Using Dependency Parsing and Machine Learning for Factoid Question Answering on Spoken Documents

Pere R. Comas, Jordi Turmo, Lluís Màrquez

TALP Research Center, Technical University of Catalonia (UPC), Spain

pcomas@lsi.upc.edu, turmo@lsi.upc.edu, lluism@lsi.upc.edu

Abstract

This paper presents our experiments in question answering for speech corpora. These experiments focus on improving the answer extraction step of the QA process. We present two approaches to answer extraction in question answering for speech corpora that apply machine learning to improve the coverage and precision of the extraction. The first one is a reranker that uses only lexical information, the second one uses dependency parsing to score robust similarity between syntactic structures. Our experimental results show that the proposed learning models improve our previous results using only hand-made ranking rules with small syntactic information. Moreover, this results show also that a dependency parser can be useful for speech transcripts even if it was trained with written text data from a news collection. We evaluate the system on manual transcripts of speech from EPPS English corpus and a set of questions transcribed from spontaneous oral questions. This data belongs to the CLEF 2009 track on QA on speech transcripts (QAst).

Index Terms: question answering, answer extraction, oral question answering, speech transcriptions

1. Introduction

Question Answering (QA) is the task of extracting short, relevant textual answers in response to natural language questions. As a subset of QA, factoid QA, focuses on questions whose answers are syntactic and/or semantic entities, e.g. organization names, person names, dates, etc. QA is different from Information Retrieval (IR) because it outputs concrete answers to a question instead of references to full documents which are relevant to a given set of query words. This difference is very important when working with spoken documents instead of written documents, since accuracy is much more desirable for a human user when searching through audio records.

Most of factoid QA systems have three main modules that work in a sequential pipeline [1]. First of all the question processing module (QP) examines the question to extract a set of query terms from it and decide what kind of answer is expected for this question. For factoid questions this reduces to guess the type of named entity (NE) that answers the question. The second module is an IR engine. It retrieves documents or shorter passages from the collection using the query terms extracted by the former module. Ideally, these passages should contain the answer we are looking for. Finally the answer extraction (AE) module extracts exact answers from the retrieved passages. First, answer candidates are identified as the set of NEs that occur in these passages and have the same type as the answer type detected by QP (e.g. for the question *"When were biometric data included in passports?"* all retrieved entities of types 'date' or 'time' are identified as candidate answers). Then, these candidates are ranked using a scoring function and the top n are returned as possible answers.

Current text–based QA systems use NLP technology that require text written in accordance with standard norms for written grammar but the syntax of speech is quite different from that of written language. Speech contains disfluencies, repetitions, restarts and corrections, moreover, any practical application of search in speech requires the transcriptions to be produced automatically, and the Automatic Speech Recognisers (ASR) introduce a number of errors. For these reasons, almost any QA system for spoken documents uses only very shallow linguistic processing (i.e. part-of-speech tagging and named entity recognition) since these are more robust than complex tools like full syntax parsers or coreference resolution systems.

The interest in question answering on spoken documents is very recent and the literature about this subject is still small. Many researchers currently working on QA on spoken documents participate in the Crosslingual Evaluation Forum workshop (CLEF). CLEF holds annual competitions on various QA tasks, including an spoken document task [1, 2, 3] named QAst (Question Answering on Speech Transcripts). The best system of the 2009 evaluation, developed by LIMSI [4], makes extensive use of hand-crafted rules and patterns. They use them for named entity (NE) recognition, question classification and answer extraction. The answer extraction module uses a chunker and a rule-based system for identifying relations between chunks, then candidates are scored according to edit distance with respect to the question.

Many works dealing with written text take a more sophisticated approach to answer extraction using syntactic or semantic information. Some use syntax or semantic roles to identify predicates that are not NEs but may be candidate answers [5]. These candidates are selected if they are semantic arguments of a suitable type and are related to a search predicate from the question. Other approaches use more informed techniques like reducing AE to a graph matching problem between semantic structures of the question and text passages [6]. Using spoken documents is a severe difficulty for the whole process of QA and would make these approaches infeasible, specially since there is no data to train semantic role labellers or full parsers on spoken documents.

In this work we use some tools designed for written text and machine learning techniques to improve the AE step of a question answering system substantially. First of all we describe a baseline system for the extraction that uses only shallow information and a manually developed heuristic combination. Then,

The work leading to this research has received funds from the Spanish Ministry of Science and Technology, projects KNOW2 (TIN2009–14715–C04–04) and OPENMT2 (TIN2009–14675–C03).

it is improved with a reranker based on machine learning. Finally we use a dependency parser to get more informed features (but with a high error ratio) for the ranking, this improves the precision and coverage of the extraction.

2. Approach

The following sections describe our baseline answer extractor and two improvements of increasing complexity. In these experiments we have used two different sets of questions as provided by the QAst evaluation, one for development (called DEV) and one for test (called TEST). Details about the document collection and question sets are provided in the description of QAst corpora in Section 3.1.

2.1. Heuristic Answer Extractor

We will refer to the baseline version of the AE as Heuristic extractor. This is based on the properties of the context where the candidate answers appear in the retrieved passages. The candidates are ranked using a scoring function based on a set of seven heuristics that measure keyword distance and density. Heuristics are based in those of [7]:

- H_1: *Same word sequence*: computes the number of words that are recognized in the same order in the answer context.
- H_2: *Punctuation flag*: 1 when the candidate answer is followed by a punctuation sign, 0 otherwise.
- H_3: *Comma words*: computes the number of question keywords that follow the candidate answer, when the later is succeeded by comma. A span of 3 words is inspected.
- H_4: *Same sentence*: the number of question words in the same sentence as the candidate answer.
- H_5: *Matched keywords*: the number of question words found in the answer context.
- H_6: *Answer span*: the largest distance (in words) between two question keywords in the given context.
- H_7: *Distance from QFW*: measures the distance between the candidate answer and the question focus word. This is enabled only for certain types of type question.

All these heuristics can be implemented without the need for any natural language processing resources outside of a basic tokenizer. For each candidate answer, these seven values are then converted into an overall answer score using the formula

$$score = H_1 + H_2 + 2H_3 + H_4 + H_5 - \frac{1}{4}\sqrt{H_6} - H_7, \quad (1)$$

where the heuristic weights were manually optimized following [7]. This score drives the final answer ranking.

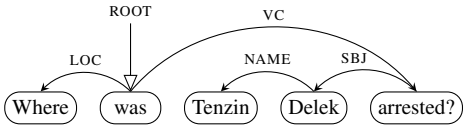
2.2. Heuristic Reranker

The previously described Heuristic extractor is dependant of the collection, since the weights must be tuned according to the characteristics of the documents, and it is not an easy task since it involves several variables at the same time. In addition, this is a rough score that doesn't make any use of simple information such as expected answer type or repetition of candidates. Thus, we have implemented a machine learning layer that takes the seven heuristic scores and reranks the candidates according to a model learned with the DEV set. This approach makes unnecessary to manually tune the heuristics' weights since it is done in the learning, and it can include more information in the extraction process.

We have used binary Support Vector Machines (SVM) for learning a classifier to distinguish correct from incorrect answer candidates.[1] Each candidate answer got from the development questions using the former Heuristic extractor is a training example. The candidates may be either correct or wrong answers, being either a positive or negative example in the SVM model. For each candidate, the following set of features is computed: 1) score value (equation 1); 2) order in the ranking according to score; 3) values of each heuristic $H_1..H_7$; 4) number of times this candidate is repeated; 5) length in number of words; 6) candidate's type of NE; 7) number of keyword in the query.

These features are converted into binary features before the learning process. To do this we procede as follows: first the range of values of each feature is split into k parts, numbered from 0 to $k-1$. Then each value is expanded with a series of inequalities framing it. For example, if the candidate answer is 3 words long (feature *clen:3*), these new features are added: *clen>0*, *clen>1*, *clen>2*, *clen<4*...*clen<k-2*, *clen<k-1*. Categorical values (i.e. type of NE) are not binarized.

Question 9: Where was Tenzin Delek arrested?



Path from *Where* to *Tenzin_Delek*: LOC - ROOT - VC - SBJ - NAME
Simplified path: LOC - VC - SBJ - NAME

Figure 1: Sample question with dependency parsing

2.3. Adding Syntactic Information

Keyword density measures from 2.1 do not capture meaning. Any good QA system should use syntactic and semantic information to understand the text and make deductions from it, but this is even more difficult in speech transcripts. In this experiment we add syntactic information to our answer extractor to improve its ability to distinguish correct and incorrect candidates.

We have labelled the collection with syntactic relations using an inhouse dependency parser [8];[2] thus any pair of words in a sentence are linked by a sequence or path of syntactic relations. We have also labelled the questions with syntactic relations. Our dependency parser has been trained with the CoNLL 2007 Shared Task collection, a collection of newspaper texts.[3]

The key assumption is that the syntactic relations between the keywords and the candidate answer (in the collection) should be similar to the syntactic relations between the keywords and the question tag in the question. This denotes that keywords in the text are framing the candidate answer with restrictions similar to those expressed in the question. For factoid questions this question tag is either *who*, *where*, *when*, *how*, *what* and *which*. Comparing the paths should help disregard candidate answers that are near the keywords but are not properly related to them and getting candidates that are closely related syntactically but far away in number of tokens (i.e. those that can not be captured with local heuristics).

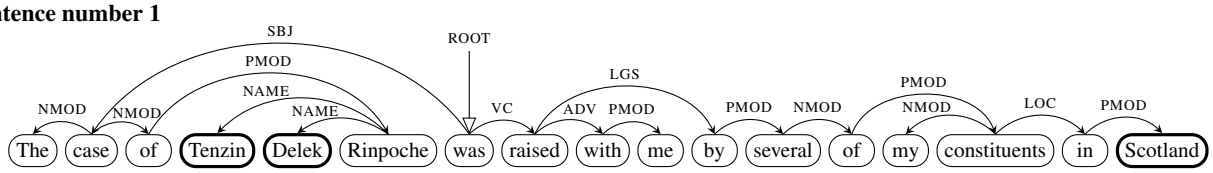
For example: consider the question "*Where was Tenzin Delek arrested?*" Figure 1 shows the path of syntactic relations joining question tag *Where* with keywords *Tenzin Delek*. Figure 2 shows the parsing of two sentences from the EPPS

[1] `http://svmlight.joachims.org`

[2] `http://www.lsi.upc.edu/~xlluis/jointparser/`

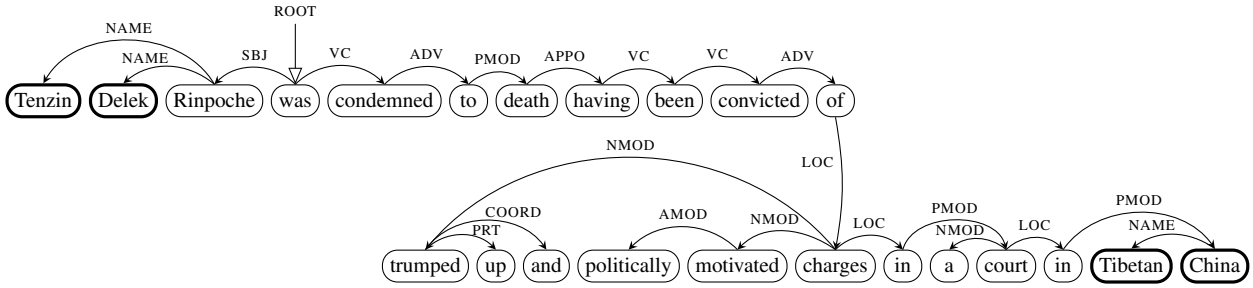
[3] Adapting the parser to speech is beyond the scope of our work. We only focus in extracting robust features to make the parser useful in speech.

Sentence number 1



Path from *Scotland* to *Tenzin Delek*: PMOD - LOC - PMOD - NMOD - PMOD - LGS - VC - ROOT - SBJ - NMOD - PMOD - NAME
Simplified path: LOC - LGS - VC - SBJ - NAME

Sentence number 2



Path from *Tibetan China* to *Tenzin Delek*: PMOD - LOC - PMOD - LOC - PMOD - ADV - VC - VC - APPO - PMOD - ADV - VC - ROOT - SBJ - NAME
Simplified path: LOC - LOC - VC - SBJ - NAME

Figure 2: Two examples of dependency parsing

collection that contain candidate answers, *Scotland* and *Tibetan China*. Our heuristic measures based on keyword density can not distinguish between the correct one (*Tibetan China*) and the totally unrelated one (*Scotland*). In Sentence 1, *Scotland* is a locative nominal modifier of *constituents* but is not related to *Tenzin Delek*. Prior to comparison, paths are simplified to avoid sparsity removing frequent labels that are of little use like name modifiers (NMOD), preposition modifiers (PMOD) and general adverbs. Sequences of contiguous verbs are representes as a single VC label. Note that label ROOT denotes the main verb of the sentence; we transform it into VC since there is no need to force the paths to contain the *main* verb of the sentence. If we compare the path from the question with the paths from Sentence 1 (shown in Figure 2) we can see that the latter differs with an extra LGS relation. LGS denotes the logical subject of a passive verb. This means that *Scotland* modifies a noun phrase that has a syntactic relation with the main verb other than a locative modifier. Thus *Scotland* is not necessary expressing a locative restriction of a verb whose subject is *Tenzin Delek*. If we look at Sentence 2 we found one extra LOC relation, so it means that *Tibetan China* is a locative modifier of a locative modifier whose subject is the keyword *Tenzin Delek*.

To compare two given paths Q_k and T_k, where Q_k has been extracted from the question for keyword k and T_k from the collection, we use a dynamic programming algorithm to align them. It finds the longest sequence (M_k) of labels that can be matched without changing it's order. Then it computes the labels from Q_k that are missing in T_k and viceversa. This information is summarized in a set of features that enrich the model described in the previous section. These features must be very robust since the result of parsing speech transcripts can be very poor. For each candidate answer, these features are added:

1. Number of keywords syntactically related and the proportion over total keywords.
2. Distance from candidate answer to keywords in number of syntactic relations between them.
3. The length of M_k and the ratio $|M_k|/|Q_k|$ for each keyword k. Also the maximum, minimum and average of each.
4. Total number of inserted labels and the count for each different label.
5. Total number of matched labels: $\sum_{\forall k} |M_k|$.

Before learning the model, the ratios are discretized to intervals and integer values are expanded to binary features as explained in the previous section.

3. Experimental Results

3.1. QAst corpus

In these experiments we have used the datasets provided by the QAst evaluation task from CLEF 2009 workshop. This data is a collection of manual transcriptions of 3 hours from the European Parliament Plenary Sessions (EPPS) in English. This is about 35,000 words. The sessions are divided in turns according to speaker. The only punctuation mark is the full stop but there are marks for hesitations and partial words and most of the names are capitalized. There are two sets of questions to be answered using the collection: a development set (DEV) of 50 questions and a test set (TEST) of 100 questions. The questions are spontaneous oral questions manually transcribed [9]. Each set contains factoid and definitional questions in a proportion of roughly 75%–25%. Some of the questions do not have an answer in the collection, therefore the correct answer for them is "nil." We have experimented only with factoid QA.

3.2. Measures

We have evaluated our answer extractor in the context of the QAst evaluation, using the TEST set questions with manual transcripts of the EPPS corpus (see Section 3.1 for details). The models for the rerankers have been trained using the DEV question set and then evaluated with the TEST set. Our evaluation

Model	T1	T5	MRR	Acc.
Upper Bound	46	46	0.6133	61.33%
Heuristic Baseline	15	35	0.3013	20.00%
Heuristic Rerank	19	34	0.3360	25.33%
Syntactic Rerank	22	37	0.3687	29.33%
LIMSI	22	42	0.3931	29.33%
INAOE	18	44	0.3824	24.00%
TOKYO TECH	5	11	0.1067	6.67%

Table 1: Experimental results

reports the same measures than QAst evaluation. The QA system outputs a ranking of 5 answers for each question, this is evaluated with two measures:

Mean Reciprocal Rank (MRR): Average of inverses of the ranking of the first correct answer for each question. It is defined as $\frac{1}{N}\sum_{i=0}^{N}\frac{1}{rank_i}$, where $rank_i$ is the position of the first correct answer in the answer's list for question number i.

Accuracy: The fraction of correct answers ranked in the first position in the list of 5 possible answers.

Additionally, we report T1 and T5 measures. T1 is the number of questions that have a correct answer ranked first, T5 denotes how many have a correct answer anywhere in the ranking.

3.3. Results

As we have exposed in Section 1, the AE module is the last one in the QA pipeline. The previous modules may introduce errors that make impossible to extract the correct answer (e.g. errors in passage retrieval, errors in the expected answer type, etc.). Due to these errors, in this experiment it is possible to extract the correct answer in only 46 of the 75 factoid questions, so this is the theoretical upper bound of our answer extractor. Table 1 contains MRR and Accuracy for each of our three answer extractors (the heuristic baseline and the two rerankers), and the upper bound.

Table 1 shows that Heuristic Reranker gives a better ranking of candidates than the Heuristic approach in MRR and Accuracy. T1 increases from 15 to 19 but coverage (T5) is not improved. This may be explained by the fact that the this reranker do not include truly new information in the features, it just makes a better use of the heuristics.

The Syntactic Rerank improves both MRR and Accuracy of Heuristic Rerank in more than 3 points. Now both T1 and T5 are improved indicating that dependency parsing incorporates useful and new information.

The last three rows of the table are the results achieved by the rest of participants in the QAst evaluation 2009. The best results in MRR and Accuracy where obtained by the LIMSI group [4]. The results of LIMSI are better than any of our three runs in MRR,[4] but our Syntactic Rerank achieves the same Accuracy. This is an important results because our approach is free of hand-crafted or language-dependant rules and it can be easily adapted to different tasks.

It is remarkable that even using a small training set of 50 training questions this is enough to get a significant impact with both methods. Table 2 show the results of a second experiment with expanded training sets. In this experiment a reranker model has been learned for each test question using a different training set for each one. The training set is formed by all examples from DEV set and all examples from TEST set but the corresponding to the question itself. Therefore the training sets contain 149 questions instead of just 50 like in the first experiment. This is a leave-one-out evaluations, mixing DEV and TEST sets do not biases the results since all questions are totally independant. As expected, the results are much better when using bigger training sets, both rerankers have a parallel improvement of 6 or more points and the coverage of the Syntactic Reranker reaches an 89% of the upper bound.

[4]It is not possible to know if this difference is due solely to AE or because modules like QC and IR are better than ours. This would require a white-box evaluation and this is beyond the scope of this paper.

Model	T1	T5	MRR	Acc.
Heuristic Rerank	26	38	0.4036	34.67%
Syntactic Rerank	27	41	0.4251	36.00%

Table 2: Results with an expanded traning set

4. Conclusions

In this paper, we have presented two approaches to answer extraction in question answering for speech corpora that apply machine learning to improve the coverage and precision of the extraction. The first one is a reranker that uses only lexical information, the second one uses dependency parsing to score similarity between syntactic structures. Our experimental results show that the proposed learning models improve our previous results using only hand-made ranking rules. Moreover, this results show also that a dependency parser can be useful for speech transcripts even if trained with written text data.

5. References

[1] J. Turmo, P. R. Comas, S. Rosset, O. Galibert, N. Moreau, D. Mostefa, P. Rosso, and D. Buscaldi, "Overview of QAst 2009," *Proceedings of the CLEF 2009 Workshop on Cross-Language Information Retrieval and Evaluation*, 2009.

[2] J. Turmo, P. R. Comas, S. Rosset, L. Lamel, N. Moreau, and D. Mostefa, "Overview of QAst 2008," in *Evaluating Systems for Multilingual and Multimodal Information Access*. Springer Berlin / Heidelberg, 2009, pp. 314–324.

[3] J. Turmo, P. Comas, C. Ayache, D. Mostefa, S. Rosset, and L. Lamel, "Overview of QAst 2007," *Proceedings of the CLEF 2007 Workshop*, 2007.

[4] G. Bernard, S. Rosset, O. Galibert, G. Adda, and E. Bilinski, "The LIMSI participation to the QAst 2009 track: Experimenting on answer scoring," *Proceedings of the CLEF 2009 Workshop on Cross-Language Information Retrieval and Evaluation*, 2009.

[5] S. Yaman, D. Hakkani-Tür, and G. Tur, "Combining semantic and syntactic information sources for 5-W question answering," in *INTERSPEECH*, 2009.

[6] D. Shen and M. Lapata, "Using semantic roles to improve question answering," in *Proceedings of the EMNLP-CoNLL*, 2007.

[7] M. Surdeanu, D. Dominguez-Sal, and P. Comas, "Design and performance analysis of a factoid question answering system for spontaneous speech transcriptions," *INTERSPEECH 2006*, 2006.

[8] X. Lluís, S. Bott, and L. Màrquez, "A second-order joint eisner model for syntactic and semantic dependency parsing," in *Proceedings of the CoNLL 2009: Shared Task*, 2009.

[9] D. Buscaldi, P. Rosso, J. Turmo, and P. R. Comas, "Towards the evaluation of voice-activated question answering systems: Spontaneous questions for QAst 2009." in *Proceedings of the III Jornadas PLN-TIMM. Madrid, Spain.*, 2009.