JAPANESE AND KOREAN VOICE SEARCH

Mike Schuster and Kaisuke Nakajima

Google Inc., USA

{schuster, kaisuke}@google.com

ABSTRACT

This paper describes challenges and solutions for building a successful voice search system as applied to Japanese and Korean at Google. We describe the techniques used to deal with an infinite vocabulary, how modeling completely in the written domain for language model and dictionary can avoid some system complexity, and how we built dictionaries, language and acoustic models in this framework. We show how to deal with the difficulty of scoring results for multiple script languages because of ambiguities. The development of voice search for these languages led to a significant simplification of the original process to build a system for any new language which in in parts became our default process for internationalization of voice search.

Index Terms— Speech recognition, voice search, Japanese, Korean

1. INTRODUCTION

Voice search, the ability to perform web searches simply by speaking them to your mobile phone, first appeared around 2008 on iPhone and Android phones in US English [1]. Soon after it was obvious that developing voice search for other languages, especially ones where keyboard input is more difficult, provided even greater benefits than for English.

While speech recognition systems are very similar across languages in terms of basic technology, especially many Asian languages pose new problems that weren't easily solved by porting traditional techniques available from English. Many Asian languages have vastly larger basic character inventories (Japanese, Korean, Chinese among others), some have several alphabets mixed (Hiragana, Katakana, Kanji and ASCII for Japanese, Hangul and ASCII for Korean), often even in a single query or sentence. This complicates generation of the pronunciation dictionary and adds confusion during decoding because of the large number of homonyms. Some parts of the basic character sets exist in multiple formats (multiple width etc.) and numbers can often be written in multiple ways, which requires appropriate normalization in some cases.

The concept of spaces between words exists not at all or only partially depending on the language (traditionally Japanese and Chinese do not have spaces between characters but Korean has some), but depends in practice on the application as well (search keywords typed into search engines in Japanese or Korean are usually but not always separated by spaces, and ASCII keywords are *always* separated). Segmenters have to be used to come up with basic word units that can be used in the dictionary and language model and spaces between these units may have to be added or removed for display after decoding. We developed a segmenter described in section 3 that is purely data driven and can be used for any language without modification. Additional problems were how to deal with the many words of English origin as well as URLs, numbers, dates, names, E-Mail addresses, abbreviations, punctuation and other special cases which mostly exist in English as well but became even more complicated with the mixture of multiple character sets.

At Google the Japanese voice search system was among the first developed after US English and led to a significant simplification and speed-up of the process to deal with new languages in general – in large parts the process described below became the standard to build and maintain any new language and was first applied to Korean without any modifications shortly after Japanese was launched in November 2009. Japanese and Korean are in terms of usage currently (9/2011) our largest international languages after US English. Voice search for Mandarin Chinese and Cantonese as launched by Google in 2009 and 2010 are described in [2] and [3].

2. ACOUSTIC DATA COLLECTION

Acoustic data is necessary to build initial acoustic models and test sets to measure performance. Publicly available databases in Japanese/Korean were either severely restricted to be used for commercial applications or the domain of the data was too different from a voice search application to be used over the data channel of a smartphone. We decided to collect data ourselves.

After several attempts with a voice recorder, Blackberry as well as iPhone-based data collection applications we converged eventually on the client/server architecture for data collection described in [4] using paid speakers.

It took a few weeks to collect 250k read utterances with 500 speakers of all ages, dialects, regions and for as many real noise conditions as possible, which was contracted out to local companies. To match the domain we used randomly sampled queries out of the top N queries from anonymized logs from the local Google search domains (google.co.jp, google.co.kr).

A decision that we made early on was to leave all data as much as possible in the original written domain and model it there as well (as opposed to converting it to the spoken domain for modeling and back to the written domain for display as we had done this for English). This had the advantage that it was not necessary to manually transcribe the collected data or to convert the language model data – we could use it as is. However, this meant also that long sequences of Japanese/Korean characters forming several words, numbers, URLs, E-Mail addresses etc. were now left in their written original form and somehow had to be modeled in the language model (LM) and recognition process efficiently as discussed in the next sections.

3. SEGMENTATION AND WORD INVENTORY

As mentioned above, many Asian languages have often no or few spaces between words. Since the whole speech recognition system is based on having some kind of word inventory the original Japanese

text has be segmented. One choice is to use the Unicode characters themselves but in experiments we found that for Japanese there are too many pronunciations per character on average to be able to run an efficient and accurate enough search. Other segmenters we tried had problems segmenting arbitrary sequences of characters as often found in web text data (resulting in many out-of-vocabulary words (OOVs)) and also were often language-dependent and complex because they typically use semantic information as well – we wanted to use a technique that depends only on the data and preferably does not produce any OOVs. We developed a technique (WordPieceModel) to learn word units from large amounts of text automatically and incrementally by running a greedy algorithm as described below. This gives us a user-specified number of word units (we often use 200k) which are chosen in a greedy way without focusing on semantics to maximize likelihood on the language model training data – incidentally the same metric that we use during decoding. The algorithm to find the automatically learned word inventory efficiently works in summary as follows:

1. Initialize the word unit inventory with the basic Unicode characters (Kanji, Hiragana, Katakana for Japanese, Hangul for Korean) and including all ASCII, ending up with about 22000 total for Japanese and 11000 for Korean.
2. Build a language model on the training data using the inventory from 1.
3. Generate a new word unit by combining two units out of the current word inventory to increment the word unit inventory by one. Choose the new word unit out of all possible ones that increases the likelihood on the training data the most when added to the model.
4. Goto 2 until a predefined limit of word units is reached or the likelihood increase falls below a certain threshold.

Training of the segmenter is a computationally expensive procedure if done brute-force as for each iteration (addition of a new word piece unit by combining two existing ones), as all possible pair combinations need to be tested and a new language model needs to be built. This would be of computational complexity $O(K^2)$ per iteration with K being the number of current word units. The training algorithm can be sped up significantly by a) testing only the pairs that actually exist in the training data, b) testing only pairs, with a significant chance of being the best (for example high priors), c) combining several clustering steps into a single iteration which is possible for groups of pairs that don't affect each other and d) to only modify the language model counts for the affected entries. Using these greedy speed-ups we can build a 200k word piece inventory out of a frequency-weighted query list in a few hours on a single machine.

This inventory is then used for language modeling, dictionary and decoding. As the segmentation algorithm builds an inverse binary tree of pairs starting from the basic character symbols the segmentation itself does not need any Dynamic Programming or other search procedures and is hence computationally very efficient – isolating the basic characters and combining them as going down the tree will give a deterministic segmentation in linear time with respect to the length of the sentence. Since the word inventory is limited to 200k units that can model every possible character sequence and does not produce any OOVs it simplifies somewhat the generation of the dictionary – in our case roughly only 4% of the words have more than one pronunciation. As described in more detail below we found that adding too many pronunciations hurts performance, likely because of the increased number of similar hypotheses during the alignment for training and decoding.

While normal Japanese text does not have spaces between words there are cases where some spaces are used in other languages (Korean for example), and also sometimes in Japanese. For example search keywords are normally separated by spaces ("京都ラーメン") as are words with ASCII characters (English words, abbreviations, numbers etc.) when embedded in Japanese text. Initially we ran the production system without the ability to glue word pieces together again – this didn't matter for popular words and short queries as those became complete units but was annoying for longer and rare queries as spaces appeared where they shouldn't have been. Initially we experimented with a bigram-based technique run on the hypotheses to decide where to put spaces but abandoned this after finding that it is much more efficient and elegant to deal with this problem right during decoding. To be able to learn where to put spaces from the data and make it part of the decoding strategy we use the following technique:

1. The original language model data is used "as written", meaning some data with, some without spaces.
2. When segmenting the LM data with WordPieceModel attach a space marker (we use an underscore or a tilde) before and after each unit when you see a space, otherwise don't attach. Each word unit can hence appear in four different forms, with underscore on both sides (there was a space originally on both sides in the data), with a space marker only on the left or only on the right, and without space markers at all. The word inventory is now larger, possibly up to four times the original size if every combination exists.
3. Build LM and dictionary based on this new inventory.
4. During decoding the best path according to the model will be chosen, which preserves where to put spaces and where not. The attached space markers obviously have to be filtered out from the decoding output for correct display. Assuming the common scenario where the decoder puts automatically a space between each unit in the output string this procedure then is:
   (a) Remove all spaces (" " -> "")
   (b) Replace double space marker by space ("__" -> " ")
   (c) Remove remaining single space markers ("_" -> "")

   The last step could potentially be also a replacement by space as this is the rare case when the decoder hypothesizes a word unit with space followed by one without space or vice versa.

Below we show an example of the segmentation and gluing procedure (note that original text and final result contain a space): Japanese original unsegmented text (1), after segmentation (2), the addition of underscores to retain location of spaces (3), a possible raw decoder result (4) with the final result after removing underscores (5):

1. " 京都 清水寺の写真" (original text)
2. " 京都 清水寺 の写真" (after segmentation)
3. " __京都__ __清水寺 の写真__" (after addition of underscores, used for LM training, dictionary etc.)
4. " __京都__ __清水寺 の写真__" (decoder result)
5. " 京都 清水寺の写真" (displayed output)

The data-driven, language-independent techniques described above with a minimum of rule-based processing are successfully used for some of our Asian languages (Japanese, Korean, Mandarin) across multiple applications, Voice Search, dictation as well as YouTube caption generation.

Test Set	Nunits	Nwords	Perplexity	Cost/Sent.
rand250k	50k	1423357	341.08	33.40
rand250k	100k	1244490	732.81	33.03
rand250k	200k	1148840	1251.66	32.96
top10k	50k	337611	55.67	13.58
top10k	100k	278800	123.92	13.45
top10k	200k	237835	281.35	13.42

Table 1. Number of words, word perplexity and average cost/sentence for several sizes of segmenter inventory.

Pronunciations added	Accuracy
0	68.2
100	67.8
200	67.7
500	67.4
750	67.4
1000	67.2

Table 2. Number of pronunciations added for names and resulting sentence accuracy on a Japanese test set.

4. LANGUAGE MODELING

Making the assumption that people say what they type we used local web queries from anonymized logs as our LM data. Once the word inventory is decided it is straight-forward to estimate LMs, in our case usually entropy-pruned 3- to 5-grams with Katz back-off after removing unwanted symbols etc. as much as possible similar to what is described in [5]. For Japanese and Korean we made no distinction between native and English loan words (about 25% of the dictionary for Japanese and 40% for Korean are ASCII words, most of them English) – we simply treated all of them as words of the native vocabulary modeled with the native phoneme set.

We used a year of data to cover all seasons and holidays as usage of some words peaks only around certain times. Typical perplexities for several sizes of the basic word inventory (50k, 100k, 200k) for the final Japanese LMs are shown in table 1, here for a 250k random set and for the top 10k queries. As the word definition is different for each inventory size the word perplexities cannot be compared between these, but average cost/sentence (log-likelihood we use during decoding) can be used.

5. DICTIONARY

Good quality pronunciation dictionaries are relatively easy to obtain for frequent English words but there is very little available for languages like Japanese or Korean. Pronunciation of Japanese character sequences is in general quite ambiguous as most Kanji have at least two readings and also we found that many high-frequency words as used in web queries are often hard to pronounce using automatic tools (for example popular Japanese pop groups “AKB48”, “W-INDS”, “Kis-My-Ft2” or mixed character words like “シャ乱Q” (three character sets mixed in a single word!) or “価格.com”). Japanese place names and family names are even for native Japanese speakers sometimes hard to pronounce. Native Korean (Hangul) has a more regular structure than Japanese but for both languages abbreviations, TV shows, radio stations, product numbers etc. are often difficult to pronounce automatically.

Building dictionaries for a large number of words (in our case 200k word units consisting of sequences of characters of the Japanese/Korean character sets including ASCII etc. as discussed above) is always a semi-automatic process requiring at least some human intervention to end up with a high-quality dictionary which is essential for good recognition rates. We used the following process: Our Japanese/Korean dictionary is based on 33/32 phonemes, respectively. We found that the selection of the phonemes is not critical as we had to add and remove phones a few times during the development process which did not make a significant difference in recognition rates. After generating an initial dictionary using various techniques and sources, among them internal IME (Input Method Editor) data (partially using MeCab [6] and “Google Japanese Input” which is now a publicly available tool [7]), extraction of readings from the web, an internal transliterator for ASCII words as well as simple rule-based approaches we reviewed by hand the for speech recognition most important groups of pronunciations: a) the top 10k words as occurring in the LM data, b) all single character (Kanji) words, c) all words with single phoneme pronunciations, d) all ASCII words with 1, 2, or 3 letters (many of those being abbreviations). Frequent numbers, ordinals, dates, alpha-digit combinations etc. were also reviewed carefully. Since our vocabulary is closed (any possible Japanese/Korean word can be represented by the units in the dictionary if the pronunciation matches) we only have to do this once (with possible fixes later on).

While trying to increase pronunciation coverage we found that adding pronunciations can be beneficial up to a point but adding very rare pronunciations will decrease average recognition rate. It is essential to measure the impact of any substantial changes to the dictionary, sometimes even requiring acoustic retraining. Especially adding pronunciations for high-frequency short words (single Kanji and short ASCII words) can lower recognition rates substantially. As a rule of thumb we found that pronunciations should only be added if they are substantially different from the ones already available and not too infrequent. Table 2 shows typical effects on accuracy when explicitly adding more pronunciations for Japanese names (one experiment out of many to increase pronunciation coverage for certain groups of words in the dictionary).

6. ACOUSTIC MODELING AND QUALITY MEASURES

The acoustic models are standard 3-state HMMs with decision-tree clustered context-dependent states representing Gaussian mixtures with a variable number of diagonal Gaussians. Feature vectors are 39-dimensional consisting of PLP cepstral coefficients with cepstral mean subtraction and their 1st and 2nd order derivatives. Linear Discriminant Analysis (LDA) and Semi-Tied Covariance modeling (STC) [8] is used in the later training iterations. The models are gender-independent Maximum-Likelihood trained followed by some boosted MMI (Maximum-Mutual-Information) iterations [9]. Decoding is a standard Finite-State-Transducer (FST)-based [10] time-synchronous beam search.

We used 250k utterances collected as described above to build the initial models, which had a total of about 100k Gaussians.

To measure quality we used *WebScore*, a technique developed by us [1] to give an estimate of whether the user found the correct web page given a spoken query. To calculate *WebScore@N* (WSC@N) we test if the search URL of the top hypothesis is in the first N search URLs of the reference – in this case it is reasonable to assume that the correct web page was found (given the search engine). WebScore@1 therefore simply tests if the top hypothesis URL and the top reference URL are equal. This measure had two advantages over regular word error rate: 1) As our word definition is defined by the

Time	Test Set	WSC@1	Training
before launch	initial (read)	73.3	250k (read)
before launch	internal (real)	63.8	250k (read)
before launch	internal (real)	69.8	250k (read)
launch	internal (real)	66.0	250k (read)
after launch	internal (real)	69.2	1.2M (prod)
after launch	prod A (real)	71.7	3M (prod)
after launch	prod B (real)	70.1	6.6M (prod)

Table 3. Typical time-line of results depending on test set and amount/type of training data, here for Japanese voice search for the first few months.

segmenter a change in word inventory will lead to different word error rates (smaller inventory results in smaller word error rate (WER)) and can therefore not be used to compare against other languages or other word inventories for the same language. 2) Many queries in Japanese/Korean can be written in multiple ways (different scripts, with or without spaces etc.) depending on context or user preference but mean the exact same thing. For example "任天堂", "ニンテンドウ", "にんてんどう", "nintendo" are all valid ways to search for the Japanese game company (all are pronounced the same and have the same meaning) – the result URL should be the same for any of these. WebScore takes care of these ambiguities (the reference will contain only one of the above) and hence gives a better estimate for the true error rate than for example sentence error rate.

7. MAINTENANCE AND UPDATES

After initial models are launched it is critical to retrain acoustic models with production data to adapt to new acoustic conditions, to retrain language models to include recent terms and to fix problems in the dictionary as they are noticed. We did this several times for Japanese/Korean using first a test set from the initial data collection and later new test sets drawn from anonymized production data. As transcribing acoustic data is a very time-extensive and expensive process we used data without human-transcribed results, simply selected by using data above a certain confidence threshold (unsupervised data), for retraining the acoustic models. Although this data contains recognition errors (we found less than 5% for Japanese) the results were very promising as they improved final recognition rates and therefore user satisfaction without a lengthy transcription process. Table 3 shows a typical progression for quality from initial launch with small models trained on 250k read data to training on $>$6M utterances from anonymized production logs months later and models with 300k Gaussians, obviously with many improvements on acoustic model, language model and dictionary on the way. Note that recognition rates are measured on different test sets starting from an initial read data test set to test sets drawn from production data, which are much harder to recognize.

8. CONCLUSIONS AND DISCUSSION

We described how we built Google's Japanese and Korean voice search systems. While developing these systems we found that the simplified process described above based on modeling in the written domain with a closed dictionary but infinite vocabulary based on the WordPieceModel segmenter resulted in relatively low-complexity systems to maintain and update. Using unsupervised data for initial training and subsequent updates of the acoustic models has cut down our development time for new languages significantly. Many of the techniques described in this paper may seem obvious to use now but weren't at the time of development.

Over time we found that while for most spoken queries and text the techniques described above work well and on average quality of the system improves with every update, there are still some problematic corner cases which can only be addressed by adding more complexity to the system. For example rare (longer) numbers are often not recognized correctly because the combined pronunciations of their segmentation will not match the actual pronunciation (example in English: "1234567" said as "one million two hundred thirty four thousand five hundred and sixty seven" – the pronunciations of a possible segmentation of "123 4567" will likely not cover the original pronunciation). In general, similar is true for any character sequences whose pronunciation depends on a wider context (think prices ("$104" -> "one hundred and four dollars"), dates etc. – any language has a significant number of high-frequency exceptions like this). However, overall these cases are relatively rare for the average user. We are committed to improving recognition for the corner cases as well in the near future.

9. ACKNOWLEDGEMENTS

We want to thank Martin Jansche for providing tools to generate pronunciations for parts of early versions of the dictionaries and the rest of the Google speech group for providing the training tools as well as many fruitful discussions and inputs.

10. REFERENCES

[1] Johan Schalkwyk, Doug Beeferman, Francoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Garrett, and Brian Strope, "Google Search by Voice: A Case Study," in *Advances in Speech Recognition: Mobile Environments, Call Centers, and Clinics*, Amy Neustein, Ed. Springer-Verlag, 2010.

[2] Jiulong Shan, Genqing Wu, Zhihong Hu, Xiliu Tang, Martin Jansche, and Pedro Moreno, "Search by Voice in Mandarin Chinese," in *Proceedings of Interspeech*, 2010.

[3] Yun-Hsuan Sung, Martin Jansche, and Pedro Moreno, "Deploying Search by Voice in Cantonese," in *Proceedings of Interspeech*, 2011.

[4] Thad Hughes, Kaisuke Nakajima, Linne Ha, Atul Vasu, Pedro Moreno, and Mike LeBeau, "Building transcribed speech corpora quickly and cheaply for many languages," in *Proceedings of Interspeech*, 2010.

[5] Ciprian Chelba, Johan Schalkwyk, Thorsten Brants, Vida Ha, Boulos Harb, Will Neveitt, Carolina Parada, and Peng Xu, "Query language model for voice search," in *Proceedings of the 2010 IEEE Workshop on Spoken Language Technology*, 2010.

[6] Mecab, "http://mecab.sourceforge.net," .

[7] "http://tools.google.com/dlpage/japaneseinput/eula.html," .

[8] Mark Gales, "Semi-tied covariance matrices for Hidden Markov Models," in *IEEE Transactions on Speech and Audio Processing*, 1999.

[9] Daniel Povey, Dimitri Kanevsky, Brian Kingsbury, Bhuvana Ramabhadran, George Saon, and Karthik Visweswariah, "Boosted MMI for model and feature space discriminative training," in *ICASSP*, 2008.

[10] OpenFST, "http://www.openfst.org," .