LANGUAGE MODELING FOR VOICE SEARCH: A MACHINE TRANSLATION APPROACH

Xiao Li, Yun-Cheng Ju, Geoffrey Zweig, and Alex Acero

Microsoft Research
One Microsoft Way, Redmond, WA, 98052, U.S.A.
{xiaol,yuncj,gzweig,alexac}@microsoft.com

ABSTRACT

This paper presents a novel approach to language modeling for voice search based on the idea and method of statistical machine translation. We propose an n-gram based translation model that can be used for listing-to-query translation. We then leverage the query forms translated from listings to improve language modeling. The translation model is trained in an unsupervised manner using a set of transcribed voice search queries. Experiments show that the translation approach yielded drastic perplexity reductions compared with a baseline language model where no translation is applied.

Index Terms— language modeling, machine translation, voice search, directory assistance

1. INTRODUCTION

With the skyrocketing popularity of search technologies, the ability to enter search queries by voice has become increasingly appealing for mobile and telephony applications. Indeed, recent years have witnessed a burgeoning development of voice search services, mostly in the domain of local search [1], With such services, users can retrieve information, such as phone numbers, addresses and driving directions, of business listings or business categories by speaking to an automated agent.

A technical framework for voice search has been proposed in [2] and further investigated in [3]. In this framework, a voice search system is decoupled into two components: voice recognition and information retrieval. First, a spoken utterance $\mathbf{o}$ is converted into text query $\mathbf{q}$ using automatic speech recognition (ASR), *i.e.*,

$$\mathbf{q}^* = \underset{\mathbf{q}}{\operatorname{argmax}}\, p(\mathbf{o}|\mathbf{q})p(\mathbf{q}) \tag{1}$$

where $p(\mathbf{o}|\mathbf{q})$ and $p(\mathbf{q})$ represent an acoustic model and a language model (LM) respectively. Statistical LMs, *e.g.* n-gram models, are often used to allow flexibility in what users can say. Next, the best (or n-best) $\mathbf{q}$ is feed into a search engine to retrieve the most relevant document $\mathbf{d}$, *i.e.*

$$\mathbf{d}^* = \underset{\mathbf{d}}{\operatorname{argmax}}\, p(\mathbf{d}|\mathbf{q}) \tag{2}$$

In the context of local search, documents $\mathbf{d}$ are in form of business listings, which are typically short, *e.g. Kung Ho Cuisine of China*.

To build the language model $p(\mathbf{q})$ for ASR, a simple and intuitive approach is to use listings themselves as training sentences. Often, however, a user refers to a listing in a different way as the original form of that listing; and there are often multiple ways of expressing the same listing. For example, the listing *Kung Ho Cuisine of China* can be formulated as *Kung Ho*, *Kung Ho Chinese Restaurant*, or *Kung Ho Restaurant*. In this regard, the LM trained using listing forms may not best predict what users will say in practice. Ideally, the LM should be trained using transcriptions of real voice search queries, but this would require a prohibitive number of training samples, considering the number listings is often in the magnitude of $10^6 \sim 10^8$ (for regional or national business search). A more realistic approach, as was adopted in [4], is to generate query forms by rules based on a combination of human knowledge and data.

In this work, we propose a statistical translation model that automatically converts the original form of a listing to its query forms, which in turn are used for building more robust LMs for voice search. The translation model is trained using a small number of transcribed queries, without the need of acquiring the true listings. Compared with rule-based approaches, our method requires the least human knowledge and can easily lend itself to be used in different voice search applications such as product search [5].

The work most related to the idea presented in this paper is the use of machine translation in web information retrieval [6], where queries are assumed to be generated or ”translated” from web documents. In our task, however, the listings and queries are both very short, and hence more akin to the concept of ”sentence pairs” in bilingual machine translation, thus enabling us to borrow some techniques used therein. The rest of the paper is organizes as follows: Section 2 and Section 3 present the details of our translation model. Section 4 discusses how we obtain listing and query pairs that are used to train our translation model. Section 5 discusses LM perplexity experiments, followed by discussions in the last section.

2. NGRAM-BASED TRANSLATION MODEL

Although a query and its intended listing may differ in forms, there usually exists a semantic correspondence, at the word level, between the two. In other words, each word in the query can be mapped to a word in the listing or to a *null* word; and it is vice versa. This motivates us to use a machine translation approach to predict possible query forms of a listing, and then to utilize the predicted query forms to improve language modeling. Specifically, we use n-grams on word pairs to model the joint probability of a listing and a query. Note that this model has a similar flavor to the *tuple* n-gram model used in bilingual machine translation [7], and to the joint multi-gram model used in letter-to-sound conversion [8, 9].

We start with the assumption that there exists a corpus of parallel text $(\mathbf{d}, \mathbf{q})$, where listings and queries serve as source and target sentences respectively. Moreover, we let $\mathbf{a}$ denote a *monotonic* alignment between $\mathbf{d}$ and $\mathbf{q}$, where *null* words are added, if necessary, to account for insertions or deletions that occur in the alignment. Once $\mathbf{d}$ and $\mathbf{q}$ are aligned, including sentence start and sentence end, we create a sequence of word pairs as follows,

$$(\mathbf{d}, \mathbf{q}, \mathbf{a}) = ((d_1, q_1), (d_2, q_2), \ldots, (d_L, q_L))$$

where we treat each (d_i, q_i) as a single semantic unit. Note that consecutive word pairs can be merged to form phrase pairs if necessary. We then train a standard n-gram model on such units. Consequently, the probability of an aligned sentence pair is computed as

$$p_M(\mathbf{d}, \mathbf{q}, \mathbf{a}) = \prod_i p((d_i, q_i)|(d_{i-n+1}, q_{i-n+1}), \ldots, (d_{i-1}, q_{i-1})) \quad (3)$$

where the subscript M is used to denote the monotonic condition. Initially, we obtain the alignment $\mathbf{a}$ by computing the Levenshtein distance between $\mathbf{d}$ and $\mathbf{q}$. Then we iteratively update the alignment and n-gram model parameters in the maximum likelihood sense.

Once the n-gram model is trained, the listing-to-query translation works as follows: given a listing form, we search for the query forms that yield the highest conditional probability. Mathematically,

$$\begin{aligned} \mathbf{q}^* &= \underset{\mathbf{q}}{\text{argmax}}\ p_M(\mathbf{q}|\mathbf{d}) \\ &= \underset{\mathbf{q}}{\text{argmax}}\ p_M(\mathbf{d}, \mathbf{q}) \\ &= \underset{\mathbf{q}}{\text{argmax}} \sum_{\mathbf{a}} p_M(\mathbf{d}, \mathbf{q}, \mathbf{a}) \\ &\approx \underset{\mathbf{q}}{\text{argmax}}\ \max_{\mathbf{a}} p_M(\mathbf{d}, \mathbf{q}, \mathbf{a}) \end{aligned} \quad (4)$$

where $p(\mathbf{d}, \mathbf{q}, \mathbf{a})$ is evaluated using Equation (3). The translation not only exploits word-level semantic correspondence as modeled by unigrams, but also takes into account word context by using higher-order n-grams. The search for the best or n-best query forms can be achieved efficiently by applying the best-first search algorithm [10], where pruning techniques are applied to further reduce computational complexity. Once the n-best query forms are obtained for all listings, we add them into our training sentences for LM estimation.

There are two implementation details worth special attention. First, allowing the use of *null* words in $\mathbf{d}$ raises a potential problem at decode time — the search space is drastically expanded since *null* can be present or absent at any position of the source sentence. To avoid this problem, we eliminate the use of $(d_i = null, q_i)$ as semantic units for all values of q_i. Specifically, in training we merge $(d_i = null, q_i)$ with its proceeding or following semantic unit, depending on which of the phrases, $q_{i-1}q_i$ or $q_i q_{i+1}$, have more occurrences in the training data. We then treat $(d_{i-1}, q_{i-1}q_i)$ or $(d_{i+1}, q_i q_{i+1})$ as a single semantic unit. At decode time, we do not explicitly insert *null* in $\mathbf{d}$, since using semantic units $(d_{i-1}, q_{i-1}q_i)$ or $(d_{i+1}, q_i q_{i+1})$ is equivalent to adding *null* in the source sentence.

The second problem is concerned with out-of-vocabulary (OOV) words in $\mathbf{d}$. When OOV occurs, we would not be able to produce any query forms, since $p(d_i = \text{OOV}, q_i) = 0$ for any value of q_i. To deal with such cases, we always assign a positive probability to unigrams $(d_i, q_i = d_i)$ whenever $d_i = \text{OOV}$. This implies that a listing word, if never seen in the training data, will always be translated to itself, which is intuitively correct.

3. REORDERING

A natural extension to the above framework is to allow non-monotonic alignments between $\mathbf{d}$ and $\mathbf{q}$. An equivalent way of formulating this problem is to reorder $\mathbf{d}$, while keeping the order of $\mathbf{q}$, before a monotonic alignment is applied. Formally, for a listing $\mathbf{d}$ with length n, we introduce a variable $s = (s(1), s(2), \ldots, s(n))$ to represent a permutation of $(1, 2, \ldots, n)$. The resulting reordered listing is represented by $\mathbf{d}_s = (d_{s(1)}, d_{s(2)}, \ldots, d_{s(n)})$. Note that our monotonic condition in Section 2 is equivalent to assuming an identity permutation, *i.e.*, $s = (1, 2, \ldots, n)$. In general, the number of permutations is factorial in n. For computational feasibility, we constrain the permutation space to contain only a subset of permutations,

$$s \in \mathcal{S} = \{(j+1, j+2, \ldots, n, 1, 2, \ldots, j) : \quad j = 1..n\}$$

In other words, we only consider permutations that are shifts of the original order. In addition, for reordering purpose, we append *null* words to a listing, if necessary, to ensure all listing forms have the same length and hence the same permutation space.

Having defined the space of s, we revisit our training and decoding algorithms that enable reordering. For training, we modify our algorithm as follows.

1. Initialize an alignment between $\mathbf{d}$ and $\mathbf{q}$ by computing their Levenshtein distance; and estimate initial n-gram model parameters using the aligned sentence pairs;
2. For each listing $\mathbf{d}$, find the best permutation s given the current model,

   $$s^* = \underset{s \in \mathcal{S}}{\text{argmax}}\ \max_{a} p_M(\mathbf{d}_s, \mathbf{q}, \mathbf{a}), \quad (5)$$

   where $p_M(\mathbf{d}_s, \mathbf{q}, \mathbf{a})$ is defined in Equation (3) under the monotonic condition.
3. Given new $(\mathbf{d}_{s^*}, \mathbf{q})$ pairs, re-estimate n-gram model parameters following Section 2 under the monotonic condition.
4. Repeat step 2 and 3 until convergence. We let s^* denote the final permutation of $\mathbf{d}$.

At decode time, on the other hand, we allow a listing to be reordered in advance. Mathematically,

$$\begin{aligned} \mathbf{q}^* &= \underset{\mathbf{q}}{\text{argmax}}\ p(\mathbf{q}|\mathbf{d}) \\ &= \underset{\mathbf{q}}{\text{argmax}} \sum_{s \in \mathcal{S}} p(\mathbf{q}|s, \mathbf{d}) p(s|\mathbf{d}) \\ &\approx \underset{\mathbf{q}}{\text{argmax}}\ \max_{s \in \mathcal{S}} p_M(\mathbf{q}|\mathbf{d}_s) p(s|\mathbf{d}) \\ &\approx \underset{\mathbf{q}}{\text{argmax}}\ p_M(\mathbf{q}|\mathbf{d}_{\hat{s}}) \end{aligned} \quad (6)$$

where $p_M(\mathbf{q}|\mathbf{d})$ is defined in Equation (4) and where

$$\hat{s} = \underset{s \in \mathcal{S}}{\text{argmax}}\ p(s|\mathbf{d}) \quad (7)$$

The translation process is essentially decoupled into two steps. First, we use a classifier $p(s|\mathbf{d})$ to find the best permutation $\hat{s} \in \mathcal{S}$. Secondly, we search for the best query form $\mathbf{q}$ based on the permuted listing form $\mathbf{d}_{\hat{s}}$ under the monotonic condition. For the first step, we build a classifier $p(s|\mathbf{d})$ using a conditional maximum entropy model [11], where we have unigrams, bigrams and trigrams as input features. The classifier is trained using $(\mathbf{d}, s^*)$ pairs obtained from step 4 of our training algorithm. Note that we empirically found that the prior probability of s being the identity permutation is very high, meaning that in most cases a listing would stay in the original order.

4. LISTING RETRIEVAL USING TF-IDF

Our discussion assumes the availability of a set of $(\mathbf{d}, \mathbf{q})$, pairs, based on which a translation model can be trained. In many applications, however, while it is relatively easy to obtain query transcriptions, finding their intended listings is a difficult task. In this work, we obtain hypothesized listings for given queries in an unsupervised manner using information retrieval techniques. More specifically, we represent each $\mathbf{d}$ and $\mathbf{q}$ by a vector $\mathbf{v_d}$ and $\mathbf{v_q}$. Each vector component corresponds to a term (word or phrase) in the listing vocabulary, and the value of which is the *tf-idf* weight of that term with respect to the listing collection [12]. On the basis of this vector space model, we measure the relevancy between $\mathbf{v_d}$ and $\mathbf{v_q}$ by the *cosine similarity*, *i.e.*,

$$\cos(\mathbf{v_d}, \mathbf{v_q}) = \frac{\mathbf{v_d} \cdot \mathbf{v_q}}{\|\mathbf{v_d}\| \cdot \|\mathbf{v_q}\|}; \quad (8)$$

and we look for the listing $\mathbf{d}$ that maximizes this similarity metric. Moreover, we empirically set a threshold on this similarity metric, and $(\mathbf{d}, \mathbf{q})$ will be excluded from training if their cosine similarity does not reach the threshold.

The reason we choose to use *tf-idf* based cosine similarity is largely due to its widely acknowledged success in the field of information retrieval, although it might introduce noise in the retrieved results. For example, the query *Bluwater Restaurant*, if no exact match exists, would retrieve lexically similar listings such as *Bluwater Consulting*; and *Consulting* and *Restaurant* would be aligned as a semantic unit. With sufficient training data, however, one would hope the probability of such linguistically incorrect semantic unit to be trivial.

5. EVALUATION

5.1. Methodology

Given a data set of transcribed voice queries, there are two ways of measuring LM perplexity. First, we collapse a query set $\mathcal{Q}$ to contain unique queries, which we denote as $\mathcal{Q}_{uniq}$. We then randomly divide $\mathcal{Q}_{uniq}$ into train and test sets. Both training set queries and the original listing collection $\mathcal{D}$ are utilized to train a LM (we will discuss this in detail shortly). The LM perplexity is then evaluated on test set queries. We let PPL_{uniq} denote the perplexity obtained in this fashion. Since there is no query overlap between training and testing, PPL_{uniq} reflects how well the language model generalizes to unseen queries.

In real-world scenarios, however, some queries occur more frequently than others, and hence should be weighted more in both training and testing. We therefore use a second way of measuring perplexity. With certain sampling rate, we randomly sample $\mathcal{Q}$ to form a training set. In this way, a query with a high frequency in $\mathcal{Q}$ is likely to be sampled more times, and hence will be weighted more. Similarly we sample $\mathcal{Q}$ again to form a test set, on which the LM perplexity PPL is evaluated. Note that in general we expect $PPL < PPL_{uniq}$ with the same amount of training data.

5.2. Data preparation

Our data sets were obtained from an in-house voice local search application. Without loss of generality, our experiments focused on the Seattle (WA, U.S.) area, for which we had a feed of 140K business listings including a few hundred category names. We denote this listing set by $\mathcal{D}$. We also had a large collection of transcribed voice search queries from this area, which formed our query sets $\mathcal{Q}$ and hence $\mathcal{Q}_{uniq}$.

First, to evaluate PPL_{uniq}, we allocated 5k unique queries from $\mathcal{Q}_{uniq}$ for testing, which we denote as $\mathcal{Q}_{uniq}^{test}$. For each of the remaining queries $\mathbf{q}$, we retrieved the most relevant $\mathbf{d}$ following the method in Section 4, and we filtered out $\mathbf{q}$ if $\cos(\mathbf{v_d}, \mathbf{v_q}) < 0.8$. After filtering, we created training sets $\mathcal{Q}_{uniq}^{train}$ with different sizes that range from 1K to 10K. Similarly, to evaluated PPL on naturally-distributed queries, we allocated 6K queries from $\mathcal{Q}$ (in which 5K are unique) for testing, and we formed training sets $\mathcal{Q}^{train}$ with different sizes from 1K to 15K.

5.3. Perplexity experiments

In this work, we use bigram language models with backoff, and we use an open vocabulary but OOVs in the test set are not included in perplexity evaluation. For translation model, we use bigrams on word (or phrase) pairs and we allow reordering. Note that this bigram model for translation should be distinguished from the above bigram language model for ASR.

We conducted two sets of experiments, corresponding to the two evaluation methods presented in Section 5.1. Both methods compare a baseline LM, which we call QAL, with LMs estimated from translation results, which we call TAL.

- QAL: *Query Augmented Listings*. A baseline LM that is trained on a combination of $\mathcal{D}$ and training set queries (either $\mathcal{Q}_{uniq}^{train}$ or $\mathcal{Q}^{train}$).

- TAL: *Translation Augmented Listings*. We first build a translation model using training set queries and the retrieved listings; then we apply the translation model to listings $\mathcal{D}$ to obtain $\mathcal{D}'$, which contains the n-best query forms as well as their weights. Finally, we combine $\mathcal{D}$ and $\mathcal{D}'$ to train a new LM. Note that below we only report experiments using top $n = 1, 2, 3$ translations, as further increasing in n did not bring significant improvement.
- QTAL: *Query and Translation Augmented Listings*. Similar to TAL except we combine all $\mathcal{D}$, $\mathcal{D}'$ and training set queries in LM estimation.

Figure 1 shows the perplexity results of QAL as well as TAL with top $n = 1, 2, 3$ translations. Here the LMs were trained on $\mathcal{Q}_{uniq}^{train}$, with different amounts of training data, and was evaluated on $\mathcal{Q}_{uniq}^{test}$. Similarly, Figure 2 shows the results where we trained on $\mathcal{Q}^{train}$ and evaluated on $\mathcal{Q}^{test}$. Initially when no transcribed data was available, we used only $\mathcal{D}$ as training sentences for LM estimation, resulting in $PPL_{uniq} = 3830$ and $PPL = 1771$ (the initial points in both figures). Not surprisingly, as we increased the amounts of transcribed queries, we observe consistent perplexity reductions from all LMs. It is interesting to see that TAL-$n$best, $n = 1, 2, 3$, have significantly lower perplexities than QAL, given the same amounts of transcribed queries. This is because the translation approach is able to learn from the observed samples and generalize to unseen samples. As one can imagine, as the number of transcribed queries further increases, the performance of QAL will approach to, or even exceed, that of TAL, since the LM trained using QAL will asymptotically converge to the real query distribution. In practice, however, as the query set will always be finite, the translation approach can be more appealing owing to its generalization ability.

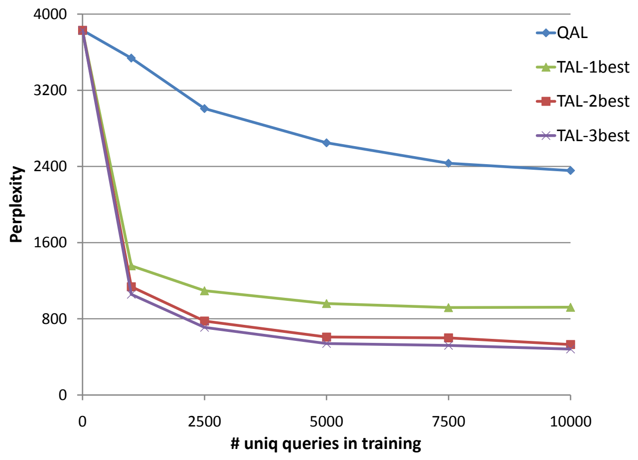

Fig. 1. Perplexities PPL_{uniq} on unique queries with different amounts of transcribed queries in training

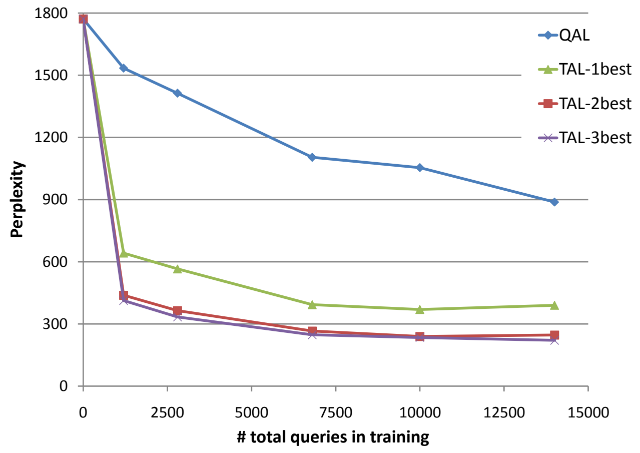

Fig. 2. Perplexities PPL on all queries with different amounts of transcribed queries in training

Lastly, we conducted experiments using QTAL, again with top $n = 1, 2, 3$ translations. They yielded only trivial improvements over TAL; here we do not include them in the figures to avoid confusion. Instead, we report the PPL values of QTAL with 3-best translations — the best configuration we obtained in all experiments — as well as their relative reductions from QAL. As shown in Table 1, QTAL achieved a relative 70-80% reduction from our baseline TAL. The perplexity reductions for PPL_{uniq} are in a similar range.

# queries	1k	3K	7K	10K	14K
QAL	1534	1414	1104	1054	888
QTAL-3best	409	314	229	217	190
relative reduction %	73.4	77.8	79.3	80.4	78.5

Table 1. PPL scores of QAL, QTAL-3best (the best LM) and the relative PPL reductions with different amounts of transcribed queries

6. CONCLUSIONS AND FUTURE WORK

This paper introduced a translation approach to language modeling for voice search. Specifically we used an n-gram based translation model to predict query forms from listing forms, and we added the query forms into the training data for language modeling. We obtained very promising results when evaluating LM perplexities on transcribed voice search queries. For future work, we would like to evaluate the impact of such an approach on recognition performance, and to exploit other features such as word class and part-of-speech tag in the translation model.

The authors would like to thank Hermann Ney, Xiaodong He, Patrick Nguyen and Mei Yang for useful discussions.

7. REFERENCES

[1] D. Miller, "Speech-enabled mobile search marches on," *Speech Technology Magazine*, July 2007.

[2] P. Natarajan, R. Prasad, R. Schwartz, and J. Makhoul, "A scalable architecture for directory assistance automation," in *Proc. ICASSP*, Orlando, U.S., 2002.

[3] D. Yu, Y.-C. Ju, Y.-Y. Wang, G. Zweig, and A. Acero, "Automated directory assistance system - from theory to practice," in *Proc. Interspeech*, Antwerp, Belgium, 2007.

[4] O. Scharenborg, J. Sturm, and L. Boves, "Business listings in automatic directory assistance," in *Proc. Eurospeech*, Aalborg, Denmark, 2001.

[5] Zweig et. al., "The voice-rate dialog system for consumer ratings," in *Proc. Interspeech*, Antwerp, Belgium, 2007.

[6] A. Berger and J. Lafferty, "Information retrieval as statistical translation," in *Proc. SIGIR*, 1999.

[7] J. M. Crego, A. de Gispert, and J. B. Marino, "The TALP ngram-based SMT system for IWSLT'05," in *Proc. International Workshop Spoken Language Translation*, Pittsburgh, 2005.

[8] M. Bisani and H. Ney, "Investigations on joint multigram models for grapheme-to-phoneme conversion," in *Proc. ICSLP*, Denver, U.S., 2002.

[9] X. Li, A. Gunawardana, and A. Acero, "Adapting grapheme-to-phoneme conversion for name recognition," in *Proc. ASRU*, Kyoto, Japan, 2007.

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, second edition, 2003.

[11] Adam L. Berger, S. Della Pietra, and V. J. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[12] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, 1988.