Question-Answering Dialog System for Large Audiovisual Archives

Adam Chýlek[1], Luboš Šmídl[2], and Jan Švec[1]

[1] NTIS - New Technologies for Information Society,
[2] Department of Cybernetics,
Faculty of Applied Sciences, University of West Bohemia, Pilsen, Czech Republic
chylek@ntis.zcu.cz, smidl@kky.zcu.cz, honzas@ntis.zcu.cz

Abstract. In this paper, we present our spoken dialog system that serves as a search interface of the MALACH archive. The voice interface and natural language input allow the users to retrieve information contained in large audiovisual archives more comfortably. Especially, finding answers to a more structured question should be easier in comparison with typical search input options. The dialog is build on top of a system that automatically annotates and indexes the archive using automatic speech recognition. These indexes were searchable so far only in a full-text search for any arbitrary text query. Our proposed approach improves this system and leverages named entity recognition to create a knowledge base of semantic information contained in the recognized utterances. We describe the design of the dialog system, as well as the automatic knowledge base generation and the approach to creating queries using a spoken natural language as an input.

Keywords: knowledge base generation, natural language processing, question answering, dialog system

1 Introduction

Nowadays it is easy to collect and store a large amount of audiovisual data. Annotating and searching such archives is a different story. Our research was done on the MALACH archive of Holocaust testimonies[3]. This archive is made of thousands of hours of video footage containing interviews and personal testimonies of Holocaust survivors and witnesses. The archive is maintained by the USC Shoah Foundation[4], its collection was initiated by the Shoah Visual History Foundation founded by Steven Spielberg. It serves as a valuable resource for contemporary historians and future generations.

Focusing on English [13] and Czech [12] part of the archive we have hundreds of hours of video for each language annotated mostly only by manually assigned keywords, hence limiting the ability to search the archives only to a sparse subset

[3] https://malach.umiacs.umd.edu/
[4] https://sfi.usc.edu/

of handpicked keywords. Although we chose the MALACH archive as our data source, the limited availability of searchable keywords is common also to other archives (e.g. TV or radio broadcast archives) and our approach can be also used in these cases.

Prior to our research, the authors of [14] developed a search engine that allowed searching for any arbitrary text in these archives. Their approach uses automatic speech recognition (ASR) of the audio sources to create a searchable index.

We further improve the indexation by incorporating spoken language understanding (SLU), namely semantic entity detection. We will describe how this creates a knowledge base that stores additional semantic information from the interviews.

We also focus on improving the accessibility of the information stored in the knowledge base. Using the SLU on the archived data alone would still require the user to have specific knowledge about the search query format (e.g. the use of ”OR”, ”NOT” keywords). To mitigate the need for such knowledge, we have developed a dialog system on top of the knowledge base that accepts queries in spoken natural language and presents the results using speech and a graphical interface.

In the following paragraphs, we give the reader an overview of related work, describe how we create our knowledge base, how we map the natural language onto the database queries and how we bind it all into a dialog system. We conclude the paper with future research plans.

2 Related Work

Our research focuses on applying information retrieval, question answering, natural language understanding, database query generation and dialog management on a contemporary problem - searching in large audio-visual archives.

The combination of these parts is usually researched in relation to question answering from the semantic web [3, 8] or text comprehension [2] using a statistical approach. The authors of [1] even show promising results learning end-to-end dialog on a common DSTC (Dialog State Tracking Challenge) benchmark [20].

In contrast, we have limited data for the statistical approach. We felt encouraged that systems based on experts' knowledge (e.g. for the dialog management, query templates) are still performing well on the DSTC tasks [6]. Therefore we will stick to the handcrafted knowledge sources and the handcrafted dialog strategy and focus on the ease of extendibility.

The author of [8] explores similarly to us the question answering using a knowledge base from readily available heterogeneous sources. In contrast, we handle the creation of the knowledge base ourselves and from homogenous sources. Constructing a homogenous knowledge base benefits the whole system, as shown in [5].

Regarding the individual parts, our approach to using spoken language understanding for spoken content retrieval comes from an extensive overview in

[7] and from the experience our research team has in SLU. The database query generation is inspired by normalized queries from [3] and the dialog management follows loosely dialog acts and flow from [15].

3 Creating the Knowledge Base

The input to our system for the knowledge base generation (Fig. 1) is video footage with an interview that has separate audio channels for the interviewer and the interviewee. In our pipeline, we extract the audio and feed it to an ASR system [17]. This ASR uses an acoustic model trained on the audio from the available footage and language model from the transcribed part of the dataset. We have processed both the English and the Czech parts of the MALACH archive.

The output of the ASR system is a word lattice that is then used as an input to our knowledge base generator. The source footage is processed only upon addition to the system. This means we can use offline processing and do not have to worry about processing times.

We are using a semantic entity detection (SED) approach described in [16] to find the entities in multiple ASR hypotheses using weighted finite state transducers. This approach uses a 1-best hypothesis that our ASR system generates and outputs a list of detected semantic entities.

The semantic entities are defined in context-free grammars by their word forms (e.g. semantic entity ”mother” is described as ”mother OR mom OR ma”). The entities can be arranged in a hierarchy, saying for example that ”mother” is a ”family member”.

We have created grammars for entity classes that we consider to be the most useful to the dialog system in our domain: cities, countries, dates, names, proper nouns, family members, life events (e.g. birth, death, injury), education and other geographical places (e.g. camp names).

We run our algorithms once an audiovisual file is added to the source file storage. Apart from the word index for the previous search engine, we create a new searchable knowledge base from the detected semantic entities.

We process both the English and Czech portion of the corpora the same way. The only language-specific parts are the ASR (trained individually for each language) and the grammars (localized word forms, but the entities and the hierarchy stay the same).

The knowledge base if formed by (subject, predicate, object) triples. Apart from the semantic entities, we also store in the knowledge base the 1-best hypothesis from the ASR, the reference to the audiovisual source file and the metadata that were available for each speaker. For example, from the utterance in Fig. 2 we would store triples that are listed in Table 1. The triples create a graph-like structure for each interview as depicted in Fig. 3.

The design of the database (predicates and types) allows us to find answers to queries that were not possible to find using just a full-text search. We will show that using this structure we can create database queries that answer questions the user may ask. For example, the user can utter a natural language query

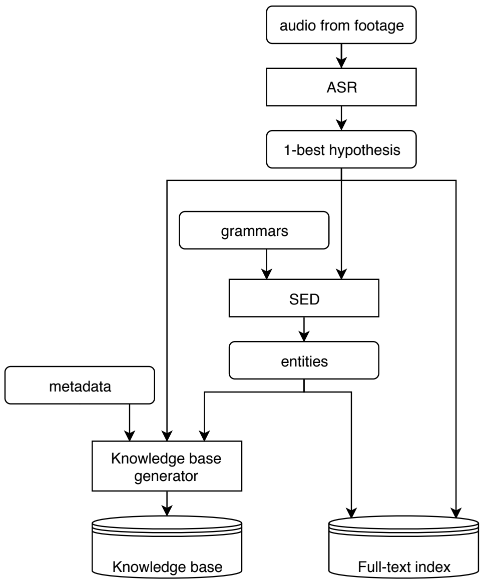

Fig. 1. The schema of the offline knowledge base generation.

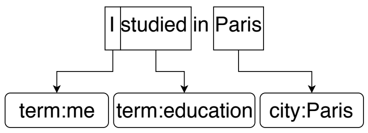

Fig. 2. In an utterance ”I studied in Paris” we will detect three entities, represented as strings ”terms:me”, ”terms:education” and ”city:Paris”. The first two entities belong to the class ”terms” that contains important terms and phrases related to the speaker. The last entity belongs to the ”city” class.

subject	predicate	object
”I”:Word	entityStart	”terms:me”:Entity
”I”:Word	next	”studied”:Word
”studied”:Word	entityStart	”terms:education”:Entity
”studied”:Word	next	”in”:Word
”in”:Word	next	”Paris”:Word
”Paris”:Word	entityStart	”city:Paris”:Entity
”file8266.1.wav”:File	recognition	”I”:Word
”id8266”:Interview	recording	”file8266.1.wav”:File

Table 1. Triples stored for the example from Fig. 2. The subject and the object are written in the form of ”value”:type tuple. For the sake of clarity of the example, we omitted the metadata and additional properties that we store, e.g. for the Word type we also store the timestamp when the word was uttered and the position in the recognized word sequence.

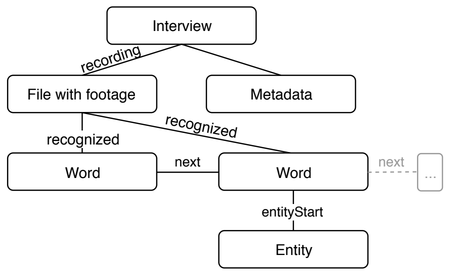

Fig. 3. A simplified schema of the graph of the objects that are stored in the knowledge base. The root of the graph is an object that represents the whole interview, connected to its metadata and files with the interview footage. These files relate to words the ASR recognized in the audio channel. The semantic entities detected in the output of the ASR keep the reference to their original word forms.

”Who was born in Paris?” and receive a spoken answer (e.g. ”sister”) together with video footage where the answer is mentioned.

On the other hand, the index is missing any deeper knowledge inference, e.g. creating instances of family members and representing the facts about them. When we recognize sentences such as ”My sister was born in April in Paris, she studied in Bern and later moved to Berlin.” we would like to understand the coreference and store that the speaker’s sister was born in April in Paris, studied in Bern and moved to Berlin. Right now, we only know that such entities were detected near each other and ignore that they belong to a single instance of the sister entity.

We are not able to extract this knowledge without any deeper syntactic and semantic analysis, but the design of the knowledge base is ready for such additional data.

Preliminary work has been done to assess whether our system will be able to infer and represent some of that knowledge using a tectogrammatical trees generated by TreeX [11]. This semantically oriented structural representation of the recognized utterance should be able capture the relationship between the entities. This expansion of knowledge base shows promises, but it will need to be researched further.

Furthermore, using output of the ASR for the TreeX algorithm is quite challenging, as the algorithm expects sentences with punctuation as its input, but we are only able to provide word sequences from the whole recognized audio segment (ca. 10 minutes) without any punctuation or sentence boundaries. Nevertheless, we have successfully enriched part of the knowledge base with these trees and created possible templates for the queries, leaving space for future research.

4 Dialog Management

With the information from the footage stored in our knowledge base, we now focus on describing the design of our agent-based dialog system. We describe how the user can use natural language to request information, ask follow-up questions or execute actions on the results. The interaction can be multimodal. We also describe the responses the system presents using a graphical user interface (GUI) and a text-to-speech synthesis (TTS).

We are using our in-house developed modular cloud-based framework Speech-Cloud for speech recognition, spoken language understanding, dialog management and text-to-speech synthesis. This framework also allows us to send messages to and from the graphical user interface. The graphical user interface is created as a web page.

As opposed to the ASR used for the knowledge base generation, this time we are using an online recognition, processing the user's utterance in real time. The ASR's acoustic model was trained on LibriSpeech [10] for the English part and on the archive footage for the Czech part. The language model was created from the transcribed portion of the MALACH archive.

The SLU module uses the same grammars as we use for the knowledge base generation (described in Sec. 3) since we know the user can look only for the things that are contained in these grammars. For the dialog, we need to include only an additional small set of grammars. They contain entities such as question type (who, when, etc.) and dialog actions (go back, repeat, play). To make the input to the dialog system more robust we leverage our ability to augment the recognizer's language model using the words from the grammars that we use for the SLU.

As mentioned in the introduction, we are enhancing an existing search platform that already had a rich graphical interface in form of a dynamic web page. This allows us to create a multimodal dialog - apart from the spoken dialog, we also show the video footage with the results on the screen and allow interaction with them. The structure of the whole system is depicted in Fig. 4. The ASR and TTS interfaces are integrated into that web page and controls to start and to stop the recognition are available to the user. We also allow textual input in natural language.

The framework is event-driven. Each module can dispatch its events and other modules can choose to react to them. The dialog manager reacts to messages from the SLU module and from the GUI. It then updates the dialog state and produces some event (if necessary). These events can be a request to the TTS modules or to the GUI (e.g. play a file, display search results).

The dialog manager itself is designed to be easily extended with event handlers and dialog agents. In the dialog manager, the events from the SpeechCloud framework are passed to the individual event-handling modules (Fig. 4). Some of the events the manager receives and processes are: recognition results with a 1-best hypothesis, results from the SLU module, the level of the input signal, the start or the end of the recognition, the start or the end of the synthesis and finally a change of the playing footage in the GUI.

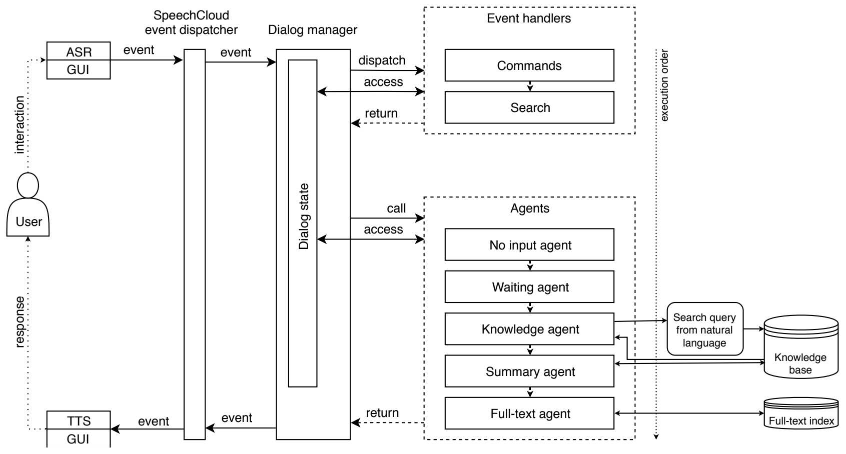

Fig. 4. The schema of the dialog manager reacting to the user's input that we receive as events from the SpeechCloud framework. The event handlers first parse the event's content, alter the dialog state and then the agents react to these changes. Finally, the system reports the responses from the agents to the user using TTS and GUI.

The purpose of the event handling modules is to update the state of the dialog. The dialog manager passes the event to these modules in a predefined order and each module can stop the propagation of the event to the modules that follow. Each module also has access to the state before the event, the current state (changed by the preceding modules) and the history of the states in the dialog.

The structure of the dialog state and the information it contains is maintained mainly by the event handlers and the dialog agents. In our current design, the state will contain most importantly the information what entities the user wants to retrieve and the 1-best hypothesis of the recognized utterance. In the state, there is also the number of "no input" events so far, information whether a search is ongoing, a reference to a currently played source file or dialog action flags that mark whether the user requests help, a repetition of a system's utterance or whether we recognized an affirmative or dismissive utterance.

We have implemented two event-handling modules. The first module handles events that represent dialog commands and GUI actions. The dialog commands are derived from the result of the SLU. The module will parse affirmative and dismissive utterances, requests for help or requests to repeat the system's utterance and change the dialog state accordingly. The events from the GUI contain information what footage is playing and again, this information is stored in the dialog state in case the user references the speaker from that footage.

The other module handles search events. It parses semantic entities that relate to the question-answering portion of the system (e.g. entities from Fig. 2) so that the dialog agents can later use them to perform the database queries.

After all the handlers process the event, we call the agents (also in a predefined order, see Fig. 4). The purpose of these agents is to react to the dialog state changes. We have created a knowledge agent that, based on the dialog state, retrieves information from the knowledge base using a retrieval algorithm described in Sec. 5. Then we are using a summary agent that queries the database for facts about the current speaker that may interest the user. We have also created other agents that improve the user experience: a no input agent reacts to the lack of input and a waiting agent that provides filler utterance when the search is taking a long time, as suggested in [4]. As a fallback search method, we have also implemented a full-text search agent.

The most important agent is the knowledge agent. This agent uses the search approach described in Sec. 5 to execute queries on our knowledge base. The search is executed only when the search event handler module parses search-related semantic entities from the user's utterance. Based on the dialog state, the knowledge agent decides what kind of query to use ("when", "what", etc. as in Sec. 5) and which entities should be filled into the template.

The knowledge agent also handles possible references to the current recording based on the dialog state. This allows the user to ask follow-up question regarding the current speaker (if there is any) as seen in Fig. 6. In case that the referenced speaker does not talk about the requested information, we search the whole knowledge base for that information. Then we inform the user that nothing was found for the speaker, but there are other speakers that talk about similar things. This strategy allows us to keep the dialog going and allows the user to explore the knowledge base.

The summary agent is designed to explore the knowledge base for the user in other cases. For example, it gives an overview of the most talked-about topics of the current speaker. This is also possible due to the design of our knowledge base, although we simply count all the classes of the entities that are linked to the speaker and present some of the most occurring ones.

Thanks to our full-text search agent we can handle user's requests that do not contain any indexed entities. In this case, we fall back to the full-text search engine, removing certain stop-words from the user's utterance and using the rest as a textual query. The results are then seamlessly presented to the user as if they were contained in our knowledge base, keeping a consistent user experience.

The combination of the event-handlers and the dialog agents results in a dialog manager that allows the user to explore the knowledge base and retrieve information that would be hard to access without it as seen in Fig. 6.

5 Search Queries from Natural Language

As mentioned in the introduction and in the description of the dialog manager, we are not only creating a knowledge base from the detected entities, but we want to query this database using spoken natural language. We are using a Neo4j [19] graph database as a storage for our knowledge base, but the SQL-like Cypher query language [9] that the database uses is far from natural (see Fig. 5).

```
MATCH (i:Interview)-->(f:File)-->(e:Entity)
WHERE e.value="city:Paris" RETURN i,f
```

Fig. 5. A Cypher query that retrieves all recordings in our knowledge base that mention the city of Paris.

We must create a mapping from the output of the dialog's ASR system onto the database queries. It would be unfeasible to write direct mapping from utterances (i.e. 1-best output from the ASR) to queries and the dialog would not be natural - the users would have to know specific structure of the phrases from the query templates.

Instead, we have based our mapping on a template selection, similarly to [18]. We had no previous examples of natural language searches in the domain and their mappings onto the Cypher queries. We had to rule out any statistical approach to creating and scoring such templates. But since our knowledge base is by design homogenous, the choice of a template is unambiguous, the templates can be created by experts and there is no need for scoring. The templates and items to be filled are selected solely from the output of the SED in the user's utterance. We base the SED on the same grammars that we use for the indexing (described in Sec. 3).

This approach has several advantages. The structure of the utterance will be more flexible, as we are looking only for entities that allow us to infer the meaning of the utterance and compile the correct query.

For each search, the algorithm requires a semantic representation of the user's input. This information will come from the dialog manager as updates to the dialog state. The state must contain information about what the user wants to retrieve and what restrictions we have received from the user. We will then choose which template to use and what needs to be filled in the template.

We have designed templates for four question types: "when", "who", "where" and "what". These templates contain queries that restrict the type of entities we retrieve as a result:

- "when" retrieves only dates,
- with "who", we are looking only for entities that represent names or family members,
- "where" restricts the results to geographical places (cities, camp names, countries, etc.) and
- "what" returns entities like life events.

If we can't categorize the utterance as one of these four types, we can use a generic template that does not restrict the result type and will return all the entities that were included in the query.

The gist of the templates is that the users are looking for something that connects entities they have uttered. In our implementation of the system, the templates only restrict to a co-occurrence of the entities. This means that the

S: Hello, what can I do for you?
U: <silence>

The agent that handles a missing input reacts for the system.

S: You can start the conversation by tapping the microphone.
U: I want to know about deaths in Auschwitz.

Utterance is parsed as a search request with entities ”death” (terms class) and ”Auschwitz” (city class). The knowledge agent executes a database query and presents the results and a question.

S: I’ve found 68 records. Do you want me to start playing them?
U: Yes.

The command handler finds affirmative keyword and the knowledge agent interprets it as the answer to its question.

S: <playback started>
U: What else is mentioned in this record?

The search handler finds the request for a summary and a reference to the current speaker. The summary agent retrieves the summary.

S: (person1) also talks about their father, sister and school.
U: Do they mention Paris?

The search handler keeps the reference to the current speaker and a search request for Paris (city class). The knowledge agent fails to find any result for the referenced speaker but searches the rest of the database.

S: (person1) doesn’t talk about that. But I have found more than 70 records from others.
S: Do you want me to start playing them?
U: No. Whose sister comes from Paris?

The command handler finds dismissive keyword and the search handler finds the request for the question type ”whose” with ”sister” and ”Paris” as the entities the user is looking for. The knowledge agent finds the records and presents the answer.

S: I’ve found 10 records. One of the people mentioned is (person2)
U: And the other ones?

The command handler parses the keyword for the next record and the knowledge agent handles the system’s reaction as a continuation of its previous search.

S: Another person is (person2)

Fig. 6. Commented transcript of a dialog. S is the system, U is the user. The commentary is in italics.

queries are restricted only to such source files that have all the entities near each other, measured by the position in the speaker’s utterance.

We are also able to restrict the search to a certain speaker (e.g. when the user uses the speaker’s name). Since the entities are linked to nodes representing each speaker, the queries are then altered to retrieve only these entities.

For example, applying our approach to the sentence ”My sister was born in April in Paris, she studied in Bern and later moved to Berlin”, we are able to find answers to questions similar to ”Where did her sister study?”, ”Who was born in Paris?” or ”When was her sister born?”, but we are not able to answer ”Tell me about her sister”.

Once we select and fill the correct template, we execute the query, retrieve information from the knowledge base and parse the results. Since we are focused on multimodal communication, we are also retrieving links to the video sources that contain the results and the word forms and timestamps of the entities we find. We make them available to the dialog manager that can e.g. present the user the answer together with the video footage in the GUI. This is easy to achieve because the entities in our knowledge base have links to their word forms and the source files.

For queries with the generic template, we retrieve the word forms of all the entities the user was looking for. For example, the query ”Find me family members and their education.” contains two entities: ”family_member”:terms, ”education”:terms. It maps onto a generic query template. The word forms of all the entity pairs will be returned, e.g. (”sister”, ”grammar school”), (”father”, ”university”).

For queries that have specific result types (when, who, where, what), we retrieve only word forms of these results. To show an example, we can say that ”When was his sister born?” contains four entities (”when”:keyword, ”reference”:keyword, ”sister”:terms, ”born”:terms). The algorithm will choose the ”when” template and fill it with the ”terms” entities while restricting the search to the speaker of the current video footage. It will return only word forms of the dates it finds e.g. ”April 1968”.

6 Conclusion and Future Work

We have presented a modular system that allows the user to retrieve information from large audio-visual archives using spoken natural language queries and explore the knowledge base. We have deployed the system for English and Czech parts of the MALACH archive. A transcript of a dialog with the system that showcases how the information can be retrieved is in Fig. 6.

We can conclude that basing the system on expert-made grammars and query templates is sufficient to create a working system with a reasonable impact on the amount of information that becomes available to the users. The semantic entity detection and augmenting the language model using the grammars results in a dialog that is less prone to errors in speech recognition.

The design of the presented system allows for a quick transfer to a different domain and easy expansion of the knowledge base. The work on using tectogrammatical trees to further enhance the knowledge base, creating a true representation of facts about the objects is still ongoing. Another expansion to our knowledge base that we are planning to incorporate will come from semantic web sources that are related to our domain with focus on keeping our knowledge base homogenous.

Acknowledgement

This work was supported by the European Regional Development Fund under the project Robotics for Industry 4.0 (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000470), by the Technology Agency of the Czech Republic, project No. TE01020197 and by the grant of the University of West Bohemia, project No. SGS-2019-027.

References

1. Bordes, A., Boureau, Y.L., Weston, J.: Learning End-to-End Goal-Oriented Dialog. In: ICLR (2017), http://arxiv.org/abs/1605.07683
2. Choi, E., He, H., Iyyer, M., Yatskar, M., Yih, W.t., Choi, Y., Liang, P., Zettlemoyer, L.: QuAC: Question Answering in Context. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. pp. 2174–2184. Association for Computational Linguistics, Brussels, Belgium (2018), https://www.aclweb.org/anthology/D18-1241
3. Dubey, M., Dasgupta, S., Sharma, A., Höffner, K., Lehmann, J.: AskNow: A framework for natural language query formalization in SPARQL. In: Lecture Notes in Computer Science. vol. 9678, pp. 300–316 (2016)
4. Gambino, S.L., Zerrieß, S., Schlangen, D.: Testing Strategies For Bridging Time-To-Content In Spoken Dialogue Systems. In: Proceedings of the Ninth International Workshop on Spoken Dialogue Systems Technology. pp. 1–7 (2018)
5. Gurevych, I., Porzel, R., Slinko, E., Pfleger, N., Alexandersson, J., Merten, S.: Less is more: Using a single knowledge representation in dialogue systems. In: Proceedings of the HLT-NAACL Workshop on Text Meaning. pp. 14–21 (2003)
6. Kadlec, R., Vodolan, M., Libovicky, J., Macek, J., Kleindienst, J.: Knowledge-based Dialog State Tracking. In: 2014 IEEE Spoken Language Technology Workshop (SLT). pp. 348–353. No. 1, IEEE (dec 2014), http://ieeexplore.ieee.org/document/7078599/
7. Lee, L.s., Glass, J., Lee, H.y., Chan, C.a.: Spoken Content Retrieval—Beyond Cascading Speech Recognition with Text Retrieval. In: IEEE/ACM Transactions on Audio, Speech, and Language Processing. vol. 23, pp. 1389–1420 (sep 2015), http://ieeexplore.ieee.org/document/7114229/
8. Lopez, V.: PowerAqua : Open Question Answering on the Semantic Web. Ph.D. thesis (2011)
9. Neo4j, Inc.: The Neo4j Cypher Manual v3.5 (2019), https://neo4j.com/docs/cypher-manual/3.5/
10. Panayotov, V., Chen, G., Povey, D., Khudanpur, S.: Librispeech: An ASR corpus based on public domain audio books. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. vol. 2015-Augus, pp. 5206–5210 (2015)
11. Popel, M., Žabokrtský, Z.: TectoMT: Modular NLP Framework. In: IceTAL, 7th International Conference on Natural Language Processing. pp. 293–304. Reykjavik (2010), https://ufal.mff.cuni.cz/treex

12. Psutka, J., Radová, V., Ircing, P., Matoušek, J., Müller, L.: USC-SFI MALACH Interviews and Transcripts Czech LDC2014S04 (2014), https://catalog.ldc.upenn.edu/LDC2014S04
13. Ramabhadran, B., Gustman, S., Byrne, W., Hajič, J., Oard, D., Olsson, J.S., Picheny, M., Psutka, J.: USC-SFI MALACH Interviews and Transcripts English (2012), https://catalog.ldc.upenn.edu/LDC2012S05
14. Stanislav, P., Švec, J., Ircing, P.: An engine for online video search in large archives of the holocaust testimonies. In: Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. vol. 08-12-Sept, pp. 2352–2353 (2016)
15. Stede, M., Schlangen, D.: Information-Seeking Chat: Dialogue Management by Topic Structure. In: Proceedings of the 8th Workshop on the Semantics and Pragmatics of Dialogue. pp. 117—-124 (2004)
16. Švec, J., Ircing, P., Šmídl, L.: Semantic entity detection from multiple ASR hypotheses within the WFST framework. In: 2013 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2013 - Proceedings. pp. 84–89 (2013)
17. Švec, J., Psutka, J.V., Trmal, J., Šmídl, L., Ircing, P., Sedmidubsky, J.: On the Use of Grapheme Models for Searching in Large Spoken Archives. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings. vol. 2018-April, pp. 6259–6263 (2018)
18. Unger, C., Bühmann, L.: Template-based question answering over RDF data. In: Proceedings of the 21st international conference on World Wide Web. pp. 639–648 (2012), http://dl.acm.org/citation.cfm?id=2187923
19. Webber, J.: A programmatic introduction to Neo4j. In: Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity. p. 217 (2012)
20. Williams, J.D., Henderson, M., Raux, A., Thomson, B., Black, A., Ramachandran, D.: The Dialog State Tracking Challenge Series. AI Magazine 35(4), 121 (2017)