Natural Language Understanding for Partial Queries

Xiaohu Liu, Asli Celikyilmaz, Ruhi Sarikaya

Microsoft Corporation, Redmond, WA 98052

{derekliu, aslicel, ruhi.sarikaya}@microsoft.com

Abstract

Typical natural language understanding systems are built based on the assumption that they have access to the fully formed complete queries. Today's natural user interfaces, however, enable users to interact with various services and agents (e.g. search engines, personal digital assistants) running on desktop computers and laptops. The system is expected to understand the user's intent while the user is typing the query with the goal of increasing system response rate and ultimately improving the user's productivity. Language understanding models built on fully formed queries perform poorly when tested on partial or incomplete queries. In this study, we consider the problem of domain detection for typed partial natural language queries. We design two sets of features in addition to lexical features to train a multi-valued domain classification model. The first feature set consists of character n-gram features, and the second is the class-based features extracted from clustering of word embeddings. Our experiments show that the two feature sets improve the model's performance by up to 52.8% in comparison to the lexical n-gram baselines.

Index Terms: natural language understanding, incomplete utterances, character n-gram

1. Introduction

Personal digital assistants has been receiving tremendous attention in the last couple of years as a means to enable information access and task completion and ultimately to improve user's productivity. When they are deployed on the mobile devices such as smart phones, a significant portion of the queries are through a voice interface, since voice could be the most effective input modality in terms of the information throughput. In addition, compared to speech, typing the information might be inconvenient for the user. However, in real applications, we have observed that when the personal assistant applications are being ported to the desktop, laptop and tablet computers, where the dominant input modality is type-in modules. However, there are differences between voice-in and type-in modules in the way the users interact with the natural user interfaces (NUI) on such devices. Concurrently, the change in the input modality effects how the statistical spoken dialog models are built, in particular, the natural language understanding (NLU) models.

Traditional NUI systems that are built on type-in modalities wait for the user to complete their query (detected by the enter key) and the NLU models are then trigged to understand the user's intent. Consider the following scenario: The user types in a natural language query and the system doesn't wait for the user to complete their query. Instead, it immediately starts executing the language understanding stack as soon the user starts typing. Such a system is more efficient since the NLU system can return an instant response regarding the user's intent with each key-stroke. This would be a great productivity boost as the user does not have to type the entire query and wait for the system to respond. This very notion is used in commercial search engines [1, 2] in the form of auto-suggestions [3, 4] where the system makes suggestions about what should be typed and at the same time showing the instant results as the user is typing the query.

State-of-the-art NLU systems [5, 6, 7, 8] consist of three key models: 1) domain detection, 2) intent detection, 3) slot tagging. The domain detection model determines which domain the query belongs to at a high level (e.g. find documents, weather, places, calendar, communications,...). The intent model determines the specific intent of the user given the domain (e.g. check_weather, find_place, get_directions, call_contact, ...). The slot tagger extracts the slots and entities contained in the query (e.g. date, location, contact,...). For multi-domain NLU systems, a top level domain detection is critical to scope the user's intent.

In this study we focus on the domain detection problem. Specifically, we address the language understanding problem for partial natural language queries to infer the user's goal. We propose a new model called a "prefix model" that can handle partial (incomplete) queries as well as fully formed complete queries. The incomplete queries have prefix characteristics, i.e. the beginning of the query is complete, but the remaining words or characters are missing. In this paper we refer to incomplete queries as *prefix queries* and hence the new domain detection model that can efficiently handle the prefix queries along with the fully formed complete queries.

Earlier work on dialog systems handled incomplete queries in human to machine dialog systems via incremental processing, in which interpretation components are activated, and in some cases decisions are made, before the user utterance is complete. Some of this work has demonstrated overall improvements in system responsiveness and user satisfaction [9], and others allowing diverse contextual information to be brought to bear during incremental processing [10]. The work in [11], builds an NLU system using partial speech recognition hypothesis to predict the semantic content, looking at length of the speech hypothesis as a general indicator of semantic accuracy in understanding. Our work is different from earlier work in that our prefix models are not trained on partial speech recognition results, but trained on the complete natural language utterances via novel feature sets.

In the next section, we give details of our prefix model and later in Section 3 we describe the word cluster features and how they are used in the prefix models. In Section 4, we present the character n-gram features to improve the robustness of the models. In Section 5 we provide an analysis of the prefix model while evaluating against different test sets with complete and incomplete queries. A discussion is presented in Section 6 followed by a conclusion and future work in Section 7.

2. Prefix model for domain classification

The main interfaces for interacting with a personal assistant is either to speak or use the search box to type in commands. These commands cover everything you expect the personal assistant to handle such as setting up reminders, meetings, sending email, finding documents, checking weather or searching the web. The user is expected to type and see the system response instantly with each keystroke. For example, when the user wants to set an alarm and intends to type "*set an alarm for 7:30 am*", the proposed domain detection model can trigger the "alarm" domain for the partial query "*set an al*." The system then gives instant feedback to the user by showing the alarm application before the user types the whole query.

There are two types of prefix queries we want to handle where traditional domain models trained with complete queries have trouble determining the domain of a query. The first type of queries contain complete words yet still lacks some words (e.g. "set an alarm"). Second type of queries usually have incomplete last word (e.g. "set an ala"). The incomplete words (e.g., "*ala*", "*alar*") are usually treated as unknown words. In fact they are similar to "*alarm*" to some degree, but standard domain detection models fail to use this similarity information efficiently. As a result, typical domain detection models are not designed to handle such prefix queries, as the training data usually do not contain prefix queries. This yields the domain detection model to perform poorly when the queries are incomplete.

The proposed prefix model is trained on a training data set that only contains complete queries (without the prefix queries), and yet it can handle both complete and incomplete queries for the domain detection task. We achieve this by using two sets of features and building an NLU system that operate at the character level, where a request is sent to NLU engine at each character stroke. It is proven that adding explanatory features extracted from different methods and sources selectively can improve model performance without additional training data [7, 12, 13]. Following this, we first build word clusters to capture the similarity between incomplete words and complete words, and use word cluster ids as explanatory features for the prefix model. We also find it being very useful to add character n-grams [14] as explanatory features for the prefix model. To our knowledge, this is the first attempt to build a prefix model to handle partial natural language queries.

3. Word class features

In [15, 16, 17], word class features are used for shrinking the size of the language models and NLU models, specifically multi-label logistic regression for intent detection and linear chain conditional random fields for slot filling tasks. The models are trained with shrinkage based features, which improve the robustness and generalization to unseen events and inputs. This is achieved by shrinking the model size while maintaining or improving performance on the training data. It was noted that the word class features are different than the typical use of class based features for modeling. In many earlier works just the word class membership (i.e. ID) for the current word is used, whereas the features used in [17] are compound word and class n-gram features. We use the following compound features to build the prefix models in the experiments:

- $w_j c_j$: the current word w_j in the query along with the corresponding class id c_j of that word.
- $w_{j-1} c_j$: the previous word w_{j-1} and the class id c_j of the current word w_j.
- $c_{j-1} w_j$: the previous word's class id c_{j-1}, and the current word w_j.
- $c_{j-1} c_j$: the previous word's class id c_{j-1} and the current word's class id c_j.
- $c_j w_{j-1} w_j$: the current word w_j, current word's class id c_j and previous word w_{j-1}
- $c_{j+1} w_{j+1}$: the next word w_{j+1} and the next word's class id c_{j+1}.

To obtain the class id for each word in the vocabulary, we use unsupervised clustering method. In this paper, we use the Brown Clustering method [18], which is the most commonly used n-gram clustering algorithm for many speech and language processing tasks. We only used full queries to cluster the words with Brown clustering.

Brown hierarchical word clustering algorithm partitions the vocabulary into a predefined number of classes to maximize the bi-gram mutual information between words and classes [18]. The algorithm first assigns each of the most frequent words to their own class and the remaining words to the final class. Then, the exchange algorithm is performed where individual words are moved to another class if this improves the class bi-gram mutual information, until no more such moves are possible. In this paper, we use the C++ implementation of Brown clustering [19].

4. Character n-gram features

Most NLU systems use word n-grams as the main features. A word n-gram is a sequence of *n* contiguous words. For example, from the query *"please set an alarm"*, we can extract four uni-grams, three bi-grams and two tri-grams if the sentence boundary is not considered as a word token for simplicity. Using only word n-grams is not sufficient for prefix queries as they may contain incomplete words. We need to use sub-word features [20] extracted from the partial queries, as well as dependency between sub-word units and the fully formed words to predict the domain. Character n-grams can serve the purpose as they are n-grams at character level.

Similar to word n-grams, a character n-gram is a sequence of *n* contiguous characters in a query. A word boundary represented by the space letter is important when extracting character n-grams. For better readability, we denote the space character with underscore '_'. For example, some interesting character five-grams extracted from the query "*please set an alarm*": "*set_a*", "*et_an*", "*an_al*", "*_alar*".

Table 1: *Sample character n-grams.*

length	character n-gram samples from "please set an alarm"
2	pl, _s, se, _a, al
3	ple, e_s, _se, t_a, _an, _al, ala
4	plea, se_s, e_se, _set, an_a, n_al, _ala, alar
5	pleas, ase_s, se_se, e_set, _an_a, an_al, n_ala, _alar

An incomplete word misses characters from the end (trailing characters). Hence, we do not extract character n-grams starting in the middle of word and ending in the same word. For example, we don't extract bi-grams "*la*" or "*ar*" from "*alarm*". The character trigram "*lar*" is not extracted either. We want to

Table 2: *Data for training and testing with frequency information.*

domains	train _ queries	test _ queries
alarm	35040	652
calendar	33533	1723
communication	151038	749
note	14952	1188
reminder	37814	500
timer	470	273
weather	44628	752
web	408155	90749

make sure each character n-gram extracted contains a word prefix. This is important since the nature of incomplete queries is that the missing characters are on the right side only.

5. Experiments

As the first step towards partial query understanding, our main goal is to build a domain model to classify queries into one of eight supported domains: *alarm, calendar, communication, note, reminder, timer, weather*, and *web*. There are reasonable number of queries for training and test in every domain as shown in Table 2.

The data is collected and annotated by crowdsourcing. Data sets were previously targeted to handle complete natural language queries. Each query is labeled with one of eight possible domains from the semantic frame. The domain classification model is a multi-class classification so we use multi-class logistic regression. We built four different models using different feature sets to benchmark them against a word n-gram baseline model as follows:

- Baseline model: this model is built using word n-gram features up to tri-gram features (e.g., uni-grams, bi-grams, and tri-grams).
- Model 1: the model is trained using word n-gram features and word class features.
- Model 2: the model is trained using both word n-gram and character n-gram features.
- Model 3: the model is trained using word n-gram, character n-gram and word class features.

Next, we present the experiment results where we evaluate the performance on prefix test queries, complete test queries, impact of the length of the test queries on accuracy as well as domain specific data.

5.1. Analysis on complete and prefix test queries

We first evaluate the models against a fixed test set which includes prefix queries only. The performance is measured using overall classification error rate and results are presented in Table 3.

Table 3: *Classification error rates on prefix query set(%)*

Baseline	Model 1	Model 2	Model 3
4.79	4.31	2.36	2.26

We observe that, when used individually along with word n-gram features, both word class and character n-gram features are effective in reducing errors. The classification error rate drops by 10% relatively comparing Model 1 with the baseline model. The model 2 reduces the error rate by 50.7% relative to the baseline model. When both feature sets are used in Model 3, the error rate is further reduced by 4.2% compared to using character n-gram features alone. Compared to the baseline model, Model 3 reduces the error rate by 52.8% relatively.

Table 4: *classification error rates on complete query test set (%)*

Baseline	Model 1	Model 2	Model 3
3.69	3.01	3.11	2.74

It should be noted that, when we increase the n-gram window size, more context is covered but a larger training data set is also required. We experimented by increasing the n-gram window size until there is no performance gain. Our experiments show that n-grams from bi-grams to six-grams are all useful.

We see that the enhanced models have lower error rates than the baseline model when handling incomplete queries, we also want to compare the performance on regular queries. We would like to have little or no degradation on complete queries, otherwise there may be a need to build two separate models. This is critical because there are numerous engineering constraints regarding latency, memory usage and cost for supporting and maintaining two sets of models as opposed to one.

We run experiments on a test set including only complete natural language queries. The models with proposed feature sets still outperform baseline models as illustrated in Table 4. All three models have lower error rates than the baseline model, and the model with both feature sets has the lowest error rate.

The take away here is that, since the proposed models have better performance in both complete and incomplete (partial) queries, we do not need to build different models for complete and incomplete queries. We can build one model with both word class features and character n-gram features to replace two separate models.

5.2. Per domain analysis of the prefix model

In addition to overall classification error rates, we are also interested in analyzing how the models perform in each domain. We measure the domain classification performance in all domains using F1 scores (F1=2* $\frac{precision*recall}{precision+recall}$ is a measure of a test's accuracy considering both the *precision* and the *recall*).

The results are shown in Table 5. We observe that in each domain, the model with proposed feature sets performs better than the baseline model. Both model 2 and 3 have the character n-gram feature set and give the best results in all domains.

Table 5: *Performance comparison across domains (F1 scores)*

domains	Baseline	Model 1	Model 2	Model 3
alarm	34.63	63.84	**77.09**	70.51
calendar	40.10	54.00	70.11	**70.76**
communication	59.44	63.14	70.22	**78.17**
note	74.18	74.18	**85.87**	81.96
reminder	66.53	68.55	82.43	**83.43**
timer	89.04	89.94	**92.82**	90.26
weather	59.44	61.32	**81.03**	78.04
web	96.93	97.88	98.72	**98.77**

5.3. Impact of prefix query length on prefix models

It is interesting to see how models perform with regarding to the length of partial queries. We test four models against queries with length ranging from 2 to 26 characters (there are very few queries longer than 26 characters). The results are shown in Figure 1. In the figure, the x-coordinate corresponds to query length, e.g. $x = 10$ represents a set of test queries with 10 or more characters. As queries get longer from 2 to 10, the model improvement over baseline (e.g. the gap between baseline and Model 1) is getting smaller. This reflects the fact that it is generally easier to classify longer queries than short ones even though they are incomplete. More features can be extracted from longer queries. The model improvement over the baseline is more than 1.7% absolute difference across all test sets. Model 3 achieves the lowest error rates consistently.

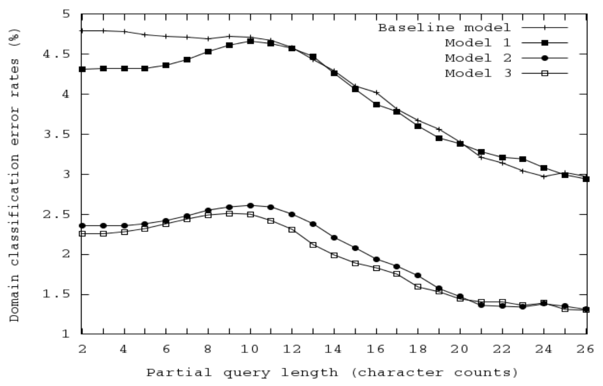

Figure 1: Performance against query length

6. Discussion

It is possible to build models to tackle partial queries by traditional supervised approaches, where we collect and annotate incomplete queries and train NLU models. The issue is that it is more expensive to collect a training set with incomplete queries than regular queries. Since we already have complete queries in each domain, ideally we should be able to reuse the data without extra cost.

A straightforward approach we tried first is to automatically generate prefix queries from complete queries and add them to the training sets. The generated prefix queries are assigned with the same labels as the original queries. However the outcome was very poor as the domain labels that are assigned to auto-generated queries can be wrong therefore too much noise is introduced by these generated queries.

7. Conclusions and future work

In this paper, we describe the new problem of partial query understanding. We build models from existing training data using additional feature sets, such as word clusters and character n-grams. In desktop search scenarios, we significantly reduce the domain classification errors and achieve reasonable performance without adding extra data. We still need to work on improving the performance of some domains. One possible future work is improving the prefix model's accuracy is by using a language modle to predict the next word along with the user's feedback to the system responses.

8. References

[1] B. Mitra, M. Shokouhi, F. Radlinski, K. Hofmann, "On user interactions with query auto-completion", *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2014.

[2] M. Shokouhi, "Learning to personalize query auto-completion", *Proceedings of the 36th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2013.

[3] S. Whiting , J. M. Jose, "Recent and robust query auto-completion", *Proceedings of the 23rd International Conference on World Wide Web*, 2014.

[4] H. Ma, M. R. Lyu, I. King, "Diversifying Query Suggestion Results", *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, AAAI 2010, Atlanta, Georgia, USA, 2010.

[5] J-P. Robichaud, P. A. Crook, P. Xu, O. Z. Khan, R. Sarikaya, "Hypotheses Ranking for Robust Domain Classification And Tracking in Dialogue Systems", *Proceedings of Interspeech*, Singapore, 2014.

[6] P. Xu and R. Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network", *Proceedings of ICASSP*, Florence, Italy, 2014.

[7] X. Liu, R. Sarikaya, C. Brockett, C. Quirk, and W. Dolan, "Paraphrase Features to Improve Natural Language Understanding", *Proceedings of Interspeech*, Lyon, France, 2013.

[8] Tur, G. and Mori, R. D., "Spoken Language Understanding: Systems for Extracting Semantic Information from Speech", John Wiley and Sons, 2011.

[9] G. Skantze and D. Schlangen. "Incremental dialogue processing in a micro-domain". *Proceedings of EACL 2009*, pages 745753, 2009.

[10] G. Aist, J. Allen, E. Campana, C. G. Gallo, S. Stoness, M. Swift, and M. K. Tanenhaus. "Incremental dialogue system faster than and preferred to its nonincremental counterpart." *Proceedings of the 29th Annual Conference of the Cognitive Science Society*,2007.

[11] K. Sagae, G. Christian, D. DeVault, and D. R. Traum. "Towards natural language understanding of partial speech recognition results in dialogue systems." *Proceedings of NAACL HLT*, 2009.

[12] A.K. Farahat, A. Ghodsi, M. S. Kamel, "Efficient greedy feature selection for unsupervised learning", Knowledge and Information Systems, May 2013.

[13] Y. Lu, I. Cohen, X. Zhou, Q. Tian, "Feature Selection Using Principal Feature Analysis", *Proceedings of the 15th international conference on Multimedia*, NY, USA, 2007.

[14] J. Houvardas, E. Stamatatos, "N-Gram Feature Selection for Authorship Identification", *Artificial Intelligence: Methodology, Systems, and Applications*, 2006.

[15] R. Sarikaya, S. F. Chen, A. Sethy, B. Ramabhadran, "Impact of Word Classing on Shrinkage-based Language Models", *Proceedings of Interspeech*, Tokyo, Japan, 2010.

[16] R. Sarikaya, S. F. Chen, B. Ramabhadran, "Shrinkage-Based Features for Natural Language Call Routing", *Proceedings of Interspeech*, Florence, Italy, 2011.

[17] R. Sarikaya, A. Celikyilmaz, A. Deoras, M. Jeong, "Shrinkage Based Features for Slot Tagging with Conditional Random Fields", *Proceedings of Interspeech*, Singapore, 2014.

[18] Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai, "Class-based n-gram Models of Natural Language.", *Computational Linguistics*, 18(4), 1992.

[19] P. Liang, "Semi-Supervised Learning for Natural Language", Masters thesis, MIT, 2005.

[20] R. Sarikaya, M. Afify, Y. Deng, H. Erdogan, Y. Gao, "Joint morphological-lexical language modeling for processing morphologically rich languages with application to dialectal Arabic", *IEEE Trans. Speech Audio and Langugage Proc.*, vol. 16(7), pp. 1330-1339, 2008.