Design and Performance Analysis of a Factoid Question Answering System for Spontaneous Speech Transcriptions

Mihai Surdeanu, David Dominguez-Sal, and *Pere R. Comas*

Technical University of Catalunya, Barcelona, Spain
{surdeanu, pcomas}@lsi.upc.edu, ddomings@ac.upc.edu

Abstract

This paper introduces a QA designed from scratch to handle speech transcriptions. The system's strength is achieved by analyzing the speech transcriptions with a mix of IR-oriented methodologies and a small number of robust NLP components. We evaluate the system on transcriptions of spontaneous speech from several 1-hour-long seminars and presentations and show that the system obtains encouraging performance.

Index Terms: question answering, natural language processing

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, dates, etc. As of today, most of the QA research has focused on written, grammatically-correct text, and most approaches are based on heavy syntactic and semantic analysis of the text, e.g. full syntactic parsing, textual entailment, and semantic role labeling [5]. The few approaches that stepped away from the written-text framework have focused mainly on spoken questions but continued to extract answers from written document collections [1].

While QA on written text has without a doubt significant real-world applications – e.g. search engines, automated customer service – many important scenarios are not addressed. For example, a QA system could be used to extract information from transcriptions of presentations, seminars, meetings or other types of reunions, news or radio broadcasts, and many more. The importance of speech transcriptions to many real-world applications motivates the work presented in this paper: we design and analyze a factoid QA system that answers (currently) written questions with snippets extracted from spontaneous speech transcriptions of seminars and presentations.

The shift from written text to spontaneous speech transcriptions means that a series of phenomena that make text processing difficult are emphasized: disfluency or stuttering, speaker corrections and specifications, and lack of grammatical structure. All these issues indicate that a corresponding shift has to take place in the design of a QA system tailored for spontaneous speech transcriptions: instead of the usual in-depth processing of the target text, the QA system must use only natural language processing (NLP) tools that are robust enough to function on speech transcriptions. In this paper we show that such a design is possible: we describe a QA system that obtains state-of-the-art performance by combining two robust NLP tools – a part-of-speech (POS) tagger and a named entity recognizer and classifier (NERC) – with several measures of keyword density and proximity that can be easily extracted from any speech transcription.

The paper is organized as follows. Section 2 overviews the architecture of the QA system. Section 3 introduces the strategy employed for the identification of candidate answers in speech transcriptions. Section 4 describes the experimental results, and Section 5 concludes the paper.

2. System Architecture

The QA system introduced in this paper uses a typical architecture consisting of three components linked sequentially: *Question Processing* (QP), which identifies the type of the input question, followed by *Passage Retrieval* (PR), which extracts a small number of relevant passages from the underlying speech transcriptions, and finally *Answer Extraction* (AE), which extracts and ranks exact answers from the previously retrieved passages. This section describes all these components. In the next section, we detail the most important module of AE: the *identification of candidate answers* in speech transcriptions.

2.1. Question Processing

The QP component detects the type of the input questions by mapping them into a two-level taxonomy consisting of 6 question types and 53 subtypes:

type	subtype
ABBREVIATION	abbreviation, expression abbreviated
ENTITY	animal, body organ, color, creative work, currency, disease, event, food, instrument, language, letter, other, plant, product, project, religion, sport, symbol, system, technique, equivalent term, vehicle, special word
DESCRIPTION	definition, description, manner, reason
HUMAN	group, individual, title, description
LOCATION	city, country, mountain, other, state
NUMERIC	angle, code, count, date, distance, money, order, other, period, percent, speed, temperature, size, weight

In our system there is a one-to-one mapping from question type to the category of the expected answer. For example, the tuple `HUMAN:individual` entails that the answer is a named entity of type `PERSON`.

The above question taxonomy is largely inspired by [2]. Nevertheless our classification mechanism is different: instead of using a hierarchy of $6 + 53$ binary classifiers (one for each type and subtype), we opted for a single Maximum Entropy multi-class classifier that extracts the best tuple $<$`type:subtype`$>$ for every question. We chose the single-classifier design because it significantly improves the classification response time, which is an paramount requirement for any interactive system. We compensate for the possible loss of accuracy with a richer feature set. For-

Table 1: *The feature extraction functions for the question classifier.* w - *token word,* l - *token lemma,* p - *token POS tag,* **sem** - *set of semantic classes (from [2]) that contain this word,* **prox** - *set of proximity-based word sets (from [3]) that contain this word,* qfw - *the QFW detection function.* $\cdot$ *stands for string concatenation.*

$\phi_{\mathbf{sequence}}(\mathbf{x})$
foreach($x_i \in \mathbf{x}$) add features:
w(x_i), l(x_i), **sem**(x_i), **prox**(x_i),
w(x_i) $\cdot$ w(x_{i+1}), l(x_i) $\cdot$ l(x_{i+1})
foreach($c \in$ **sem**$(x_i), c' \in$ **sem**(x_{i+1})): $c \cdot c'$
foreach($c \in$ **prox**$(x_i), c' \in$ **prox**(x_{i+1})): $c \cdot c'$

$\phi_{\mathbf{qfw}}(\mathbf{x})$
add features:
w(qfw(**x**)), l(qfw(**x**)), **sem**(qfw(**x**)), **prox**(qfw(**x**)),
w(x_0) $\cdot$ w(qfw(**x**)), w(x_0) $\cdot$ p(qfw(**x**))
foreach($c \in$ **sem**(qfw(**x**))): w(x_0) $\cdot$ c
foreach($c \in$ **prox**(qfw(**x**))): w(x_0) $\cdot$ c

mally, our question classifier assigns a question class – i.e. tuple <type:subtype> – to each question, using the function:

$$qc(\mathbf{q}) = \arg\max_{c \in \mathcal{C}} \text{score}(\phi(\mathbf{q}), c) \quad (1)$$

where **q** is the sequence of all the question words, e.g. {"What", "is", "the", "Translanguage", "English", "Database", "also", "called"}, $\mathcal{C}$ is the set of all possible question classes, score is the classifier confidence, and ϕ is a feature extraction function, defined as a composition of several base feature extraction functions:

$$\phi(\mathbf{q}) = \phi_{sequence}(\mathbf{q}) + \phi_{sequence}(\mathbf{h}) + \phi_{qfw}(\mathbf{q}) \quad (2)$$

where **h** is the sequence of heads of the basic syntactic phrases in the question, e.g. {"What", "is", "Database", "called"} for the above example, $\phi_{sequence}$ extracts n-gram features from a sequence of words, and ϕ_{qfw} extracts features related to the question focus word (QFW). The QFW, which indicates the question emphasis, is usually the head of the first noun or verb in the question, skipping stop words, auxiliary and copulative verbs. For example, for the above question, the QFW is "database". We have implemented the detection of the QFW with 7 surface-text patterns. We detail the feature extraction functions in Table 1.

2.2. Passage Retrieval

Our passage retrieval algorithm is inspired by the query relaxation algorithm of [4], which adds or drops query keywords depending on the quality of the information retrieved. Our implementation is capable to adjust not only the set of keywords used, but also the proximity between the keywords. While the original algorithm can handle several important issues, i.e. missing keywords or keywords replaced with semantic equivalents, we found that the addition of the proximity adjustments is crucial for our target speech documents. For example, the name of the speaker typically appears only at the beginning of a presentation, potentially far away from topic relevant keywords. Generally, the distance between keywords is larger than regular texts but varies from speaker to speaker due to speaker corrections, repetitions or specifications.

The retrieval algorithm consists of two main steps: (a) in the first step all non-stop question words are sorted in descending order of their priority, and (b) in the second step, the set of keywords used for retrieval and their proximity is dynamically adjusted until the number of retrieved passages is sufficient.

Keyword priorities are assigned solely based on their POS tags and lexical context. In descending of the assigned priority, all non-stop keywords are grouped as follows: (a) words that appear within quotes, (b) proper nouns, (c) numbers, (d) contiguous sequences of nouns and adjectives, (d) contiguous sequences of nouns, (e) other adjectives, (f) other nouns, (g) verbs, (h) adverbs, (i) the QFW, (j) other words. For example, for the question "What is a measure of similarity between two images?", the set of sorted keywords extracted by this algorithm is: {"two", "images", "similarity", "measure"}.

In the second step, the actual passage retrieval is implemented using the following algorithm:

(1) retrieve passages using keyword set **K** and proximity p.
(2) if number of passages $< MinPass$:
  if $p < MaxProx$
    increment p; goto step (1)
  else
    reset p; drop the least-significant keyword from **K**; goto step (1)
(3) else if number of passages $> MaxPass$:
  if $p > MinProx$
    decrement p; goto step (1)
  else
    reset p; add the next available keyword to **K**; goto step (1)
(4) return the current set of passages.

where the set of keywords **K** is initialized with all keywords with priority larger than the priority assigned to verbs, and the current proximity is initialized with some default value (20 words in our experiments). The algorithm is configured with four parameters: $MinPass$ and $MaxPass$ – lower and upper bounds for the acceptable number of passages (currently 1 and 50), $MinProx$ and $MaxProx$ – lower and upper bounds for keyword proximity (currently 20 and 60 words).

The actual information retrieval (IR) step of the algorithm (step (1)) is implemented using a Boolean IR system that fetches only passages that contain *all* keywords in **K** at a proximity $\leq p$.

2.3. Answer Extraction

The answer extraction component ranks candidate answers based on the properties of the context where they appear in the retrieved passages. We consider as candidate answers all entities of the same type as the answer type detected by the question processing component (we discuss candidate answer identification in detail in the next section). Candidate answers are ranked using a set of seven heuristics, inspired from [4]:

(H1) *Same word sequence* - computes the number of words that are recognized in the same order in the answer context;

(H2) *Punctuation flag* - 1 when the candidate answer is followed by a punctuation sign, 0 otherwise;

(H3) *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. The last two heuristics are a basic detection mechanism for appositive constructs, a common form to answer a question;

(H4) *Same sentence* - the number of question words in the same sentence as the candidate answer.

(H5) *Matched keywords* - the number of question words found in the answer context.

(H6) *Answer span* - the largest distance (in words) between two question keywords in the given context. The last three heuristics quantify the proximity and density of the question words in the answer context, which are two intuitive measures of answer quality.

(H7) *Distance from QFW* - measures the distance between the candidate answer and the QFW. This heuristic is enabled only for questions of type `NUMERIC`, where typically the QFW appears very close to the answer. For example, the answer to the question: "How many stories does the tower of Pisa have?" is "8 stories".

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

$$score = \mathbf{H1} + \mathbf{H2} + 2\mathbf{H3} + \mathbf{H4} + \mathbf{H5} - \frac{1}{4}\sqrt{\mathbf{H6}} - \mathbf{H7}. \quad (3)$$

where the heuristic weights were previously optimized for a set of 200 questions [4]. The above score drives the answer ranking that is reported to the user.

3. Identification of Candidate Answers

Our QA system recognizes a battery of 20 types of answer entities using several methodologies. For clarity, we group the answer types into 3 categories:

Open-domain entities: this group contains 5 types of *named* entities: – *person names*, *location names*, *organization names*, *other names* (e.g. creative works, events), and *languages* – and 10 types of *numeric* entities – *angle measures*, *dates*, *distance measures*, *money*, *numbers*, *percents*, *speed measures*, *surface measures*, *temperatures*, and *weights*. The named entities are recognized with a NERC based on Support Vector Machines that we previously tailored for speech transcriptions [6]. Currently, the NERC obtains an F1 measure of 90.31 on written text (on the CoNLL development corpus [7]) and of 75.50 on the Switchboard speech transcriptions [6]. The numeric entities are identified with an in-house system based on a regular expression grammar with 60 rules. Note that this grammar includes rules for the recognition of letter-spelled numbers and dates, which are typical in speech transcriptions, e.g. "nineteen eighty".

Domain-specific entities: these types were required because the transcriptions used in the experiments reported here (see next section) are a mixture of open-domain and domain-specific information from the speech/language/image processing domain. To handle domain-specific questions, we added three new answer types: *method/technique* - names of algorithms or methods (e.g. "HMM"), *project* - names of projects (e.g. "CHIL"), and *system* - names of actual software or hardware systems (e.g. "XDMLTool"). Additionally, we had to expand the existing *organization* type with names specific to the current domain. All these issue are addressed with the same methodology: (a) We harvested a large number of technical articles from the domain of interest[1]; (b) Human annotators extracted all the entities that fit into one of the above 4 categories; (c) Linguists developed regular expression rules to match the extracted entities, e.g. the rule: `(noun|adjective)+ (''system'' | ''prototype'' | ''toolkit'')`, with the constraint that at least one modifier must be upper case, extracts system names; and (d) Entities that could not be matched by rules (121 in our data) were added to a gazetteer.

[1]We harvested all the articles published by the LIMSI Spoken Language Processing group between 2000 and 2005.

Table 2: *Overall results.*

	TOP 5	TOP 1	MRR
exact	$35/50 = 70\%$	$31/50 = 62\%$	0.66
context	38/50 = 76%	$35/50 = 70\%$	0.73

Constraint-based entities: this set of answer types is triggered by AE only when additional answer constraints are detected in the question. We currently implemented two such constraints:

(C1) *Abbreviation expansion constraint* - this constraint is instantiated when the question demands the expansion of an abbreviation. For example, for the question "What does TREC stand for?", the constraint created limits the candidate answers to entities that can be abbreviated as "TREC". In our current implementation, any sequence of words whose prefixes (of length 1 or 2) equal the abbreviation, e.g. "Text Retrieval Conference", is considered as a candidate answer.

(C2) *QFW constraint* - this constraint is triggered when the question type does not match any known answer type and the QFW is a noun. For example, the type of the question "What is the Translanguage English Database also called?" is `ENTITY:equivalent_term`, which is not matched to any of the previous answer types. In this case, we search for candidate answers that are semantically related to the QFW noun. Currently, we consider as candidates all the noun phrases that share the same head word with the QFW, e.g. "Chairable English Database"[2] for the previous question.

The last constraint is crucial to increase the coverage of our QA system, because we are now not limited to the previous 19 answer types, but can (theoretically) answer any factoid questions that have a noun QFW. We consider this an exciting avenue for future research: we will explore more QFW-based constraints, e.g. semantically-related phrases (e.g. synonyms) and phonetically-similar candidates.

4. Experimental Results

The QA evaluation described here took place under the auspices of the European Union project "Computers in the Human Interaction Loop" (CHIL)[3]. The data was transcribed and annotated by the Evaluations and Language resources Distribution Agency (ELDA)[4] and consisted of: (a) a development set of 2 transcriptions and 10 questions, and (b) a test set of 8 transcriptions and 50 questions. All documents are transcriptions of 1-hour-long technical presentations or seminars from the domains of speech/language/image processing. All questions were factoid and had one of the types described in the previous section.

Table 2 summarizes our overall results. We report two types of scores: (a) TOP k, which scores a question as correct if the system provided a correct answer in the top k returned (in our evaluation we use $k = 5$ and $k = 1$); and (b) Mean Reciprocal Rank (MRR), which assigns to a question a score of $1/k$, where k is the position of the first correct answer, or 0 if no correct answer is returned. We show results for "exact" answers, where we evaluate only the text

[2]"Chairable" is actually a transcription error: should be "Terrible".

[3]`http://chil.server.de`

[4]`http://www.elda.org`

Table 3: *Results per NE category.* NIL *indicates questions that have no answer in the collection.*

	org	per	loc	time	measure	method	system	language	other	NIL
answered: TOP 5, exact	5/11	8/9	8/8	2/2	5/5	0/2	0/1	2/3	3/4	2/5

Table 4: *Number of system errors per component, using TOP 1 scoring of exact answers.*

	Open-domain	Domain-specific
Question processing	3	2
Passage retrieval	2	0
Candidate identification	2	5
Answer ranking	4	0
Other	0	1
Total	**11**	**8**

snippet returned by the system, and "context" answers, where we evaluate a context of 250 bytes surrounding the returned answer. Table 2 shows that our results are quite promising: we provide the exact answer on the first position for 62% of the questions, and provide an answer in the top 5 contexts extracted for 76% of the questions. In comparison, the two best factoid QA systems at the last TREC evaluation had a TOP 1 score of 71.3% and 66.6% for exact answers extracted from written text. Arguably, the two evaluations are not directly comparable: both the question sets and the document collections are different. Nevertheless, the fact that our system obtains approximately the same performance on speech transcriptions as other, more complex systems on written text is very encouraging.

Table 3 shows a distribution of the questions and of the correct answers per named entity (NE) category. Table 3 indicates that our effort in customizing our candidate identification module with domain-specific entity types was insufficient: we fail to answer any question where the answer type is method or system. From the open-domain entity types, we perform worse on organization names. From the NERC perspective this is explainable: due to their ambiguity, organization names are consistently harder to classify than person or location names [6].

Nevertheless, not all the errors on open-domain answer types come from the candidate answer identification module. We show a more detailed error analysis in Table 4, which lists the number of errors per system component. Furthermore, we split the errors in two sets: *open-domain* errors, which appeared in open-domain technology, and *domain-specific* errors, which were caused by the lack of domain-specific information. The analysis in Table 4 indicates that the largest number of errors appears indeed in candidate answer identification, where we failed to extract both domain-specific entities, e.g. "Eurospeech", "Icsi", or "LDA", and open-domain entities, e.g. "southern methodist university". QP has the next largest number of errors, but at least 2 questions are misclassified because their meaning changed in the current domain. For example, in an open-domain setup the question "Where does X work?" typically expects an answer of type location, but in our current domain the expected answer type is always organization. In the "other" errors category we assigned 1 question where we miss the name of a speaker that appears only at the beginning of the presentation and is linked to the context extracted only through first-person pronouns. We currently do not perform any form of coreference resolution, so we miss this answer. The biggest failure of the proposed open-domain algorithms is in answer ranking, where our ranking heuristics ranked incorrect answers higher than the correct ones for 4 questions. This can be blamed on the (intentional) lack of syntax analysis in AE.

Table 4 indirectly answers another important question: what is the theoretical performance upper limit that can be achieved with the proposed approach, which employs limited natural language analysis of the target texts? The ideal amount of syntactic and semantic analysis required in an QA system is still open for discussion, so the answer to the above question is definitely important. We calculate this performance upper limit in our setting by adding all the necessary domain knowledge to our system, and leaving unaddressed only the core open-domain issues (second column of Table 4). This hypothetical system achieves a TOP 1 score of 78% for exact answers. meaning that the proposed approach has the potential to answer roughly 4 out of 5 factoid questions with the correct exact answer on the first position.

5. Conclusions

This paper introduces a QA designed from scratch to handle speech transcriptions. The system's strength is achieved by mixing IR-oriented methodologies, i.e. keyword density and proximity, with a small number of robust NLP components: POS tagging and NE recognition. We evaluate the system on transcriptions of spontaneous speech from several 1-hour-long seminars and presentations and show that: (a) the system obtains excellent performance: we answer 62% of the questions with the correct exact answer on the first position, and (b) the theoretical performance upper limit is high, which justifies the claim that the combination of NLP and IR technologies increases overall robustness.

The proposed system can be adapted to automatic transcriptions with a relatively small research effort: both NLP components can be trained for automatic transcriptions and the keyword matching necessary for the answer ranking heuristics can be extended with approximate matches based on phonetical distance.

6. Acknowledgements

This work has been partially funded by the European project CHIL (IP-506808) and the Spanish Ministry of Science and Technology project TIN2004-0171-E. Mihai Surdeanu is a research fellow within the Ramón y Cajal program of this ministry. We also thank ELDA for annotating the data and organizing the QA evaluation.

7. References

[1] Harabagiu S., Moldovan D., and Picone J., "Open-Domain Voice-Activated Question Answering", Proceedings of COLING, 2002.

[2] Li X. and Roth D., "Learning Question Classifiers: The Role of Semantic Information", Natural Language Engineering, 1(1), 2004.

[3] Lin D., "Proximity-based Thesaurus", `http://www.cs.ualberta.ca/~lindek/downloads.htm`, 2006.

[4] Paşca M., "Open-Domain Question Answering from Large Text Collections", CSLI Publications, CSLI Studies in Computational Linguistics, 2003.

[5] NIST, "TREC 2005 Question Answering Main Task Results", `http://trec.nist.gov/pubs/trec14/appendices/qa.main.results.html`

[6] Surdeanu M., Turmo J., and Comelles E., "Named Entity Recognition from Spontaneous Open-Domain Speech", Proceedings of Interspeech, 2005.

[7] Tjong Kim Sang E. and De Meulder F., "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition", Proceedings of CoNLL, 2003.