Procesamiento del Lenguaje Natural, Revista nº 44, marzo de 2010, pp 115-122 recibido 20-01-10 revisado 19-02-10 aceptado 05-03-10

Multilingual Access to Online Help Systems and Databases

Acceso Multilingüe a Sistemas de Ayuda y Bases de datos En Línea

Alejandro Revuelta-Martínez, Luis Rodríguez and Ismael García-Varea
Departamento de Sistemas Informáticos
Universidad de Castilla-La Mancha
02071 Albacete, Spain
{Alejandro.Revuelta,Luis.RRuiz,Ismael.Garcia}@uclm.es

Resumen: En este artículo presentamos el sistema AMSABEL, un sistema de acceso a información basado en lenguaje natural. AMSABEL proporciona acceso al contenido de una base de datos relacional, a partir de consultas realizadas en lenguaje natural, tanto en castellano como en inglés. De esta manera se permite un acceso directo a la base de datos por parte de usuarios no expertos. El sistema utiliza aprendizaje automático para traducir la consulta original al lenguaje de consulta estructurado siendo, por lo tanto, independiente del contenido de la base de datos siempre que se disponga de datos de entrenamiento.
Palabras clave: Acceso a información multilingüe, procesamiento del lenguaje natural, SQL, traducción automática estadística

Abstract: In this paper we present the AMSABEL system, a natural language based information retrieval system. AMSABEL provides access to data stored in a relational database using unrestricted natural language. Queries can be expressed in either, English or Spanish, thereby providing direct access to the database for non-expert users. The system relies on machine learning techniques to translate the original query from natural language into a structured query language and, therefore, does not depend on the contents of the database, as long as training data is available.
Keywords: Multilingual information access, natural language processing, SQL, statistical machine translation

1 Introduction

Using natural languages to access to databases has important advantages. Most users are not able to directly use formal query languages and its learning can be difficult and time-consuming. The use of other kind of interfaces can solve some of these problems but, typically, at the expense of limiting flexibility. On the contrary, the user could employ the interface in a very natural way if he is able to directly request data in his native language.

The idea of querying a database using natural language goes back to the late sixties and early seventies (Androutsopoulos, Ritchie, and Thanisch, 1995). These systems, also called *Natural Language Interfaces to Databases*, were built to be used with a particular database and, therefore, were not easily adaptable to other databases. Besides, those early systems only allowed to use a subset of natural language.

Database access using natural language was improved in the following years by increasing the accuracy of the results as well as reducing the complexity of the adaptation of the systems to a new database. In the early nineties the Air Travel Information Service (ATIS) task (Price, 1990) was developed, consisting of a set of queries to an air travel database, expressed in both, spoken and written English. Human-generated results of the queries were also included. The ATIS task was created to be used in speech recognition evaluation but it has also been widely used in natural language database access systems. Some of the systems developed for this task were AT&T's CHRONUS (Pieraccini et al., 1992), BBN, MIT's TINA (Seneff, 1992), CMU's Phoenix (Ward and Issar, 1994) or SRI's Gemini (Dowding et al., 1993).

ISSN 1135-5948

Modern approaches are aimed at reducing the cost to access to new databases, this is mainly achieved by reducing the configuration effort required by non-expert users (Minock, 2010) or by using machine learning, as in He and Young (2003), Griol et al. (2006) or PRECISE (Popescu, Etzioni, and Kautz, 2003) systems.

In this paper we present a system which allows *multilingual access to online help systems and databases* (AMSABEL)[1]. Our proposal is based on statistical methods but, unlike any of the previous approaches, AMSABEL uses techniques applied in the field of Statistical Machine Translation.

Modern SMT approaches appeared in the early nineties (Brown et al., 1993) and achieved high translation precision. SMT systems are built by means of machine learning, this way avoiding the need of adding expert knowledge, which is a common requirement of previous machine translation approaches.

We propose the use of SMT to translate from English or Spanish into Structured Query Language (SQL), instead of another natural language. This entails new and interesting problems since the semantics of a formal language is, in general, more affected by translation errors. On the other hand, by using SMT methods our system inherits their portability and does not depend on the underlying database.

We introduce, as well, a simple dialogue process for this kind of systems and a result-oriented evaluation methodology. Finally, we have implemented and tested a prototype for the AMSABEL system.

The system is expected to be greatly useful. For example, it could be used by governments to provide access to official databases. The AMSABEL system could be used with little (or no) training by most citizens and its multilingual architecture will also help foreigners.

This paper is organized as follows. Section 2 describes the task, the data and the corpus used for building and evaluating the system. Section 3 presents the AMSABEL system in detail. Section 4 describes the evaluation procedure and, finally, Section 5 presents conclusions and future work.

[1]http://amsabel.albacete.org

2 Task

The implemented system is used to access to information about train routes among several cities. The structure of that information is extracted from the Spanish railway company[2] and contains, for each route, the following data: Train identification number, train type/service, departure and arrival cities, days of the week when the route is active, starting and ending route dates, departing and arriving times, ticket prices and ticket classes.

A relational database has been designed including all the information previously described. The actual data has been automatically generated by choosing random values. Some constraints have been considered in the data generation to keep the semantics of the database (e.g., a route starting date should take place before its ending date). However, some other constraints have been ignored, since they are not important from the system point of view (e.g., the real distance between cities has not been considered when calculating trip lengths). The generated database stores information on 1094 routes covered by 50 different trains.

The generation of the parallel, training corpora has been performed automatically using context-free grammars by means of syntax directed translation schemes (Amengual et al., 2000) which are typically used to develop language processors or compilers.

This process has been performed starting from the grammar initial symbol, and using random rules until one pair of sentences is generated. As a result, two parallel sentences (in two different languages) will be generated, and this will be iterated until reaching a reasonable number of sentences.

Three different grammars have been designed for the system, having each of them 90 rules. One grammar generates Spanish-SQL sentence pairs, other generates English-SQL sentences and the last one generates Spanish-English sentences. The latter will be used to translate an English query into Spanish when a user requests the help of a human operator.

Table 1 shows an example of three parallel sentences in English, Spanish and SQL. Table 2 describes the training, validation and test sets features. Test values for the pair English-Spanish are not provided due to that

[2]http://www.renfe.es/horarios

Eng.	*How much is a train from Barcelona to Madrid?*
Spa.	*¿Cuánto cuesta un tren desde Barcelona a Madrid?*
SQL	*SELECT DISTINCT Precio_estacion, FROM Tren JOIN Billetes ON (Billetes.Tren=Tren.Id_tren) WHERE Origen='barcelona' AND Destino= 'madrid'*
This SQL sentence returns a column showing the prices of all the tickets (when purchased at the station) of the trains that go from Barcelona to Madrid.	

Table 1: Example of three parallel English-Spanish-SQL sentences.

		Spa–SQL		Eng–SQL		Eng–Spa	
Train	# sentences	10000		10000		10000	
	vocabulary size	4539	4032	4635	4132	4605	4161
	running words	138672	224265	139558	224947	139417	151743
Validation	# sentences	1000		1000		1000	
	running words	14873	20918	14494	21438	14659	15272
Test	# sentences	748		752		–	
	running words	9413	14558	9763	15039	–	–

Table 2: Training, validation, and test task statistics. Test values for the pair English-Spanish are not provided due to that experiment is not reported.

experiment is not reported.

3 The AMSABEL system

A system has been implemented to interact with the database described in Section 2. This system allows a user to query information in either English or Spanish. In addition, it keeps constant communication with the user, requesting, if necessary, further information.

3.1 System overview

The whole system is designed following a client-server architecture with two kinds of clients and two kinds of servers (see Figure 1).

- *The user client* gets the input from the user sending it to the translation server. In addition, it receives the SQL sentence and access to a specific database management system. Finally, it provides a platform for user-system dialogue.
- *The translation server* translates natural language queries into SQL sentences. It takes part in the user-system dialogue by sending feedback to the user and attending his requests and, if the user requests a human operator, this server forwards the query to the operator server.
- *The operator server* receives queries from users through the translation server sending them to a free operator and, finally, sending back the operator's answer. This server can translate the input query when it is not in the operators' native language.
- *The operator client* is used to translate user queries into SQL sentences. The operator can also send back to the user a comment that will be displayed as part of the user-system dialogue.

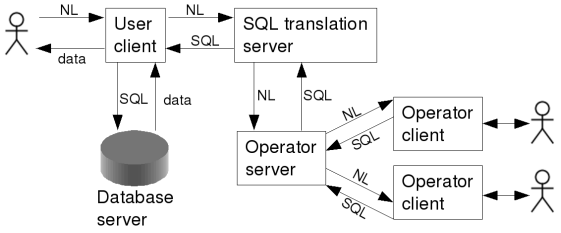

Figure 1: The AMSABEL system. The user queries in natural language (NL) are translated into SQL, returned and used to access to the database.

The access to the database using natural language is divided in four different processes: preprocessing, translation, postprocessing and the dialogue system.

3.2 Preprocessing

In this step, the system input is prepared to be sent to the SMT engine by rewriting some parts of the sentence in order to make it readable by the translation engine or to reduce translation complexity.

1. *Character-level processing*: Characters are written in lowercase. Consecutive spaces are collapsed into a single one and special characters (e.g., accents or the 'ñ' letter) are formatted properly.

2. *Language identification*: To this end, the words in the query are compared to each considered language vocabulary to determine which one presents less out-of-vocabulary (OOV) words. However, if the query has many OOV words in all of them an error message is displayed (this filtering prevents the users from querying information outside the database domain or using non-supported languages).

3. *Spell checking*: OOV words are replaced by the closest word in the task vocabulary according to Levenshtein distance if a specific threshold is not exceeded.

4. *Word-level processing*: Regular expressions are used to identify dates, numbers, days of the week and similar expressions which are converted to the format expected by the translation engine.

5. *Word categorization*: A total of 8 categories are used to replace dates, departure cities, arrival cities, hours, days of the week, ticket classes and numeric values in the sentence.

3.3 Translation

This step relies on SMT to translate either, the processed input query into an SQL sentence, or the input query to the operator's mother language. From a formal point of view, SMT can be stated as the problem of finding a target sentence $\hat{\mathbf{e}}$ which maximizes the probability $Pr(\mathbf{e}|\mathbf{f})$ for a given source sentence $\mathbf{f}$:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}}\, Pr(\mathbf{e}|\mathbf{f}) \qquad (1)$$

Using Bayes' theorem, Eq.-(1) can be restated as:

$$\hat{\mathbf{e}} = \underset{\mathbf{e}}{\operatorname{argmax}}\, Pr(\mathbf{e}) \cdot Pr(\mathbf{f}|\mathbf{e}) \qquad (2)$$

In Eq.-(2), $Pr(\mathbf{e})$ is known as the *language model* probability and represents the probability of being $\mathbf{e}$ a sentence in the target language. Most systems use smoothed n-gram models (Jelinek, 1998) to estimate this probability:

$$\begin{aligned} Pr(\mathbf{e}) &= Pr(e_1 \cdot e_2 \cdots e_n) \\ &\approx \prod_{i=1}^{n} p(e_i|e_{i-n+1}, \cdots, e_{i-1}) \end{aligned} \qquad (3)$$

This factorization significantly reduces the parameters of the model, which are trained from data using maximum likelihood estimation.

The AMSABEL system uses a trigram language model trained using the SRILM toolkit (Stolcke, 2002) on the SQL training corpus described in Section 2.

The second probability, $Pr(\mathbf{f}|\mathbf{e})$, is the *translation model* probability which represents the probability that $\mathbf{f}$ is a translation of $\mathbf{e}$. In this paper this probability is obtained through phrase-based models (Koehn, Och, and Marcu, 2003).

Phrase-based models split the input sentence $\mathbf{f}$ into segments (phrases) of consecutive words. Then, each source phrase is translated into a target language phrase and, finally, the target phrases are reordered to obtain the translation $\mathbf{e}$. This model is formalized in Eq.-(4).

$$\begin{aligned} Pr(\mathbf{f}|\mathbf{e}) &= Pr(\bar{f}_1^I|\bar{e}_1^I) \\ &= \prod_{i=1}^{I} \phi(\bar{f}_i|\bar{e}_i) d(a_i - b_{i-1}) \end{aligned} \qquad (4)$$

Where $\bar{f}_i$ is the i-th phrase in $\mathbf{f}$, $\bar{e}_i$ is the i-th phrase in $\mathbf{e}$, $\phi(\bar{f}_i|\bar{e}_i)$ is the probability of being $\bar{f}_i$ a translation of $\bar{e}_i$ and $d(a_i - b_{i-1})$ is the distortion model used for reordering target phrases. The order of a target phrase $\bar{f}_i$ depends on a probability distribution calculated using its start position (a_i) and the end position of $\bar{f}_{i-1}$ (b_{i-1}).

Phrase-based models used by the AMSABEL system are obtained from word alignments trained using the GIZA++ toolkit (Och and Ney, 2003) on the bilingual training corpora (English-SQL and Spanish-SQL) described in Section 2.

Once the model has been trained, a search algorithm is employed to find the best translation for the source sentence. The

AMSABEL system uses Moses beam-search decoder (Koehn et al., 2007) to translate the input. This technique can also efficiently compute the n-best translations, allowing the user to request different translations of a single input sentence (see Section 3.5).

3.4 Postprocessing

The postprocessing step firstly restores the changes performed in the preprocessing step. Initially, special characters are expressed in SQL format. Secondly, categories are replaced by their original values.

After those changes, the sentence should be in SQL format (see Table 1) but some errors can arise in the process. Any error in the sentence will cause that the system will not be able to return any result. Therefore, the SQL sentence is parsed to remove any syntactical error while trying to keep the semantics of the original query. To this end, the following steps are performed.

- A syntactic parser checks that the sentence is a correct SELECT SQL query.
- The parser also checks that every table and field name exists in the database.
- If any error is found, the parser tries to correct it by synchronizing with the next field or table, or by inserting the expected keyword.
- If a fatal error between SELECT and FROM keywords occurs, this fragment of the query is dropped and replaced by all the relevant fields of the tables that appear in the rest of the sentence.
- If any error between FROM and WHERE keywords occurs, this part is dropped and completely regenerated from all the tables needed to form a correct SQL query.

At the end of this step the system, hopefully, has a complete and syntactical error-free SQL sentence.

3.5 Dialogue system

The user-system interaction is not limited to typing a query and receiving an answer. Instead, a simple dialogue system assists the user in the query process.

Firstly, the system reads a query from the user which is then processed and translated into SQL, as explained in Sections 3.2, 3.3 and 3.4. If no errors occurred in the user query it is translated into SQL and used to access to the database. Otherwise, an error message is generated and different solutions are suggested.

Once a query has been correctly translated, it is used to retrieve the data, and the interface shows a description in natural language of the information retrieved. This way, the user can check the answer.

In case the returned information is not correct, the user is allowed to choose one of these three new operations:

- *Request more information.* In case of some fields are missing in the result, this option adds additional fields to the system outcome. This option can be selected twice: the first time it adds fields directly connected with the user query and the second time all the fields of the database are added (this information could still be useful due to the domain of the task is very specific). This operation does not modify the constraints of the original query.
- *Request another answer.* Returns an alternative translation for the original query. Each time this option is selected the next translation in the n-best list is returned.
- *Request an operator.* Users may need some information that the system is not able to return. In addition, fully automatic systems are not perfect and can introduce errors. Therefore, the system allows the user to send his query to a human operator that will type a correct SQL sentence and, if necessary, can also return some feedback to the user.

The user can select the previous options in any order.

3.5.1 Feedback

Users can easily feel frustrated if the system does not work as they expect, even if they are not using it properly. Another issue is that, although natural language is the most natural way for human interaction, users can also feel lost due to the lack of restrictions in the interface. To prevent these and other similar problems, the user can receive feedback information to keep him informed on the query status and to get explanations about what caused the errors.

The following information is returned as feedback:

- If the number of OOV words in the query is too high, the system returns a message explaining the task and the types of expected queries.
- After each query, a natural language sentence is returned describing the generated SQL sentence. This sentence describes the fields and tables retrieved, as well as the restrictions considered.
- The system informs the user if there is no more data available for a query.
- The user is also informed if no more translation alternatives are available.
- A message is shown if the user requests an operator but no one is available at that moment.
- The system can also return the actual SQL sentence generated. This option is intended for advanced users and can be either, activated or not.
- If the system detects an invalid formatted value (e.g., dates), it asks the user to express it in a valid format.

Feedback is part of the user-system dialogue. Messages are returned and showed only when necessary, in the appropriate context and they do not interrupt the interaction with the system. Instead, they try to assist the user and facilitate the use of the dialogue options to find a correct answer, trying to reduce the user's frustration if the system was not able to find a correct answer.

4 Experimental framework

4.1 Evaluation

The evaluation has been performed using the test data sets described in Table 2. Those sets have been generated following the same process used to build the training corpora (see Section 2) and considering the same vocabulary but filtering out sentences that returned an empty data set.

The evaluation performed in this paper tries to assess the usefulness of the system in the whole process of information retrieval. Hence, common machine translation metrics (e.g., WER or BLEU) can not be really informative in the context of information retrieval, since they compare the differences between two sentences but can not check the correctness of the result of the query. For example, just by removing a keyword, a correct SQL sentence can be turned into an incorrect one and, on the contrary, the order of the constraints can be completely modified without changing the result of the query.

From all this, comparison is based on the results of the SQL queries. The answers that show the same information as the corresponding reference result are considered useful, even if they include additional fields. Too much information at once can overwhelm the user but could also be really helpful if it includes the expected results.

The exact procedure for the experiments carried out can be described as follows. For each sentence in the test set, a translation into SQL is performed and the database is accessed. Finally, the information returned by the database is used to classify the query in the following categories:

Q1 *Exact information*: The query shows the user exactly the same information as the reference.

Q2 *More fields*: The query returns the same rows as the reference and, at least, the same fields.

Q3 *More rows*: The information returned by the query includes all the information returned by the reference but more rows are added.

Q4 *Incorrect*: The query does not include all the information of the reference.

Query types Q1 and Q2 can be considered as successful outcomes, since they return what the user requested (Q2 just adds more details).

Q3 (and, to a lesser extent, Q2) queries can be useful for navigational purposes because they return more information than requested. This fact can be used to refine a query, expand the data returned by a certain query, or just give an idea of the database contents in a first approach to the system.

Finally, Q4 queries do not help the user and, what is more, they can even mislead him by providing wrong answers.

4.2 Experimental results

Table 3 shows the results of English and Spanish test queries when the user selects

	Times	Q1	Q2	Q3	Q4
Eng.	0	66.6	3.7	5.1	24.6
	1	66.6	12.4	8.9	12.1
	2	66.6	14.0	16.5	2.9
Spa.	0	81.4	0.3	1.3	17.0
	1	81.4	4.2	5.4	9.1
	2	81.4	4.3	10.2	4.1

Table 3: Percentages of query types when the user selects, for each query, the option "more info" 0, 1 or 2 times.

the "request more information" option (explained in Section 3.5) zero, one or two times. These results show that this option does not modify the number of Q1 queries although it substantially increases Q2 and Q3 queries. In other words, this option increases the results useful for both, navigation and information.

If this option is not used, Q2 and Q3 queries are a small percentage of the test queries. However, selecting this option once causes a great increase in Q2 queries and selecting it twice yields the greater increase in Q3 ones.

At first, Spanish results are more accurate than English ones but the use of this option has a greater impact on English outcome. Using this option twice reduces the percentage of Q4 (useless) queries to less than 5% for Spanish and less than 3% in the case of English.

Table 4 shows the results of English and Spanish test queries when the user selects "request another answer" option (explained in Section 3.5) from zero to four times.

This option causes an important increment in the number of Q1 (perfect) queries. The first use yields the greater improvement in the results for both languages, increasing Q1 queries approximately by 10%. On the other hand, Q2 and Q3 queries are not significantly modified. This option has a greater impact on Spanish sentences reducing its Q4 queries to less than 3%.

Table 5 contains the results obtained when "request another answer" and "more info" options are considered simultaneously. These results show that SMT combined with an SQL checker can obtain useful answers for most of the queries. The use of the proposed dialogue system can return the requested information (Q1 and Q2 results) in more than 92% of the queries, achieving more than 99% when also looking for navigational information (i.e., considering, as well, Q3 results).

5 Concluding remarks

In this paper, the problem of accessing to databases using natural language has been presented, and a train timetable database access task has been described.

The task has been used to develop a system that allows the retrieval of information using English or Spanish natural languages (SMT techniques have been used to generate SQL queries). The process can be enhanced by means of a dialogue system that assists the user and simplifies the information retrieval.

The system has been evaluated to assess its correctness and usability. The results obtained show that SMT can be considered when translating from natural languages into the formal SQL. By adding an SQL parser and a simple dialogue system AMSABEL can achieve correct and specific answers, in more than 92% of the test queries and useful responses in more than 99% of the test queries.

As future work, it could be interesting to study the implementation and performance costs of the AMSABEL system in a different and larger database to assess the database-

	Times	Q1	Q2	Q3	Q4
Eng.	0	66.6	3.7	5.1	24.6
	1	75.1	3.9	4.4	16.6
	2	78.3	3.5	3.3	14.9
	3	81.0	2.8	2.7	13.6
	4	81.8	3.1	2.7	12.5
Spa.	0	81.4	0.3	1.3	17.0
	1	92.0	0.3	0.8	7.0
	2	94.5	0.1	0.4	5.0
	3	95.9	0.1	0.3	3.7
	4	96.9	0.3	0.3	2.5

Table 4: Percentages of query types when the user selects, for each query, the option "request another answer" from 0 to 4 times.

	Q1	Q2	Q3	Q4
Eng.	81.8	10.4	7.1	0.8
Spa.	96.9	1.5	0.8	0.8

Table 5: Percentages of query types when the user selects, for each query, the option "request another answer" 4 times and the option "more info" two times.

independence. In addition, an evaluation of the system using real users could be carried out to compare to the results of the automatic evaluation techniques used in this paper. Finally, the interface could be even more natural by using speech recognition.

Acknowledgements

Work supported by the European Social Fund and the Spanish Consejería de Educación y Ciencia de la Junta de Comunidades de Castilla-La Mancha under PBI08-0210-7127 research project. Additionally, the authors wish to thank the anonymous reviewers for their valuable comments.

References

Amengual, J.C., A. Castaño, A. Castellanos, V.M. Jiménez, D. Llorens, A. Marzal, F. Prat, J.M. Vilar, J.M. Benedí, F. Casacuberta, M. Pastor, and E. Vidal. 2000. The EuTrans-I spoken language translation system. *Machine Translation*, 15(1–2):75–103.

Androutsopoulos, I., G.D. Ritchie, and P. Thanisch. 1995. Natural language interfaces to databases — an introduction. *Natural Language Engineering*, 1(1):29–81.

Brown, P.F., S.A. Della Pietra, V.J. Della Pietra, and R.L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311.

Dowding, J., J.M. Gawron, D. Appelt, J. Bear, L. Cherny, R. Moore, and D. Moran. 1993. Gemini: A natural language system for spoken-language understanding. In *Proc. of the Workshop on Human Language Technology*, pages 43–48.

Griol, D., F. Torres, L. Hurtado, S. Grau, F. García, E. Sanchis, and E. Segarra. 2006. A dialog system for the DIHANA project. In *Proc. of the Int. Conference on Speech and Computer*, pages 131–136.

He, Y. and S. Young. 2003. A data-driven spoken language understanding system. In *Workshop on Automatic Speech Recognition and Understanding*, pages 583–588.

Jelinek, F. 1998. *Statistical Methods for Speech Recognition*. The MIT Press, Cambridge, MA, USA.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 177–180.

Koehn, P., F.J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 48–54.

Minock, M. 2010. C-Phrase: A system for building robust natural language interfaces to databases. *Data & Knowledge Engineering*, 69(3):290–302.

Och, F.J. and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Pieraccini, R., E. Tzoukermann, Z. Gorelov, J.L. Gauvain, E. Levin, C.H. Lee, and J.G. Wilpon. 1992. A speech understanding system based on statistical representation of semantics. In *Proc. of the 1992 Int. Conference on Acoustics, Speech and Signal Processing*, pages 193–196.

Popescu, A.M., O. Etzioni, and H. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proc. of the 8th Int. Conference on Intelligent User Interfaces*, pages 149–157.

Price, P.J. 1990. Evaluation of spoken language systems: The ATIS domain. In *Proc. of the Workshop on Speech and Natural Language*, pages 91–95.

Seneff, S. 1992. Tina: A natural language system for spoken language applications. *Computational Linguistics*, 18(1):61–86.

Stolcke, A. 2002. SRILM – an extensible language modeling toolkit. In *Proc. of the 2002 Int. Conference on Spoken Language Processing*, pages 901–904.

Ward, W. and S. Issar. 1994. Recent improvements in the CMU spoken language understanding system. In *Proc. of the workshop on Human Language Technology*, pages 213–216.