Information Extraction via Heuristics for a Movie Showtime Query System

Martin Jansche

Department of Linguistics
The Ohio State University, Columbus, OH 43210–1298, USA
jansche.1@osu.edu

Abstract

Semantic interpretation for limited-domain spoken dialogue systems often amounts to extracting information from utterances. For a system that provides movie showtime information, queries are classified along four dimensions: question type, and movie titles, towns and theaters that were mentioned. Simple heuristics suffice for constructing highly accurate classifiers for the latter three attributes; classifiers for the question type attribute are induced from data using features tailored to spoken language phenomena. Since separate classifiers are used for the four attributes, which are not independent, certain errors can be detected and corrected, thus increasing robustness.

1. Introduction

With several new movies being released each week the options of what to see are constantly changing. This gives rise not only to a demand for showtime information services, but at the same time to a problem in creating such a service: an spoken dialogue system that responds to movie showtime queries has to be highly adaptable in light of the frequent changes to the database of current movie titles.

We discuss the implications of this and other requirements for the NLP component of a movie showtime query system [1]. The input of the NLP component consists of transcribed speech, i. e., a transcription of a user query produced by an automatic speech recognition (ASR) component. This input is transformed into a semantic representation, before it is handed on to a dialogue management component. The semantic interpretation step is very simple for the domain at hand and is conceptualized as an information extraction (IE) task. The difficulty of the IE task depends on properties of the input and the level of detail encodable in the semantic representation.

In our case the input is spoken language with its typical disfluencies, ranging from false starts to filled pauses; see the top part of Table 1 for examples, taken from the development corpus described below. The transcribed input is prone to contain errors introduced by the ASR component, such as misrecognized words or misplaced speech/noise boundaries; see the middle part of Table 1. A precise taxonomy of these phenomena is not required, since for purposes of information extraction it is not important to know what might have gone wrong in which place, as long as the user's intention can somehow be determined.

Disfluent input and propagated errors can make information extraction difficult. On the other hand, the meaning representation to be constructed is quite simple. A flat frame representation, i. e. a list of attributes and their values, is all that is required [1]. Figure 1 shows the semantic representation for the utterance *"When is The River Wild playing at Ogden Six in Naperville?"* No "deep" understanding is required and the IE task is expected to be fairly easy.

TASK	*when*
MOVIE	*The River Wild*
TOWN	*Naperville*
THEATER	*Ogden Six*

Figure 1: *Example of a semantic representation.*

The IE component must satisfy the following requirements:

Adaptability The database of current movie titles changes frequently. Routine changes should not require any adjustment of the IE component.

Portability Adapting the system to a new location with different town and theater names should be straightforward.

Accuracy Extract all and only relevant information (success can be measured directly).

Robustness Perform adequately in the presence of disfluent or ill-formed input, possibly due to errors introduced by other components (can be measured indirectly in terms of accuracy).

Automatic construction of the IE component for the movie showtime domain has been explored, using vector-based classification [2] with some success. In general, this kind of approach is desirable, since it is easily reusable and potentially very robust and accurate, given a training procedure that minimizes error, and given enough training data. And therein lies the crux of that approach – not only is it difficult to get sufficient training data, the nature of the domain at hand precludes this altogether for the task of identifying movie titles. It would be hopelessly impractical to gather new training data each time a new movie is released, which is when the system would have to be retrained.

In the rest of this paper we explore several options for constructing the components used for information extraction automatically. Since the number of question types is small and not likely to change, machine learning from data is feasible. The other classifiers are also constructed automatically, although they are not learned from data; instead they are built rationally, using simple heuristics. Development of those components is described in Section 2. In Section 3 we evaluate our approach and discuss properties of the domain it was applied to that contributed to its performance. Section 4 briefly summarizes our findings.

This work was carried out with support from Lucent Technologies as part of a summer internship under the direction of Bob Carpenter. I would like to thank him and Jennifer Chu-Carroll for helpful discussions and feedback. Many thanks also to Chris Brew for comments on earlier drafts of this paper. The usual disclaimers apply.

False starts	*Where is the – When is The Client playing in Glen Ellyn?*
Restarts	*What's playing – What's playing near Carol Stream?*
Repairs	*Where is Citizen Plane – Kane playing near Cineplex?*
Word repetitions	*What's playing at at Ogden Six?*
Filled pauses	*Um, where is Fox Valley Theater?*
Misrecognized words	*Is Shawshank Redemption playing where Naperville?*
Misplaced speech boundary	*At what time is Legends of the Fall playing at*
Don't know/care	*When is The Lion King play?*

Table 1: *Problematic aspects of the transcribed input.*

2. Development

2.1. The Illinois corpus

For development we used a corpus of 2929 utterances (1823 unique, expressing 696 unique queries) collected in 1994/5 in the Chicago area. Of the more than 22,000 words in the corpus, only 506 are unique. Each utterance is annotated with four attributes: TASK, MOVIE, TOWN, and THEATER. The latter three require no further explanation, except that a special value 'null' indicates that no movie title etc. was mentioned. The TASK attribute can take on the following values:

what if the utterance requires a response providing information about movies, e. g.:

What's playing at Ogden Six?

Please give me the listing for the movies at the Ogden Six movie theater, please.

when if the utterance requires a response providing information about showtimes, e. g.:

When is Nell playing at Ogden Six?

Please give me the times Little Rascals playing at Wheaton theater.

where if the utterance requires a response providing information about where a movie is playing, e. g.:

Where can I see The Client?

I'm looking for Nell.

location if the utterance requires a response providing information about the location of a theater, e. g.:

Where is the Tivoli?

Where are the Fine Arts Theaters playing?

yes/no if the utterance requires a response confirming whether a particular movie is playing at a particular theater, e. g.:

Is Nell playing at Wheaton?

Examples of annotated utterances are given in Table 2 ('n/a' indicates that a particular value is 'null' by necessity). Information extraction amounts to predicting the values of the four attributes, which we call the *class attributes,* for a given utterance.

2.2. Predicting the attribute TASK

The TASK attribute mixes information about the question type and the predicate of a sentence. All utterances labeled with a TASK value of *location* have the form of *where*-questions; in fact, they invariably start with the word 'where' (ignoring filled pauses). What distinguishes utterances with TASK *where* from those with TASK *location* is the main predicate, which may be ambiguous/identical, as in the following examples:

utterance	TASK
Where is Star Trek?	*where*
Where is Westridge Court?	*location*

In order to distinguish the two values reliably, information about the values of the MOVIE and THEATER attributes is needed. In other words, accurately predicting TASK without also predicting MOVIE and THEATER is not possible. The extent of this problem will become clear below. To deal with it we adjusted the labeling of the corpus as follows. A new attribute QUESTION was created which is identical to TASK, except that it lost the *location* value; whenever the TASK value is *location*, the QUESTION value is *where*.

We can predict QUESTION more easily than TASK, but in the end we need to recover the distinction that was lost by going from TASK to QUESTION. The IE module thus consists of the following five components:

1. A classifier for the QUESTION attribute
2. A classifier for the MOVIE attribute
3. A classifier for the TOWN attribute
4. A classifier for the THEATER attribute
5. A post-processing component that maps quadruples of predicted values to potentially different values for TASK, MOVIE, TOWN and THEATER

2.3. Predicting QUESTION

A simple idea for a non-constant baseline classifier is to use a dictionary of keywords for prediction. A small dictionary is shown in Figure 2.

keyword	QUESTION
what	*what*
what time(s)	*when*
when	*when*
where	*where*

Figure 2: *Keyword dictionary for* QUESTION *baseline classifier.*

The sentence below contains three matches from this dictionary, indicated by underlining:

what tie what time is the brady bunch playing at westridge court

It is clear from this and the following examples that prediction should be based on the rightmost longest match:

utterance	TASK	MOVIE	TOWN	THEATER
what's in wheaton	*what*	n/a	*Wheaton*	null
when is red playing	*when*	*Red*	null	null
where is iq playing	*where*	*IQ*	null	null
where's the tivoli	*location*	n/a	null	*Tivoli*
is nell playing at wheaton	*yes/no*	*Nell*	null	*Wheaton*

Table 2: *Examples of annotated utterances.*

where is the when is the client playing in glen ellyn
what romantic movies where are romantic movies playing

The rightmost longest match can be determined efficiently by a variant of a well-known algorithm [3] for matching finite sets of keywords.

Rightmost longest pattern matching based on the dictionary in Figure 2 (we predict *yes/no* by default, in case none of the keywords in the dictionary matched) achieves a baseline accuracy of 99.0% on the development corpus.

This can be improved by training a decision tree classifier [4] based on several attributes, including information about the rightmost question word, the rightmost compound of *any*, and the presence of other words (including *movie(s)*, *theater(s)*, *time(s)*, *looking*, and others). Repeated 10-fold stratified cross-validation estimates classification accuracy at 99.6%.

For comparison we also trained a decision tree based on binary attributes indicating the presence of question words and certain bigrams containing a question word (such as *what time*). This time the goal was to predict the TASK attribute directly, and without using the rightmost longest match strategy. Accuracy determined by the same cross-validation procedure was 98.6%, lower than the baseline for QUESTION. Using rightmost longest pattern matching, this could be improved to 99.2% accuracy, still below the best result for QUESTION.

2.4. Predicting MOVIE

Using rightmost longest pattern matching based on a dictionary containing full movie titles performs at 96.7% accuracy overall, 93.6% on utterances containing a movie title. There are two easy ways to improve on this baseline result.

First, if we use a dictionary containing titles with leading determiners (*a(n)* and *the*) removed, we achieve 98.9% accuracy overall, 97.9% on utterances containing a movie title.

Second, in addition we put into the dictionary unique unigrams, i. e., single words that occur in exactly one movie title and do not appear on a stop list of frequent words. Pattern matching based on this enlarged dictionary performs at 99.8% accuracy overall, 99.6% on utterances containing a movie title.

Note that all these classifiers are constructed without looking at training data, per the requirements discussed above.

2.5. Predicting TOWN

Rightmost longest match based on a dictionary containing town names performs at 82.5% accuracy overall, 99.6% on utterances containing a town name. This discrepancy is due to the fact that a large number of utterances labeled as not containing a town name are erroneously predicted to contain one, because a theater description, e. g., *the Wheaton theater*, has been mistaken for the town name it contains.

Such accidental matches can easily be prevented by using a different keyword dictionary: for each town name π put entries into the dictionary with keywords *in (the)* π, *near (the)* π, *around (the)* π, and π *area*. Pattern matching based on this dictionary achieves 99.7% accuracy overall, 99.2% on utterances containing a town name.

This successfully blocks accidental matches of theater names when looking for a town name, since if a theater is mentioned, it is typically not preceded by *in* or *near*, never by *around*, and theater names do not contain and are never followed by *area*.

2.6. Predicting THEATER

Rightmost longest match based on dictionary containing theater names performs at 65.7% accuracy overall, 84.7% on utterances containing a theater name. There are two reasons for this low baseline: overlap between theater and town names as before, and lack of unique names for theaters. For example, the Rice Lake Square Cinemas are referred to as the *Rice Lake Square theater*, the *Rice Lake theater*, the *Rice Square theaters*, the *Rice theater* etc.

To address these issues, construct a larger keyword dictionary as follows:

1. Put a theater name θ in the dictionary only if θ does not occur in a town name. If it does, put *at (the)* θ, *is/are* θ, θ *cinema(s)*, and θ *theater(s)* in the dictionary.
2. If a theater name ends in a word indicating a mall (*court*, *mall*, *square*, ...), in a number, or in the words *cinema(s)* or *theater(s)*, repeat the first step with all of those trailing words removed, unless the entire theater name consists only of such words; in that case, let λ be the first word of the theater name and add *the* λ to the dictionary.
3. Also add unique unigrams that do not appear in another theater name, a town name, or on the stop list.

It is possible to make manual adjustments, since this component will not be modified frequently. For example, due to a common misunderstanding we added a dictionary entry *Showcase 12* to predict *Showplace 12*. Matching based on this dictionary achieves 99.7% accuracy, both overall and on utterances containing a theater name.

Accidental misclassification of town names is avoided, since occurrences of town names are not preceded by *at* etc. or followed by *theater* etc. when they refer to towns.

2.7. Post-processing

Recovering the *where* vs. *location* distinction represented in the TASK attribute is easy: if the value for QUESTION is *where*, MOVIE is null, and THEATER is not null, predict a TASK value of *location*; otherwise predict the same TASK value as the QUESTION value that was determined (modulo error correction).

There are external constraints on semantic representations: for example, a *what* TASK precludes the presence of a movie

	Class attributes					error correc'n	
	Q6N/TASK	M3E	T2N	T5R	product	no	yes
previous	94.3	98.9	84.8	81.7	64.6		
baseline	99.0	96.7	82.5	65.7	51.9	50.1	50.4
best	99.6	99.8	99.7	99.7	98.8	99.0	99.2

Table 3: *Comparison of classification accuracies for Illinois data in percent.*

title, whereas a *where* or *when* TASK requires it. This makes it possible to detect certain kinds of errors. For example, the following utterance is predicted to have a TASK value of *what*:

when is hoop dreams playing at showplace twelve what

Because a movie title was recognized, the semantic representation violates the above constraint, so we know that an error has occurred. Since both a movie and a theater were mentioned, the TASK value can be corrected to *when*.

2.8. Putting everything together

Table 3 summarizes our baseline and best results from the preceding sections, and compares them with previous results (Chu-Carroll, p. c., 1999) for vector-based topic identification [2] evaluated on a test set of 263 utterances. Care must be taken in interpreting these results, since in the QUESTION/TASK column Chu-Carroll's previous result is the accuracy for predicting TASK directly, whereas our figures are for predicting QUESTION; nevertheless, it is surprising that Chu-Carroll's results for TASK are below our baseline attempt (98.6% accuracy) at predicting TASK, for which we had been careful to use only a subset of the features Chu-Carroll used, namely salient unigrams and bigrams. The figures in column 'product' are the products of the numbers in the preceding four columns, which is the overall accuracy we would expect if errors were independent. There are two columns for the actually observed overall accuracies, one without error correction (other than recovering TASK) and one with.

3. Evaluation

The evaluation was carried out using a corpus of utterances collected in 1999 in New Jersey, originally for evaluating dialogue management strategies [5]. The utterances had been transcribed automatically (70% of all utterances contain one or more transcription errors) and by a human. We took queries to the variable-initiative system that do not depend on dialogue state (mostly from the user's first turn) and that were not truncated by the acoustic model, for a total of 133 utterances. These were partitioned into a held-out set (33 utterances) and a test set (100 utterances).

The held-out data were used as additional training data and to check for any systematic errors of the classifiers, which did not become apparent. We decided to run the classifiers without any modifications on the test set, except that the QUESTION classifier had now been trained on both the development corpus and the held-out set (the latter with an instance weight of 100). Table 4 lists the absolute (out of 100) and thus relative (in percent) accuracy on the test data. An inspection of the classification errors revealed a systematic problem with theater names, the most noticeable difference between New Jersey and Illinois being the prevalence of schematic names in New Jersey, e. g., *Loews East Hanover*, referred to by users as *Loews theater in East Hanover*, a construction absent from the Illinois data. This turned out to be the source for almost all systematic mistakes.

	Class attributes				Total
	Q6N	M3E	T2N	T5R	
human xscrpn	99	100	100	93	92
automatic xscrpn	90	91	84	89	71

Table 4: *Classification accuracy on the New Jersey test data.*

There are several reasons for why our approach works as well as it does. First of all, the good performance of some of the very simple heuristics indicates that the task at hand is quite easy. This is not surprising, since the domain is rather small, both in terms of total vocabulary, but also, and more importantly, in the number of predicates that are implicitly extracted. Only two kinds of predicates are understood, corresponding to two kinds of database queries: finding movie showtime information, and theater locations. The semantic sorts of the noun phrases determine the predicate (*playing* vs. *located*) completely, hence no information about verbs was extracted or used for prediction (when we tried to use information about verbs, overall accuracy went down due to the presence of yet another potential source of errors).

4. Conclusions

In conclusion, we want to emphasize the following points. First, if neither writing rules by hand nor automatic rule induction is an option, automatic construction of rules based on heuristics can be a viable alternative, producing classifiers that are easy to reason about. Second, information can be extracted most accurately and reliably when it can be predicted from local syntactic features. Third, the redundancy in flat semantic representations can be used to detect and correct classification errors.

5. References

[1] Jennifer Chu-Carroll, "MIMIC: An adaptive mixed initiative spoken dialogue system for information queries," in *Proc. ANLP 6*, 2000, pp. 97–104.

[2] Jennifer Chu-Carroll and Bob Carpenter, "Vector-based natural language call routing," *Computational Linguistics*, vol. 25, no. 3, pp. 361–388, 1999.

[3] Alfred V. Aho and Margaret J. Corasick, "Efficient string matching: An aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, 1975.

[4] J. Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, CA, 1993.

[5] Jennifer Chu-Carroll and Jill S. Nickerson, "Evaluating automatic dialogue strategy adaptation for a spoken dialogue system," in *Proc. NAACL 1*, 2000, pp. 202–209.