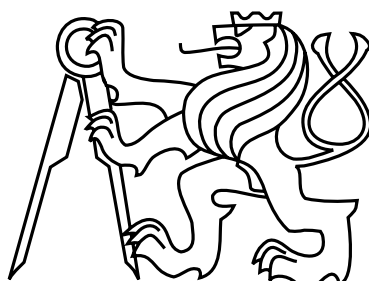


Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering



Master's Thesis

Text Summarization for Chatbots

Matyáš Lustig

Supervisor: Ing. Jiří Spilka, Ph.D.

Study Programme: Cybernetics and Robotics
Branch of Study: Cybernetics and Robotics
Master's Degree

May 23, 2019

I. Personal and study details

Student's name: **Lustig Matyáš** Personal ID number: **434921**
Faculty / Institute: **Faculty of Electrical Engineering**
Department / Institute: **Department of Control Engineering**
Study program: **Cybernetics and Robotics**
Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Text summarization for chatbots

Master's thesis title in Czech:

Sumarizace textu pro chatboty

Guidelines:

Text summarization is a natural language processing task of producing a shorter version of a document that preserves most of the original document's meaning while staying grammatically correct. The aim of this work is to incorporate a selected method into a conversation agent capable of communicating the document's summary to the end user.

The following tasks are defined:

1. Study the algorithms used for text summarization.
2. Select candidate algorithms (argument selection), implement them or use existing implementations.
3. Provide systematic experiments, analyze the results (complexity, performance, etc.) on different datasets.
4. Try to improve the performance of the selected algorithm or propose another one.
5. Implement the chosen method into a conversation agent providing a real-life use case.

Bibliography / sources:

- [1] R. Nallapati, B. Zhou, C. N. dos Santos, Çağlar Gülçehre, and B. Xiang, Abstractive text summarization using sequence-to-sequence RNNs and beyond, in CoNLL, 2016.
- [2] M. Allahyari, S. A. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut, "Text summarization techniques: A brief survey.," CoRR, vol. abs/1707.02268, 2017.
- [3] S. Gupta and S. Gupta, "Abstractive summarization: An overview of the state of the art," Expert Systems with Applications, vol. 121, 12 2018.
- [4] S. Gehrmann, Y. Deng, and A. M. Rush, "Bottom-up abstractive summarization," arXiv preprint arXiv:1808.10792, 2018.

Name and workplace of master's thesis supervisor:

Ing. Jiří Spilka, Ph.D., Department of Biomedical Engineering and Assistive Technology, CIIRC

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **18.02.2019** Deadline for master's thesis submission: **24.05.2019**

Assignment valid until:

by the end of summer semester 2019/2020

Ing. Jiří Spilka, Ph.D.
Supervisor's signature

prof. Ing. Michael Šebek, DrSc.
Head of department's signature

prof. Ing. Pavel Ripka, CSc.
Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student's signature

Acknowledgements

I would like to express my gratitude to my supervisor, Jiří Spilka for all the suggestions and advice he has offered me throughout the work on this thesis. I am grateful to my parents for their continuous encouragement and to the hearty people of Tel Aviv.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague on May 23, 2019

.....

Abstract

Recent advances in speech recognition and natural language understanding have brought an edge to the communication between humans and computers. With personal assistants and chatbots slowly becoming parts of our lives, we implement a conversational platform built on Amazon Alexa Skills, providing a user with summarized content of a Wikipedia website. A comprehensive overview of the state of the art in dialogue systems and unstructured text summarization areas is provided, and an extractive latent semantic analysis and abstractive discourse-aware attention model are evaluated on publicly available datasets. Word2vec word embeddings are further proposed and validated as a possible improvement. The selected approach is implemented into the chatbot providing a practical use case.

Abstrakt

Zpracování přirozeného jazyka proměňuje, jak jako lidé komunikujeme s počítači. Hlasoví asistenti a chatboti se pomalu ale jistě stávají nedílnými součástmi našich životů. Práce pojednává o návrhu konverzační platformy založené na Amazon Alexa Skills, která uživateli zprostředkuje souhrn webové stránky z Wikipedie. Současné přístupy k vývoji dialogových systémů a sumarizace nestrukturovaného textu jsou popsány a dvě vybrané metody, extraktivní “latent semantic analysis” a abstraktivní “discourse-aware attention” model, porovnány na veřejně dostupných datasetech. Na základě získaných zkušeností je navrženo a otestováno možné vylepšení zahrnující word2vec model. Vybraná metoda je integrována do platformy chatbota poskytující reálný příklad užití.

Contents

1	Introduction	1
I	Chatbots	3
2	State of the Art	5
2.1	Dialogue Systems	6
2.1.1	Frame Based	6
2.1.2	Machine Learning	8
2.2	Available Platforms	8
2.2.1	Amazon Alexa	9
2.2.2	Google Assistant (Home)	10
3	Platform Development	11
3.1	Design	11
3.1.1	Knowledge Base Acquisition	11
3.1.2	Navigation	12
3.1.3	System Architecture	13
3.2	Implementation	14
3.2.1	Alexa Skill	14
3.2.2	Switch	15
3.2.3	Controller	15
3.2.4	Chatbot Prototype	17
II	Text Summarization	19
4	State of the Art	21
4.1	Extractive Approach	21
4.1.1	Topic Words	22
4.1.2	Frequency Driven	23
4.1.3	Latent Semantic Analysis	24
4.2	Abstractive Approach	25
4.2.1	Structure Based	25
4.2.2	Semantic Based	26
4.2.3	Deep Learning	26

5	Implementation	29
5.1	Latent Semantic Analysis	29
5.2	Discourse-Aware Attention Model	30
5.3	Cluster	31
6	Experiments	33
6.1	Datasets	33
6.1.1	CNN/Daily Mail	34
6.1.2	PubMed	34
6.2	Metrics	35
6.2.1	Human	35
6.2.2	ROUGE	35
6.2.3	BLEU	37
6.2.4	Run Time	37
6.3	Results	37
6.3.1	Latent Semantic Analysis	38
6.3.2	Discourse-Aware Attention Model	43
7	Improvement Proposals	47
7.1	Implementation	48
7.2	Results	48
7.3	Chatbot Integration	52
8	Conclusion	53
A	Contents of the Attached CD	55

List of Figures

2.1	Structure of a voice assistant	5
2.2	Developer console of Amazon Alexa Skills	9
2.3	Developer console of Dialogflow, Google NLP platform	10
3.1	Structure of the crawled content from Wikipedia page about Earth with sections and subsections on the left and the actual contents as showed on the website ¹ . The structure obtained by scraping is identical to the original one.	12
3.2	Diagram of the designed architecture solution for our application.	13
3.3	A sample conversation between the end user and the chatbot	17
4.1	Baseline seq2seq model with attention [26]	27
4.2	Pointer generator model with the ability to copy rare words directly from the source text while still generating most of the summary from scratch [26]	27
4.3	A discourse-aware attention model built on top of a pointer generator seq2seq architecture by <i>See et al.</i> [27]	28
6.1	Attention Visualizer showing a sample PubMed article, reference abstract and generated summary produced by the DAAM	46
7.1	Results of the LSA word embeddings experiments on CNN/DM dataset compared with the original LSA and DAAM implementations. Run time per document is not stated for DAAM due to incomparable hardware.	50
7.2	Results of the LSA word embeddings experiments on PubMed dataset compared with the original LSA and DAAM implementations. Run time per document is not stated for DAAM due to incomparable hardware.	51
7.3	A sample conversation between the end user and the chatbot employing LSA	52

List of Tables

2.1	Transcript of a dialogue with GUS from <i>Bobrow et al.</i> [1] with additional highlighting of possible intents and entities by authors	7
6.1	Quantitative overview of the used datasets. Number of documents, average document length in sentences and words and average summary length in sentences and words for CNN/Daily Mail and PubMed dataset.	34
6.2	Results of LSA method on CNN/Daily Mail and PubMed datasets according to ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-SU*, recall and F1 scores and BLEU metric.	38
6.3	Example of a good summary of CNN/DM article produced by the LSA. Scores ROUGE-1 F1 0.4898 and ROUGE-SU* F1 0.1803.	39
6.4	Example of a bad summary of CNN/DM article produced by the LSA. Scores ROUGE-1 F1 0.5319 and ROUGE-SU* F1 0.2192.	40
6.5	Example of a good summary of Wikipedia page section produced by the LSA	41
6.6	Example of a bad summary of Wikipedia page section produced by LSA . . .	42
6.7	Results of DAAM method on CNN/Daily Mail and PubMed datasets according to ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-SU*, recall and F1 scores and BLEU metric.	43
6.8	Example of a summary of PubMed article produced by the DAAM. Scores ROUGE-1 F1 0.2277 and ROUGE-SU* F1 0.0426.	44
6.9	Example of a summary of Wikipedia page section produced by the DAAM . .	45
7.1	Configurations of the LSA word embeddings experiments	49
7.2	Results of the LSA word embeddings experiments on CNN/DM dataset compared with the original LSA and DAAM implementations.	49
7.3	Results of the LSA word embeddings experiments on PubMed dataset compared with the original LSA and DAAM implementations.	50

Chapter 1

Introduction

Conversation is a key instrument for developing and maintaining mutual relationships. Talking to someone is probably the most effective way how to transfer the necessary information, apart from nonverbal communication. People started to think that having a conversation with a computer would be equally convenient.

Recent advancements in artificial intelligence and natural language processing made possible by abundant computational power and large datasets proved that the objective is not unreachable as there are already many applications of chatbots and voice assistants in our daily lives. Nevertheless, it might take a long way for verbal communication to become the first choice when dealing with a computer. The famous Turing test focusing on the machine's ability to replace a human in their behavior indistinguishably remains a high aim.

One of the most attractive use cases for a conversation with a computer is to obtain new information. It can be a weather forecast for tomorrow, the population of a certain city or the state of traffic on the way to work. However, a lot of the information on the web is in the form of an unstructured text be it paragraphs, articles or essays. While reading a news article in a web browser is easy; comprehension just by listening to it proves much more demanding. The long text passages are not suitable for recitation by voice assistants. Moreover, with the current accelerating tempo of data being generated, a necessity to filter only the essential is rising.

Both problems can be solved with the help of automatic text summarization, one of the many tasks of natural language processing (NLP) field. The aim is to provide a coherent summary of the source text or multiple documents while maintaining the original context and readability. There are many possible applications such as summarizing news articles in a couple of bullet points, forming an abstract of a scientific article or condensing information about a certain topic from dozens of source documents. While humans perform naturally very well at these tasks, shifting the responsibility to a computer might allow us to focus on more creative work.

In this work, we propose a platform allowing a user to absorb virtually any text document through a voice-enabled chatbot. The first part of this work deals with the development of an extensible conversational platform filled with information extracted from the web. Wikipedia website is used as an example of the source document providing a great range of the possible topics. Text is processed, summarized and provided to the user via an agent built on top

of Amazon Alexa. The dialogue management system is based on a frame-based architecture represented by a state automaton introduced first by *Bobrow et al.* [1]. The solution can serve as a basis for various NLP experiments in the future.

The second part proceeds with an overview of text summarization techniques. Most of the approaches can be roughly divided into two classes, extractive and abstractive. Extractive methods aim to extract the most important sentences from the source document to represent its message. Three relatively independent steps [2] can be identified ensuring a great variety between the final solutions – intermediate representation captioning the main aspects of the source text, scoring individual sentences and finally a selection of the candidate ones for the summary. Abstractive methods on the other hand try to summarize the source text using phrases originally not appearing in the document mimicking a human approach better. Deep learning methods inspired by successful applications in machine translation mainly by *Bahdenau et al.* [3] are recently reaching state of the art results.

Implementations of an extractive latent semantic analysis and abstractive discourse aware attention model are described in this work and tested on two distinguish datasets, CNN/Daily Mail news articles and longer scientific works from the PubMed database. Experiments are thoroughly evaluated using defined metrics and human assessment. Further, we propose and validate a possible improvement by incorporating word2vec word embedding model into the latent semantic analysis. Finally, the method best fitting our designed criteria is implemented into the chatbot providing a tool useful in daily life. The user can absorb a text document via an interactive conversation instead of reading it alone.

Part I

Chatbots

Chapter 2

State of the Art

The art of maintaining a meaningful conversation is one of the many heavily pursued branches of natural language processing field. One might even call the conversational agents, chatbots or dialogue systems, a holy grail of NLP as they combine and build atop of its many subfields. The high level structure of every voice assistant system can be broken down into three independent parts: natural language understanding transcribing the user's voice into text, the dialogue manager or system containing the processing logic and response generation and third comes the natural language generating step turning the chatbot's textual response into an output voice utterance.

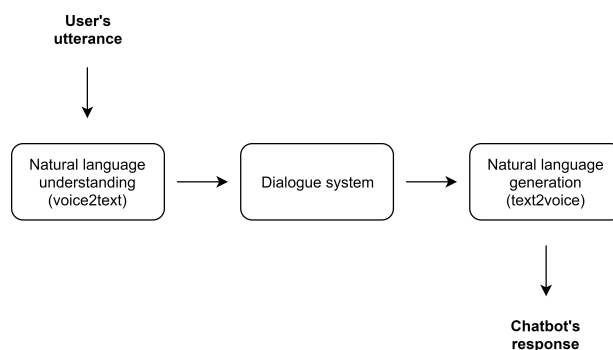


Figure 2.1: Structure of a voice assistant

The current state of the dialogue systems research and applications is described in this chapter. Section 2.1 deals with the second step in the chain, the core of every chatbot, a dialogue navigation system. Different approaches to maintaining a conversation structure are presented. The supporting natural language parts are taken care of in Section 2.2 where we provide a brief overview of publicly available platforms for implementing such solutions from which one is further used in the work.

2.1 Dialogue Systems

Most of the dialogue applications developed nowadays fall into two more or less separated classes. The first class is referred to as **task oriented agents**, an analogy to expert systems, highly specialized bots operating in limited number of domains very often equal to one. They are designed to complete a particular task based on the input information provided by the user. An example of such a bot can be a digital assistant in your mobile phone, Siri by Apple, Amazon Alexa or Google Assistant. The particular task might be to answer your simple question such as, “What is the weather in Geneva going to be tomorrow?”

In a simplified way, a set of actions happen after you ask such a question. Foremost, the agent has to convert your recorded speech to text and rightly detect the intent, that you are curious about weather. Then it identifies the relevant entities in your sentence, such as the city you take interest in and the particular day in the future. Based on this set of information the agent can extract the proper knowledge from its own database or a third party weather forecasting web service. Let’s say the model indicates temperature of 23 °C and a clear sky. Only now, after carrying out all the necessary steps which altogether took a couple of milliseconds, can the assistant construct and utter the answer to your question such as, “Tomorrow in Geneva it is going to be 23 °C with a clear sky.”

The second group of dialogue systems can be called **general agents** and in the complexity of development they are very much comparable to a general AI. Although the public perceives the term chatbot as basically any type of conversational agent, parts of the NLP science community see it as this second class of dialogue systems designed for extended and casual conversation [4]. General agent’s highest aim is to mimic inter human dialogues with their abrupt topic switching, fluency and maybe even subjective points of view. Example of such an interaction might be a casual small talk with your voice assistant asking you about your day at work as soon as you come home.

2.1.1 Frame Based

From now on, we will mostly focus on the first class of dialogue systems and refer to them as chatbots. The task oriented agents operate based on the information and commands provided by the user which means they must be able to extract such data from the utterances. The dialogue scenario is usually described by distinct frames where each frame can be perceived as one turn by the bot and one by the user. Navigation between the frames is either firmly set or driven by user’s response. In any way the whole system can be represented as a state automaton with individual frames representing states and user’s commands as transitions between them. One option is to allow the user only to interact with the agent through buttons and predefined so called quick replies. As soon as the user responds with a whole sentence as is in human nature, the bot runs into problems. However, such a way of dialogue does not offer the end users much freedom and is thus not very cherished by them.

More user friendly technique which most of the assistants employ nowadays is a frame driven dialogue system with slot filling introduced first by *Bobrow et al.* [1]. The goal here is to understand user’s intentions from plain human to human like utterances. The bot knows what kind of information about the particular topic it needs to extract and thus several frames, slots and their possible values can be predefined to simplify the task. In case the

chatbot deals with more fields, the individual *domain* user is interested in has to be identified at first. A domain is an area covering similar actions dealing with the same subject, i.e. GPS navigation, personal calendar or booking a flight. Then the concrete *intent* has to be determined, what in particular is the user trying to accomplish in the already selected domain, i.e. finding the fastest way to the office, adding a new meeting to a calendar or purchasing additional baggage on the booked flight. Some actions might be performed without any details but in most cases, additional information has to be provided to the agent by filling the prepared slots on the background. The individual piece of information is called *entity* and has to be extracted from the utterances including possible synonyms [4]. Extending former example, those entities can be destination, time of the meeting or the flight booking number.

Finally the agent has collected all necessary information to perform the action on user's behalf. An illustration of the whole process can be drawn from a transcript of an actual dialogue with GUS (Genial Understander System), the first frame driven system specializing in the domain of booking a flight trip [1]. Possible intents and entities are highlighted in blue and orange respectively.

GUS: Hello. My name is GUS. I can help you plan a simple trip by air. Where do you want to go?

USER: I want to go to San Diego on May 28

GUS: What time do you want to leave?

USER: I must be in San Diego before 10 am

GUS: Would you like P.S.A. flight 102 that arrives at 9.15 am?

USER: What is the next flight?

GUS: Air California flight 310 that leaves at 8.30 am and arrives at 10.00 am

USER: I'll take the first one

GUS: In what name should I make a reservation, please ?

USER: Dan Bobrow

GUS: I have confirmed the following flight: P.S.A. flight 102 on Wednesday May 28 from San Jose to San Diego that leaves at 7.30 am and arrives at 9.15 am. What date do you want to return on ?

USER: On Friday in the evening.

GUS: Would you like the flight that leaves at 7.45 pm ?

USER: That's fine.

GUS: I have confirmed the following flight: P.S.A. flight 307 on Friday May 30 from San Diego to San Jose that leaves at 7.45 pm and arrives at 9.30 pm. Thank you for calling. Goodbye.

Table 2.1: Transcript of a dialogue with GUS from *Bobrow et al.* [1] with additional highlighting of possible intents and entities by authors

2.1.2 Machine Learning

The dialogues systems benefit recently from the heavy employment of machine learning (ML) algorithms. The main difference compared to the frame based systems is that the conversation can be more open and evolve in more or less new ways every time. Both approaches can however be combined. For example, the navigation between states of an automaton in the frame based dialogue system can be based on learned probabilities. The area of dialogue systems development resembles more an alchemy than a strict science.

In the first class of ML based solutions, the dialogue control system is perceived as a sequential process where different subsystems process the obtained information one after another – user utterance understanding, dialogue frame/state tracking, appropriate action selection, assistant response generation. Neural network (NN) or other machine learning models can be used at each stage of the pipeline. For example *Levin et al.* [5] have used a combination of supervised and reinforcement learning for stochastic Markov Decision Process model representing the dialogue structure and navigation.

An emerging approach aims to learn dialogue navigation as an end-to-end model using usually the Recurrent neural networks (RNN), which are able to produce a sequence of the most probable output words based on conversation history. All of which happens in a neural network monolith and does not rely on hand crafted subsystems as in the first case of sequential processing. The models are trained on large datasets and try to produce utterances similar to human conversation. *Sordoni et al.* have opened the road to data driven dialogue navigation with RNNs which are by their definition able to keep track of the conversation context [6].

Another interesting approach is Hybrid coded network (HCN) by *Williams et al.* [7]. It is based on a RNN and adds a mechanism for injecting domain knowledge into the assistant together with constraints on actions in the form of utterances templates. Advantage of limiting the possible number of options simplifies the model and reduces its learning complexity.

2.2 Available Platforms

A supporting structure is necessary around the dialogue managing system to take care of the communication between user's device and the server, transcribing their speech into textual form and sending back the generated reply converted into speech. The two biggest players in the field nowadays are Amazon and Google. Both of the platforms are fully functional standalone but also offer tools for own application development, a feature interesting for our intentions.

The voice assistant Amazon Alexa was first released in November 2014 empowered by underlying technology Amazon Lex, a service for building conversation interfaces in both voice and text form. It provides the developer with speech recognition and language understanding capabilities. Thanks to many possible endpoints, the multi-platform development effort is highly reduced [8]. Google Home was initially released two years after Alexa in November 2016 with similar features. Developers are allowed to build applications for Google Home standalone devices or Google Assistant running on mobile phones with the help of Dialogflow (formerly API.AI), a tool for creating voice and conversational interfaces [9]. Description of both of the platforms' features and development process follows.

2.2.1 Amazon Alexa

Amazon provides developers with an interactive console for creation and management of so called Alexa Skills¹. Initially, a developer has to either select one of the prepared templates or enter the custom mode providing the most freedom. In order to be able to run the new skill, all items on the builder checklist informing about the necessary steps of development have to be checked. At first the invocation name of the skill is chosen. That is the word which wakes the assistant up when pronounced. The next step is defining intents – mappings between user’s queries and bot’s actions – assistant should recognize in conversations. They can be either chosen from more than 20 predefined ones or introduced newly with sample utterances, different entities, intent slot types and the option for required comprehension confirmation. Slot types or entities – important pieces of information to be extracted from the utterances – can be defined including synonyms. Another possibility is to upload a JSON file defining the whole interaction model schema.

Various interfaces or peripherals can be connected to the skill (audio, display, video, gadgets), each providing its directives and requests for content streaming and control. Last but not least, the service endpoint has to be chosen. Two options exist, either an AWS Lambda, a serverless computing service by Amazon Web Services, or a HTTPS Web service of the developer’s own choice. These endpoints receive POST requests whenever a user interacts with the Alexa Skill and can enrich the conversation with data obtained from the application’s backend.

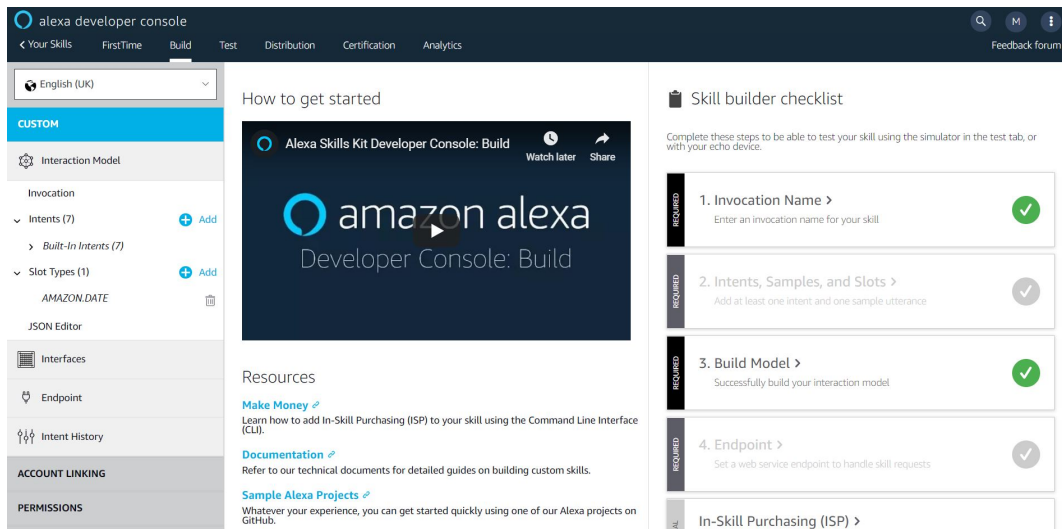


Figure 2.2: Developer console of Amazon Alexa Skills

¹Alexa Skills developer console, <https://developer.amazon.com/alexa/>

2.2.2 Google Assistant (Home)

Actions on Google is a developer platform offering the opportunity to extend abilities of Google Assistant. The implemented actions can be accessed through Google Home or any mobile phone/device with Google Assistant application installed. Very similarly to Alexa Skills, developer can choose invocation name of their agent and its voice from a short list of options. As Google Assistant deployed on mobile phones also features a graphical interface, the look of an agent on the screen can be customized.

The real power can be however found while defining various actions built into the assistant. Here Google introduces another of its services, Dialogflow, a platform for building NLP based solutions². The service is under abrupt developed and some of its latest released features are still in beta version. Developer is allowed to define intents and entities similarly to Alexa Skills. Very interesting feature still in beta version is the ability to create knowledge base just by uploading a PDF document, text in Q/A format or a website URL. Agent is then supposedly able to answer user's questions based on the provided unstructured text. In order to enhance agent with higher functionality, it can be connected through POST requests from webhooks or Cloud Functions³, serverless compute platform equivalent to AWS Lambda, to application's backend. On the frontend side, various messaging platforms can be connected to the developed application as Google Assistant is only one of the options. Many prebuild agents, small talk regime for chit chat with the user and the chance to utilize machine learning for training are also offered.

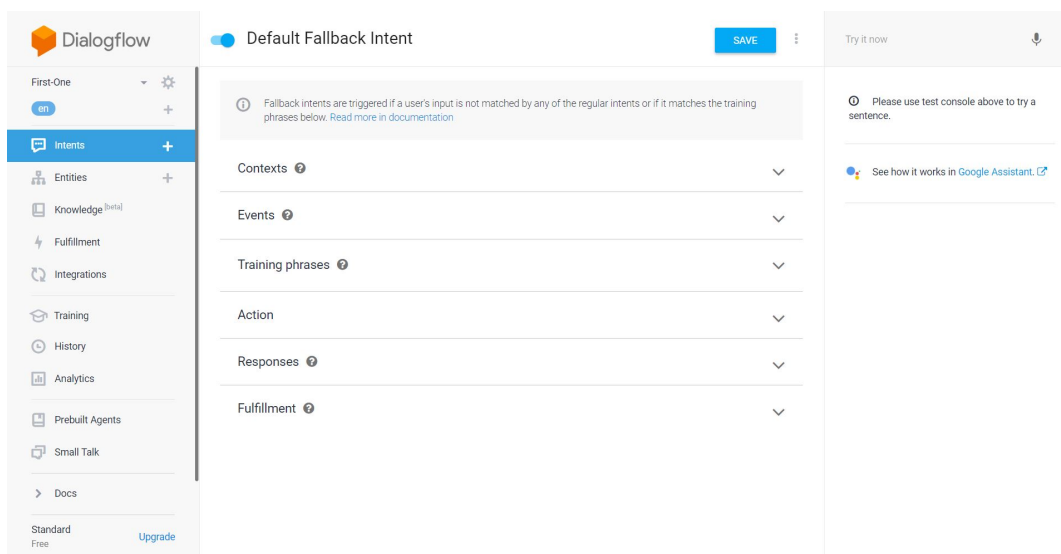


Figure 2.3: Developer console of Dialogflow, Google NLP platform

²Dialogflow developer console, <https://console.dialogflow.com/>

³Google Cloud Functions, <https://cloud.google.com/functions/>

Chapter 3

Platform Development

Based on the acquired knowledge from previous chapter, the conversation platform is developed here. Its functionality is designed in a bottom up approach followed by implementation of the individual parts. A functional prototype of the chatbot is presented and discussed.

3.1 Design

The design phase is an important step before the implementation itself as many problems and challenges can be detected and addressed at a lower cost. Main ideas of the chatbot's structure are described here including its knowledge base acquisition and dialogue navigation. We further dive into the decentralized system architecture and design the main components functionality as well as communication in between them.

3.1.1 Knowledge Base Acquisition

In order to build a chatbot some form of a knowledge base or a fixed dialogue structure has to be present. At first our idea was to obtain such information by scraping text documents in different formats and converting their text into a hierarchy of headings and paragraphs, from which a dialogue scenario could be build. While converting text documents into a form suitable for a conversation structure looked like a great idea, the reality turned out to be much more complicated. For example PDFs are very well readable for humans and their formatting remains the same no matter the platform they are viewed on. However, their machine readability is very limited and extracting a firm structure out of the text turned out to be almost impossible. Thus while searching for an alternative data source, Wikipedia came to our minds.

An important advantage of Wikipedia websites is that they all have the same structure. Not only they are noticeably all divided into a set of similar headings and sections, even their HTML source code is structured in the same way which makes content scraping much easier. Moreover, many websites could be scraped making the chatbot more universal by extending its possible knowledge base virtually to the whole Wikipedia.

We employ the fact that all headings, subheadings or normal paragraphs are written in their distinct ways using different HTML tags. It is possible to look for example only for top

level headings of sections on the page and build the content structure of the website. Next, headings of a lower level subsections can be found and assigned to the appropriate top level sections. The same process is then repeated with the textual paragraphs and in the end the complete textual content of the website is arranged in a hierarchic tree.

3.1.2 Navigation

The website's structure is further represented as a state automaton and serves as a knowledge base for the chatbot. Each (sub)section of the website represents a state with links to both sections on the same level and to its own subsections. Apart from the section title, a corresponding paragraph and other meta data are assigned to each state. The idea is that the agent will be able to use this automaton and navigate through the states, an approach very similar to the frame based chatbots described in Section 2.1.1.

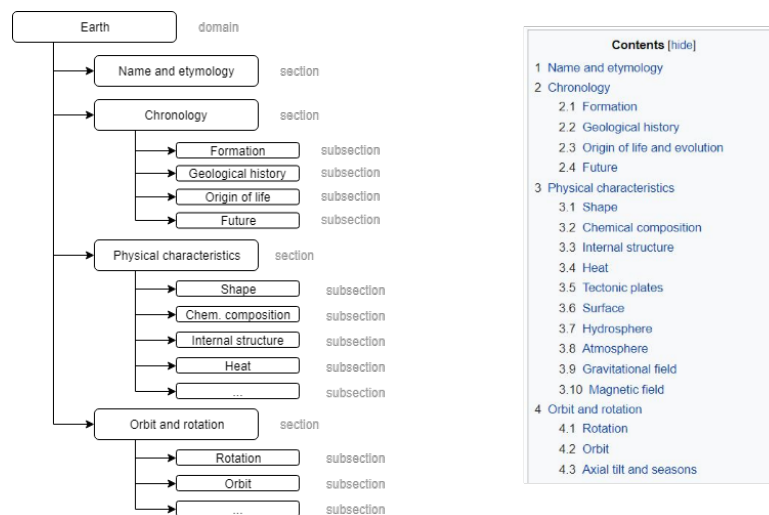


Figure 3.1: Structure of the crawled content from Wikipedia page about Earth with sections and subsections on the left and the actual contents as showed on the website¹. The structure obtained by scraping is identical to the original one.

In the diagram above we can see the sections and subsections, which represent the states of the automaton. In order to keep track of the mutual connections between them, several relations are defined – parent state, children states and sibling states. For illustration, let's imagine that the initial state is *Name and etymology*. It does not have any subsection or otherwise children states. However, from there it can be proceeded to a sibling state *Chronology* which already has several children states such as *Formation* or *Origin of life*. Their parent state is again *Chronology* whose sibling states are *Name and etymology* and *Physical characteristics*.

¹Wikipedia page about Earth, <https://en.wikipedia.org/wiki/Earth>

The navigation in the state automaton is determined by the three relations between its states that have been described above. The agent can pass between the states on the same level (sibling states) as well as between the parent state and children states or vice versa. In the example above that means transfer from *Future* is permitted to *Chronology* but not to *Chemical composition* as that is a children state of a different parent and thus not a sibling state. Such navigation through the dialogue structure might seem a bit lengthy as we might have to pass through a particular state several times. However, this design choice allows for navigation problem simplification while maintaining a sufficiently rich frame based model for future experiments.

3.1.3 System Architecture

The aim of this work is not to design a flawless software application but rather to use a simple system to demonstrate and experiment with various NLP techniques. The design is kept uncomplicated and functional. Despite author's experience with Google Dialogflow, Amazon development platform has been selected in the end due to accessibility to a physical Amazon Alexa device. Most of the architecture choices are made with the idea of solution transferability in mind.

The first element closest to the end user is the Amazon Alexa with its Skills platform. It functions as the frontend interface uttering the chatbot's responses but also taking care of the speech recognition and basic intent detection. In order to process the transcribed text, our Alexa Skill has to be connected to an endpoint. Out of the two options the serverless compute service AWS Lambda has been selected as it is officially recommended. The choice can also be perceived as an opportunity to explore microservices architecture. One of the many good traits of Lambda functions is that they scale automatically according to the current traffic with a default limit of one thousand concurrent executions.

The Alexa Skill is connected to AWS Lambda function called *Switch*, where all the top level logic takes action. The transcribed text is analyzed and detected intents and entities govern the next actions. Switch sends further requests to another AWS Lambda function, the *Controller*, and creates user friendly text responses based on the data received back. While different voice assistant platforms have various interfaces, the Switch endpoint can be easily rewritten for any of them making the chatbot transferable.

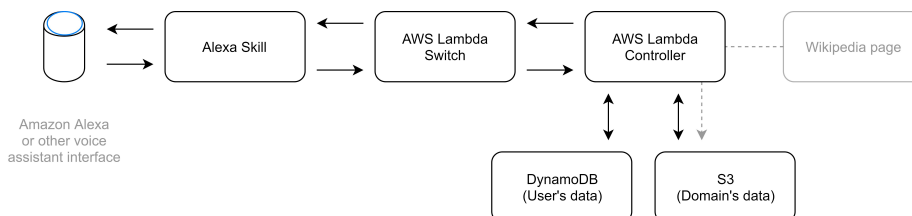


Figure 3.2: Diagram of the designed architecture solution for our application.

Controller represents the state automaton built upon the website's content. It receives requests from Switch such as "go to the next section" and returns data related to the next state. In order to be able to do so, it is connected to an AWS S3, an object data storage service where the website's content, the domain is stored. When the domain is not yet in memory, Controller can scrape the particular website to obtain it. Controller also keeps track of the user's profile to maintain continuity of the conversations. It does so with the help of AWS DynamoDB which is a key-value NoSQL database.

While a vendor lock-in might be a danger for a business company, the simplicity of running different services at AWS highly accelerates the development and provides us with more time to focus on the real objective of this work.

3.2 Implementation

Now when the architecture and agent's functionality have been designed it is time to move to the implementation phase. Every part of the solution's chain is described starting with the Alexa Skill followed by methods of the two main Lambda functions Switch and Controller as well as the structure of several objects used throughout the design and access to the data storage services. The goal here is not to comment on every line of code but rather to explain the general concepts and interesting choices made along the way.

3.2.1 Alexa Skill

Although Alexa Skill is the only part which does not required nor allow writing own code, for integrity the setup through the graphical interface described in Section 2.2.1 should not be omitted. Apart from setting the invocation name of our skill to "askwiki" and connecting it to the Lambda functions, most of the work lies in defining various intents which together with their entities, slot values control the behavior of the chatbot.

Every new skill comes with a predefined set of intents controlling the highest level logic, such as activation and termination of the certain skill or handling exceptions. For the purposes of navigation through a domain scraped from a Wikipedia website, we have further defined three intents called *next-section*, *previous-section* and *specific-section*. These intents recognized with the help of predefined examples of user's sentences drive the logic of the state automaton moving between adjacent sections or jumping over to a specific one. In the case of a *specific-section* it is also necessary to extract the name of the section a user is interested in. That is where slot values come up. Through the console it is possible to define certain values of various types that should be extracted from the user's utterances such as date *AMAZON.Date* or the name of an airport *AMAZON.Airport*. A list of possible values of the slot can be predefined or any detected input can be passed to a slot type *AMAZON.SearchQuery*. As the idea is to keep the chatbot as universal as possible, the second option is employed for section name detection.

The detected intent and possibly the relevant slot value are passed with much more of the request's meta data to the connected Switch AWS Lambda function in a form of a JSON.

3.2.2 Switch

According to the design, the Switch Lambda function is sorting the requests received from the skill based on the detected intent and forwards them to the Controller. When it immediately receives back information from Controller, it generates a human readable and a voice suitable message which is sent to the skill.

The decision logic of the chatbot follows guidelines provided by Alexa Skills Kit SDK for Python² sample skill using decorators approach. Decorator is a design patten allowing extension of an existing object functionality without modifying its structure. At first the entrance method `lambda_handler()` deals with the incoming request type and decides whether the Skill has been launched, a new session with the user is to be started or terminated and of course handles request exceptions. As the request's type is determined, relevant method is called taking care of the necessary steps. These can be for example `on_launch()`, `on_exception()`, or `on_intent()`.

As the supporting methods are implemented, the main focus shifts to the last of the mentioned methods where the agent's behavior is guided by the detected intents. At first the Amazon built in intents such as *AMAZON.HelpIntent* or *AMAZON.StopIntent* are treated and then we implement our own ones, *next-section*, *previous-section* and *specific-section*.

In most of the cases a request to the Controller is generated based on the extracted data from the user's utterance. As Controller replies, a final textual response for the user is built and forwarded to the Skill.

The Alexa Skill response is supposed to be provided in the form of a JSON containing some meta data, a parameter whether it is the final agent's utterance or the dialogue session is to be continued and several types of textual response – the speech to be pronounced by the device, a card visible on devices with graphical interface and alternatively a reprompt asking the user to provide additional information.

As the domain's data about the particular section is received from the Controller, the information is inserted into prepared string templates and passed in the form of parameters to the response building method `build_response()` where the whole JSON for the Skill is created.

3.2.3 Controller

Controller is an AWS Lambda function hidden from the user taking care of the state automaton logic, scraping content from Wikipedia websites and building the new domain. Necessary operations further include text preprocessing and cleaning as well as handling the two connected data storages.

According to the design, Controller receives a message from Switch requesting a particular state automaton operation for the given user, for example *next-section*. It extracts the current user's state from the Dynamo database or creates a new profile with default initial state. In the next step the domain is loaded from S3 in the form of a custom Domain object supporting the state automaton navigation. Based on the requested operation (next, previous or specific section), the automaton transition is carried out. That basically means the user's profile is

²Alexa Skills Kit SDK for Python, <https://alexa-skills-kit-python-sdk.readthedocs.io/>

updated in the DynamoDB to reflect the new current state and the actual section name with the processed textual content extracted from the website section are sent in a JSON response back to the Switch. The text processing steps and summarization of unstructured text are addressed in Part II.

The best way how to extract content from a website is utilizing its provided API. Nevertheless, to maintain generality of the proposed solution, it has been decided to **scrape the content** utilizing Python library Beautiful Soup³ instead. As every Wikipedia page holds the same HTML structure, the crawler can be deployed on any of them. The page usually consists of sections and their subsections of multiple levels.

At first the crawler loops through the top level section headers omitting those bringing little value to the conversation and occurring on almost every page such as “Notes and references”, “External links”, “Bibliography”, etc. For each of the meaningful sections an instance of object State is created with section’s name, textual content and information about its adjacent sections (*parent* on higher level, *siblings* on the same level just beside, and a list of *children* states on the lower level) used by the state automaton. Wikipedia links citing other pages are removed from the text at this point. For each of the sections, the crawler goes one level deeper and loops over its subsections replicating the process.

In the end, an instance of object Domain is returned as a pair of the website’s name and an array of interconnected State object instances representing the state automaton which Controller steps through. The final result highly resembles the structure in Figure 3.1.

Once the Domain instance is formed by scraping, it is saved in the **S3 data storage** in a form of a Python serialized object Pickle⁴. Every domain does not have a fixed amount of sections/states as every website has different length. This would cause many problems during saving the structure in a SQL database with defined columns. Thus it has been decided to employ object serialization and Pickle is the first choice in Python. Working with byte streams in DynamoDB showed to be problematic and the S3 service was used.

Setup and communication with the S3 data storage is thanks to the interconnected AWS services very straightforward. The Domain object is paired with a unique key identifier and saved in a bucket representing some space in the storage. Two situations might arise: either the domain has been already crawled and saved in the data storage or it still has to be obtained first. We thus try to request the domain by it’s known key being the website’s name and the domain is either loaded straight from S3 or scraped and saved first.

DynamoDB database is used for keeping track of users’ profiles. Each profile consists of the user-id retrieved from the Alexa Skill request, the domain a user is currently located in and their current state in it.

Several operations are necessary to maintain this information up to date. Method `get_user()` is defined retrieving the user’s profile from the database based on their user-id which represents the key in the NoSQL schema. Next, method `update_user()` creates a new user profile in case the provided user-id is not found in the database with default domain

³Beautiful Soup: Python library for pulling data out of HTML and XML files, <https://www.crummy.com/software/BeautifulSoup/bs4/doc/>

⁴Pickle: Object serialization in Python, <https://docs.python.org/3/library/pickle.html>

and initial state in it. If some user is matched, their domain and current state are updated based on the parameters provided. Finally, `remove_user()` deletes the profile record from database in case a conversation session ends. The user's profile is kept in memory only during the individual dialogues so that their history does not interfere with latest session.

3.2.4 Chatbot Prototype

When all the sections described above are successfully combined together, the objective of building an experimental conversational platform is fulfilled. A prototype of an agent providing information from Wikipedia page about Earth is operating allowing basically three possible operations – moving back and forth through the page's content and jumping to a specific section. In order to determine which section to go to, the cosine distance has been used to identify the section name closest to the requested section by the user.

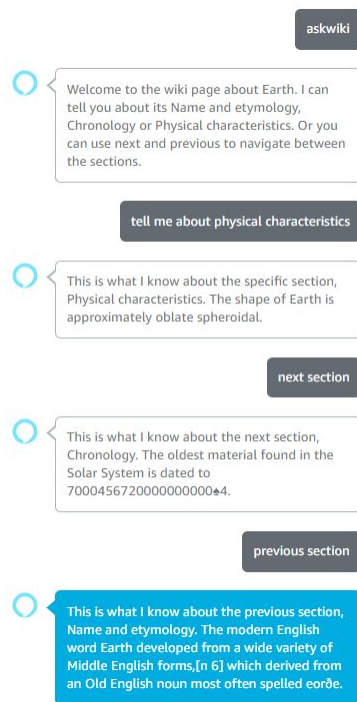


Figure 3.3: A sample conversation between the end user and the chatbot

As can be noticed, some Wikipedia links persisted the cleaning efforts and the number format is damaged. Such issues will be addressed together with the implementation of text summarization method in the following part allowing for better content selection than simply selecting the first sentence of every section.

Part II

Text Summarization

Chapter 4

State of the Art

With internet being nowadays an inherent part of our lives the amount of available information is skyrocketing. It is hard to find reliable evidence but some of the sources quote that alone over the span of last two years 90 % of the data in the world has been generated and the pace is still accelerating. Most of the data is unstructured and text is representing a substantial part. Naturally a strong desire arises to distinguish between relevant and useless data to reduce the volume of information we have to absorb. Although the research dates back to the end of last century [10], automatic summarization techniques are booming recently with the increased need and availability of cheap computational power necessary for machine learning algorithms, mainly NNs.

According to *Radev et al.* [11] a summary is defined as a text that is composed of one or more source documents, that conveys important information in the original text(s), while being no longer than half of the original text(s) and usually even less. As humans we have developed an exceptional sense for natural language and ability to easily spot even minor deviations from the established norm, both semantic and syntactic. That makes any NLP task very challenging and automatic text summarization is no exception. Let's have a look at several techniques used for shortening and extracting the significant information from text. Most of the current approaches can be categorized into two classes, *extractive* and *abstractive* ones with more and more algorithms combining recently both worlds. Further, the text summarization applications can be divided by the number of source documents. Either a single or multi document summaries are produced. This work deals with the first case.

4.1 Extractive Approach

Extractive summarization methods try to identify the most significant sentences in the original text and build a summary solely from such selection. Following the methodology of *Nenkova and McKeown* [2], three relatively independent steps can be identified in the process – intermediate representation, sentence scoring and summary sentence selection.

As the first step, the **intermediate representation** of the document is created. It captures the main aspects of the source text. The most popular methods employ topic

representation and identification of topics discussed in the text. Others use indicator representation which assigns to each sentence a features such as its position in the document or paragraph, its length or the number of named entities.

Further, based on the provided representation all original text **sentences are scored**. For mentioned topic representation, each sentence is assigned a score related to how well it represents various topics from the document. Several metrics can be combined at once with different weights. External context of the source document such as links pointing to the document on web might also be considered as shown by *Hingu et al.* [12].

Lastly, the final **selection of sentences** has to be conducted based on their score. That can be accomplished in several ways. Either the desired length of the summary is predefined and the appropriate number of sentences or a ratio of the original text length is chosen. Another way is to iteratively select sentences in a greedy way until a certain level of match is reached between the source text and its summary while maximizing coherence and minimizing redundancy [2].

An advantage of such division into three separate steps is that different methods at the individual stages can be combined together providing a vast space of alternatives to be explored. Some of the existing approaches are described below in greater detail in accordance with *Nenkova and McKeown* [2].

4.1.1 Topic Words

Probably one of the earliest works focusing on automatic text summarization was done by *Luhn* already in 1958 [13]. He proposed a method aiming to identify descriptive words in the source text representing the document's topic. Words occurrence in the document was employed to distinguish between those probably of low importance and those that are key for the topic. The excluded words might be those occurring very often such as prepositions or domain specific words as well as words on the other side of the frequency scale which are very rare.

An updated approach based on the *Luhn's* method [13] has been developed by *Dunning* [14] utilizing the power of log-likelihood ratio test for identification of words that are highly descriptive of the input, also called *topic signatures*. Topic signature words occur in the source text more often than one would expect by chance and are thus rare in other documents [2]. The χ^2 distribution table can be used to look up probabilities of obtaining certain value by chance. Adoption of this statistical significance approach led to great improvement in selecting relevant content in multi-document news summarization [15].

Once the topic signature words are identified, the salience of every sentence in the document is computed either from the number of topic signatures it contains or from the ratio of topic signatures with respect to the other words. Although both heuristics are based on the same intermediate representation, they can produce very different results. The first approach favors longer sentences simply because they contain more words and thus theoretically more topic signatures. The second method on the other hand measures density of key words which can be more precise.

4.1.2 Frequency Driven

In topic words representation, the words are assigned a binary value, one or zero. However, the possibilities here are much broader and that is where frequency driven methods come to light. The simplest approach assigns each word a raw frequency of occurrence in the source corpus. That score can be highly influenced by the length of the document as a word occurring twice in one sentence might be important, but not necessarily so in hundred sentences. **Word probability** overcomes that issue by computing score of each word w as a ratio between the number of the word's occurrences $n(w)$ and the total number of words N in the whole source document [2].

$$p(w) = \frac{n(w)}{N} \quad (4.1)$$

Such word probability has been used for example in the work of *Vanderwende et al.* [16] expanding an existing SumBasic system. SumBasic assigns each sentence s with total number of words n_s a salience score $S(s)$ equal to the average probability of the contained words $p(w)$.

$$S(s) = \frac{\sum p(w_i)}{n_s} \quad (4.2)$$

Selection of sentences for the final summary is then conducted in an iterative greedy way selecting the top scored sentence containing the word with current highest probability assuming that word represents the most salience topic [2]. For each word contained in the chosen sentence the probability is adjusted to deal with redundancy and reflect the iterative buildup of summary until its desired length is reached.

The word probability methods rely on a stop word list to eliminate the most common words carrying little information. Formation of such a list is controversial and demanding. Frequency method called **TF-IDF** (Term Frequency – Inverse Document Frequency) tackles the challenge of scoring corpus words without the need for such a list. It does so by comparing the number of occurrences of a certain word to its expected frequency learnt from a (multi-document) source corpus

$$TF-IDF = n(w) \log \frac{D}{d(w)} \quad (4.3)$$

where $d(w)$ represents the number of documents in the corpus containing the word w and D the total number of corpus documents. Instead of the nominal number of words in the document $n(w)$, a probability $p(w)$ can be used to compensate for the document's length. Based on the TF-IDF score, descriptive words are those that appear more often in the current document than in the whole corpus as words contained in most of the documents have very low IDF [2].

As TF-IDF provides a very good indication of word's salience and is not computationally demanding, it is often employed by many current systems as the initial words representation on which more complex sentence selection mechanisms are built.

4.1.3 Latent Semantic Analysis

Latent Semantic Analysis (LSA) is an unsupervised method representing the source text based on co-occurrence of words in sentences. Several variants of the algorithm exist today. *Gong and Liu* [17] initially proposed the method for identifying important topics in both single and multi-document cases. The intermediate representation is built by creating a matrix A from the source text of dimensions n words \times m sentences. Each element of the matrix a_{ij} represents a score of the i word in the sentence j . The score can be for example equal to TF-IDF and set to zero if the sentence does not contain the particular word. Linear algebra singular value decomposition (SVD) is then applied to the matrix A resulting in product of three distinct matrices.

$$A = U\Sigma V^T \quad (4.4)$$

Matrix U holds information about words and topics. Its dimensions are n words \times k topics and each element contains the weight of the word (row) in relation to a certain topic (column). Matrix Σ is a diagonal descending $k \times k$ matrix giving each topic from matrix U a certain weight with respect to the whole source document. Finally, matrix V^T ties topics in k rows to sentences in m columns.

From these matrices it is possible to extract an indication of how much a certain sentence represents a given topic resulting in matrix D where each element d_{kj} contains the score of topic k for sentence j .

$$D = \Sigma V^T \quad (4.5)$$

This approach allows for setting a certain threshold of importance under which all marginal topics in the source text can be omitted. Such dimensionality reduction can be done by removing last t rows of U , last t rows and columns from matrix Σ and last t rows of V^T simplifying the problem and focusing only on the information rich topics.

Original implementation of *Gong and Liu* [17] selects one sentence for each topic performing dimensionality reduction down to the number of sentences to be kept in the summary. This approach however does not count with cases where several sentences are needed to sufficiently describe a certain document's topic. Other approaches try to assign an overall weight to each of the topics and select the appropriate number of sentences based on that value or choose sentences which discuss several topics at once.

One of the most promising sentence selection heuristics has been introduced by *Ozsoy et al.* [18] under the name of *cross method*. They include a preprocessing step between the SVD and summary formation with the aim to remove less related sentences in each of the topics while keeping the most informative ones. For every topic represented by a row in V^T , the average score over individual sentences is calculated and then the values below this threshold are set to zero [18]. Length of a sentence vector represented by a column of V^T is then summed up and the longest sentence vectors are selected for the resulting summary.

The richness of various implementations nicely illustrates the advantages of splitting the summarization into three distinct steps. While retaining the exactly same intermediate

representation and employing different sentence selection algorithms, very diverse results can be achieved.

4.2 Abstractive Approach

As extractive methods build summary only from sentences already present in the source document. They do not create any new content nor paraphrase the existing one. Due to their ability to achieve satisfactory results despite relatively simple implementation, most of the research has been historically focused on extractive approaches. However, some of their typical drawbacks are lack of coherency and unnatural sentence composition. With the abundance of computational power and advancements in NLP, the research has been recently shifting towards the abstractive methods. They aim to reformulate the source text in an unseen form using vocabulary not originally present and thus better mimicking the work of humans. *Jing* [19] has shown that in 300 human written summaries, 19% of sentences were written from scratch and the rest rather by cutting and pasting, paraphrasing or reducing several sentences into one.

The abstractive methods are usually further classified into two main categories, the structure based and the semantic based approach. Moreover, most of the recent research is based on the uprise of deep learning. A basic overview of the main methods is provided below in accordance with [20], [21] and [22].

4.2.1 Structure Based

All structure based techniques employ prior knowledge and schemas such as templates, structures like tree or ontologies and extraction rules to encode the most salient information from the source document. The individual methods are named according to the information storing structure they utilize for representing the content.

Tree based methods represent the source document by a dependency tree consisting of phrases from the text. The summary is then curated with the help of various algorithms such as local alignment across pair of parsed sentences combined with a language generator [20]. The method can also be easily applied to multi-document summaries when the multiple trees are aligned with each other.

Template based methods have a set of linguistic patterns and extraction rules at their heart which together serve as a template representing the source text. Corresponding snippets from the document are matched with the template's slots to form a kind of a database of the content from which the final summary is generated. An advantage of template methods is that summaries are highly coherent. Example of this approach can be found in the project of *Gistexter* [23] which identifies topic related information from the document and appends them to a database from which it forms a summary.

Ontology based approach exploits the fact that most documents are confined to a certain topic domain. Each domain has its knowledge structure which can be represented by an ontology. We can create an ontology from the information provided by the source document and complete the missing or uncertain data with information from an existing knowledge base from the particular domain which results in a fuzzy ontology.

Graph based solutions utilize a structure where each node represents a word or an n-gram with directed edges connecting them together into sentences and paragraphs. Such graph is then searched for valid sub-graphs encoding a syntactically valid sentence while maximizing the contained information. From a set of found paths the final summary is then extracted. While the extraction might make it look like an extractive method, new phrases can be generated thanks to the exploration of new paths in the graph classifying it as an abstractive method [21].

4.2.2 Semantic Based

A second class of abstractive methods consists of algorithms using the semantic representation of a document. They identify noun and verb phrases which are then fed into a language generation module. Several subclasses can be distinguished and are briefly reviewed here.

Multimodal semantic methods build a representation of multiple topic documents by capturing main concepts and relations in between them. The trick is to employ the right heuristics scoring the salience of concepts which are then turned into sentences forming the final summary. The rating can be computed based on information density, completeness, links to other concepts or number of the expression's occurrences [21].

Information item based solutions generate the summary based on an abstract representation of the document rather than from its raw sentences. The unit of such structure is an information unit, a smallest element of coherent information in the text. For example a triplet of subject – verb – object retrieved from the source text by a syntactic analysis and a parser. As these units do not usually form a whole sentence, they have to be combined together. These newly built sentences are ranked based on their frequency score or other metric and arranged in a concise summary [22].

Semantic graph based models build in the first step a rich semantic graph where verbs and nouns of the input text are represented as graph nodes and the edges show semantic and topological relations among them. The created graph is then reduced based on a certain importance heuristics and the final summary is generated [22].

4.2.3 Deep Learning

Large part of progress in text summarization is heavily inspired by the advances in machine translation (MT) and speech recognition. The main idea is to map an input sequence to an output sequence representing the source as accurately as possible. These models are called sequence-to-sequence (seq2seq) models and they have proven to be successful in tasks ranging from machine translation or speech recognition to video captioning. Opposed to translation which is essentially a n-to-n problem regarding the relations between the input and output sequences length, summarization produces a text much shorter than the input. Finding the correct mapping is thus less straightforward. While most of the deep learning approaches to summarization utilize one or more of the structure and semantic methods introduced above, a unique group of applications is highlighted here.

Several sources providing the overview of current deep learning research state exist [24]. It shows that most of the approaches are based on seq2seq architecture of RNNs and LSTMs introduced for machine translation by *Bahdenau et al.* [3] and adopted by *Nallapati et*

al. [25] for summarization. They employ a bidirectional RNN as an encoder creating a new representation of the input text and uni-directional RNN as a decoder producing the summary. Several other tricks improving the results can be employed, such as introduction of a large vocabulary or an attention mechanism. Another great contribution for the research community has been their introduction of a CNN/Daily Mail dataset heavily used since then.

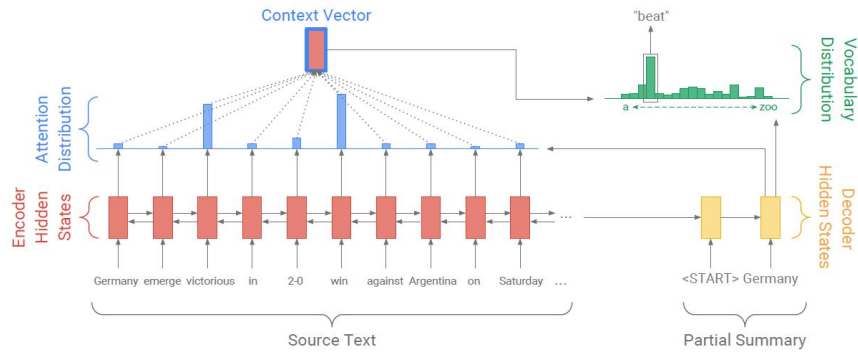


Figure 4.1: Baseline seq2seq model with attention [26]

See et al. [26] further improved the algorithm to be able to handle repetition of facts and detail inaccuracies in the summary. The mechanism named pointer generator networks allows for copying of unique words from the source text via pointing to them while most of the output text is still newly generated. The upgrade is clearly visible in Figures 4.1 and 4.2 showing the original and improved architecture respectively.

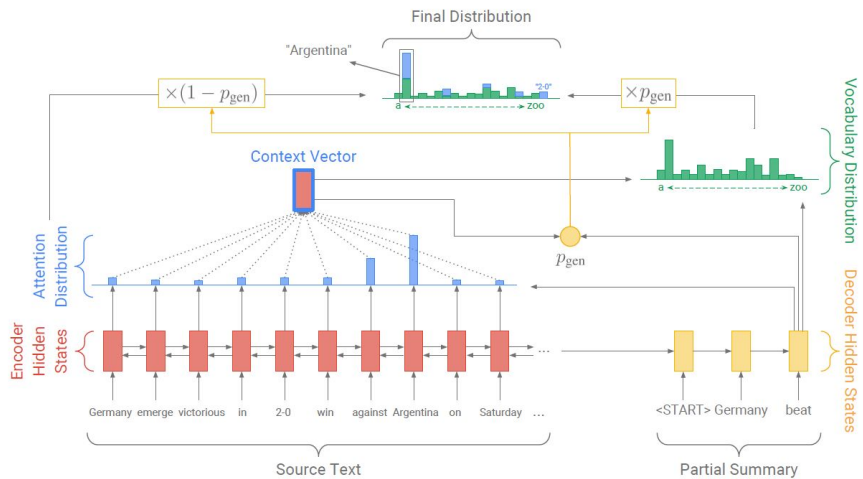


Figure 4.2: Pointer generator model with the ability to copy rare words directly from the source text while still generating most of the summary from scratch [26]

Attention mechanism has recently become a great part of the researched solutions. A recent paper published by *Cohan et al.* [27] focuses on summarization of long documents as opposed to shorter news texts in the preceding works. They do so by combining the RNN encoder of a seq2seq model described above with an attentive decoding mechanism. In order to deal with the longer texts, the encoder has to be able to capture hierarchy of the document's structure. It does so by encoding the source text at first at the word level and further on the whole section level. The attentive mechanism implemented also at both levels prevents repetition of the same phrases in the summary. The authors have named their model a *discourse-aware attention model* and its architecture can be explored in Figure 4.3. Some of the used mechanisms are however missing, for example the pointer copying. Due to the lack of longer documents summarization datasets, the authors have publicly released two new datasets collected from science article repositories arXiv.org and PubMed.com.

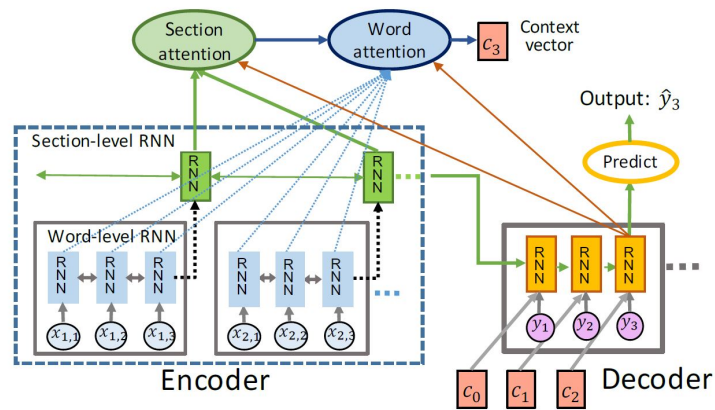


Figure 4.3: A discourse-aware attention model built on top of a pointer generator seq2seq architecture by *See et al.* [27]

Chapter 5

Implementation

One of the aims of this work is to experiment with text summarization algorithms and based on the obtained results propose an improvement suitable for the area of voice assistants. Two recent state of the art methods have been chosen, representatives of both extractive and abstractive methods. Following their description and implementation, our experience with a computation cluster employed throughout the testing is mentioned.

The extractive approach is represented by LSA method introduced by *Ozsoy et al.* [18]. Although the algorithm's results have been overcome many times since its release, it clearly illustrates the principles of extractive methods and sometimes still performs on par with more sophisticated implementations by elegantly utilizing the basic principles of linear algebra.

The pool of abstractive methods is represented by the discourse-aware attention model from *Cohan et al.* [27]. This algorithm represents according to the authors state of the art and has outperformed most of the current competition on selected datasets. Its function is not purely abstractive but rather benefits from a fusion of both extractive and abstractive approaches.

5.1 Latent Semantic Analysis

While the theory behind LSA method has been already deeply described in Section 4.1.3, only several implementation technicalities are mentioned here. The code is adopted from a Python module PyTLDR¹ by *Jai Juneja* and adjusted to our needs.

Python NLP platform NLTK² is utilized at the first step to tokenize sentences of the input text into words. From the words a matrix A is built with the help of `TfidfVectorizer` from scikit-learn, a machine learning library for Python³ which builds a sparse representation of words TF-IDF scores per sentences. After the source matrix is built, it is split by SVD into the three matrices, U , S and V^T . The topic average score is calculated and used to filter sentences below it. By a dot product between the squares of S and V^T the length vector is calculated and top sentences selected to be next returned in their original order.

¹PyTLDR: Automatic text summarization in Python, <https://github.com/jaijuneja/PyTLDR>

²NLTK: Python natural language toolkit, <https://www.nltk.org/>

³scikit-learn: Machine learning in Python, <https://scikit-learn.org/stable/>

A list of stop words from NLTK is used to filter out frequent and unimportant words during the representation matrix creation. Examples from the list can be “yourself”, “I”, “but”, “again”, “which” or “those”. Obviously, the words bear none or very little information about the meaning. The list of stop words has been further extended by words specifically occurring in the used datasets. While LSA works on the individual document level, we run it in a loop over the individual articles from a dataset. Two additional parameters are to be set for our method. The length of the final summary can be given either in the form of a ratio with respect to the original length such as 1/9 or by the desired number of sentences. Number of topics to be detected in the source document can also be set.

5.2 Discourse-Aware Attention Model

Authors of the article presenting the unique seq2seq architecture enhanced with an attention decoder [27] have open sourced their whole code for easier results reproduction. A part of the release are also instructions for obtaining the PubMed and arXiv datasets [28]. We have adopted their implementation and refer to the method by abbreviation DAAM.

The original code is run with identical configuration provided by the authors with the aim to reproduce their results. As the discourse-aware attention model code is originally based on *See et al.* [26] work which is also publicly available, the number of tips for running the code is quite high and helps during our setup. The model’s hyper parameters had been also already suggested there. With some variation they are defined by *Cohan et al.* [27] as follows: two bidirectional LSTMs with cell size of 256 and embedding dimensions of 128. Embeddings were trained from scratch (i.e. without pre-trained embeddings). Vocabulary size constrained to 50,000. Mini-batches set to size 16, the document length limited to 2000 and section length to 500 tokens, and number of sections to 4. Batch-padding and dynamic unrolling were used to handle variable sequence lengths in LSTMs. Training was done using Adagrad optimizer with learning rate 0.15 and an initial accumulator value of 0.1. The maximum decoder size was set to 210 tokens. The model is trained initially without coverage which is added at the last two epochs to help the model converge faster. Training is performed for about 10 epochs and each training step takes about 3.2 seconds. Beam search is used at decoding time with beam size of 4. The abstractive baselines are trained for about 250K iterations [27].

TensorFlow 1.4 was used for implementing the model’s architecture but the current TensorFlow’s version is 2.0⁴ and the API has changed greatly. Hence, we had to experiment with our development environment setup and finally ended with TensorFlow 1.8, which still supported the authors’ code. The batch size had to be decreased to 8 instead of 16 because of full memory issues. Due to the model’s complexity and limited space we suggest the reader to go through the well commented code by themselves and consult the original paper for theory and design choices whenever necessary.

Regarding data preprocessing, the PubMed dataset is one of the two originally used by the authors so no bigger problems have arisen during the training or evaluation phase. Preprocessing of the CNN/DM dataset into a format suitable for the model turned out

⁴Tensorflow: An end-to-end open source machine learning platform, https://www.tensorflow.org/api_docs/python/tf

to be a much bigger challenge. Both datasets are completely different, not only regarding the content but also in their form and length. *Cohan et al.* [27] take advantage of the clearly divided sections of PubMed scientific articles which is lacking in the CNN/DM news dataset. While we did not want to extensively alter the code in order to maintain result comparability, most of the work has been put into the preprocessing stage. For example, at least three sections are necessary for a document to be taken into consideration by the model. The news articles were thus split into several parts containing equal number of sentences. Minimum length of a section and abstract has also been decreased to accommodate the bullet points headings. Each article was then transformed into a JSON object containing abstract, article text, sections with their headings. The JSONs are then written in one of the training, evaluation or testing source files line by line.

5.3 Cluster

Training of neural networks and especially LSTMs utilized in the abstractive summarization approach is computationally demanding and cannot be performed on regular laptop. Instead a cluster of GPU (graphics processing units) has to be employed. We have gained access to one located at CIIRC, Czech Technical University in Prague, consisting of several high performance compute nodes.

To run the desired code on the cluster, two types of software are used. Environment module system Lmod⁵ simplifies the management of dependencies such as libraries or packages on the cluster. Slurm workload manager⁶ takes care of the queue of jobs submitted to the cluster and distributes the workload between the available nodes. Although a piece of code can be tested on the master node accessed over SSH, it is not recommended to run heavy computations there. The training has been carried out on nodes containing four GPUs GTX 1080 Ti 11 GB and RAM 192 GB providing enough computational power.

Before a job can be run on the cluster, resources have to be allocated via Slurm. To run a job, we need to submit a job script, a Bash file with special directives at the beginning specifying the number of CPUs, GPUs, required memory or the maximal runtime. Next the desired dependencies of the code are loaded via Lmod and the right environment is setup. Finally the command to run the code itself follows which can naturally include arguments to be specified. They make the cumbersome operation on the cluster much easier and are thus highly desirable.

When a job script is prepared, the job can be submitted to the queue by command `sbatch job_script.sh` which eventually runs the code in non-interactive batch mode. The output is written to a specified log file. An option to run jobs in an interactive mode also exists but is not requisite for our purposes.

⁵Lmod: New environment module system, <https://lmod.readthedocs.io/>

⁶Slurm: Workload manager, <https://slurm.schedmd.com/>

A reduced example of a job script might look like this:

```
#!/bin/bash
#SBATCH --job-name=pubmed_train
#SBATCH --output=pubmed_train.log
#SBATCH --cpus-per-task=1
#SBATCH --gres=gpu:4
#SBATCH --mem=80G
#SBATCH --time=2-00:00:00

module load CUDA/9.0.176-GCC-6.4.0-2.28
module load cuDNN/7.1.4.18-fosscuda-2018b
module load Anaconda3/5.0.1

conda activate seq2seq

python home/summ/run_summarization.py \
--mode=train \
--data_path=home/data/pubmed/train.bin \
--vocab_path=home/data/pubmed/vocab \
--log_root=pubmed_log \
--exp_name=exp_11 \
--batch_size=8 \
--vocab_size=50000 \
--optimizer=adagrad \
--coverage=False \
..
```

Chapter 6

Experiments

What is tricky about text summarization is that no ideal abstract of a text exists and everyone defines a good summary differently [29]. If we gave a document or a collection of documents to several people and ask them to summarize its content, the summaries would be far from identical. The lack of ground truth makes any evaluation of text summarization very challenging. Certain desirable qualities can still be identified and used for assessment.

A good summary should contain all the important information from the source text in a shorter form while not repeating itself. The definition of *importance* might further vary depending on the reader’s perspective. The final text has to be coherent, maintain context of the original document as well as correct grammatical structure. Readability should not be impaired. That is a lot of criteria to be fulfilled simultaneously. Datasets and evaluation metrics are presented here and later used for assessment of the tested methods’ results.

6.1 Datasets

To be able to compare results of different methods, two requirements must be fulfilled – an evaluation metrics and fixed conditions during all the experiments. The second ingredient is typically represented by a publicly available dataset consisting of samples and corresponding ground truth which can be used by any scientist to verify their method. Further, NNs and especially deep learning techniques are very greedy regarding the number of samples which can go to millions.

It turned out that not many curated datasets exist in the area of text summarization. What is needed is a raw text to process and at least one example of its summary written by a human so that an automatic evaluation and comparison is possible between the model and reference summaries. A good overview of available datasets has been provided by *Dernoncourt et al.* [30] with metadata about each of them. Based on the summarization methods chosen above, availability of the datasets and an attempt to introduce some variability in the source text type and summary length, a set of news articles and a database of medical scientific research papers have been selected.

6.1.1 CNN/Daily Mail

CNN and Daily Mail dataset is a collection of news articles paired with multi sentence bullet style summaries written by journalists which has been created by *Hermann et al.* [31] for question answering task and later modified by *Nallapati et al.* [25] for text summarization. In a whole, the corpus consists of 286,817 training pairs, 13,368 (4.7%) validation pairs and 11,487 (4%) test pairs and the source documents have on average 766 words spanning 29.7 sentences while the reference summaries consist on average of 53 words in 3.7 sentences [25], see Table 6.1. We use the original version of the dataset where name entities are kept and not replaced by IDs as in the anonymized version.

6.1.2 PubMed

As the news articles are relatively short, a second dataset is introduced in order to asses the methods' ability to summarize several pages documents which is where text summarization brings the greatest benefit. *Cohan et al.* [27] have introduced the PubMed dataset of medical scientific paper together with more technical arXiv archive. We use the first one of the two. They have downloaded scientific papers from the public repositories, discarded too short/long articles and utilized the more or less standard structure of the articles to extract typical sections hierarchy. Abstract is then used as the ground truth summary of the source text. The corpus consists of 119,924 training, 6,633 (5%) validation and 6,658 (5%) test pairs with the average document length of 3016 words split between 86.4 sentences and summary of 203 words in 6.7 sentences on average.

Although it was not intended since the beginning, a similarity between Wikipedia pages and scientific articles showed up. Both sources contain a lot of numerical values and expressions which are not typical for non-technical documents such as news articles. This will allow us to test our Wikipedia use case and evaluate it on the PubMed corpus.

Dataset	number of docs	avg doc length (sent)	avg doc length (words)	avg summ length (sent)	avg summ length (words)
CNN/Daily Mail	311 672	29.7	766	3.7	53
PubMed	133 215	86.4	3016	6.7	203

Table 6.1: Quantitative overview of the used datasets. Number of documents, average document length in sentences and words and average summary length in sentences and words for CNN/Daily Mail and PubMed dataset.

The variety between the datasets is obvious. Table 6.1 shows that PubMed articles are roughly four times longer word wise than CNN/DM. Their sentences are longer as well corresponding to the different writing style. Abstracts of scientific articles cannot be compared to bullet points form of news. Most of the current research focuses on one type of source documents and optimize their methods for one single use case. We believe that evaluating algorithms on such diverse test scenarios will provide us with better understanding of their advantages and weaknesses and make our results more convincing.

6.2 Metrics

Without the ability to measure, it is not possible to precisely assess, compare and quantify any results. As we can see, most of the criteria defining a good summary are purely qualitative and subjective. Four main evaluation methods trying to overcome this issue and score a produced summary have been selected.

6.2.1 Human

When the quality of a summary is so subjective and hard to quantify, evaluation by a human critic is an obvious option. There have been several campaigns in the US since late 1990s trying to establish standards and assess the quality of machine summaries. They include SUMMAC (1996-1998), DUC (Document Understanding Conference, 2000-2007) where judges were evaluating the coverage of the original document by a summary, and more recently TAC (Text Analysis Conference, 2008-present) with query based summaries being evaluated based on to what extent they answer the given question [32]. An assessment by a judge or preferably an aggregation over a plenary of judges results in the most natural human evaluation. However, its complexity and resource demands are very high. That is why researchers have been trying to develop an automatic evaluation metric similar to the human rating.

6.2.2 ROUGE

The most used set of metrics for summarization nowadays have been introduced in 2004 by *Lin* [33] under the name of Recall-Oriented Understudy for Gisting Evaluation (ROUGE). It measures the quality of an automatically generated summary by comparing it to the ground truth one(s) provided by humans. The score is based on the number of overlapping text units such as n-grams, word sequences or word pairs [33]. The original paper presents four main similar measures: ROUGE-N, ROUGE-L, ROUGE-W and ROUGE-S. All three main scores are implemented in default – recall, precision and resulting F1.

ROUGE-N is a measure comparing the number of n-grams (usually bi-grams and tri-grams) between the generated model summary and a reference summary or alternatively a set of them. The recall score is computed as

$$\text{ROUGE-n} = \frac{\text{number of matching n-grams}}{\text{number of n-grams in reference summaries}} \quad (6.1)$$

ROUGE-L employs longest common sequence (LCS) of both texts. The intuition tells the longer the LCS between two sentences, the more similar they are. It captures well similarity at the sentence level but suffers from the fact that it counts only the in-sequence words and thus alternative or shorter LCSs are not reflected by the score [33].

ROUGE-W is rewarding consecutive LCSs by adding weight to the longer sequences. It improves the ROUGE-L score to reflect similarity between texts better.

ROUGE-S or a skip bi-gram co-occurrence statistics allows for arbitrary gaps between words and thus provides more flexibility in the sentence structure. An extension is **ROUGE-SU** adding the possibility to skip uni-grams apart from bi-grams and thus allows insertion of multiple words between the first and the last words of the bi-grams. They do not have to form a consecutive sequences of words [32]. The maximum skip distance (number of inserted words) can be limited. Star symbol (*) signifies no set limit.

Lin [33] compared ROUGE results to three years of DUC conference data. From his conclusions it can be drawn for our purposes that (1) ROUGE-2, ROUGE-L, ROUGE-W, and ROUGE-S work well in single document summarization tasks, (2) ROUGE-1, ROUGE-L, ROUGE-W, ROUGE-SU4/9 perform great in evaluating very short headline-like summaries, (3) using multiple references usually improves correlation to human judgment [33]. Each test case is evaluated with the help of ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-SU*, both recall and F1 measures. We have found out that especially ROUGE-1 and ROUGE-SU* F1 scores serve as good indicators of performance in our experiments.

It might be worth to remind ourselves of how different accuracy measures are computed to understand why exactly recall and F1 have been chosen. Let's define True Positive (TP) samples as n-grams occurring both in reference and model summary, True Negatives (TN) which are in neither, False Positives (FP) as n-grams not in the reference but occurring in the model summary and False Negatives (FN) those in the reference which are not included in our produced model summary. Then accuracy, precision, recall and F1 scores can be computed as follows.

$$\text{accuracy} = \frac{TP + TN}{TP + FP + FN + TN} \quad (6.2)$$

$$\text{precision} = \frac{TP}{TP + FP} \quad (6.3)$$

$$\text{recall} = \frac{TP}{TP + FN} \quad (6.4)$$

$$\text{F1} = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}} \quad (6.5)$$

Extractive methods only select sentences already contained in the source document so we are more interested in the FN than FP ratio. It is desirable to construct summaries as similar to the reference ones as possible and the LSA method does not contain any mechanism substituting human creativity. Thus accuracy and precision are of lower importance to

us than recall, a sensitivity measure of correctly selected n-grams from the gold standard summary. As distribution of the four observation classes is unequal, F1 score is employed for compensation of different importance of FP and FN as explained above.

6.2.3 BLEU

BiLingual Evaluation Understudy (BLEU) was originally introduced as a method for automatic evaluation of machine translation by *Papineni et al.* [34]. However, the principle is very similar to evaluating summarization results and the metrics is thus often used for such purposes as well. BLEU measure is precision based and ranges from 0 to 1 similarly to ROUGE, lowest to highest which is demanding to achieve even for a human translator. The idea behind is to count the ratio of n-grams in the candidate translation overlapping with the reference [33] with addition of penalizing texts which are of different length than the reference.

The difference between ROUGE-N precision and BLEU score is the brevity penalty for long texts and the fact that BLEU computes matches between n-grams in model and reference for several n values as opposed to one fixed n in ROUGE. Both metrics are used in this work providing a complete palette of evaluation options.

6.2.4 Run Time

Integration of a text summarization algorithm with a chatbot place demands not only on precision and coherency of the summary but also on low latency of the communication. Another evaluation metric is thus introduced here, the measurement of summary generation computation time.

As LSA method is tested on regular personal laptop and discourse-aware attention model trained on a cluster with variable configuration, the comparison of their run times cannot be exact. It is also very well possible that other tasks are putting load on the CPUs during the experiment and thus interfering with the in-line measurements. Nevertheless, the run time values mentioned with every set of results will help us when deciding about suitability of each method for the chatbot.

6.3 Results

The results of several experiments are presented here. Extractive LSA and abstractive discourse-aware attention model are evaluated on the CNN/Daily Mail and PubMed datasets with the help of ROUGE and BLEU metrics together with qualitative human assessments. Computation time is measured in all cases. Further, the selected methods are fed with a Wikipedia web page content to evaluate their suitability for chatbot integration.

Different datasets give preference to extractive or abstractive methods. Each summarization method is suitable for either short or long texts. ROUGE scores are said to favor extractive methods [26]. It is hard if not impossible to present a fair comparison between the methods. Nevertheless, we consider the various testing conditions presented here to be a unique added value of this work.

6.3.1 Latent Semantic Analysis

A great advantage of Latent Semantic Analysis is its low computational requirements. As each source text is summarized independently on others, there is no need to provide any huge dataset for a model training. Vocabulary is built only from the article being summarized and does not use the whole corpus. The length of the produced summary is set to two and five sentences for CNN/DM and PubMed datasets respectively. We have also experimented with the parameter of number of topics to be identified in the source document. The highest scores have been obtained with just one topic although that may not sound reasonable at the first sight.

Dataset	ROUGE-1		ROUGE-2		ROUGE-L		ROUGE-SU*		BLEU
	R	F1	R	F1	R	F1	R	F1	
CNN/DM	0.2611	0.2547	0.0762	0.0739	0.1753	0.1706	0.0783	0.0678	0.0630
PubMed	0.3230	0.3123	0.0928	0.0910	0.1856	0.1802	0.1230	0.1061	0.0809

Table 6.2: Results of LSA method on CNN/Daily Mail and PubMed datasets according to ROUGE-1, ROUGE-2, ROUGE-L and ROUGE-SU*, recall and F1 scores and BLEU metric.

Several observations can be made about the results in Table 6.2. ROUGE-1 score is naturally higher than ROUGE-2 as single words are more probable to match between the model and reference summaries than are bi-grams. As ROUGE-SU* allows for insertion of unlimited number of words between the first and last word of every bi-gram, the score is higher than ROUGE-2. Better results are generally achieved for PubMed than for CNN/DM dataset. That might be caused by the fact that scientific articles are longer and contain more technical terms than news articles which which LSA might identify better as important.

Looking at the pure numbers does not say much about the quality of the summaries produced by LSA but serve as a good basis for comparison between the different methods. The results achieved by our implementation are comparable to what can be found in other sources [35]. ROUGE score is based only on exact text overlap and neglects content overlap potentially obscured by humans choosing original words of similar meaning when writing their summaries. The coherency of the produced snippets also cannot be assessed by the quantitative metrics. From those reasons two particular outcome examples are mentioned here for illustration. Each of them consists of the original article, the reference human summary and a model summary produced by the LSA algorithm.

In the first case in Table 6.3 the algorithm has very well mimicked the human author and included the very same pieces of information found in the source text. However, as an extractive method, it naturally lacks the ability to reformulate the original sentences. The resulting model summary is thus slightly longer than the reference one containing additional information. In general, it highly depends on the specific context of the reader what kind of data and thus parts of the sentences are unnecessary and should be omitted and which are important. This time the reader can clearly understand the message of the article. As is desirable for a sport event coverage, all important scores and players' names are included.

Not only the individual sentences are well chosen, also their order seems natural and the summary is quite coherent.

Such conclusions are only supported by ROUGE scores being much higher for the particular article than the average values in Table 6.2. ROUGE-1 F1 scores 0.4898 which is roughly two times higher than the average across the whole dataset. However, when we look at ROUGE-2, the difference already significantly increases. Even metrics focused on match of longer phrases from the text prove that LSA is able of producing summaries of relatively high quality such as the example in Table 6.3 with ROUGE-SU* F1 equal to 0.1803.

SOURCE	<p>andy murray marched into the third round of the bnp paribas open at indian wells with a routine 6-1 6-3 win over canadian vasek pospasil . the scot raced through the opening set , carving out five break points and taking three , before another two breaks of serve in the second earned him victory in one hour and 19 minutes . his opponent , the world number 62 , did not trouble the 27-year-old , making only 46 per cent of first serves . andy murray is all smiles after sealing his progress into the last 32 of indian wells on saturday . andy murray -lrb- right -rrb- shakes hands with vasek pospasil having beaten the canadian 6-1 , 6-3 at indian wells . murray took full advantage to set up a meeting with germany 's philipp kohlschreiber . ' i thought i did quite well , ' the british number one said on sky sports 3 . ' he did n't serve so well today . ' i thought i played quite solid . i was using my forehand well . ' murray put in an assured performance to defeat the world no 62 in just under 80 minutes . murray admitted the contrast in atmosphere from great britain 's davis cup tie against the united states in glasgow last weekend took some getting used to . ' it 's a completely different vibe on the court , ' he said . ' it 's extremely different to what it was like in davis cup . i tried to give myself some positive energy – that helped a little bit , but it was tough . ' murray , who earlier this week revealed he was set to add jonas bjorkman to his coaching team , said he would like the davis cup quarter-final with france in july to be held at queen 's club . the british no 1 celebrates his comprehensive win by hitting some signed tennis balls into the crowd .</p>
REF	<p>the british no 1 defeated the canadian youngster 6-1 , 6-3 . moves into the indian wells third round to play philipp kohlschreiber . murray is set to add jonas bjorkman to his coaching team .</p>
MODEL	<p>andy murray marched into the third round of the bnp paribas open at indian wells with a routine 6-1 6-3 win over canadian vasek pospasil . murray , who earlier this week revealed he was set to add jonas bjorkman to his coaching team , said he would like the davis cup quarter-final with france in july to be held at queen 's club .</p>

Table 6.3: Example of a good summary of CNN/DM article produced by the LSA. Scores ROUGE-1 F1 0.4898 and ROUGE-SU* F1 0.1803.

Unfortunately, LSA does not always perform so well. Below in Table 6.4 we can see a summary produced by the method which bears resemblance to the reference resulting in high ROUGE scores. ROUGE-1 F1 overcomes the average more than twice with

value of 0.5319, ROUGE-2 reaches even higher meaning many more than usual matching bi-grams were found between the reference and model summaries. With some leeway in the word structure, ROUGE-SU* F1 reaches 0.2192.

SOURCE	<p>by . matt blake . published : . 06:08 est , 3 october 2012 . . updated : . 08:56 est , 3 october 2012 . these ghastly photographs are likely to cause ripples across the art world . they were taken by andrey antov who spent ten years building his collection of extraordinary images by photographing rock formations reflected in water . one , taken at the rila and pirin mountains in bulgaria , earned the nickname the ‘ devil ’ when mr antov spotted a demonic face after he rotated it 90 degrees . devilish : this picture , taken at the rila and pirin mountains in bulgaria , earned the nickname the ‘ devil ’ when mr antov spotted a demonic face in the rock . another bears an uncanny resemblance to a llama , with tufts of grass making up the animal ’s furry coat . mr antov , from oxford in connecticut , america , said he did n’t rely on the computer to enhance the shots , simply using his camera on a tripod . the 36-year-old explained : ‘ it all started when i went on a field trip and took the image of the ‘ devil ’ . ‘ i noticed the face by chance a few days later after developing the image and turning it 90 degrees on one side . art in nature : this picture , left , bears an uncanny resemblance to a llama , with tufts of grass making up the animal ’s furry coat while the other , right , could be a bird ? reflections of another world : mr antov , from oxford in connecticut , . america , said he did n’t rely on the computer to enhance the shots , . simply using his camera on a tripod . ‘ afterwards this i began to find more faces in the rocks . ‘ it ’s taken me more than 10 years to build up the collection because the images are not easy to spot . ‘ i try to spot the faces when there is a perfect reflection . this is more likely in a lake and when there is no wind . ‘ then i place my camera on a tripod and set the self timer . ‘ it ’s pretty simple , it does n’t involve photoshop or other alteration techniques . ’ he added : ‘ i ’ve had a great reaction to the pictures . people have been very surprised that it is possible . ’</p>
REF	<p>andrey antov , from oxford , connecticut , spent ten years on the collection . he did n’t rely on the computer to enhance the shots , simply a tripod . they are all shots of rocks reflected in water then rotated 90 degrees .</p>
MODEL	<p>one , taken at the rila and pirin mountains in bulgaria , earned the nickname the ‘ devil ’ when mr antov spotted a demonic face after he rotated it 90 degrees . mr antov , from oxford in connecticut , america , said he did n’t rely on the computer to enhance the shots , simply using his camera on a tripod .</p>

Table 6.4: Example of a bad summary of CNN/DM article produced by the LSA. Scores ROUGE-1 F1 0.5319 and ROUGE-SU* F1 0.2192.

Despite the exceptionally high ROUGE scores, the text is incomprehensible. The main motive of the source article stays unclear as the algorithm gives higher priority to long sentences describing specific details instead of focusing on the main idea behind the story. That corresponds to what has been said about the first example in Table 6.3, reader’s context is very important for the final impression of the summary. And while it might be desirable to

mention exact score and names of the players in the case of a sport match coverage, it does not make much sense in a more descriptive kind of an article as in the second case. Sense for such fine nuances are inherent to humans but pose a great challenge for artificial solutions. The example also illustrates the gap between automatic scoring and human qualitative evaluation which should thus never be omitted.

According to the author’s knowledge, there is no dataset allowing for quantitative evaluation of Wikipedia pages summarization. Pure human qualitative assessment is the only way how to test suitability of the LSA method. The algorithm is applied to the individual subsections of the Wikipedia page about Earth with the aim to identify their most characteristic sentences. They will be then used by the chatbot from Part I to return relevant information about the topic. Although sections and subsections are often further divided into paragraphs, this differentiation is not considered initially as the topic of individual paragraphs is hard to identify without a provided heading. The outcomes are again presented through two examples consisting of a section name, its original content and a summary provided by our LSA model with a set length of two sentences.

SECTION	Name and etymology
SOURCE	<p>The modern English word Earth developed from a wide variety of Middle English forms, which derived from an Old English noun most often spelled <i>eorðe</i>. It has cognates in every Germanic language, and their proto-Germanic root has been reconstructed as <i>erþō</i>. In its earliest appearances, <i>eorðe</i> was already being used to translate the many senses of Latin <i>terra</i> and Greek <i>gē</i>: the ground, its soil, dry land, the human world, the surface of the world (including the sea), and the globe itself. As with <i>Terra</i> and <i>Gaia</i>, Earth was a personified goddess in Germanic paganism: the Angles were listed by Tacitus as among the devotees of Nerthus, and later Norse mythology included <i>Jörð</i>, a giantess often given as the mother of Thor.</p> <p>Originally, earth was written in lowercase, and from early Middle English, its definite sense as “the globe” was expressed as the earth. By Early Modern English, many nouns were capitalized, and the earth became (and often remained) the Earth, particularly when referenced along with other heavenly bodies. More recently, the name is sometimes simply given as Earth, by analogy with the names of the other planets. House styles now vary: Oxford spelling recognizes the lowercase form as the most common, with the capitalized form an acceptable variant. Another convention capitalizes “Earth” when appearing as a name (e.g. “Earth’s atmosphere”) but writes it in lowercase when preceded by the (e.g. “the atmosphere of the earth”). It almost always appears in lowercase in colloquial expressions such as “what on earth are you doing?”</p>
SUMMARY	<p>As with <i>Terra</i> and <i>Gaia</i>, Earth was a personified goddess in Germanic paganism: the Angles were listed by Tacitus as among the devotees of Nerthus, and later Norse mythology included <i>Jörð</i>, a giantess often given as the mother of Thor. Originally, earth was written in lowercase, and from early Middle English, its definite sense as “the globe” was expressed as the earth.</p>

Table 6.5: Example of a good summary of Wikipedia page section produced by the LSA

In the first case in Table 6.5, LSA selected the last sentence of the first paragraph and last sentence of the second paragraph. That was probably caused by the simultaneous occurrence of several unique words and names with low frequency in the source text, such as *paganism* or *Thor*. If a human was to mimic an extractive approach and select two sentences representing the section, these would probably end up between the hot candidates. However, the first one is possibly getting too much into the detail and thus for example the first sentence of the section might be a better choice.

The second example in Table 6.6 illustrates how LSA is not at all flawless. The algorithm has selected two sentences containing very specific words, such as *mantle* or *lithosphere* which do occur often in the source text. However, the context is missing in the produced summary and reader is served with specific details that are hard to comprehend without the general overview of the section. An human adopting an extractive approach would probably choose the first sentence of the paragraph accompanied maybe by the last one leaving the technicalities out of the short summary.

SECTION	Internal structure
SOURCE	Earth's interior, like that of the other terrestrial planets, is divided into layers by their chemical or physical (rheological) properties. The outer layer is a chemically distinct silicate solid crust, which is underlain by a highly viscous solid mantle. The crust is separated from the mantle by the Mohorovičić discontinuity. The thickness of the crust varies from about 6 kilometres (3.7 mi) under the oceans to 30–50 km (19–31 mi) for the continents. The crust and the cold, rigid, top of the upper mantle are collectively known as the lithosphere, and it is of the lithosphere that the tectonic plates are composed. Beneath the lithosphere is the asthenosphere, a relatively low-viscosity layer on which the lithosphere rides. Important changes in crystal structure within the mantle occur at 410 and 660 km (250 and 410 mi) below the surface, spanning a transition zone that separates the upper and lower mantle. Beneath the mantle, an extremely low viscosity liquid outer core lies above a solid inner core. The Earth's inner core might rotate at a slightly higher angular velocity than the remainder of the planet, advancing by 0.1–0.5° per year. The radius of the inner core is about one fifth of that of Earth.
SUMMARY	The crust is separated from the mantle by the Mohorovičić discontinuity. The crust and the cold, rigid, top of the upper mantle are collectively known as the lithosphere, and it is of the lithosphere that the tectonic plates are composed.

Table 6.6: Example of a bad summary of Wikipedia page section produced by LSA

Overall, the performance of LSA highly depends on the source text. We have shown that it is able to capture the general concepts of a source text at some occasions. It seems like the text has to be more descriptive and contain less technicalities and numbers, see Table 6.5. Even in the opposite cases, LSA might achieve great results if the exactness in the final summary is desirable, such as in the sport event coverage in Table 6.3. However, in other cases the general concept and overview are blurred by the article's specific terminology, a good instance of such behavior can be found in Tables 6.4 and 6.6. Example summaries of

PubMed datasets are not included from space saving reasons. From the comparison of our good and worse examples of LSA application we can roughly assume the performance limits.

A criterion which should not be neglected is the computational demands and mainly run time. LSA method summarizes each source document independently on any grater corpus which serves as a great advantage in domains where large datasets are missing. Due to its relative simplicity and no model training required, LSA implementation has been run on a standard laptop with configuration Intel i5-6200U 2.30 GHz, RAM 8GB. We have been able to summarize CNN/DM articles at the pace of 0.0056 *s/doc* and more than three times longer PubMed documents at roughly two times slower rate of 0.0136 *s/doc*. Run time is not directly proportional to the source text length which can pose an advantage in case of long input documents.

6.3.2 Discourse-Aware Attention Model

We run the model with similar configurations on both CNN/DM and PubMed datasets varying only when necessary. As has been already indicated in Section 5.2 the model is not optimized for CNN/DM dataset as opposed to the original PubMed. Better results might have been obtained after some tuning. However, the mutual comparison would not have been very apt anymore and particularly better results were not obtained during later tuning experiments.

Dataset	ROUGE-1		ROUGE-2		ROUGE-L		ROUGE-SU*		BLEU
	R	F1	R	F1	R	F1	R	F1	
CNN/DM	0.2565	0.2697	0.0687	0.0722	0.2351	0.2473	0.0692	0.0629	0.0464
PubMed	0.2250	0.2815	0.0617	0.0785	0.2043	0.2557	0.0582	0.0779	0.0425

Table 6.7: Results of DAAM method on CNN/Daily Mail and PubMed datasets according to ROUGE-1, ROUGE-2, ROUGE-L, ROUGE-SU*, recall and F1 scores and BLEU metric.

For all metrics in Table 6.7, recall score is higher for the CNN/DM dataset and F1 on the other hand for PubMed. However, the difference is not significant enough to make any conclusions. ROUGE-1 is again naturally higher than ROUGE-2. ROUGE-SU* scores are at a similar level as ROUGE-2 indicating that allowing unlimited number of words to extend bi-grams does not increases the match between reference and model summaries. Interestingly, both ROUGE-L scores are quite high compared to other metrics. That might be explained by the model creating long common sequences similar to the reference human summaries.

We were not able to fully replicate the results achieved in the original paper, specifically for PubMed dataset ROUGE-1 F1 0.3893, ROUGE-2 F1 0.1537 and ROUGE-L F1 0.3521. ROUGE-SU* and recall scores are not stated by the authors. Finding a reason why the same performance was not achieved was challenging. Consulting other researchers experience and the authors' suggestions at the GitHub forums did not help. The model might have been trained with slightly different coefficients or their values were altered during the process. According to the tips provided by *See et al.* [26] and *Cohan et al.* [27] we have tried to rerun

the training several times with different setups. For example, the dimension of encoder and decoder can be reduced at the beginning of the training phase. After every couple of epochs it can be gradually increased up to the indicated values. Such approach speeds up the learning process and allows for more iterations resulting potentially in better results.

From those reasons the much more sophisticated abstractive approach actually achieved results comparable to the extractive LSA which scores can be found in Table 6.2. Discourse-aware attention model scores slightly higher in most of the ROUGE metrics with the exception of ROUGE-L where the advantage compared to LSA is significant. However, training of the model and decoding the outcome summaries themselves is very time consuming which gives advantage to a much quicker LSA. The training of the model took around four full days on the cluster, see Section 5.3. Each training step took around 3.5 s. The smaller encoder/decoder dimensions accelerated the training but the loss settled again after three days and did not bring any improvement to ROUGE scores. The summary generation took tens of seconds for both datasets which is incomparable to the LSA.

REF background : whole - genome comparisons can provide great insight into many aspects of biology . until recently , however , comparisons were mainly possible only between distantly related species . complete genome sequences are now becoming available from multiple sets of closely related strains or species. results:by comparing the recently completed genome sequences of vibrio cholerae , streptococcus pneumoniae and mycobacterium tuberculosis to those of closely related species – escherichia coli , streptococcus pyogenes and mycobacterium leprae , respectively – we have identified an unusual and previously unobserved feature of bacterial genome structure . scatterplots of the conserved sequences (both dna and protein) between each pair of species produce a distinct x - shaped pattern , which we call an x - alignment . the key feature of these alignments is that they have symmetry around the replication origin and terminus ; that is , the distance of a particular conserved feature (dna or protein) from the replication origin (or terminus) is conserved between closely related pairs of species . statistically significant x - alignments are also found within some genomes , indicating that there is symmetry about the replication origin for paralogous features as well.conclusions:the most likely mechanism of generation of x - alignments involves large chromosomal inversions that reverse the genomic sequence symmetrically around the origin of replication . the finding of these x - alignments between many pairs of species suggests that chromosomal inversions around the origin are a common feature of bacterial genome evolution .

MODEL background : the aim of this study was to evaluate the effects of closely related bacterial genomes in the duplication of whole genomes . methods : this study was performed in this cross - sectional study . the subjects were divided into two groups (n = 10) , which was treated with the x - whitney u test (n = 30) , which was performed in the study . the results showed that the genomes were significantly higher in the group than in the group than those with related bacterial species . in this study , we found that large - scale genomic duplications might be used to promote arabidopsis thaliana .

Table 6.8: Example of a summary of PubMed article produced by the DAAM. Scores ROUGE-1 F1 0.2277 and ROUGE-SU* F1 0.0426.

The quality of the produced summaries is illustrated by an example from the PubMed dataset in Table 6.8. For the length of the original scientific articles, only the reference and model summaries are shown without the source text itself. Even from the abstract it is obvious how technical the articles are. The trained model is thus very domain specific as can be seen later. The algorithm has proven its abstractive abilities to form new sentences unseen in the original document. Summary nicely follows the structure of a scientific article abstract. Sentences are not necessarily fluently connected to each other which would be unwelcome in other types of texts but is typical for scientific articles. The ROUGE scores for this particular document fall generally below average with ROUGE-1 F1 value 0.2277 or ROUGE-SU* F1 score 0.0426 showing that although the generated model summary might seem reasonable, the match with the reference abstract is relatively low. Possible cause might be the fact that model summary is shorter than the reference one as set number of sentences of the summary is fixed upfront and stays the same for every document in the dataset. In other documents frequent repetition of certain words or whole phrases was observed even the well known problem of abstractive methods should have been solved by the employment of attention coverage in the decoder according to *Cohan et al.* [27].

SECTION	Name and etymology
SOURCE	<p>The modern English word Earth developed from a wide variety of Middle English forms, which derived from an Old English noun most often spelled eorðe. It has cognates in every Germanic language, and their proto-Germanic root has been reconstructed as erþō. In its earliest appearances, eorðe was already being used to translate the many senses of Latin terra and Greek gē: the ground, its soil, dry land, the human world, the surface of the world (including the sea), and the globe itself. As with Terra and Gaia, Earth was a personified goddess in Germanic paganism: the Angles were listed by Tacitus as among the devotees of Nerthus, and later Norse mythology included Jörð, a giantess often given as the mother of Thor.</p> <p>Originally, earth was written in lowercase, and from early Middle English, its definite sense as “the globe” was expressed as the earth. By Early Modern English, many nouns were capitalized, and the earth became (and often remained) the Earth, particularly when referenced along with other heavenly bodies. More recently, the name is sometimes simply given as Earth, by analogy with the names of the other planets. House styles now vary: Oxford spelling recognizes the lowercase form as the most common, with the capitalized form an acceptable variant. Another convention capitalizes “Earth” when appearing as a name (e.g. “Earth’s atmosphere”) but writes it in lowercase when preceded by the (e.g. “the atmosphere of the earth”). It almost always appears in lowercase in colloquial expressions such as “what on earth are you doing?”</p>
SUMMARY	we report a case of a giantess - old man with a history of middle word Earth word Earth , which was treated with of the of the earth on the basis of the was performed

Table 6.9: Example of a summary of Wikipedia page section produced by the DAAM

Despite the lack of available data did not allowed for training the model particularly for the Wikipedia use case, not so precise summaries of its sections might still be generated by a model trained on a different dataset. After processing the webpage into a suitable format, we run the decoding process whose results can be assessed at least qualitatively. Same section of the page about Earth is chosen for comparison with the LSA outcome in Table 6.5. The model trained on PubMed dataset fails utterly to produce any acceptable summary of the source text as we can see in Table 6.9. This single example is no exception among other sections of the webpage. Phrases used in scientific articles comically sneak into the text and no context or information from the original documents are present. Additional experiments and different setups would be necessary to better assess the abilities of the discourse-aware attention model. The original authors have shown that it is able to achieve state of the art results on newly presented datasets in comparison with several other methods. Transferability of the trained model into new domains seems to be at least challenging based on our tests with Wikipedia. The model is built on top of a certain vocabulary and semantic relations which differ widely between such specialized documents.

Although the mechanism of neural networks is understood, the intermediate results and their final outcomes are hard to predict upfront. That is applicable to one of the recent advancements in deep learning, the attention mechanism employed by *Cohan et al.* [27]. Any option to grasp the decision process inside the algorithm is thus warmly welcomed. A publicly available attention visualization tool developed by *See* [26] is mentioned here with the hope to inspire the reader to get more insight into the results.



Figure 6.1: Attention Visualizer showing a sample PubMed article, reference abstract and generated summary produced by the DAAM

The visualizer shows the distribution of attention in a text based sequence-to-sequence tasks such as translation or summarization [36]. As the user hovers their mouse over the decoded summary words, the source text is overlaid with an attention heat map. For networks utilizing the pointer mechanism, generation probability of each decoded word is displayed as well [36]. The original DAAM code produces source files needed for attention visualization so we can run the tool and observe the behavior of the decoder.

Chapter 7

Improvement Proposals

After experimenting with selected text summarization methods, we have arrived at the stage where possible improvements might be proposed based on the gained experience. The aim is to try to perfect a chosen method in accordance with the overall objective of this work, summarization of text for chatbot, namely a Wikipedia page.

Although the DAAM approach is much more recent and theoretically promising better results, both training and summary generation proved to be very time consuming. Moreover, its focus on one particular domain defined by the training dataset does not make it a good candidate for our further work. The chatbot should be as universal as possible regarding a source text. From those reasons LSA method has been chosen as the candidate for enhancement attempts offering clear functioning principle, domain transferability and low computation time.

Two main ways of altering the LSA algorithm appear. The heuristics for selecting sentences for the final summary can take on many forms from which several have been already described by *Ozsoy et al.* [18]. We have thus decided to experiment with the construction of an input matrix A representing relations between sentences and words. In this work, the matrix is originally built from TF-IDF scores. While that method provides information about the word occurrence in the document, it does not capture any semantic information and mutual relations between the words.

We propose a new representation of the sentences and words matrix by employing well proven word embedding language models. Instead of representing a word by its frequency, it can be embedded into a continuous vector space where semantically similar words are mapped nearby each other. The principle is often illustrated with the help of a symbolic equation $\text{KING} - \text{MAN} + \text{WOMAN} = \text{QUEEN}$ showing how capable is the model of learning semantic relations between words simply from raw text. According to our best knowledge, it is the first attempt to combine LSA method with word2vec model. Several variants consisting of various pre-trained models and sentence vector construction are presented here. Implementation highlights are mentioned together with comparison of results with the existing solutions.

7.1 Implementation

The proposed improvement is extending an already implemented LSA algorithm and the code should be altered as little as possible for later results comparison. Preprocessing step is preserved from earlier. Too short sentences or stopwords are filtered from the source text which is then tokenized into individual sentence and words. While the length of the documents from CNN/DM or PubMed dataset does not allow for training a word embedding model, two pre-trained ones are tested: **glove-wiki-gigaword-50** built from vectors of length 50 based on Wikipedia 2014 dump and Gigaword containing 5.6B tokens and 400K vocab¹ and **word2vec-google-news-300** trained on 3M words and phrases from the Google News dataset with dimension 300² based on the word2vec model first introduced by *Mikolov et al.* [37]. The two models have been chosen with respect to their distinct and far dimensions as well as different corpora used during training. The variety of conditions should make the assessment of results more credible. Both of the models can be found aside many others in a repository uniting data for Gensim library³. Gensim is a Python NLP library focusing on long texts processing which can be used for topic modeling, document clustering or similarity assessment [38]. The library is employed in the work for the word embeddings manipulation.

Remembering matrix A , it represents the relations between sentences from the source text in columns and words in rows. A score is assigned to each of the intersections, in this case the similarity between word and sentence vector. Words present in the pre-trained model already have their vectors. Regarding vectors of sentences, we experiment with two ways of obtaining them. Our so called sent2vec method is computed either as an **average of word vectors** or as an **average of word vectors weighted by each word's TF-IDF**. Average of the vectors is used in both cases instead of pure sum to compensate for varying sentences length. The TF-IDF algorithm has been reimplemented based on the theory from Section 4.1.2 in order to have a complete control over its function.

Individual values of matrix A are set to the cosine similarity between the intersecting sentence and word vectors. The resulting value is scaled instead of a usual interval $(-1, +1)$ to $(0, 1)$ accommodating former conventions. It may happen that some unique words are not present in the word embeddings pre-trained model and thus their vector is unknown. These out of vocabulary words are omitted from the sentence vector construction and relevant cells of A are set to zero. From the constructed matrix, standard procedure of the LSA follows with SVD and summary sentences selection as in the original implementation.

7.2 Results

Two choices of a word2vec embedding model and two variants of sent2vec computation give four possible test cases in total described in Table 7.1. They are compared among each other and with the original LSA and DAAM implementations evaluated in Section 6.3 on

¹GloVe: Global vectors for word representation, <https://nlp.stanford.edu/projects/glove/>

²Word2vec: Tool for computing continuous distributed representations of words, <https://code.google.com/archive/p/word2vec/>

³Data repository for pre-trained NLP models and NLP corpora, <https://github.com/RaRe-Technologies/gensim-data>

the CNN/DM and PubMed datasets with the help of introduced ROUGE metrics, mainly ROUGE-1 and ROUGE-SU* F1 scores.

Config	embedding model	sent2vec method
config 1	glove-wiki-gigaword-50	averaging
config 2	glove-wiki-gigaword-50	TF-IDF weighted averaging
config 3	word2vec-google-news-300	averaging
config 4	word2vec-google-news-300	TF-IDF weighted averaging

Table 7.1: Configurations of the LSA word embeddings experiments

The results are summarized in Tables 7.2 and 7.3. It can be seen that the more complex the approach gets, the better ROUGE F1 scores are obtained in all metrics. Highest match between the reference and model summary is secured by word2vec-google-news-300 embedding model with the combination of sent2vec computed via averaging of word vectors weighted by words' TF-IDF. However, outcomes of all four tested approaches are very similar and insignificant. The difference stays in the range of units of percents. Both ROUGE-1 and ROUGE-SU* F1 scores also correlate strongly with each other in all of the six tested cases and no anomalies are observed. DAAM performs relatively better on PubMed dataset than on CNN/DM compared to the various LSA word embedding configurations similarly to the original LSA implementation which stems from its domain specialization.

Method	ROUGE-1		ROUGE-2		ROUGE-L		ROUGE-SU*		BLEU
	R	F1	R	F1	R	F1	R	F1	
LSA	0.2611	0.2547	0.0762	0.0739	0.1753	0.1706	0.0783	0.0678	0.0630
DAAM	0.2565	0.2697	0.0687	0.0722	0.2351	0.2473	0.0692	0.0629	0.0464
config 1	0.2686	0.2495	0.0703	0.0650	0.1746	0.1619	0.0791	0.0625	0.0571
config 2	0.2708	0.2525	0.0723	0.0671	0.1762	0.1640	0.0807	0.0641	0.0605
config 3	0.2826	0.2517	0.0740	0.0653	0.1812	0.1608	0.0867	0.0622	0.0540
config 4	0.2847	0.2555	0.0760	0.0674	0.1829	0.1633	0.0883	0.0643	0.0562

Table 7.2: Results of the LSA word embeddings experiments on CNN/DM dataset compared with the original LSA and DAAM implementations.

Utilization of the word embeddings introduces additional steps in the processing such as computation of the word and sentence vectors and finding their similarity. Altogether they slow down the average processing time per document to 0.6625 *s/doc* for CNN/DM and 3.6287 *s/doc* for PubMed dataset on the same personal laptop as before. Loading of the pre-trained embedding model into memory is also very time consuming reaching tens of seconds. It is done only once at the beginning of processing the whole set of data and is thus not included in the average run times. The relative difference between CNN/DM and PubMed articles processing times is roughly two times higher when compared to the original LSA implementation.

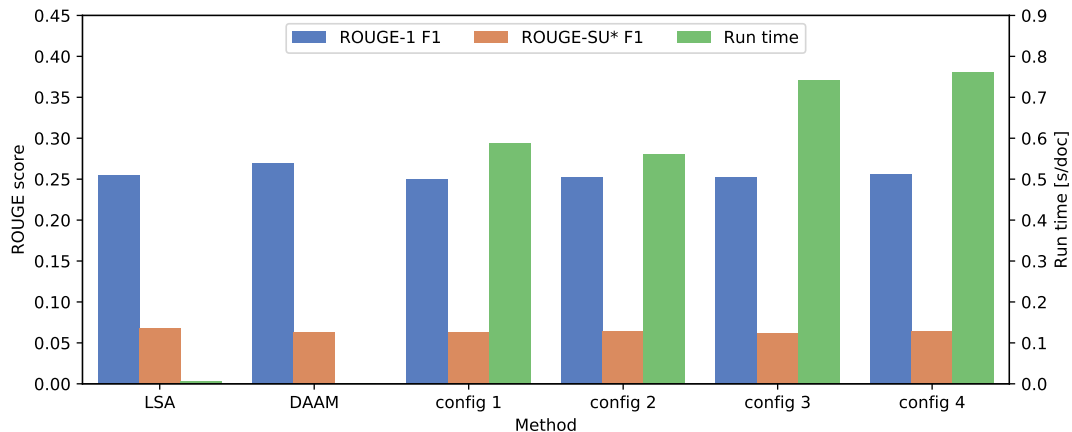


Figure 7.1: Results of the LSA word embeddings experiments on CNN/DM dataset compared with the original LSA and DAAM implementations. Run time per document is not stated for DAAM due to incomparable hardware.

Although the increasing complexity of test configurations does not significantly improve the ROUGE scores, the step up in the word2vec embedding model brings obviously longer processing times. Summarization of a document from CNN/DM dataset using a model of dimension 50 takes 0.5739 *s/doc* in average and model with vector length 300 already 0.7512 *s/doc*, a rise of roughly 30 % visible in Figure 7.1. The increase is observable in a similar way for the PubMed dataset in Figure 7.2. On the other hand, averaging with TF-IDF weighting was presumed to take more time than simple averaging but it did so only in half of the test configurations proving nothing.

Method	ROUGE-1		ROUGE-2		ROUGE-L		ROUGE-SU*		BLEU
	R	F1	R	F1	R	F1	R	F1	
LSA	0.3230	0.3123	0.0928	0.0910	0.1856	0.1802	0.1230	0.1061	0.0809
DAAM	0.2250	0.2815	0.0617	0.0785	0.2043	0.2557	0.0582	0.0779	0.0425
config 1	0.3314	0.3180	0.0856	0.0842	0.1823	0.1756	0.1222	0.1047	0.0731
config 2	0.3327	0.3178	0.0899	0.0871	0.1836	0.1759	0.1243	0.1052	0.0761
config 3	0.3521	0.3190	0.0972	0.0893	0.1941	0.1767	0.1404	0.1051	0.0845
config 4	0.3480	0.3209	0.0966	0.0901	0.1912	0.1780	0.1369	0.1064	0.0829

Table 7.3: Results of the LSA word embeddings experiments on PubMed dataset compared with the original LSA and DAAM implementations.

Comparing coherency and readability of summaries produced by the original LSA implementation and the proposed word embedding configurations on both articles and Wikipedia page sections, both selected with minor exceptions the very same sentences from the source texts. As no clear winner can be selected through a human quality assessment, we have

to rely on the quantitative metrics. In the best case, configuration 4 including word2vec-google-news-300 word embedding model and sentence vectors computed by TF-IDF weighted averaging improved the original LSA ROUGE scores by roughly 10 % for both datasets. The higher score comes at a cost of soaring computation time, $135\times$ than pure LSA for CNN/DM and even double that difference for PubMed dataset! The step change in run time is obvious in Figures 7.1 and 7.2.

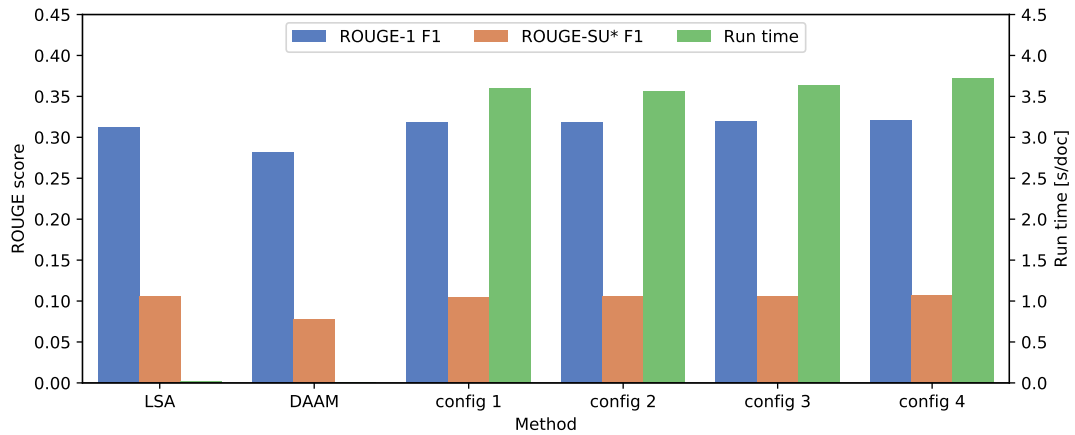


Figure 7.2: Results of the LSA word embeddings experiments on PubMed dataset compared with the original LSA and DAAM implementations. Run time per document is not stated for DAAM due to incomparable hardware.

Although combining the word2vec model with LSA might be desirable in certain cases as it provides a slight increase in the performance regarding ROUGE scores, we believe the cost of longer computation time is too high for the real time chatbot use case in this work.

7.3 Chatbot Integration

To bring this work to conclusion, the chatbot platform developed in Part I and a selected text summarization algorithm from Part II are joined together for easier comprehension of long documents. Based on the experiments, we have selected the LSA method with TF-IDF weighting to take care of the summarization due to its reasonable performance on CNN/DM and PubMed datasets as well as the lowest run time achieved. The deployed solution is based on implementation described in Chapter 3 with the addition of the LSA summarization method into the source text processing pipeline in the AWS Lambda function Controller block. In order to keep the original set of information in the database untouched and complete, summarization is integrated as the last step in the chatbot's response formulation before it is sent back via Switch to the end user. Thanks to the extensible design and self-containment of the various methods, the integration proved itself to be easy and straightforward.

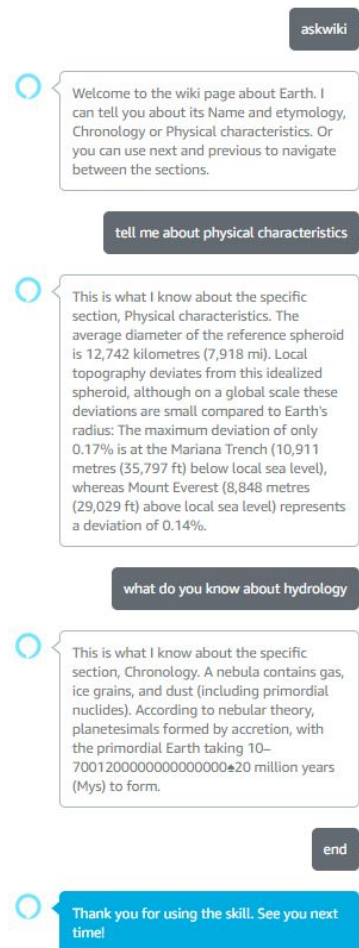


Figure 7.3: A sample conversation between the end user and the chatbot employing LSA

Chapter 8

Conclusion

Is there a better way how to conclude a work about text summarization than by summarizing the work itself? Following is a five sentence summary of this very own conclusion generated by the implemented latent semantic analysis algorithm.

Possible improvements of the approaches are suggested and a suitable method is implemented into the chatbot binding the two areas together. The current state of the art of dialogue systems has been depicted in Chapter 2 in combination with available development platforms. After a thorough overview of various approaches to unstructured text summarization in Chapter 4, two of them are selected as representatives of extractive and abstractive streams for further experiments illustrating the richness of possible solutions. Based on the experiments and analysis of existing solutions, we have proposed possible improvements of the method by incorporating word2vec word embeddings. However, taking all essential parameters into account, mainly low latency requirements of the chatbot, the original LSA method has been selected as the most suitable candidate and implemented into the chatbot providing a user with curated content of the source documents.

A unique problem has been tackled in this work – chatbot able of discussing the content of a long text document or web page with a human. The final solution incorporates two distinct areas of NLP. In Part I the dialogue system is developed based on the presented theoretical basis. Part II then focuses on experiments with automatic summarization of unstructured text. Possible improvements of the approaches are suggested and a suitable method is implemented into the chatbot binding the two areas together. All initially set objectives of the work have been accomplished.

Dialogue management problem still has not been satisfactorily solved until today. Nevertheless, we have managed to design and implement an experimental platform based on a state automaton. It can describe a selected Wikipedia website by communicating its scraped content to the end user talking to an Amazon Alexa device. The current state of the art of dialogue systems has been depicted in Chapter 2 in combination with available development platforms. From there an inspiration has been drawn for our solution design. Implementation based on AWS services is comprehensibly documented to provide the reader with insight into the development choices made along the way.

After a thorough overview of various approaches to unstructured text summarization in Chapter 4, two of them are selected as representatives of extractive and abstractive streams for further experiments illustrating the richness of possible solutions. Implementation of latent semantic analysis (LSA) algorithm and discourse-aware attention model (DAAM) is further covered in greater detail. The solutions are evaluated on documents of various nature from two datasets, news articles of CNN/Daily Mail and scientific articles from PubMed database, providing both qualitative human assessment and quantitative ROUGE metrics evaluation in Chapter 6. Despite much higher computation time demands, the DAAM was not able to outperform LSA. Due to more promising results and better source document domain transferability, the LSA method has been chosen as a candidate for future improvements.

Based on the experiments and analysis of existing solutions, we have proposed possible improvements of the method by incorporating word2vec word embeddings. Four distinct configurations of pre-trained models and sentence vectors computation are presented. The effort has been successful and led to achieving better ROUGE scores. However, taking all essential parameters into account, mainly low latency requirements of the chatbot, the original LSA method has been selected as the most suitable candidate and implemented into the chatbot providing a user with curated content of the source documents.

There is plenty of opportunities for future research in the area of dialogue control making our chatbot more user-friendly and fluent in conversations. At least a couple of years will pass until a universal text summarization algorithm comparable to a human is developed. By designing the chatbot platform as an extensible one consisting of independent modules, we can make sure that the final solution providing users with a unique way of absorbing text content through a chatbot will always benefit from the most recent research.

Appendix A

Contents of the Attached CD

- `README.txt` – enclosed files overview and instructions
- `lsa.py` – implementation of LSA
- `lsaw2v.py` – implementation of LSA with word embeddings
- `pipeline.py` – testing pipeline for LSA and LSA word embeddings experiments
- `rouge.py` – evaluation script for ROUGE metrics
- `bleu.py` – evaluation script for BLEU metric
- `tfidf.py` – implementation of TF-IDF score for LSA
- `pubmed_run.sh` – example of a cluster job Bash script for DAAM
- `switch.py` – main logic of Switch
- `resp.py` – response generation for Switch
- `controller.py` – main logic of Controller
- `crawler.py` – crawling structure and content of a Wikipedia page
- `domain.py` – storing acquired knowledge base for Controller
- `state.py` – representation of individual states for Controller
- `database.py` – access to DynamoDB and S3 for Controller

Bibliography

- [1] Daniel G Bobrow et al. “GUS, a frame-driven dialog system”. In: *Artificial intelligence* 8.2 (1977), pp. 155–173.
- [2] Ani Nenkova and Kathleen McKeown. “A survey of text summarization techniques”. In: *Mining text data*. Springer, 2012, pp. 43–76.
- [3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014).
- [4] Dan Jurafsky and James H Martin. *Speech and language processing*. Vol. 3. Pearson London, 2014.
- [5] Esther Levin, Roberto Pieraccini, and Wieland Eckert. “A stochastic model of human-machine interaction for learning dialog strategies”. In: *IEEE Transactions on speech and audio processing* 8.1 (2000), pp. 11–23.
- [6] Alessandro Sordani et al. “A neural network approach to context-sensitive generation of conversational responses”. In: *arXiv preprint arXiv:1506.06714* (2015).
- [7] Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. “Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning”. In: *arXiv preprint arXiv:1702.03274* (2017).
- [8] Amazon Web Services. *AWS Lex FAQs*. URL: <https://aws.amazon.com/lex/faqs/>.
- [9] Wikipedia. *Dialogflow*. URL: <https://en.wikipedia.org/wiki/Dialogflow#Api.ai>.
- [10] Eduard Hovy, Chin-Yew Lin, et al. “Automated text summarization in SUMMARIST”. In: *Advances in automatic text summarization* 14 (1999).
- [11] Dragomir R Radev, Eduard Hovy, and Kathleen McKeown. “Introduction to the special issue on summarization”. In: *Computational linguistics* 28.4 (2002), pp. 399–408.
- [12] Dharmendra Hingu, Deep Shah, and Sandeep S Udmale. “Automatic text summarization of wikipedia articles”. In: *2015 International Conference on Communication, Information & Computing Technology (ICCICT)*. IEEE, 2015, pp. 1–4.
- [13] Hans Peter Luhn. “The automatic creation of literature abstracts”. In: *IBM Journal of research and development* 2.2 (1958), pp. 159–165.
- [14] Ted Dunning. “Accurate methods for the statistics of surprise and coincidence”. In: *Computational linguistics* 19.1 (1993), pp. 61–74.
- [15] Sanda Harabagiu and Finley Lacatusu. “Topic themes for multi-document summarization”. In: *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2005, pp. 202–209.

- [16] Lucy Vanderwende et al. “Beyond SumBasic: Task-focused summarization with sentence simplification and lexical expansion”. In: *Information Processing & Management* 43.6 (2007), pp. 1606–1618.
- [17] Yihong Gong and Xin Liu. “Generic text summarization using relevance measure and latent semantic analysis”. In: *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM. 2001, pp. 19–25.
- [18] Makbule Gulcin Ozsoy, Ferda Nur Alpaslan, and Ilyas Cicekli. “Text summarization using latent semantic analysis”. In: *Journal of Information Science* 37.4 (2011), pp. 405–417.
- [19] Hongyan Jing. “Using hidden Markov modeling to decompose human-written summaries”. In: *Computational linguistics* 28.4 (2002), pp. 527–543.
- [20] CS Saranyamol and L Sindhu. “A survey on automatic text summarization”. In: *Int. J. Comput. Sci. Inf. Technol* 5.6 (2014), pp. 7889–7893.
- [21] Sciforce. *Towards Automatic Summarization. Part 2. Abstractive Methods*. 2019 (accessed March 20, 2019). URL: <https://medium.com/sciforce/towards-automatic-summarization-part-2-abstractive-methods-c424386a65ea>.
- [22] Atif Khan and Naomie Salim. “A review on abstractive summarization methods”. In: *Journal of Theoretical and Applied Information Technology* 59.1 (2014), pp. 64–72.
- [23] Sanda M Harabagiu and Finley Lacatusu. “Generating single and multi-document summaries with gistexter”. In: *Document Understanding Conferences*. 2002, pp. 11–12.
- [24] Papers With Code. *State of the Art of Automatic Text Summarization*. 2019 (accessed March 20, 2019). URL: <https://paperswithcode.com/task/text-summarization>.
- [25] Ramesh Nallapati et al. “Abstractive text summarization using sequence-to-sequence rnns and beyond”. In: *arXiv preprint arXiv:1602.06023* (2016).
- [26] Abigail See, Peter J Liu, and Christopher D Manning. “Get to the point: Summarization with pointer-generator networks”. In: *arXiv preprint arXiv:1704.04368* (2017).
- [27] Arman Cohan et al. “A discourse-aware attention model for abstractive summarization of long documents”. In: *arXiv preprint arXiv:1804.05685* (2018).
- [28] Arman Cohan et al. *Resources for the paper A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents*. 2018 (accessed March 27, 2019). URL: <https://github.com/acohan/long-summarization>.
- [29] Horacio Saggion and Thierry Poibeau. “Automatic text summarization: Past, present and future”. In: *Multi-source, multilingual information extraction and summarization*. Springer, 2013, pp. 3–21.
- [30] Franck Deroncourt, Mohammad Ghassemi, and Walter Chang. “A Repository of Corpora for Summarization”. In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*. 2018.
- [31] Karl Moritz Hermann et al. “Teaching machines to read and comprehend”. In: *Advances in neural information processing systems*. 2015, pp. 1693–1701.

- [32] Mehdi Allahyari et al. “Text summarization techniques: a brief survey”. In: *arXiv preprint arXiv:1707.02268* (2017).
- [33] Chin-Yew Lin. “Rouge: A package for automatic evaluation of summaries”. In: *Text Summarization Branches Out* (2004).
- [34] Kishore Papineni et al. “BLEU: a method for automatic evaluation of machine translation”. In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.
- [35] Pranay Mathur, Aman Gill, and Aayush Yadav. *Text Summarization in Python: Extractive vs. Abstractive techniques revisited*. 2017 (accessed March 31, 2019). URL: <https://rare-technologies.com/text-summarization-in-python-extractive-vs-abstractive-techniques-revisited/>.
- [36] Abigail See. *Attention Visualizer Tool*. 2017 (accessed April 18, 2019). URL: https://github.com/abisee/attn_vis.
- [37] Tomas Mikolov et al. “Efficient estimation of word representations in vector space”. In: *arXiv preprint arXiv:1301.3781* (2013).
- [38] Radim Řehůřek and Petr Sojka. “Software Framework for Topic Modelling with Large Corpora”. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, 2010, pp. 45–50.