SpeechForms: From Web to Speech and Back

Luciano Barbosa, Diamantino Caseiro, Giuseppe Di Fabbrizio, Amanda Stent

AT&T Labs Research
180 Park Avenue, Florham Park, NJ 07932 - USA
{lbarbosa,dcaseiro,pino,stent}@research.att.com

Abstract

This paper describes *SpeechForms*, a system that uses novel techniques to automatically identify form element semantics and form element content, and to semi-automatically generate language models that allow users to fill out each web form element by voice. Preliminary experimental results show that simple per-element language models are faster and may be more accurate than statistical n-gram language models trained on large amounts of web text data.

Index Terms: language modeling, form understanding, information retrieval

1. Introduction

The main method to manually input data into web applications is via typing text into HTML-based forms. Web forms range from simple one-field search forms to multiple-field, or even multiple-form forms. While typed interaction is natural with a regular-sized keyboard and desktop display, it becomes a real challenge with a small-screen mobile device. Overcoming these limitations is particularly important to improve accessibility for people with visual or print disabilities who currently have limited ability to use web applications on mobile devices.

Speech input provides an attractive opportunity to overcome these limitations and dramatically improve the user experience on mobile phones [1]. Unfortunately, unconstrained input is not a realistic capability of current speech recognition technology. Furthermore, the lack of uptake of technologies such as SALT [2] and X+V [3] has shown that users cannot rely on developers speech-enabling web forms. However, web forms may convey enough context information for an automated system to infer the expected input in each form element [4, 5]. For example, when searching `amazon.com`, the generic search field is filtered by a specific search category from a drop-down selection box such as *Books, Appliances, Automotive*, etc. By interpreting the selected category, a generic ASR language model (LM) can be pruned or re-weighted to the actual words allowed in the specified category. Furthermore, there are several form elements that are commonly used and fairly similar across web sites, like, "date", "time", or "street address". In this cases, LM perplexity can be dramatically reduced by reusing existing and well optimized models from spoken dialogue systems [6].

This paper describes *SpeechForms*, a system that uses novel techniques to automatically identify form element **semantics** and form element **content**, and to semi-automatically generate (on the server side) language models that allow users to fill out each web form element by voice.

The reminder of the paper is organized as follows. In Section 2 we discuss previous work in this area. In Section 3 we present the architecture of SpeechForms. In Section 4 we describe our preliminary experiments with this system. We conclude in Section 5.

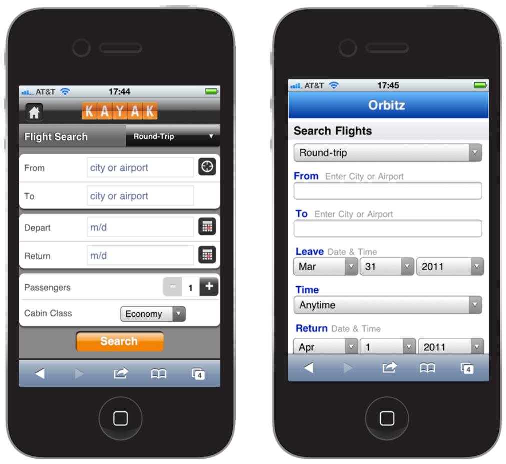

Figure 1: Two examples of flight search applications for mobile

2. Related work

Previous work on web-based language model construction has mostly focused on automatic acquisition of training data for statistical n-gram language models from web corpora (e.g., [7, 8, 9, 10, 11]). This research is relevant to our task; however, in order to acquire language model training data a seed corpus or vocabulary is typically necessary. In our work, we obtain that seed vocabulary from element values in web forms; we also use the seed vocabulary to constrain the contents of element-specific language models.

Only a few researchers have looked at automatic acquisition and tuning of domain-specific, concept-specific language models. In one recent paper, Gruenstein *et al.* [12] describe a method for mining web-based data sources to obtain language models for a web-based interactive restaurant guide. Their method uses a crawler to mine web-based data sources, rules to extract data (e.g., restaurant names, locations, and types), and an algorithm for building a hierarchical language model, including several context-dependent submodels, from the extracted data; in other words, considerable domain knowledge is required. In another recent paper, Ballinger *et al.* [13] present a method for just-in-time form element-specific language model interpolation for mobile interaction; this method presumes that the component language models already exist. In our work, we focus on automatically extracting element-specific concept vocabularies for building language models.

Our ultimate goal is to automatically generate spoken dialog interfaces from input web forms. HTML can be thought of as an abstract representation of a dialog flow, making our work similar to work on automatic dialog system generation [14, 15, 16, 2, 17].

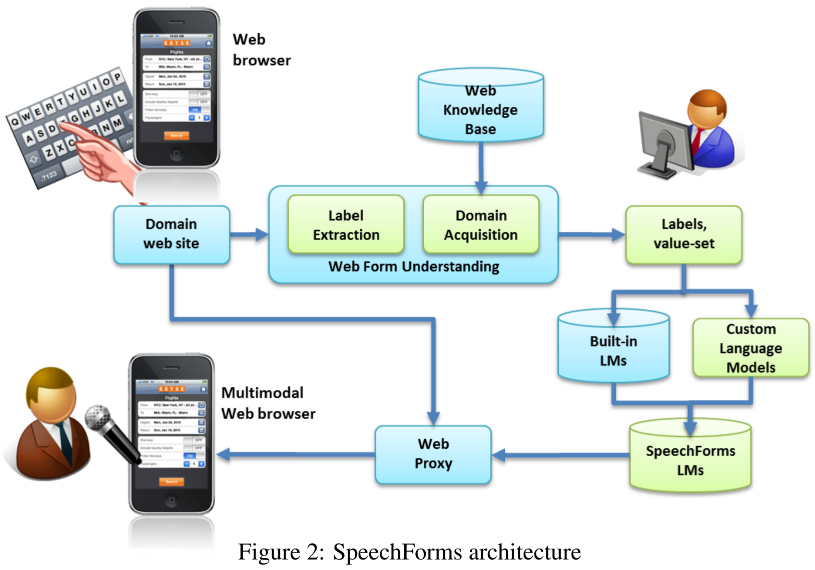

Figure 2: SpeechForms architecture

3. System description

SpeechForms is a system that semi-automatically creates and assigns language models to web form elements following the architecture displayed in Figure 2. A wizard (e.g., a website developer) points the system at one or more web forms about a particular task (e.g., flight booking, hotel booking). The system processes the web forms, extracting $\{label, value\text{-}set\}$ pairs for each form element. If the value-set for a form element is non-empty, the system automatically builds a language model from it. Otherwise, the system may: (1) use the label to automatically assign a pre-built language model to the element; (2) take the value-set from a matching element in one of the other input forms; (3) ask the wizard to select a pre-built language model for the form element; or (4) ask the wizard to point the system to a set of lexical values for the element (e.g., a list of hotel chain names from Wikipedia). The output includes labels and language models assigned to all form elements. The form can then be processed using speech through a speech-enabled web browser such as the MTALK browser [18]. Wizard involvement is minimal, consisting of (1) identifying input forms, (2) possibly connecting related form elements having different labels in different forms (e.g., *departure city* and *from*), and (3) possibly choosing a built-in language model or supplying a list of lexical values for a form element having no value-set in the form. None of these requires technical expertise.

3.1. Web form understanding

The goal of this module is to understand the structure of web forms in order to create language models specific to form elements and, ultimately, enable speech-driven form filling. A web form is composed of a set of input *elements* such as text fields/areas, checkboxes, selection lists, radio and submit buttons. Each element is usually associated with a *label*, and optionally a *value-set* of allowed inputs. Thus, we need to extract, for each form's element, the pair $\{label, value\text{-}set\}$.

In an ideal world, web developers would carefully follow style guidelines for implementation of web forms, making sure that each label was associated with its corresponding element using the appropriate tags, that element values were written in a way that could easily be spoken, etc. Unfortunately, actual web forms often do not even use the form tag, consisting instead of tables or lists, with fields and values linked visually but not in the underlying markup. The Information Integration community has proposed several techniques for form understanding (e.g., [19, 20, 21]). These techniques either require substantial human input or are very complex to implement. In this paper, we use an alternative, easy-to-implement, two-step method. First, we automatically assign labels to form elements in each input web form. Second, we semi-automatically assign value-sets and build language models for each form element.

Label extraction. The first step of our approach is to identify the labels in the form. We consider the user-viewable text inside the HTML form as a text document, and try to identify in this document words that correspond to labels. An important characteristic of Web forms is that, in a given domain, they contain a well-defined and restricted vocabulary [22]. Figure 1 illustrates this. It presents two web forms for the air travel task. We can see that there is considerable overlap in vocabulary (element labels and value-sets) across the two forms. Additionally, no single form has much textual content. Consequently, we take as labels words in the visible text of the form that are contained in a list of the most common labels in the form's domain provided by the DeepPeep website [1].

Value-set acquisition. Once we identify a form element and its label, we examine the form structure to get the value-set associated with the form element. For form elements such as selection lists and check boxes, value-sets are obtainable from the form structure itself. For open form elements such as text fields, the task is much harder. Some approaches [23, 24] have been able to identify value-sets for these elements by probing exploratory queries, for instance, from frequent words in the web site that contains the form. Here, we obtain value-sets for text fields by using information in the form itself (e.g., help popup dialogs), or by querying the user to select a pre-built language model or provide a list of values from another resource (e.g., Geonames[2], WordNet [25], Wikipedia[3] or YAGO [26]).

In the travel domain, we use the algorithm shown in Algorithm 1 to select the language model for each field. In future work, we plan to generalize the open *"text"* field case to use WordNet or another lexical taxonomy.

[1]`http://deeppeep.org`
[2]`http://geonames.org`
[3]`http://wikipedia.org`

```
1: Input: field
   switch field.type do
   case checkbox
       if exists(field.label) then
           return binary-values model

   case select
       extract list from html, build model and return
       model
   case text
       switch field.label do
           case leave OR return
               return date-time model
           case from OR to
               return airport model

       endsw
   case radio
       extract list from multiple radios with same
       name, build model and return model

   endsw
```

Algorithm 1: Algorithm for value-set acquisition in air travel domain

Corpus	Words	Sentences
Orbitz Airport List	10,062	3,856
Wikipedia IATA Airport List	61,997	18,430
Tourism 100k	42,539,932	5,795,081
Tourism 1M	792,762,712	91,634,211

Table 1: Training corpora sizes for airport language models.

3.2. Language modeling

Given a list of field values mined from the previous step, the system first normalizes the lexical items in the value-set using very general criteria that expand common abbreviations, numbers, dates, times, etc. From this normalized text, a bigram Katz's backoff n-gram language model is built. For some fields such as dates and times, preexisting rule-based W3C SRGS grammars are used.

4. Experiments

SpeechForms is designed to be a comprehensive end-to-end dialog generation system. In this paper we only evaluate one part of this process: language model creation. We compare the performance of language models built from individual web form elements to the performance of language models semi-automatically acquired from web text data (as in [7, 8, 9, 10, 11]). For this experiment we use the travel domain, a domain for which there is pre-existing, publicly available test data.

4.1. Test data

Filling out web forms is a system-directed activity; for example, without a dialog interface, the user cannot choose to supply all relevant element values in one form input. So for these experiments we looked for data from system-initiative dialogs in the air travel domain. We extracted all dialogs from the 2001 COMMUNICATOR LDC corpus[4] for the two most system-initiative COMMUNICATOR systems, the CMU (numdia) and AT&T (numdia2) systems. From these dialogs, we automatically extracted all utterances labeled with named entities corresponding to elements in the web forms listed in the previous section. The named entity types relevant to our web forms are CITY and AIRPORT (for the city/airport fields) and DATE_TIME (for the date/time fields), so these are the two fields on which we conducted our evaluation. The resulting test data included 532 utterances: 336 utterances containing at least one date/time, and 196 containing at least one city or airport name[5].

[4]LDC catalog numbers LDC2003S01 and LDC2004T16

4.2. Language model building

For the airport/city form elements, we built language models as described in the previous section. For the date/time-related form elements, we pointed the system to a pre-built date/time language model from the WATSON speech recognizer [27].

As baseline, we report recognition results for both airport/city and date/time fields using a language model trained on Web data. We used a focused crawler [28] to collect two different sets of Web pages in the travel domain. The first set contains 100,000 pages (Tourism 100k) and the second 1.5 million pages (Tourism 1M). Regarding the specialized airport/city language model, we used two different data sets: the Orbitz Airport List, obtained from the air travel form on the Orbitz web site[6]; and the Wikipedia IATA Airport List[7], which is bigger and contains more information. Statistics on the size of our language model training data sets, and the resulting language models, are reported in Table 1. Bigram Katz's backoff language models were created from each data set.

4.3. Results and discussion

In Figures 3 and 4 we show the results of our experiments. For both date/time and city/airport fields, the value-set language models are significantly faster than the Web data language models; the value-set language model for dates/times also gives better recognition accuracy. Best performance on value-set city/airport utterances is 87.3% (at 0.3 times real time), and on value-set date/time utterances is 66.5% (at 0.1 times real time). By contrast, best performance on the Web data language models is 87.0% (at 0.6 times real time) for cities/airports, and 52.2% (at 0.5 times real time) for dates/times.

Common errors made by the best-performing model for cities/airports include tokenization differences (e.g., *ST* vs *Saint*). Common errors made by the best-performing model for dates/times involve misrecognizing an ordinal number as a cardinal number (e.g., *seventh* as *seven*).

Although these results are encouraging, they are very preliminary. We have a long way to go in order for SpeechForms to be a scalable solution for speech enabling web forms. We are currently working to improve the system along several dimensions. In terms of automatic processing of web forms, we plan to: (1) incorporate additional taxonomic and lexical information (from sources including GeoNames, YAGO and WordNet), to reduce the amount of work required by the wizard; and (2) add functionality to obtain value-sets for free text fields by probing exploratory queries [23, 24]. In terms of speech enabling web forms, we plan to add functionality to improve the spoken interaction, providing better help to users in case of misrecognition, and providing better spoken guidance through the form using form labels. Finally, we plan to conduct more comprehensive evaluations of the SpeechForms system, involving several transactional domains (air travel, hotel booking, car rental/buying,

[5]A list of our test utterances is available upon request.

[6]http://www.orbitz.com/pagedef/content/air/airportCodes.jsp

[7]http://en.wikipedia.org/wiki/List_of_airports_by_IATA_code

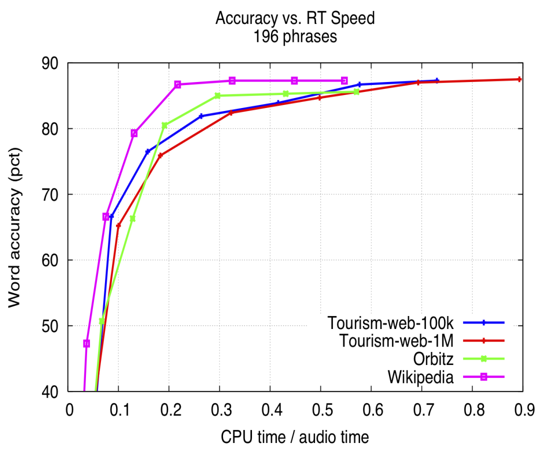

Figure 3: Accuracy vs. real time factor of airport LM

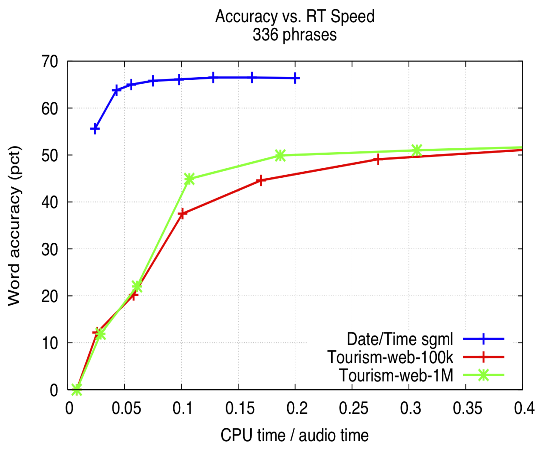

Figure 4: Accuracy vs. real time factor of date/time LM

product purchasing) and more complex web forms.

5. Conclusions

In this paper we present early work on SpeechForms, a system for semi-automatic creation of interactive spoken interfaces to web forms. This system will be of use to people interacting with the web over small form-factor devices such as mobile phones, and to people with visual or print disabilities. SpeechForms will also lead to the gradual compilation of an extensive list of language models for the most common web form elements.

6. References

[1] M. Hu *et al.*, “Speech-enabled web services for mobile devices,” in *Proc. the International Conference on Semantic Web and Web Services*, 2006.

[2] K. Wang, “SALT: A spoken language interface for web-based multimodal dialog systems,” in *Proc. ICSLP*, 2002.

[3] J. Axelsson, “XHMTL+VOICE in action,” October 2006, http://dev.opera.com/articles/view/xhtml-voice-in-action/.

[4] D. Goddeau *et al.*, “A form-based dialogue manager for spoken language applications,” in *Proc. ICSLP*, 1996.

[5] S. Issar, “A speech interface for forms on WWW,” in *Proc. EUROSPEECH*, 1997.

[6] G. Di Fabbrizio *et al.*, “Bootstrapping spoken dialogue systems by exploiting reusable libraries,” *Natural Language Engineering*, vol. 14, no. 3, pp. 313–335, July 2008.

[7] I. Bulyko *et al.*, “Web resources for language modeling in conversational speech recognition,” *ACM Transactions on Speech and Language Processing*, vol. 5, no. 1, 2007.

[8] T. Misu and T. Kawahara, “A bootstrapping approach for developing language model of new spoken dialogue systems by selectingweb texts,” in *Proc. INTERSPEECH*, 2006.

[9] R. Sarikaya, A. Gravano, and Y. Gao, “Rapid language model development using external resources for new spoken dialog domains,” in *Proc. ICASSP*, 2005.

[10] A. Tsiartas, P. Georgiou, and S. Narayanan, “Language model adaptation using WWW documents obtained by utterance-based queries,” in *Proc. ICASSP*, 2010.

[11] X. Zhu and R. Rosenfeld, “Improving trigram language modeling with the world wide web,” School of Computer Science, Carnegie Mellon University, Tech. Rep. CMU-CS-00-171, 2000.

[12] A. Gruenstein, S. Seneff, and C. Wang, “Scalable and portable web-based multimodal dialogue interaction with geographical databases,” in *Proc. INTERSPEECH*, 2006.

[13] B. Ballinger, C. Allauzen, A. Gruenstein, and J. Schalkwyk, “On-demand language model interpolation for mobile speech input,” in *Proc. INTERSPEECH*, 2010.

[14] J. Larson, “VoiceXML and the W3C speech interface framework,” *IEEE Multimedia*, vol. 10, no. 4, pp. 91–93, 2003.

[15] M. McTear, “Modelling spoken dialogues with state transition diagrams: Experiences with the CSLU toolkit,” in *Proc. ICSLP*, 1998.

[16] A. Pargellis, J. Kuo, and C.-H. Lee, “Automatic dialogue generator creates user defined applications,” in *Proc. EUROSPEECH*, 1999.

[17] J. Allen *et al.*, “PLOW: A collaborative task learning agent,” in *Proc. AAAI*, 2007.

[18] M. Johnston, G. Di Fabbrizio, and S. Urbanek, “mTalk - A Multimodal Browser for Mobile Services,” in *submission*, 2011.

[19] Z. Zhang, B. He, and K. Chang, “Understanding web query interfaces: Best-effort parsing with hidden syntax,” in *Proc. SIGMOD*, 2004.

[20] S. Raghavan and H. Garcia-Molina, “Crawling the hidden web,” in *Proc. VLDB*, 2001.

[21] H. Nguyen, T. Nguyen, and J. Freire, “Learning to extract form labels,” *Proc. the VLDB Endowment*, vol. 1, no. 1, pp. 684–694, 2008.

[22] B. He, T. Tao, and K. Chang, “Organizing structured web sources by query schemas: a clustering approach,” in *Proc. CIKM*, 2004.

[23] L. Barbosa and J. Freire, “Siphoning hidden-web data through keyword-based interfaces,” in *Proc. SBBD*, 2004.

[24] A. Ntoulas, P. Zerfos, and J. Cho, “Downloading textual hidden web content through keyword queries,” in *Proc. the 5th ACM/IEEE-CS Joint Conference on Digital Libraries*, 2005.

[25] C. Fellbaum, Ed., *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press, 1998.

[26] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: A core of semantic knowledge,” in *Proc. WWW*, 2007.

[27] V. Goffin *et al.*, “The AT&T WATSON speech recognizer,” in *Proc. ICASSP*, 2005.

[28] S. Chakrabarti, M. van den Berg, and B. Dom, “Focused crawling: A new approach to topic-specific web resource discovery,” *Computer Networks*, vol. 31, no. 11-16, pp. 1623–1640, 1999.