What's The Latest? A Question-driven News Chatbot

Philippe Laban
UC Berkeley
phillab@berkeley.edu

John Canny
UC Berkeley
canny@berkeley.edu

Marti Hearst
UC Berkeley
hearst@berkeley.edu

Abstract

This work describes an automatic news chatbot that draws content from a diverse set of news articles and creates conversations with a user about the news. Key components of the system include the automatic organization of news articles into topical chatrooms, integration of automatically generated questions into the conversation, and a novel method for choosing which questions to present which avoids repetitive suggestions. We describe the algorithmic framework and present the results of a usability study that shows that news readers using the system successfully engage in multi-turn conversations about specific news stories.

1 Introduction

Chatbots offer the ability for interactive information access, which could be of great value in the news domain. As a user reads through news content, interaction could enable them to ask clarifying questions and go in depth on selected subjects. Current news chatbots have minimal capabilities, with content hand-crafted by members of news organizations, and cannot accept free-form questions.

To address this need, we design a new approach to interacting with large news collections. We designed, built, and evaluated a fully automated news chatbot which bases its content on a stream of news articles from a diverse set of English news sources. This in itself is a novel contribution.

Our second contribution is with respect to the scoping of the chatbot conversation. The system organizes the news articles into chatrooms, each revolving around a *story*, which is a set of automatically grouped news articles about a topic (e.g., articles related to Brexit).

The third contribution is a method to keep track of the state of the conversation to avoid repetition of information. For each news story, we first generate a set of essential questions and link each question with content that answers it. The motivating idea is: *two pieces of content are redundant if they answer the same questions.* As the user reads content, the system tracks which questions are answered (directly or indirectly) with the content read so far, and which remain unanswered.

We evaluate the system through a usability study. The rest of this paper follows the following structure: Section 2 describes the system and the content source, Section 3 details the algorithm for keeping track of the conversation state, Section 4 provides the results of a usability study evaluation and Section 5 presents relevant prior work.

The system is publicly accessible at https://newslens.berkeley.edu/chatbot/ and a demonstration video is available at this link: https://www.youtube.com/watch?v=eze9hpEPUgo.

2 System Description

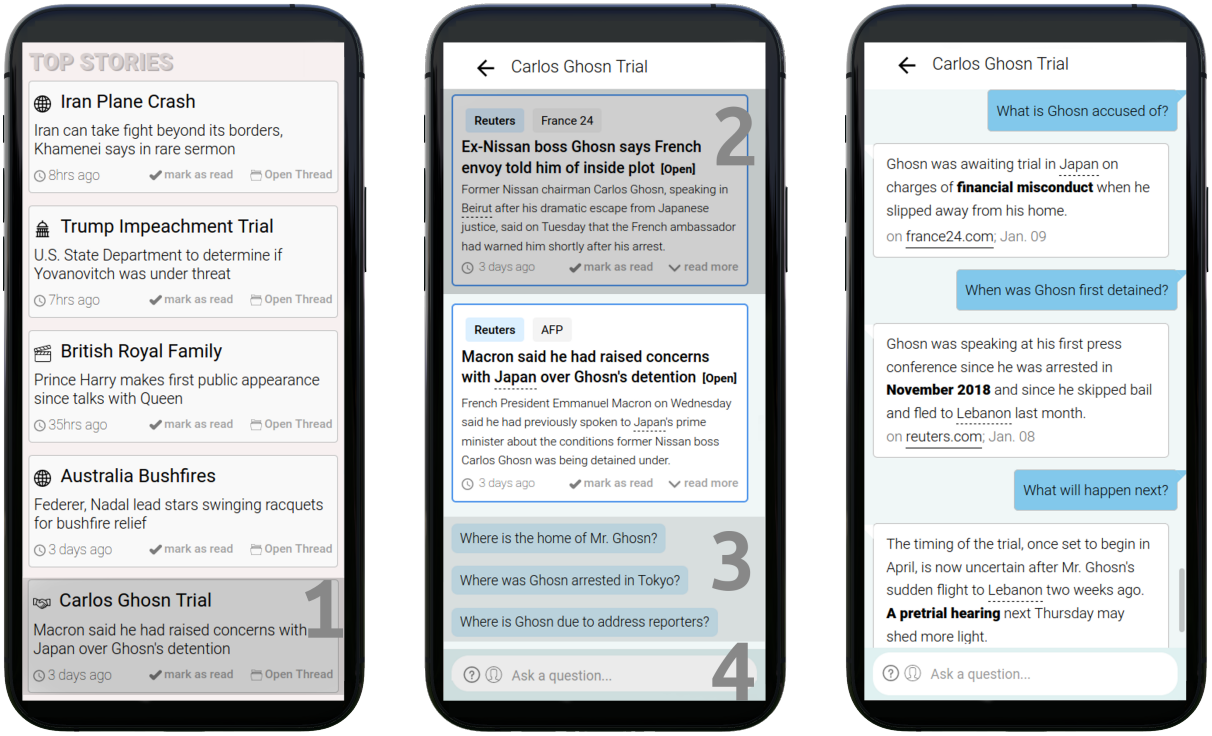
The chatbot supports information-seeking: the user is seeking information and the system delivers information in the form of news content. Because of the difference in respective roles, we expect user turns to be shorter than system turns, which we aim to be around 30 words.

We now describe the components of the chatbot: the content source, the supported user actions and the computed system answers.

Appendix A lists library and data resources used in the system.

2.1 Content Source

We form the content for the chatbot from a set of news sources. We have collected an average of 2,000 news articles per day from 20 international news sources starting in 2010. The news articles are clustered into *stories*: groups of news articles about a similar evolving topic, and each story is



(a) Homepage (b) Initiating a Chatroom (c) Chatroom Q&A

Figure 1: **Screenshots of the news chatbot** (a) Homepage lists most recently active chatrooms (Zone 1 is an example chatroom) (b) Newly opened chatroom: Zone 2 is an event message, Zone 3 the Question Recommendation module, and Zone 4 a text input for user-initiated questions. Event messages are created via abstractive summarization. (c) Conversation continuation with Q&A examples. Sentences shown are extracted from original articles, whose sources are shown. Answers to questions are bolded.

automatically named (Laban and Hearst, 2017). Some of the top stories at the time of writing are shown in Figure 1(a).

The homepage (Figure 1(a)) lists the most active stories, and a user can select a story to enter its respective chatroom (Figure 1(b)). The separation into story-specific rooms achieves two objectives: clarity to the user, as the chatrooms allow the user to exit and enter chatrooms to come back to conversations. Second, limiting the scope of each dialogue is helpful from both a usability and a technical standpoint, as it helps reduce ambiguity and search scope. For example, answering a question like: "What is the total cost to insurers so far?" is easier when knowing the scope is the Australia Fires, compared to all of news.

Articles in a story are grouped into events, corresponding to an action that occurred in a particular time and place. For each event, the system forms an *event message* by combining the event's news article headlines with an abstractive summary generated by a summarizer model (under review).

Zone 2 in Figure 1(b) gives an example of an event message. The event messages form a chronological timeline in the story.

2.2 User Actions

During the conversation, the user can choose among different kinds of actions.

Explore the event timeline. A chatroom conversation starts with the system showing the two most recent event messages of the story (Figure 1(b)). These messages give minimal context to the user necessary to start a conversation. When the event timeline holds more than two events, a "See previous events" button is added at the top of the conversation, allowing the user to go further back in the event timeline of the story.

Clarify a concept. The user can ask a clarification question regarding a person or organization (e.g., Who is Dennis Muilenburg?), a place (e.g., Where is Lebanon?) or an acronym (e.g., What does NATO stand for?). For a predetermined list of questions, the system will see if an appropriate Wikipedia entry exists, and will respond with the first two paragraphs of the Wikipedia page. For

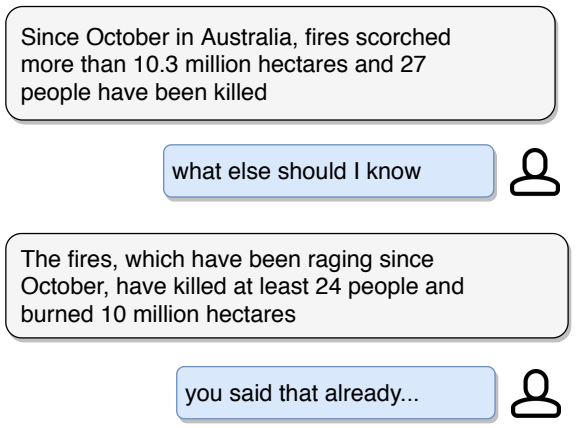


Figure 2: **Example of repetition from the system.** Repeating facts with different language is undesirable in a news chatbot. We introduce a novel question tracking method that attempts to minimize repetition.

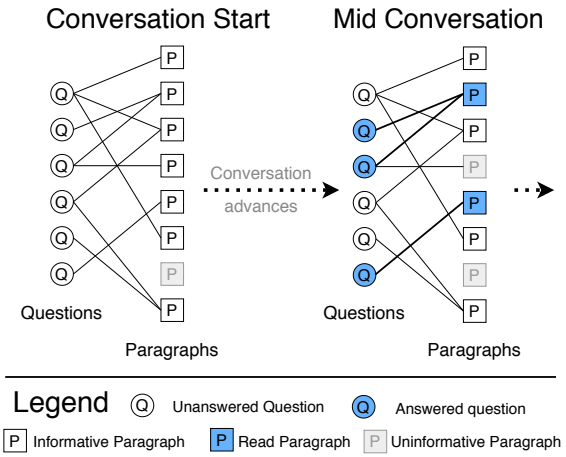


Figure 3: **Conversation state is tracked with the P/Q graph.** As the conversation advances, the system keeps track of answered questions. Any paragraph that does not answer a new question is discarded. Questions that are not answered yet are recommended.

geographical entities, the system additionally responds with a geographic map when possible.

Ask an open-ended question. A text box (Zone 4 in Figure 1(b)) can be used to ask any free-form question about the story. A Q&A system described in Section 3 attempts to find the answer in any paragraph of any news article of the story. If the Q&A system reaches a confidence level about at least one paragraph containing an answer to the question, the chatbot system answers the question using one of the paragraphs. In the system reply the Q&A selected answer is bolded. Figure 1(c) shows several Q&A exchanges.

Select a recommended question. A list of three questions generated by the algorithm described in Section 3 is suggested to the user at the bottom of the conversation (Zone 3 in Figure 1(b)). Clicking on a recommended questions corresponds to asking the question in free-form. However, because recommended questions are known in advance, we pre-compute their answers to minimize user waiting times.

3 Conversation State

One key problem in dialogue systems is that of keeping track of conveyed information, and avoiding repetition in system replies (see example in Figure 2). This problem is amplified in the news setting, where different news organizations cover content redundantly.

We propose a solution that takes advantage of a Question and Answer (Q&A) system. As noted above, the motivating idea is that two pieces of content are redundant if they answer the same questions. In the example of Figure 2, both system messages answer the same set of questions, namely: "When did the fires start?", "How many people have died?" and "How many hectares have burned?", and can therefore be considered redundant.

Our procedure to track the knowledge state of a news conversation consists of the following steps: (1) generate candidate questions spanning the knowledge in the story, (2) build a graph connecting paragraphs with questions they answer, (3) during a conversation, use the graph to track what questions have been answered already, and avoid using paragraphs that do not answer new questions.

Question Candidate Generation. We fine-tune a GPT2 language model (Radford et al.) on the task of question generation using the SQuAD 2.0 dataset (Rajpurkar et al., 2018). At training, the model reads a paragraph from the training set, and learns to generate a question associated with the paragraph. For each paragraph in each article of the story (the paragraph set), we use beam search to generate K candidate questions. In our experience, using a large beam-size (K=20) is important, as one paragraph can yield several valid questions, and beam search enforces some exploration. For instance, the first step of beam search often contains several interrogative words (what, where...).

We reduce the set of questions by deduplicating questions that are lexically close (differ by at most 2 words), and removing questions that are too long (>12 words) or too short (<5 words).

Building the P/Q graph. We train a standard

Q&A model, a Roberta model (Liu et al., 2019) finetuned on SQuAD 2.0 (Rajpurkar et al., 2018), and use this model to build a paragraph / question bipartite graph (P/Q graph). In the P/Q graph, we connect any paragraph (P node), with a question (Q node), if the Q&A model is confident that paragraph P answers question Q. An example bipartite graph obtained is illustrated in Figure 3, with the question set on the left, the paragraph set on the right, and edges between them representing model confidence about the answer.

Because we used a large beam-size when generating the questions, we perform a pruning step on the questions set. Our pruning procedure is based on the realization that two questions are redundant if they connect to the same subset of paragraphs (they cover the same content). Our objective is to find the smallest set of questions that cover all paragraphs. This problem can be formulated as a standard graph theory problem known as the set cover problem, and we use a standard heuristic algorithm (Caprara et al., 1999). After pruning, we obtain a final P/Q graph, a subgraph of the original consisting only of the covering set questions.

The P/Q graph embodies interesting properties. First, the degree of a question node measures how often a question is answered by distinct paragraphs, providing a measure of the question's importance to the story. The degree of a paragraph node indicates how many distinct questions it answers, an estimate of its relevance to a potential reader. Finally, the graph can be used to measure question relatedness: if two questions have non-empty neighboring sets (i.e., some paragraphs answer both questions), they are likely to be related questions, which can be used as a way to suggest follow-up questions.

Using the P/Q graph. At the start of a the conversation, no question is answered, since no paragraph has been shown to the user. Therefore, the system initializes a blank P/Q graph (left graph in Figure 3). As the system reveals paragraphs in the conversation, they are marked as *read* in the P/Q graph (blue paragraphs in the right graph of Figure 3). According to our Q&A model, any question connected to a read paragraph is *answered*, so we mark all neighbors of read paragraphs as answered questions (blue questions on the right graph of Figure 3). At any stage in the conversation, if a paragraph is connected to only answered questions, it is deemed *uninformative*, as it will not reveal the answer to a new question.

As the conversation moves along, more paragraphs are read, increasing the number of answered questions, which in turn, increases the number of uninformative paragraphs. We program the system to prioritize paragraphs that answer the most unanswered questions, and disregard uninformative paragraphs. We further use the P/Q graph to recommend questions to the user. We select unanswered questions and prioritize questions connected to more unread paragraphs, recommending questions three at a time.

4 Study Results

We conducted a usability study in which participants were assigned randomly to one of three configurations:

- TOPQR: the recommended questions are the most informative according to the algorithm in Section 3,
- RANDQR: the recommended questions are randomly sampled from the questions TOPQR would not select (however, near duplicates will appear in this set),
- NOQR: No questions are recommended, and the Question Recommendation module (Zone 3 in Figure 1(b)) is hidden.

These are contrasted in order to test (a) if showing automatically generated questions is beneficial to news readers, and (b) to assess the question tracking algorithm against a similar question recommendation method with no conversation state.

4.1 Study Setup

We used Amazon Mechanical Turk to recruit participants, restricting the task to workers in English-speaking countries with more than 1500 HITs and an acceptance rate of at least 97%. Each participant was paid a flat rate of $2.50 with the study lasting a total of 15 minutes. During the study, the participants first walked through an introduction to the system, then read the news for 8 minutes, and finally completed a short survey.

During the eight minutes of news reading, participants were requested to select at least 2 stories to read from a list of the 20 most recently active news stories.[1] The participants were prompted to choose stories they were interested in.

[1]We manually removed news stories that were predominantly about politics, to avoid heated political questions, which were not under study here.

Measured Value	TOPQR	RANDQR	NOQR
# participants	18	16	22
# chatrooms opened	3.2	2.9	3.1
# turns / chatroom	24.9 *	15.3 *	8.1
# rec. questions asked	11.9 *	8.2 *	-
# own questions asked	1.5	1.1	2.2
# total questions asked	13.4 *	9.3 *	2.2
latency (seconds)	1.84 *	1.88 *	4.51

Table 1: **Usage statistics of the news chatbot during the usability study.** Participants either saw most informative recommended questions (TOPQR), randomly selected recommended questions (RANDQR) or no recommended questions (NOQR). $*$ signifies statistical difference with NOQR ($p < 0.05$).

Measured Value	TOPQR	RANDQR	NOQR
(1) dull ... stimulating (7)	5.28 *	5.06	4.20
(1) frustrating ... satisfying (7)	5.00 *	4.43	4.00
(1) rigid ... flexible (7)	4.71	4.66	4.14
(1) terrible ... wonderful (7)	4.79	4.69	4.20
exploring new features	5.80	5.50	5.14
learning to operate	5.40	5.25	5.06
performing task is straightforward	5.40	5.56	5.20
system reliability	5.80	5.19	5.67
system speed	6.20	5.87	5.44
rec. questions are clear	5.78 *	4.87	4.28
answers are informative	5.07 *	4.44	3.64

Table 2: **QUIS satisfaction results.** $*$ signifies statistical difference with NOQR ($p < 0.05$).

The survey consisted of two sections: a satisfaction section, and a section for general free-form feedback. The satisfaction of the participants was surveyed using the standard Questionnaire for User Interaction Satisfaction (QUIS) (Norman et al., 1998). QUIS is a series of questions about the usability of the system (ease of use, learning curve, error messages clearness, etc.) answered on a 7-point Likert scale. We modify QUIS by adding two questions regarding questions and answers: "Are suggested questions clear?" and "Are answers to questions informative?" A total of fifty-six participants completed the study. We report on the usage of the system, the QUIS Satisfaction results and the general feedback frmo the users.

4.2 Usage statistics

We observed that participants in the QR-enabled interfaces (TOPQR and RANDQR) had longer conversations than the NOQR setting, with an average chatroom conversation length of 24.9 turns in the TOPQR setting.

This increase in conversation length is mostly due to the use of recommended questions, which are convenient to click on. Indeed, users clicked on 8.2 questions on average in RANDQR and 11.9 in TOPQR. NOQR participants wrote on average 2.2 own questions, which was not statistically higher than TOPQR (1.5) and RANDQR (1.1), showing that seeing recommended questions did not prevent participants from asking their own questions.

When measuring the latency of system answers to user questions, we observe that the average wait time in TOPQR (1.84 seconds) and RANDQR (1.88 seconds) settings is significantly lower than NOQR (4.51 seconds). This speedup is due to our ability to pre-compute answers to recommended answers, an additional benefit of the QR graph precomputation.

4.3 QUIS Satisfaction Scores

Overall, the systems with question recommendation enabled (TOPQR and RANDQR) obtained higher average satisfaction on most measures than the NOQR setting. That said, statistical significance was only observed in 4 cases between the TOPQR and NOQR, with participants judging the TOPQR interface to be more stimulating and satisfying.

Although not statistically significant, participants rated the suggested questions for TOPQR almost 1 point higher than RANDQR, providing some evidence that incorporating past viewed information into question selection is beneficial.

Participants judged the answers to be more informative in the TOPQR setting. We interpret this as evidence that the QR module helps teach users what types of questions the system can answer, enabling them to get better answers. Several NOQR participants asked "What can I ask?" or equivalent.

4.4 Qualitative Feedback

Thirty-four of the fifty-six participants opted to give general feedback do so. We tagged the responses into major themes:

1. 16 participants expressed interest in the system (e.g., *I enjoyed trying this system out. I particularly liked that stories are drawn from various sources.*)

2. 11 participants mentioned the system did not correctly reply to questions asked (e.g., *Some of the questions kind of weren't answered exactly, especially in the libya article*),

3. 7 participants found an aspect of the interface confusing (e.g., *This system has potential, but as of right now it seems too overloaded and hard to sort through.*)

4. 6 participants thought the questions were useful (e.g., *I especially like the questions at the bottom. Sometimes it helps to remember some basic facts or deepen your understanding*)

The most commonly mentioned limitation was Q&A related errors, a limitation we hope to mitigate as automated Q&A continues progressing.

5 Related Work

News Chatbots. Several news agencies have ventured in the space of dialogue interfaces as a way to attract new audiences. The chatbots are often manually curated for the dialogue medium and advanced NLP machinery such as a Q&A systems are not incorporated into the chatbot.

On BBC's Messenger chatbot[2], a user can enter search queries, such as "latest news" or "Brexit news" and obtain a list of latest BBC articles matching the search criteria. In the chatbot produced by Quartz[3], journalists hand-craft news stories in the form of pre-written dialogues (aka choose-your-own adventure). At each turn, the user can choose from a list of replies, deciding which track of the dialogue-article is followed. CNN[4] has also experimented with choose-your-own adventure articles, with the added ability for small talk.

Relevant Q&A datasets. NewsQA (Trischler et al., 2016) collected a dataset by having a crowd-worker read the summary of a news article and ask a follow-up question. Subsequent crowd-workers answered the question or marked it as not-answerable. NewsQA's objective was to collect a dataset, and we focus on building a usable dialogue interface for the news with a Q&A component.

CoQA (Reddy et al., 2019) and Quac (Choi et al., 2018) are two datasets collected for questions answering in the context of a dialogue. For both datasets, two crowd-workers (a student and a teacher) have a conversation about a piece of text (hidden to the student in Quac). The student *must ask* questions of the teacher, and the teacher answers using extracts of the document. In our system, the questions asked by the user are answered automatically, introducing potential errors, and the user can choose to ask questions or not.

In this work, the focus is not on the collection of naturally occurring questions, but in putting a Q&A system in use in a news dialogue system, and observing the extent of its use.

Question Generation (QG) has become an active area for text generation. A common approach is to use a sequence to sequence model (Du et al., 2017), encoding the paragraph (or context), an optional target answer (answer-aware (Sun et al., 2018)), and decoding a paired question. This common approach focuses on the generation of a single article, from a single piece of context, often a paragraph. We argue that our framing of the QG problem as the generation of a series of questions spanning several (possibly redundant) documents is a novel task.

Krishna and Iyyer (2019) build a hierarchy of questions generated for a single document; the document is then reorganized into a "Squashed" document, where paragraphs and questions are interleaved. Because our approach is based on using multiple documents as the source, compiling all questions into a single document would be long to read, so we opt for a chatbot.

6 Conclusion

We presented a fully automated news chatbot system, which leverages an average of 2,000 news articles a day from a diverse set of sources to build chatrooms for important news stories. In each room, the system keeps track of the state of the conversation using generated questions that are answered so far, to minimize repetition of information to the news reader.

A usability study reveals that when the chatbot recommends questions, news readers can have long conversations with the system, with an average of 24 turns. These conversation consist of combination of recommended and user-created questions. The inaccuracies of the Q&A system remain a limitation for the system, with 32% of the participants mentioning that answers are often off-track and irrelevant, suggesting that further improvements in Q&A systems are still needed to be used in a practical situation.

Acknowledgements. This research was supported in part by a Bloomberg Research Grant and an AWS ML Research Award.

[2]https://www.messenger.com/t/BBCPolitics
[3]https://www.messenger.com/t/quartznews
[4]https://www.messenger.com/t/cnn

References

Alberto Caprara, Matteo Fischetti, and Paolo Toth. 1999. A heuristic method for the set covering problem. *Operations research*, 47(5):730–743.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. *arXiv preprint arXiv:1808.07036*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*.

Kalpesh Krishna and Mohit Iyyer. 2019. Generating question-answer hierarchies. In *ACL*.

Philippe Laban and Marti A Hearst. 2017. newslens: building and visualizing long-ranging news stories. In *Proceedings of the Events and Stories in the News Workshop*, pages 1–9.

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

Kent L Norman, Ben Shneiderman, B Harper, and L Slaughter. 1998. Questionnaire for user interaction satisfaction. *University of Maryland (Norman, 1989) Disponível em*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

A Resources Used

The libraries and data sources used in the described system are as follows:

Transformers library[5] used to train the GPT2-based Question Generation model and the Roberta-based Q&A model.

spaCy library[6] used to do named-entity extraction, phrase and keyword extraction.

Wikidata[7] for entity linking and collection of textual content of relevant Wikipedia pages used in special case questions.

MongoDB[8] and **Flask**[9] for storing and serving the content to the user.

SetCoverPy[10] for its implementation of standard set cover algorithms in Python.

List of news sources present in the dataset used by the system, in alphabetical order: Aa.com.tr, Afp.com, Aljazeera.com, Allafrica.com, Apnews.com, Bbc.co.uk, Bloomberg.com, Chicagotribune.com, Chinadaily.com.cn, Cnet.com, Cnn.com, Foxnews.com, France24.com, Independent.co.uk, Indiatimes.com, Latimes.com, Mercopress.com, Middleeasteye.net, Nytimes.com, Reuters.com, Rt.com, Techcrunch.com, Telegraph.co.uk, Theguardian.com, Washingtonpost.com

[5] https://github.com/huggingface/transformers
[6] https://github.com/explosion/spaCy
[7] https://www.wikidata.org/
[8] https://www.mongodb.com/
[9] https://flask.palletsprojects.com/en/1.1.x/
[10] https://github.com/guangtunbenzhu/SetCoverPy