DocTalk: Combining Dependency-based Text Graphs and Deep Learning into a Practical Dialog Engine

Yifan Guo, Weilun Sun, Ali Khan, Tam Doan and Paul Tarau

Department of Computer Science and Engineering
University of North Texas

Abstract. Today's deep learning dominates the field of natural language processing (NLP) with text graph-based approaches being another promising approach. However, both have inherent weaknesses. We present our system called DocTalk that brings together a model that combines the strength of the two approaches. DocTalk's symbiotic model widens its application domain, enhanced with automatic language detection and effective multilingual summarization, keyword extraction, and question answering on several types of documents. Taking advantage of DocTalk's flexibility, we built it into a dialog engine, coupled with an easy-to-use web interface.

Keywords: symbiotic graph-based and neural NLP, multilingual web-based dialog engine, dependency-graph-based summary and keyword extraction, document-centered query-answering system

1 Introduction

Deep learning systems have been successful at a wide variety of tasks ranging from parsing to factoid question answering on short documents. However, they have limited success on complex and lengthy documents, for which summarization and query answering have more practical uses. On the other hand, text graph-based systems have been used successfully for extractive summarization and query answering on lengthy documents and short documents alike. Their success, however, is limited on high level information extraction and knowledge representation tasks, requiring inference steps. To make NLP more practical and accessible, our system utilizes both approaches and combines their respective strengths, circumventing many limitations intrinsic to each approach and consequently remaining viable for a much wider variety of applications.

Taking advantage of Doctalk's flexibility, we build it into a dialog agent that digests a text document (e.g. a textbook, a scientific paper, a story, a legal document) and enables the user to interact with its most relevant content elements. We will start with an overview of the system, including the main tools and functions of each module.

Fig.1 summarizes the architecture of our system. Our system implements a visually-appealing web interface that allows the users to not only easily adjust several parameters of the system, but also input documents of their choice. After a document is uploaded, it is forwarded to the Python-based document processor, which creates a text graph representation of the text that could be used for summarization, keyword extraction,

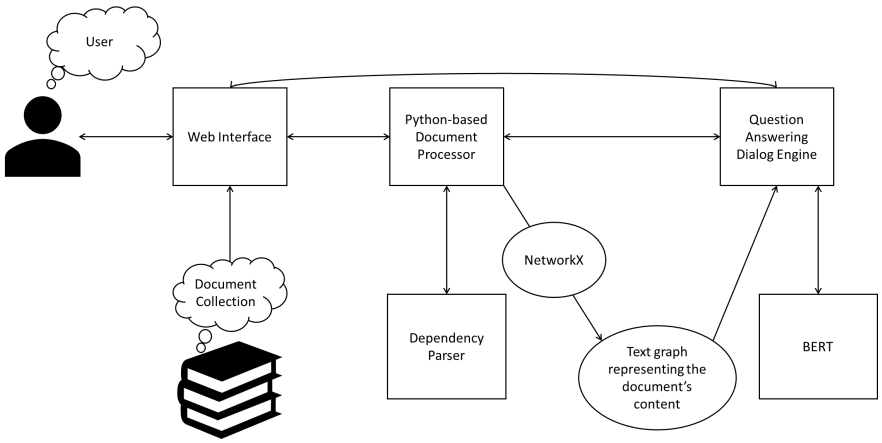

Fig. 1. System Architecture

and query answering. Afterwards, the text graph is forwarded to the Dialog Engine, enabling direct interaction via its web interface. To avoid query answering using solely machine learning, which has limited success with long documents, the engine first uses ranking algorithms on text graphs built from dependency trees to retrieve sentences that most likely contain the answer. BERT [4], a deep learning model that works well on small text documents, then determines the answer using the retrieved sentences. Both retrieved sentences and BERT's answer are directed to the web interface and presented to the user as "long answer" and "short answer" respectively.

The rest of the paper is organized as follows: Section 2 describes the Python-based document processor. Section 3 describes the question answering dialog engine. Section 4 describes the web interface and includes interaction examples. Section 5 discusses some real-world applications. Section 6 overviews related work and background information. Section 7 concludes the paper.

2 The Python-based Document Processor

The Python-based document processor relies on Stanford Stanza [9][1] for extracting dependency trees that it aggregates together into a document graph along the lines of [13]. Using the lemmatization, part-of-speech tagging, named-entity-recognition, and dependency parsing from Stanza, we derive various types of edges and incorporate the edges into a text graph representation of the text using Python's *NetworkX* library. Besides dependency links between lemmas of the words in the text, the text graph also connects lemmas to the sentences in which they occur. By selecting the most salient sentences

[1] working as a client to the CoreNLP server for English documents

via additional filtering mechanisms and PageRank [8], our document processor provides summarization and keyword extraction. The text graph is also utilized later for query answering.

2.1 Creating TextGraph

Unlike approaches that develop special techniques for each individual task, we present a unified algorithm to obtain graph representations of documents and show that this unified representation is suitable for keyphrase extraction, summarization and question answering. Our implementation is available at `https://github.com/Yifan-G/DocTalk`. In our implementation, we organize the information gleaned from Stanford Stanza's dependency trees into various types of edges, which would form a unified text graph.

Entity-oriented Text Graph Representation In order to represent the main idea or plotline of a document, we focus on the entities present in the document and their influence on each other by extracting *SVO* and *has_instance* edges.

SVO edges, short for subject-verb-object edges, play an important role in the text graph. Each SVO edge connects two nodes representing a subject and an object with a verb, all in a lemmatized form. An example would be 'senate' have 'power' in Figure 2. We redirect some dependency edges towards nouns because we consider nouns to representative of the content of the text when focusing on summarization and extracted keyphrases and when asking questions about a document.

In order to orient the text graph around key content elements, we try to recognize the nodes that represent real-world entities and identify them with *has_instance* edges that connect a node representing an entity to another node representing the category of the entity. An example is 'organization' *has_instance* 'senate' in figure 2. Through the inclusion of *has_instance* edges, we add weight to nodes representing real-world entities, increasing their influence over the keyword extraction and summarization process.

Text Graph Expansion using Syntactic Relations To incorporate relational edges that are overlooked by Stanza, we extract WordNet relations between lemmas and include them as edges in the text graph. For every token that is not a noise word, we utilize the *NLTK* toolkit's WordNet package to find its *synonyms*, *hypernyms*, and *meronyms*. If the related word is in the same sentence as the original token, we organize them as follows:

1. synonym relations into the form of (original token, is_like, synonym)
2. hypernym relations into the form of (original token, kind_of, hypernym)
3. meronym relations into the form of (original token, part_of, meronym).

Some of these relations, along with other types of relations, are shown in Fig. 2. As a result of expanding the text graph with edges reflecting accurately syntactic and semantic relations between words, the text graph is completed with both entity-oriented edges and relational edges.

So far, text graph expansion is only utilized for English documents. However, thesaurus in other languages could be implemented to enable multilingual text graph expansion.

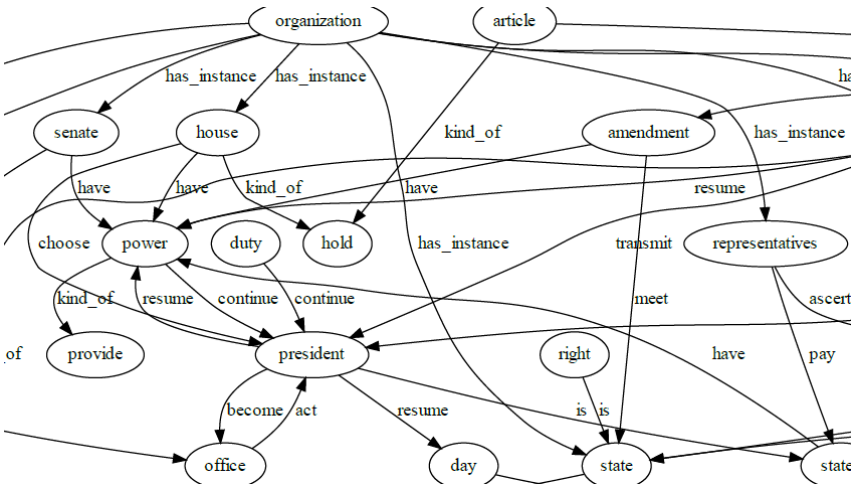

Fig. 2. Part of the dependency graph for the Constitution of the United States

2.2 Summarization and Keyword Extraction

Summarization We rely on *NetworkX's* digraph centrality algorithms (by default, PageRank) to provide an initial ranking value indicating the importance of each sentence in regards to the document. Since the initial ranking value provided by *NetworkX* depends on the number of nodes adjacent to the sentence node, longer sentences are intrinsically prioritized due to containing more nodes. Consequently, short and medium-sized sentences containing important information would be overlooked if only the PageRank ranking value is used.

To prevent that, a post-ranking normalization is applied to the ranking value, deprioritizing both excessively long and excessively short sentences. We compute the new ranking value as:

$$\text{new ranking value} = \frac{\text{ranking value}}{1+|\text{sentence length}-\text{average sentence length}|+\text{sentence length}}$$

The scaling ensures that sentences that are as close as possible to the average sentence length are prioritized. This also helps with discarding the noisy or fragmented sentences incorrectly extracted by our *pdftotext*[2] converter.

[2] `https://poppler.freedesktop.org/`

Keyword extraction Given their position in the dependency graph, with links pointed to them from their dependents, verbs and nouns are usually the highest ranked. With the intuition that texts more likely center around nouns and keyphrases always contain a nominal component, we only center a keyphrase around a lemma if more than half of its occurrences in the text are tagged as nouns.

Moreover, we prioritize compounds, with the intuition that compounds could reveal more detailed information than single words. After the first round of lemma filtering, we attempt to expand lemmas with sufficiently high ranking values into compounds by collecting their adjacent nodes, which could be compounds containing the lemmas (connected to the original lemmas through *as_in* edges). As long as the ranking value of a compound is higher than 1/16 of that of the original lemma, we collect the compound instead of the single-word lemma.

3 The Question Answering Dialog Engine

After a question is inputted, it is parsed using Stanford Stanza to extract its tokens, lemmas, POS tags and dependency edges. The nouns, verbs, and adjectives in the original question are stored as our original question words. We additionally find words related to the original question words using WordNet. Afterwards, we feed both the expanded question words and original question words into our ranking algorithm (e.g., the NetworkX implementation of Personalized Pagerank) which provides a ranking value for each sentence that indicates the relevance of the sentence to the original question words and expanded question words, discovering sentences that likely contain the answer. Lastly, sentences with the highest ranking values are combined in the order in which they occur in the text and forwarded to BERT, which returns a short answer to the question.

3.1 Question word expansion

We find the synonyms, hypernyms, hyponyms, meronyms, and holonyms of each original question word using NLTK's WordNet and save the words that exist in the original text as expanded question words. The intuition behind expanding the original question words is twofold. First, the procedure counters the effect of grammatical mistakes and ambiguity. By retrieving and considering the related words, the model could potentially 'guess' the user's intended meaning and answer the question even when the question is not grammatically or linguistically accurate. Second, the expansion of original question words possibly recalls more information relevant to the answer. Due to the short length of questions, they often contain little information, which makes it difficult to find the answers. However, by finding words related to the original question words, we gain more information relevant to the question and consequently have more to work with. The approach is especially critical when an entity is referred to using multiple names in a document.

3.2 Ranking Algorithm

In our ranking algorithm, the ranking value of each sentence is dependent on three aspects (*lshared* , *important* and *unusual*) and calculated by the equation:

ranking value $= lshared * important * unusual$,

where *lshared* is the number of original question words and expanded question words in the sentence, which measures the relevance of the sentence to the question and *important* is calculated by the equation:

$$important = e^{1+|\text{sentence length}-\text{average sentence length}|+\text{sentence length}}$$

The original ranking value of the sentence is calculated in the summarization process and uninfluenced by the question asked. It indicates the general importance of a sentence based on its influence. The denominator below the original ranking value serves to scale the ranking value against the sentence's length, ensuring that the sentence is informative as opposed to just lengthy (with the intuition described in the summarization section).

Lastly, *unusual* is calculated using the following equation:

$$unusual = sigmoid(1 - \frac{\text{the harmonic mean of(the occurrences of question word)}}{\text{number of sentences}})$$

The *unusual* value prioritizes sentences that do not contain a large amount of original question words and expanded question words. It serves to mitigate the dominance of sentences with high values in the previous two aspects and consequently foster the discovery of sentences that do not appear relevant to the question but still contain important information.

Through taking all three aspects into account, Talker fosters both the discovery of sentences closely related to the question and less relevant sentences that could provide interesting information.

3.3 BERT Short Answer

After importing the BERT transformer[3] in our Python code base, we combine a maximum of 4 highest ranked sentences into a document that is short enough for effective processing using BERT. We then feed the shortened document and the question to the pretrained BERT model, which then generates a short answer to the question. The short answer is outputted along with a confidence level, which is the estimated probability that the short answer is correct.

If no short answer is outputted or if the confidence level is lower than the threshold value of 0.1, we set the short answer displayed to "No Answer". Otherwise, we display the short answer generated by BERT.

[3] `https://huggingface.co/transformers/main_classes/pipelines.html#transformers.QuestionAnsweringPipeline`

4 The Interactive Web Interface

Our dialog agent is built around both the extractive summarization algorithm as well as the question answering algorithm. The dialog agent is built with *streamlit*[4], a Python package that allows us to turn Python code into a web app without writing additional HTML or Javascript. The main window in the center of the web interface holds the results of the analysis. The sidebar at the left allows users to upload a document to process and select a task (either summarization or question answering).

Users can upload documents up to 200 megabytes. Once a document is uploaded, the interface will automatically detect the language that the document is written in. The interface will then process the document and output the summary and keywords in the main window.

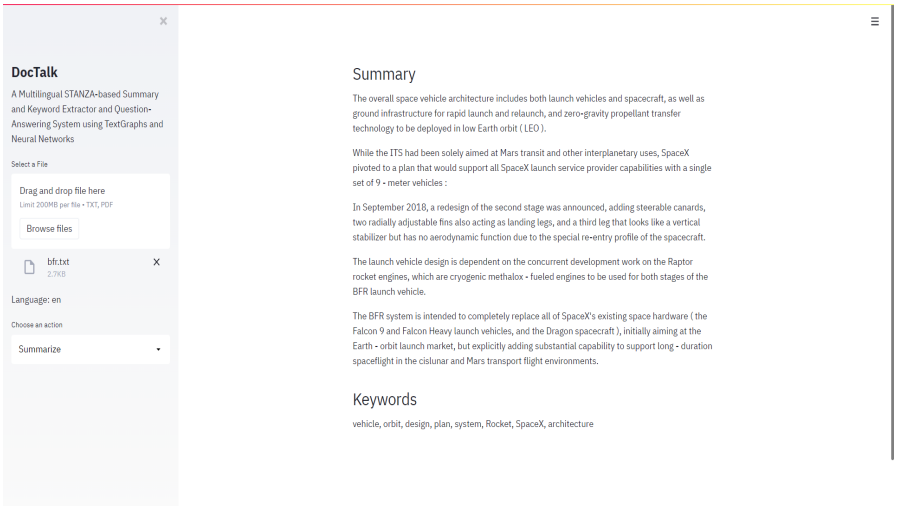

Fig. 3. Summary of an article about SpaceX's "Big Falcon Rocket"

[4] `https://streamlit.io/`

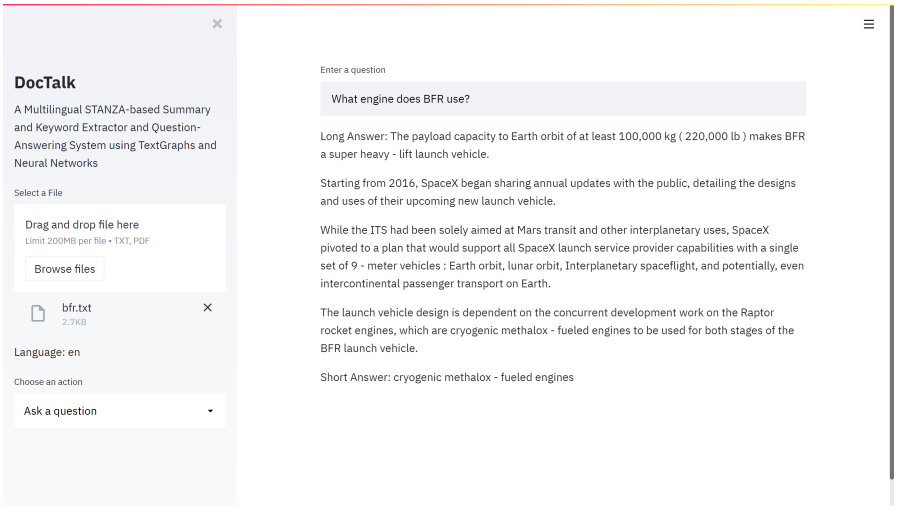

Fig. 4. Question answering based on an article about SpaceX's "Big Falcon Rocket"

When the question-answering task is chosen from the select box at the bottom-left corner, a text box appears prompting the user to enter a question about the content of the uploaded document. The interface then outputs the answer to the question.

As shown above, the algorithm provides both a short and long answer. Moreover, the system is capable of multilingual summarization and question answering, in languages including English, Chinese, Russian, and Spanish.

Fig. 5. Summarization of a Chinese article about blackholes

Language is automatically detected using the *langdetect*[5] Python package. We plan as future work to enable a user to upload a document in any of the 68 languages covered by Stanford Stanza and answer questions in the user's preferred language using Google's translate API.

5 Empirical Evaluation

5.1 Query-specific Information Retrieval

To enable effective question answering using the combined approach, DocTalk retrieves sentences relevant to the query in the first step to provide concise and important information to BERT in the second step.

In order to assess our system's ability for extracting information about a specific topic, we test the first step of our system on the proof-extraction version of Liar-PLUS [6]. We pick the top 4 sentences as our focused summary. The performance of our model and other models is shown in table 1, evaluated using *py-rouge 1.1*[6].

Model	**Validation**			**Test**		
	ROUGE-1	**ROUGE-2**	**ROUGE-L**	**ROUGE-1**	**ROUGE-2**	**ROUGE-L**
Lead-4	30.85	8.08	22.48	31.28	8.17	22.75
Biased TextRank [6]	44.73	26.29	36.58	44.88	26.65	36.69
DocTalk	**44.50**	**26.10**	**38.06**	**44.49**	**26.10**	**38.19**

Table 1. Comparison with other models on Liar-PLUS dataset

Our system's performance on extracting information relevant to a specific topic is comparable to recent models, proving the first step of our system to be effective. An effective first step enables successful question answering using the extracted information.

5.2 Question Answering

In order to simulate real-world applications of our system, which would likely be on relatively long documents (e.g., research papers) we test our system on long documents. Due to a lack of datasets containing long documents, we modify SQuAD 1.1 [11] into a synthetic dataset containing long documents. The original SQuAD 1 dataset is composed of question answering pairs on paragraphs of text. Those paragraphs are retrieved by splitting long articles. When answering a question, models only need to process one short paragraph of information. We recombine the pre-split paragraphs into complete articles and answer questions using full articles, simulating our intended real-world use-case. The average length of the paragraphs and articles are shown in table 2.

In order to evaluate the advantage of our combined approach, we test both DocTalk with BERT and BERT by itself on the modified dataset. When testing DocTalk with

[5] `https://github.com/Mimino666/langdetect`

[6] https://github.com/Diego999/py-rouge

Dataset	Avg Sentence Count	Avg Word Count	Num Questions	Num Articles
SQuAD-paragraphs	5.03	143.33	10570	2067
SQuAD-articles	217.35	6174.56	10570	48

Table 2. Average sentence count/document, average word count/document, total number of questions, and total number of documents for the two versions of SQuAD 1.1

BERT, we feed the 4 highest ranked sentences to BERT. The F1 and EM score of the tests are shown in table 3.

Setup	F1	EM
DocTalk with BERT	68.21	61.46
BERT	57.72	52.12

Table 3. Performance of different setups on SQuAD-articles

As shown in table 3, the performance of our combined approach surpasses that of BERT by itself, proving our symbiotic approach to be effective.

6 Discussion

Due to taking a symbiotic deep learning and graph based approach, DocTalk circumvents the intrinsic limitations in each. It produces a question specific summary of a long document using a graph-based approach from which it generates a salient short answer using deep learning. Consequently, it achieves a synergy that makes it applicable to a much wider variety of documents than the two approaches alone.

Furthermore, DocTalk utilizes three separate answer sentence rankers, each based on different criteria, when selecting the answer sentences. This allows for the discovery of a wide variety of sentences as candidate answer sentences, which subsequently increases the practicality of the question answering dialog engine. As a result of the underlying dependency graph-based processor and WordNet-based query expansion, DocTalk has access to richer semantic information, such as various types of relations, a feature also helping focusing in key content elements of long documents.

One potential application of the DocTalk system is as a teaching aid. In an online learning setting, it can function as a virtual teaching assistant [15] by expediently and accurately answering questions regarding content covered, saving both instructors and students time.

DocTalk is not limited to the classroom setting, since it provides summarization and answers regarding long, technical documents. Ranging from researchers who want to find something specific within a lengthy research paper to car owners who need to identify a specific feature of their vehicle from the user's manual, DocTalk's generated summary and answers provide additional clarification and save time. Specifically, the question-answering feature enables interaction with the content and quick retrieval of information, facilitating in-depth understanding.

7 Related work

The DocTalk dialog agent is novel as it contains a text graph built from dependency links that integrates words and sentences in the same text graph, resulting in a unified algorithm that enables keyword extraction, summarization, and interactive question answering. A comprehensive overview of graph-based natural language processing can be found in [10].

As the simplest but possibly still the most popular graph-based algorithm, TextRank [7] extracts keyphrases using word co-occurrence relations controlled by distance between words and computes sentence similarity as content overlap giving weights to the links that refine the original PageRank algorithm [8]. Our main innovations with respect to TextRank, besides the use of dependency-tree based graphs, is the addition of SVO relations extracted from the text as well as relations extracted from WordNet.

In extractive text summarization, sentences are ranked based on their relative importance to understanding the overall document before being considered for use in a summary. The various methods to accomplish this task can be separated into two main categories: supervised machine learning and graph-based text processing. An example in the former category is [12], which ranks sentences obtained during extractive summarization through reinforcement learning. Recent advances in natural language processing have also enabled abstractive summarization (e.g., [2]), usually tested on the CNN/DailyMail dataset, which consists of very short news articles.

Question Answering, like summarization, comes in many forms, including multiple choice and short answers. An example of multiple choice answering is [1], which incorporates both shallow lexical methods and logical reasoning into a unified framework, achieving strong results. On the other hand, [14] first obtains the question-aware passage representation and then refines the representation through a self-matching attention mechanism.

Dialog engines have been used extensively in many fields, including e-commerce, customer service, and virtual assistants. For instance, [3] presents a customer service chatbot that leverages large-scale and publicly available e-commerce data, such as data from in-page product descriptions and user-generated content from e-commerce websites. [5] presents a chatbot that aims to foster conversations and emphasizes user-centric and content-driven design. Like most of the dialog engines, their focus is primarily on providing information about relatively short documents (such as product descriptions and online logs based on conversations with real-world users).

While neural networks are prevalent in the field of summarization, question answering, and dialog engines, the DocTalk system presents a symbiotic neural and graph-based approach to the common tasks of summarization and question answering with an emphasis on longer documents (ex. scientific papers, legal documents, textbooks) as opposed to the short datasets used by the models overviewed.

DocTalk's dependency-based text graph is built in a way similar to [13]. However, instead of using a bridge to a logic-programming system for relational reasoning tasks, we rely here on graph algorithms and a deep-learning component to achieve similar outcomes. While the system described in [13] interacts via English text or voice, DocTalk provides a Web-app interface supporting user-uploaded documents in multiple languages.

8 Conclusion

The key idea of the paper evolved from the pursuit of a synergy between graph-based approaches and deep learning-based approaches in NLP. By using a combination of the two approaches, we effectively avoid many of their intrinsic limitations and take advantage of their respective strengths. Consequently, our model is capable of effective question answering on a wider variety of texts compared to solely deep-learning based approaches, and it is also capable of more intelligent question answering than solely graph-based approaches.

Moreover, we create a unified graph representation of the text that contains both words and sentences from dependency links. Due to the various types of edges present and the sentences contained, the unified text graph is simultaneously applicable to summarization, keyword extraction, and question answering. For question answering, we utilize three algorithmic approaches (text graph weight analysis, subgraph centrality, and closeness weight analysis) to extract a comprehensive collection of answer sentences.

We couple our model with a visually appealing web interface and automatic language detection. It supports summarization, keyword extraction, and question answering in languages including English, Spanish, Chinese, and Russian. Its applications range from assistive technologies to social media monitoring, education, and interactive knowledge extraction from complex legal and technical documents.

References

1. Angeli, G., Nayak, N., Manning, C.D.: Combining natural logic and shallow reasoning for question answering. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 442–452. Association for Computational Linguistics, Berlin, Germany (Aug 2016). https://doi.org/10.18653/v1/P16-1042, https://www.aclweb.org/anthology/P16-1042
2. Choi, H., Ravuru, L., Dryjański, T., Rye, S., Lee, D., Lee, H., Hwang, I.: VAE-PGN based abstractive model in multi-stage architecture for text summarization. In: Proceedings of the 12th International Conference on Natural Language Generation. pp. 510–515. Association for Computational Linguistics, Tokyo, Japan (Oct–Nov 2019). https://doi.org/10.18653/v1/W19-8664, https://www.aclweb.org/anthology/W19-8664
3. Cui, L., Huang, S., Wei, F., Tan, C., Duan, C., Zhou, M.: SuperAgent: A customer service chatbot for E-commerce websites. In: Proceedings of ACL 2017, System Demonstrations. pp. 97–102. Association for Computational Linguistics, Vancouver, Canada (Jul 2017), https://www.aclweb.org/anthology/P17-4017
4. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423, https://www.aclweb.org/anthology/N19-1423

5. Fang, H., Cheng, H., Sap, M., Clark, E., Holtzman, A., Choi, Y., Smith, N.A., Ostendorf, M.: Sounding board: A user-centric and content-driven social chatbot. In: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations. pp. 96–100. Association for Computational Linguistics, New Orleans, Louisiana (Jun 2018). https://doi.org/10.18653/v1/N18-5020, https://www.aclweb.org/anthology/N18-5020
6. Kazemi, A., Pérez-Rosas, V., Mihalcea, R.: Biased TextRank: Unsupervised graph-based content extraction. In: Proceedings of the 28th International Conference on Computational Linguistics. pp. 1642–1652. International Committee on Computational Linguistics, Barcelona, Spain (Online) (Dec 2020). https://doi.org/10.18653/v1/2020.coling-main.144, https://www.aclweb.org/anthology/2020.coling-main.144
7. Mihalcea, R., Tarau, P.: TextRank: Bringing order into text. In: Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing. pp. 404–411. Association for Computational Linguistics, Barcelona, Spain (jul 2004), https://www.aclweb.org/anthology/W04-3252
8. Page, L., Brin, S.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems **30**, 107—-117 (1998), http://citeseer.nj.nec.com/brin98anatomy.html
9. Qi, P., Zhang, Y., Zhang, Y., Bolton, J., Manning, C.D.: Stanza: A Python natural language processing toolkit for many human languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (2020), https://nlp.stanford.edu/pubs/qi2020stanza.pdf
10. Rada, M., Dragomir, R.: Graph-based natural language processing and information retrieval. Cambridge University Press (2011), https://www.academia.edu/2958437/Graph_based_natural_language_processing_and_information_retrieval
11. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAD: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 2383–2392. Association for Computational Linguistics, Austin, Texas (Nov 2016). https://doi.org/10.18653/v1/D16-1264, https://www.aclweb.org/anthology/D16-1264
12. Shashi, N., Shay, B.C., Mirella, L.: Ranking sentences for extractive summarization with reinforcement learning. North American Chapter of the Association for Computational Linguistics 2018 (2018), https://arxiv.org/abs/1802.08636
13. Tarau, P., Blanco, E.: Interactive Text Graph Mining with a Prolog-Based Dialog Engine. Theory and Practice of Logic Programming pp. 1–20 (2020). https://doi.org/10.1017/S1471068420000137
14. Wang, W., Yang, N., Wei, F., Chang, B., Zhou, M.: Gated self-matching networks for reading comprehension and question answering. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 189–198. Association for Computational Linguistics, Vancouver, Canada (Jul 2017). https://doi.org/10.18653/v1/P17-1018, https://www.aclweb.org/anthology/P17-1018
15. Zylich, B., Viola, A., Toggerson, B., Al-Hariri, L., Lan, A.: Exploring automated question answering methods for teaching assistance. In: Bittencourt, I.I., Cukurova, M., Muldner, K., Luckin, R., Millán, E. (eds.) Artificial Intelligence in Education. pp. 610–622. Springer International Publishing, Cham (2020)