes increasing, the timelines pay more attention to reporting victims, relief workers and rescue teams than celebrities in the disaster, which is more important to readers. And it can be found that the summariser learned a certain sentence pattern, *As the scope of the Haiti disaster becomes clear*, which implies this sentence may contain multiple keywords and have a high reward, thus being generated over and over again in the output.

Another exciting finding is that some format control characters that were not filtered out when building the dataset contaminated the first version of the output. However, with more rounds of learning, those characters are abandoned. And the system timeline seems to be highly fluent.

Date	Episode=2	Episode=3
2010-01-12	As the scope of the Haiti earthquake becomes clear, one thing is becoming clear: The toll is likely to be in the tens of thousands, reports the New York Times.	Relief workers from around the world are en route to Haiti to help in the wake of the country's devastating earthquake. The first foreign rescue team is en route to Port-au-Prince from the Dominican Republic, the BBC reports.
2010-01-19	The death toll from last week's earthquake in Haiti now stands at 100,000, according to the country's president, who says the real toll is closer to 200,000.	The death toll from the Haiti earthquake now stands at 50,000, according to the country's interior minister, and that number is sure to rise.
2010-01-21	As the scope of the Haiti disaster becomes clear, one thing is becoming clear: The toll is going to be a lot higher than previously thought.	Relief workers in Haiti are still struggling to get food, water, and medical supplies to survivors, and the country's interior minister says 100,000 people will initially be sent to 10 settlements near Port-au-Prince.

FIGURE 4.2. Some generated timeline examples from the news topic, Haiti earthquake (ii).

With further exploration, we found that the agent sought a high reward by various means, which made the weights of the sub-reward functions particularly important. We tested several settings of sub-reward weights, as shown in Table 4.12. For example, we set $r_2 = 1$ and $r_{1,3,4} = 0$. The system was tuned to generate a keyword sequence instead of a fluent sentence. And we found that r_3 should be pretty small, or the summariser will learn to generate repeated '-'. The reason is that the '-' sequence receives a zero-like loss in PEGASUS, which results in very a high linguistic fluency reward R_3 even if it is not a sentence.

Weights	**Output**
$[r_1, r_2, r_3, r_4] = [0, 1, 0, 0]$	'- workers workers workers workers workers workers...'
$[r_1, r_2, r_3, r_4] = [0.3, 0.3, 0.15, 0.25]$	'- - - - - - - - - - - - - - ...'

Table 4.12: Different reward weights and corresponding output.

4.4.5 Run time and Hyperparameters

CPU	**GPU**	**RAM**	**OS**
Intel(R) Xeon(R) CPU @ 2.20GHz	NVIDIA Tesla P100	13GB	Ubuntu 18.04.5 LTS

Table 4.13: Experimental environment (Google Colab Pro).

The proposed system involves several different neural network models, such as sentence-BERT and Transformer. So, comparing it to the baseline, our system requires a GPU to finish the tensor computation and much more RAM to store the neuron variables. When running our system on Google Colab Pro (Environment is shown in Table 4.13), it consumes about 2 minutes

to generate a sample on a cluster. The total time that running the system would consume depends on the size of input document collection and some hyperparameters.

The main hyperparameter affecting running time is the number of episodes, which is tested in the last subsection. Other hyperparameters contribute to the system performance in different ways, as described in previous sections. All hyperparameters are listed in Table 4.14.

Hyperparameter	**Value**	**Description**
state_size	512	The maximum length of a single generated summary
episodes	2	The number of episodes that the RL agent samples on a cluster
α	15	The normalisation factor of the sub-reward function $R3$
γ	0.99	The discount factor for future rewards in RL
lr	0.005	The learning rate of the optimisation step
$[r_1, r_2, r_3, r_4]$	$[0.20, 400, 0.05, 0.4]$	The weight of sub-reward functions designed by experience

Table 4.14: Hyperparameters.

CHAPTER 5

CONCLUSION

5.1 Contributions

The goal of the project is to refine abstractive timelines via Reinforcement Learning interactively. The quality of summaries is subjective, which relies on different users' distinctive demands. In general circumstances, those informational demands are abstract and challenging to quantify. The proposed interactive summarisation system is capable of evaluating how much these requirements are satisfied by computing cosine similarities of the word embedding of generated summaries and keywords. After each time of interaction, the Reinforcement Learning algorithm manages to tune the policy that instructs the summariser to generate tailored timelines.

As conclusion, my contributions are listed below:

- I reviewed and reproduced the current state-of-the-art extractive timeline summarisation method, CLUST.
- Following the workflow of CLUST, we replaced some components in its event detection stage and built a more precise pipeline to detect events in a bunch of source documents.
- I introduced Interactive Learning into the timeline summarisation system, which enabled the system to receive the user's feedback and present timelines containing more appealing contents.
- I implemented the interaction and learning process via Reinforcement Learning. And we defined a reward function which is summed up by four sub-reward functions evaluating topical coherence, factual consistency, language quality, and token repetition, respectively. The topical coherence sub-reward can update itself automatically according to the user's feedback.

- I implemented the policy-based Reinforcement Learning algorithm, actor-critic. It improved the learning efficiency of the system.
- I tested the performance of the timeline summarisation system on two benchmark datasets. The proposed system outperformed CLUST on Timeline17. On the other dataset, Crisis, applying the abstractive summariser PEGASUS decreased the ROUGE score of the system, but Interactive Learing pulled the performance back in some degree.

5.2 Project Status

This project aims to build an interactive timeline summarisation system that is capable of receiving the user's feedback and generating timelines with the user's favourable content. Since the system has outperformed the prior state-of-the-art method on Alignment-based ROUGE, and Table 4.1 and 4.2 shows that the content did adjust towards simulated feedback, which means the system achieved the goal. All the code can be found in my GitHub repository.

5.3 Implications and Future Direction

Although the proposed timeline summarisation system outperformed the prior state of the art, it is still imperfect and has potential for further improvements.

First, we used a pre-trained word embedding model without any fine-tuning to cluster source documents and calculate the score of generated summaries and informational requirements. There are some possibilities that we could tune the embeddings by training a classifier to classify events in a training set? However, the lack of training data is a big problem in the way there.

Secondly, detecting events in source documents only applied unsupervised clustering algorithms in both CLUST and the proposed system. Reinforcement Learning could be utilised regarding the date selection as the action to improve the precision of event detection or important date identification, which can also decide the timeline's length automatically. Then the system performance could be improved by combing two Reinforcement Learning processes.

In addition, there are some possible improvements on designing the reward function:

- Currently, both topical coherence and factual consistency are evaluated by cosine similarities. However, some metrics in Information Theory has been proven to be more effective as the evaluation metric of some summarisation tasks. So they could be integrated into the reward function to intensify the learning efficiency.
- For the language quality reward, I used the original pre-trained PEGASUS-Multi_news to compute the score. However, repetitive tokens and meaningless '-' have been a known problem of it. Applying a better language model to mark the generated summaries might be a better reward function.

- Although some prior works has shown that cosine similarity performed well on evaluating consistency of two contextual vectors, it is still worth exploring that whether there is a better method. Recent advances in Natural Language Inference enables us to employ a neural network to judge whether two sentences entails or conflicts to each other. It provides some chances to improve the performance of our system.

Last, human evaluation is still worthy even though the proposed system has outperformed state of the art on the benchmark dataset, Timeline17. Humans can evaluate text from different angles that programs can hardly do, such as the writing style and logical rigorousness. With human effort, the interactive timeline summarisation system could receive a more comprehensive evaluation. Another point is that human evaluation can help us to know that whether keywords are a good way for users to interact and how easy and quick it is for users to receive their expected results. Therefore, human evaluation is the next pivotal step for this system to improve.

BIBLIOGRAPHY

Cristina Barros, Elena Lloret, Estela Saquete, and Borja Navarro-Colorado.
Natsum: Narrative abstractive summarization through cross-document timeline generation.
Information Processing & Management, 56(5):1775–1793, 2019.

Richard Bellman.
Dynamic programming.
Science, 153(3731):34–37, 1966.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin.
A neural probabilistic language model.
The journal of machine learning research, 3:1137–1155, 2003.

Giang Binh Tran, Mohammad Alrifai, and Dat Quoc Nguyen.
Predicting relevant news events for timeline summaries.
In *Proceedings of the 22nd International Conference on World Wide Web*, pages 91–92, 2013.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio.
Learning phrase representations using rnn encoder-decoder for statistical machine translation.
arXiv preprint arXiv:1406.1078, 2014.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova.
Bert: Pre-training of deep bidirectional transformers for language understanding.
arXiv preprint arXiv:1810.04805, 2018.

Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev.
Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model, 2019.

Brendan J Frey and Delbert Dueck.
Clustering by passing messages between data points.
science, 315(5814):972–976, 2007.

Yang Gao, Christian M Meyer, and Iryna Gurevych.

April: Interactively learning to summarise by combining active preference learning and reinforcement learning.
arXiv preprint arXiv:1808.09658, 2018.

Yang Gao, Christian M Meyer, Mohsen Mesgar, and Iryna Gurevych.
Reward learning for efficient reinforcement learning in extractive document summarisation.
arXiv preprint arXiv:1907.12894, 2019.

Demian Gholipour Ghalandari.
Revisiting the centroid-based method: A strong baseline for multi-document summarization.
arXiv preprint arXiv:1708.07690, 2017.

Demian Gholipour Ghalandari and Georgiana Ifrim.
Examining the state-of-the-art in news timeline summarization.
arXiv preprint arXiv:2005.10107, 2020.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith.
Don't stop pretraining: adapt language models to domains and tasks.
arXiv preprint arXiv:2004.10964, 2020.

Sepp Hochreiter and Jürgen Schmidhuber.
Long short-term memory.
Neural computation, 9(8):1735–1780, 1997.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White.
Multilayer feedforward networks are universal approximators.
Neural networks, 2(5):359–366, 1989.

Muhammad Imran, Shady Elbassuoni, Carlos Castillo, Fernando Diaz, and Patrick Meier.
Practical extraction of disaster-relevant information from social media.
In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 1021–1024. International World Wide Web Conferences Steering Committee, 2013.

Muhammad Imran, Prasenjit Mitra, and Carlos Castillo.
Twitter as a lifeline: Human-annotated twitter corpora for nlp of crisis-related messages.
In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA). ISBN 978-2-9517408-9-1.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy.
Spanbert: Improving pre-training by representing and predicting spans.
Transactions of the Association for Computational Linguistics, 8:64–77, 2020.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut.
Albert: A lite bert for self-supervised learning of language representations.
arXiv preprint arXiv:1909.11942, 2019.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer.
BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.
CoRR, abs/1910.13461, 2019.
URL http://arxiv.org/abs/1910.13461.

Chin-Yew Lin.
Rouge: A package for automatic evaluation of summaries.
In *Text summarization branches out*, pages 74–81, 2004.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov.
Roberta: A robustly optimized bert pretraining approach.
arXiv preprint arXiv:1907.11692, 2019.

S. Lloyd.
Least squares quantization in pcm.
IEEE Transactions on Information Theory, 28(2):129–137, 1982.
doi: 10.1109/TIT.1982.1056489.

Sebastian Martschat and Katja Markert.
Improving rouge for timeline summarization.
In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 285–290, 2017.

Sebastian Martschat and Katja Markert.
A temporally sensitive submodularity framework for timeline summarization.
arXiv preprint arXiv:1810.07949, 2018.

Mohsen Mesgar, Edwin Simpson, and Iryna Gurevych.
Improving factual consistency between a response and persona facts.
arXiv preprint arXiv:2005.00036, 2020a.

Mohsen Mesgar, Edwin Simpson, Yue Wang, and Iryna Gurevych.
Generating persona-consistent dialogue responses using deep reinforcement learning.
arXiv e-prints, pages arXiv–2005, 2020b.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu.
Asynchronous methods for deep reinforcement learning.
In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

Frederick Mosteller.
Remarks on the method of paired comparisons: I. the least squares solution assuming equal standard deviations and equal correlations.
In *Selected Papers of Frederick Mosteller*, pages 157–162. Springer, 2006.

Kiem-Hieu Nguyen, Xavier Tannier, and Véronique Moriceau.
Ranking multidocument event descriptions for building thematic timelines.
In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1208–1217, 2014.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu.
Bleu: a method for automatic evaluation of machine translation.
In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

Matthew E Peters, Sebastian Ruder, and Noah A Smith.
To tune or not to tune? adapting pretrained representations to diverse tasks.
arXiv preprint arXiv:1903.05987, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al.
Language models are unsupervised multitask learners.
OpenAI blog, 1(8):9, 2019.

Nils Reimers and Iryna Gurevych.
Sentence-bert: Sentence embeddings using siamese bert-networks.
arXiv preprint arXiv:1908.10084, 2019.

Ehud Reiter.
A structured review of the validity of bleu.
Computational Linguistics, 44(3):393–401, 2018.

Cody Rioux, Sadid A Hasan, and Yllias Chali.
Fear the reaper: A system for automatic multi-document summarization with reinforcement learning.
In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 681–690, 2014.

Seonggi Ryang and Takeshi Abekawa.
Framework of automatic text summarization using reinforcement learning.
In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 256–265, 2012.

Roser Saurí, Jessica Moszkowicz, Bob Knippen, Rob Gaizauskas, Andrea Setzer, and James Pustejovsky.
Timeml annotation guidelines version 1.2.1.
01 2006.

Edwin Simpson, Yang Gao, and Iryna Gurevych.
Interactive text ranking with bayesian optimization: A case study on community qa and summarization.
Transactions of the Association for Computational Linguistics, 8:759–775, 2020.

Artem Sokolov, Julia Kreutzer, Stefan Riezler, and Christopher Lo.
Stochastic structured prediction under bandit feedback.
Advances in neural information processing systems, 29:1489–1497, 2016.

Haoyu Song, Wei-Nan Zhang, Jingwen Hu, and Ting Liu.
Generating persona consistent dialogues by exploiting natural language inference.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8878–8885, 2020.

Julius Steen and Katja Markert.
Abstractive timeline summarization.
In *Proceedings of the 2nd Workshop on New Frontiers in Summarization*, pages 21–31, 2019.

Jannik Strötgen and Michael Gertz.
Multilingual and cross-domain temporal tagging.
Language Resources and Evaluation, 47(2):269–298, 2013.

Richard S Sutton and Andrew G Barto.
Reinforcement learning: An introduction.
MIT press, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour.
Policy gradient methods for reinforcement learning with function approximation.
In *Advances in neural information processing systems*, pages 1057–1063, 2000.

Hiroya Takamura, Hikaru Yokono, and Manabu Okumura.
Summarizing a document stream.
In *European conference on information retrieval*, pages 177–188. Springer, 2011.

Louis L Thurstone.
A law of comparative judgment.
In *Scaling*, pages 81–92. Routledge, 2017.

Giang Tran, Mohammad Alrifai, and Eelco Herder.
Timeline summarization from relevant headlines.
In *European Conference on Information Retrieval*, pages 245–256. Springer, 2015.

Giang Binh Tran, Tuan Tran, Nam-Khanh Tran, Mohammad Alrifai, and Nattiya Kanhabua.
Leverage learning to rank in an optimization framework for timeline summarization.
In *TAIA Workshop at SIGIR*, 2013.

Stijn Marinus Van Dongen.
Graph clustering by flow simulation.
PhD thesis, 2000.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin.
Attention is all you need.
In *Advances in neural information processing systems*, pages 5998–6008, 2017.

William Yang Wang, Yashar Mehdad, Dragomir Radev, and Amanda Stent.
A low-rank approximation approach to learning joint embeddings of news stories and images for timeline summarization.
In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 58–68, 2016.

Christopher John Cornish Hellaby Watkins.
Learning from delayed rewards.
1989.

Joseph Weizenbaum.
Eliza—a computer program for the study of natural language communication between man and machine.
Commun. ACM, 9(1):36–45, January 1966.
ISSN 0001-0782.
doi: 10.1145/365153.365168.
URL https://doi.org/10.1145/365153.365168.

Wei Xu and Alexander Rudnicky.
Can artificial neural networks learn language models?
2000.

Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang.
Timeline generation through evolutionary trans-temporal summarization.
In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 433–443, 2011a.

Rui Yan, Xiaojun Wan, Jahna Otterbacher, Liang Kong, Xiaoming Li, and Yan Zhang.
Evolutionary timeline summarization: a balanced optimization framework via iterative substitution.
In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 745–754, 2011b.

Yuxuan Ye and Edwin Simpson.
A proposal: Interactively learning to summarise timelines by reinforcement learning.
In *Proceedings of the First Workshop on Interactive Learning for Natural Language Processing*, pages 25–31, Online, August 2021. Association for Computational Linguistics.
doi: 10.18653/v1/2021.internlp-1.4.
URL https://aclanthology.org/2021.internlp-1.4.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu.
PEGASUS: pre-training with extracted gap-sentences for abstractive summarization.
CoRR, abs/1912.08777, 2019.
URL http://arxiv.org/abs/1912.08777.

Yuhui Zhang.
Evaluating the factual correctness for abstractive summarization.
CS230 Project, 2019.