interspeech 2008 incorporating SST 08

A Language-Modeling Approach to Inverse Text Normalization and Data Cleanup for Multimodal Voice Search Applications

Yun-Cheng Ju, Julian Odell

Microsoft Corporation, Redmond WA, USA

{yuncj, juliano}@microsoft.com

Abstract

In this paper we address two related challenges in multimodal local search applications on mobile devices: first, correctly displaying the business names, and second, harvesting language model training data from an inconsistently labeled corpus. We investigate the impact of common text normalization and the quality of language model training corpus on the accuracy of displayed results. We propose a new language model framework that eliminates the need for explicit inverse text normalization. The same framework can be applied to sift through corrupted language model training data. Our new language model is 25% more accurate while 25% smaller in size.

Index Terms: text normalization, inverse text normalization, language model, multimodal, voice search, transduction, language resources.

1. Introduction

Text Normalization (TN) has become a common practice in the development of various applications with a voice user interface, such as *automated directory assistance* (ADA) [1,2]. While this process [3,4] (e.g. "*Kwik Kopy*" -> "*Quick Copy*", "*4X4*" -> "*Four by Four*") improves speech recognition efficiency and accuracy, as well as the recall of the search, it poses challenges on *Inverse Text Normalization (ITN)*, or *Pretty Print*, in multimodal voice search applications[5] where recognized utterances have to be displayed in original business names for users to verify. For example, if the word "Rite" is text-normalized (TNed) to "Right", then the task of ITN is to convert the recognized phrase "Right Aid" back to "Rite Aid". Without correct ITN, the text normalized name "*Right Aid*" will be displayed and create a confusing user experience of not knowing if the incorrect display form is really what the user wanted.

The adoption of *Statistical Language Models (SLM)* in voice search [5,6,8,9] has shown great potential in adding robustness and in improving task completion rates. Unfortunately, the data available (source feed, web queries, human transcribed utterances, caller confirmed recognition results, etc,) is never "clean", that is, the data is either inconsistently labeled, incompatible, or irrelevant to the task. It requires substantial development efforts to clean up the data, and thus limits this approach from reaching its potential.

In this paper, we present a novel ITN framework to prevent context information from being lost in TN so as to preserve original business names. We then show that ITN can also be used to clean up a corrupted corpus to achieve better system performance.

This paper is organized in three parts. Section 2 describes the *TN* and *ITN* process in our voice search application. We compares the baseline unigram-based ITN with our proposed new approach using longer context and then shows a novel **implicit ITN** LM architecture that uses the unified LM to eliminate the need for a decoder after speech recognition. Section 3 discusses why salvaging a corrupted corpus can be viewed as an ITN task, and demonstrates that our new implicit ITN framework provides a principled approach to this problem. Finally, section 4 reports some findings, challenges the common belief of the benefit of traditional TN, suggests effective remedies we learned from our experiments, and concludes with the discussion.

2. Text normalization (TN) and inverse text normalization (ITN)

2.1. Text normalization (TN)

Text Normalization determines the word units in the speech recognition and/or synthesis systems. It is the mostly used procedure that speech application developers take to control speech recognition performance.

For ADA and other voice search applications, three types of normalization are usually applied:

- Symbols & Digits (e.g. "#" -> "*Number*", "24/7" -> "*Twenty Four Seven*") - as defined by rules,
- Homonyms & Abbreviations Replacement (e.g. "*G8T*" -> "*Great*", both "*Lowe's*" and "*Loews*" -> "*Lows*") - as defined in a dictionary, and
- Word Breaking (e.g. "*Accuvision*" -> "*Accu Vision*") – as defined using a dictionary and common pre- and post-fixes.

Using more popular words produces a more consistent and smaller set of vocabulary and practically eliminates deficiency of the *letter-to-sound (LTS)* module from guessing the pronunciations for unseen words. In addition, in speech user interface, it also saves users from answering questions like "Are you looking for *Alan* or *Allen*?" in situations where they can't see the different spellings.

For telephony-based directory assistance [1,6] and voice search[5], all three TN categories resulted in significant improvements in both CPU resources and SR accuracy.

2.2. Inverse text normalization (ITN)

For our application, we noticed that roughly 30% of business names (e.g. 55,515 out of the 181,438 Seattle based business listings) were modified by our TN procedure. Since we decided to display the recognition results on mobile devices for users to confirm before sending them to a separate local search engine, it quickly became obvious that we needed an ITN module to recover the original names. While users or search engines might tolerate some incorrect word breakings (e.g. "*Outback*" vs. "*Out Back*"), they may not accept homonyms like "*Right Aid Pharmacy*" or "*Lows Hardware*".

Accepted after peer review of full paper

A straightforward implementation based on unigram popularity was examined but proven to be of insufficient accuracy. As an example, based on word frequency, all the words in the "Less Popular" column of Table 1 were "mis-ITNed" to the "More Popular" form. Overall, this approach made mistakes more than 23% (12,798) of the time (Table 2). This motivated us to conduct further investigation and to use more context information to improve ITN accuracy.

Less Popular	More Popular	TN rule
Right(55)	Rite(109)	Right
Allan(32), Alan(110)	Allen(130)	Allan
Loews(11)	Lowe's(34)	Lows
Arc(12)	Ark(15)	Ark
Kraft(18)	Craft(74)	Craft
Total Mistakes (238)	Correct(362)	

Table 1: Sample TN distribution & Base Line ITN

2.3. ITN using N-Gram language model

We formulated the ITN task in Equation (1) as a typical optimization problem similar to the one performed by an automatic speech recognition (ASR) system

$$\hat{w} = \arg\max_{w} p(w \mid s) = \arg\max_{w} p(s \mid w)\, p(w) \quad (1)$$

where s is the recognized text (spoken form), and w is the word sequence hypothesis in display form. $p(s|w)$ is the TN rule (implemented as a dictionary), and $p(w)$ is the language model (LM) probability. Since w is the word in the display form (i.e. *pretty print*), this framework calls for two separate language models: one for speech recognition and the other for ITN. We used standard trigram to estimate $p(w)$.

The new approach performed much better than the baseline approach and made mistakes on only 3% of all business listings (Table 2). A closer examination found that most of the 1898 cases were not actually true mistakes, but that the data feed of the listings was not consistent. For example, among the 161 listings of the *KeyBank* branches, 60 were listed as "*Key Bank*" (as two words) and 101 were listed as "*KeyBank*" (one word). Our new ITN converted all of them to the more popular form, "*KeyBank*". We examined the 600 sample cases listed in Table 1 and found only three mistakes as compared to 238 mistakes in the previous approach. All three were due to very similar contexts in the listings.

Description	Number	(%)
Total number of Entries	181,438	100
Entries modified by TN	55,515	30.6
Unigram ITN Mistakes	12,798	23.1
LM ITN Mistakes	1,898	3.4

Table 2: TN & ITN Performance Statistics

2.4. Implicit ITN architecture with unified LM

While our new approach successfully addresses the issue of ITN, maintaining and accommodating two separate language models and decoders requires unnecessary burden and makes this approach less appealing. In this section, we present a new infrastructure that merges the two LMs and eliminates the need for a second ITN decoder.

We chose to keep the ITN LM because it captures context necessary for ITN disambiguation. The word unit is chosen as the common denominator of the TN and ITN rules. For example, since "*Air Lines*" and "*Blockbuster*" are TNed into "*Airlines*" and "*Block Buster*", we use "[*Air Lines*]" and "[*Blockbuster*]" as word units in our new LM framework to indicate they need additional TN pronunciations for the purpose of speech recognition. In our TN process, we apply the TN rules for each listing and then align the spoken form with the original display form. A few sample listings with their corresponding LM training sentences and spoken forms are shown in Table 3.

Listing	LM Training Sentences	Spoken Form
Delta Air Lines	Delta [Air Lines]	Delta Airlines
United Airlines	United Airlines	United Airlines
Blockbuster Video	[Blockbuster] Video	Block Buster Video

Table 3: Sample LM training Sentences

Figure 1 shows an example bigram LM in the form of a finite state machine CFG. We augmented our N-Gram CFG tool [7] to support the TN pronunciations for these new word units in square brackets using grammar sub-rules. In addition, the display form of the special tokens was embedded as the name of sub-rules so we could use the parse tree to recover the pretty print from the recognition result. In this grammar example, it was clear that the contexts for the words "Rite" and "Right" were preserved.

This unified architecture provides the same ITN results as the previous two-LM approach that requires two language models, provided that the LM training and decoding settings are the same. However, we do need to maintain additional ITN contexts and embed pronunciations as sub-rule and that incurs additional computation. Compared with the original CFG w/o ITN capability, the new CFG is roughly 20% larger in size and almost 40% more expensive in recognition CPU time. Nevertheless, the unified LM architecture is still much more affordable than the two-LM approach and is preferred by our product group for its simpler deployment story w/o the need of a second ITN LM and decoder.

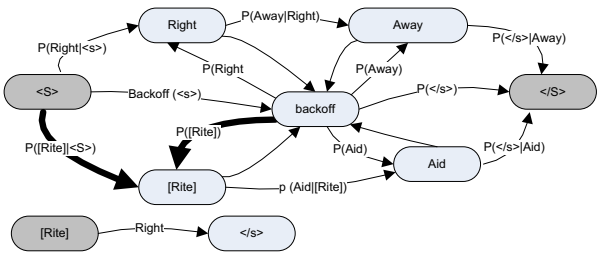

Figure 1: Bigram CFG for "[Rite] Aid" and Right Away". Thick links are TN sub-rule references. Links w/o label are unigram back offs

3. Cleanup of a corrupted training corpus

In our application, statistical Language Models need training data to provide synonyms people often use when referring to business listings as well as the popularity of each listing. Data comes in different forms from different sources and with one thing in common: it is either inconsistently

labeled or transcribed. Since both the quantity and quality of the data are important, there is a great need to find an automatic framework for cleaning up the corpus as much as possible.

In this section, we first argue that cleaning up corrupted LM training corpus can be treated as an ITN process. We then provide quantitative analysis of the improvements in both accuracy and CPU resources possible using our proposed approach.

3.1. Data cleaning using ITN

Knowing how and where the data was collected helps us understand the common problems in a corrupted corpus. Most of the data came from system logs and were engine-recognized utterances implicitly confirmed by callers. Different systems or even the same system from different versions have different TN rules, bugs, or other limitations that created the inconsistency (e.g. "*Rite Aid*" might be left untouched, or "*Rite*" might be TNed to "*Right*" in some versions). Other data, either transcribed by operators or harvested from web queries, is subject to human errors in typos, word breakings, homonyms, and styles (e.g. "*PF Chang's*", "*P F Changs*", "*P-F Chang*"). We also found some completely irrelevant data that should have been rejected.

No matter where the data is from, we see great similarity of the task of data cleanup (Equation 2) and the task of ITN in Equation (1), with the exception that ***d*** is the display form of *w* in the corrupted corpus.

$$\hat{w} = \arg\max_{w} p(w \mid d) = \arg\max_{w} p(d \mid w)\, p(w) \qquad (2)$$

We used the same word units in both tasks so we wouldn't need any further transformation or intermediate data forms. Both tasks share the same language model *p(w)*, but the new task uses a slightly different dictionary *p(d|w)* which lists the common, or potential, observed inconsistencies (e.g. *Chang's* has been written as *Chang* or *Changs*) for each word unit. In this dictionary, words, whether they require text normalization or not, can have multiple definitions. Sentences failed to be parsed, perhaps due to OOVs, can be examined further to decide whether they should be discarded, or fixed by adding new entries into the dictionary.

3.2. Experimental results

Our subsidiary Tellme has been developing and hosting their telephony-based, end-to-end, directory assistance applications for years [9]. They have collected a significant amount of transcribed calls and accumulated sophisticated TN rules over the years to improve their performance. Their LM training sentences data is of high quality and is very rich in terms of synonym expansion and listing priors. We used the complete set of their data for the Seattle metropolitan area as our "corrupted LM training corpus". The data is considered corrupted, in some senses, to our application because 1) potentially different word units were used; and 2) some of the transcriptions, unfortunately, did not follow their TN guide lines completely. Notice that it is still much cleaner than most data available. However, as we'll show later in this section, we still gained a significant amount of performance improvement after applying our data cleaning algorithm.

We transcribed 8,697 business listing queries nationwide from our multimodal local search application log, and found that 400 were located in the Seattle metropolitan area. In order to acquire a larger test set for both speech recognition and ITN, we decided to include all queries as long as they could be covered by our local listing vocabulary. Unfortunately, this created instances in which, queries in Chicago for "*Sear's Tower*" were included because Both *Sear's* and *Tower* are in our vocabulary, but these cases were far and few. Altogether, we had 7,409 queries.

In table 4, we compare the *strict phrase-level* top 1 and N-best (N<10) accuracy, as well as the grammar sizes for these three configurations: 1) baseline w/ only the business listings in our data feed; 2) baseline + the LM training corpus as it is; and finally, 3) baseline + the cleaned up LM training corpus. For all experiments, we used bigrams w/o pruning or smoothing. We can see even the corrupted corpus gave us a 12% relative gain in the Top1 accuracy, while the cleaned up corpus yielded another 25.3% relative gain. A closer examination found that the cleaned up corpus contained 36% fewer words, reducing the size of our CFG significantly. The clean corpus doesn't contain any new words that aren't found in our baseline vocabulary. The 42% increase in CFG size (20.1MB -> 28.6MB) came completely from the richer synonyms (as in more new bigrams). We believe the big accuracy improvement from cleaning up the corpus is attributed to the more consistent word units and a smaller vocabulary.

Description	Top1	TopN	Size
Baseline (feed)	32.9%	39.7%	20.1MB
w/ Corrupted Corpus	36.7%	50.5%	38.8MB
w/ Clean Corpus	46.0%	55.5%	28.6MB

Table 4: Accuracy & Resource Comparison

4. Discussion

In this section, we share our observations, discuss new issues we have identified, propose pragmatic solutions, and suggest future research work.

4.1. Does multimodal need all TN practices?

While common TN practices are very useful for speech-only systems, we felt that it was uncertain whether we are getting the same amount of benefit for multimodal applications where ITN is needed. In the last experiment, we tried to remove the TN rules gradually to observe the impact on both the accuracy and CPU time. As we can see from Table 5, removing the homonym rules, hyphens, and the breakings from simple compound word altogether didn't affect the accuracy much. On the contrary, the CFG size was smaller and recognition ran 20% faster. We suspect that the LTS module in our SR engine didn't have any trouble pronouncing those words.

Description	Top1	TopN	Size (MB)	CPU (Sec)
All TN	46.0%	55.5%	28.6	55.0
-Homonym	46.0%	55.4%	27.7	52.5
-Hyphen	46.0%	55.4%	27.5	49.4
-Compound word	46.0%	55.5%	26.4	44.4
-&, Letter	45.5%	54.8%	25.7	44.3
No TN	43.6%	52.6%	27.0	157.4

Table 5: Accuracy & Cost of TN configuration

However as the last row shows, completely removing all TN rules, leaving listings like "wwwtopcoat.com" or "A-

24/7" as they are, hurt the accuracy and made the recognition 3 times more expensive. The increase in CFG size suggested that without the help from TN, numerous poor pronunciations were added for these words by the LTS module.

4.2. How much ITN is necessary?

Despite the obvious benefits of TN/ITN in telephony-based applications, overdoing it could be a potential problem. Not only might ITN be useless to users and search engines beyond a certain point, but, as mentioned earlier, the feed itself is not clean and consistent (e.g. "*Keybank*" vs. "*Key Bank*" and "*Center*" vs. "*Ctr*."), so attempting to preserve and/or present all forms may actually pose difficulties for users.

We explored a pragmatic approach to this problem by collecting a *synonym* list. When we processed the feed, we ignored variations in the display form within the same set (for example, "*Mc Donalds*," "*Mc-Donald's*," "*McDonald's*," etc.) and replaced them with the most popular form, resulting in a cleaner vocabulary for the following steps. The same list was also used in cleaning up the LM training corpus.

Currently, this step is largely a manual process and new entries can be easily added in as required. In the mean time, we are investigating a more automatic approach by examining the navigational queries from the web search and from the click-through data.

4.3. Redundancy in the N-best alternatives

A related issue was the quality of the N-best list presented by our integrated ITN LM. Usability studies indicated that there were many redundant or useless variations even after we had the previous issue ("*Macy's*" vs. "*Macys*") addressed. Since we preserved the display form for everybody, there were homonyms in the word units and most of them showed up in the recognition N-best list (e.g. "*In-**N**-Out*", "*In **'N** Out*", "*In **and** Out*", "*In **&** Out*", …etc). Fortunately, they appeared in the correct order.

The two-LM approach described in section 3.1 guarantees that the recognition N-best list contain no homonym phrases, and that the ITN LM only return a single best display form for each phrase. However, in the unified ITN framework, not much can be done at the grammar level to prohibit competing homonyms or similar phrases from appearing on the list. A post-processing step can be taken to discard the N-best entry of a display form if it contains the same words as the spoken form. Another pragmatic approach, if proven affordable, would be to take a post-processing step to send the N-best list to a search engine or to a listing look up table, and to only show the lower alternatives if they generate different final search results.

4.4. ITN needs more training data

Our ITN uses LM context to disambiguate homonyms both for displaying recognition results and for cleaning additional training data. However, having only the business listings alone is sometimes insufficient. For example, as shown in Table 1, both "*Lowe's*" and "*Loews*" are normalized to "*Lows*". The business listing data feed only includes instances like "*Lowe's*", "*Lowe's Home Improvement*", "*Lowe's Hardware*" and "*Loews Cineplex*". Our ITN could not disambiguate context words like *Cinema* and *Movie Theater*, from *Building Supply* and *Construction Supply* in the data corpus and speech queries. As a result, it fell back to use unigram popularity, making the same mistakes that our baseline ITN makes.

Fortunately, just a few additional training sentences is enough to completely address this issue. We have been investigating the use of translation models [10] to expand the synonym training set. The expanded synonyms can provide adequate contexts to train the ITN LM more effectively.

5. Conclusions

Multimodal applications that display recognition results to users require ITN for *pretty print,* which creates new challenges for common TN practices. We proposed a unified and self-contained LM framework, which uses the common denominator of the display form and the spoken form of the listings as word units. The LM captures the context for disambiguation, while the CFG topology embeds the display form. We reported significantly improved ITN accuracy in our ADA application using our new approach.

We also showed that this framework provides a principled and systematic approach for harvesting and cleaning data corpus. In our application with very limited training data, the original data yielded a 12% relative gain in accuracy and the cleaned up data yielded a 40% gain.

Finally, while most common TN practices add benefits to speech user interface, not all of them are necessary or beneficial for multimodal applications with the *pretty print* requirement. Our investigations also suggested the need to preserve the display form with discretion.

6. Acknowledgements

The authors would like to thank our colleagues Shawn Chang at Tellme for helping us with TN rules and training corpus, Rob Chambers for developing the SR engine server and ITN plumbing, Oliver Scholz for suggesting the research topic, and Geoffrey Zweig, Xiao Li, Tim Paek, Li Deng, and Dan Bohus for participating in many useful discussions.

7. References

[1] D. Ollason, Y. Ju, S. Bhatia, D. Herron, and J. Liu, "MS Connect: A Fully Featured Auto-attendant: System Design, Implementation and Performance", Proc. InterSpeech, 2845-2848, 2004.

[2] E. Levin and A. M. Mane, "Voice User Interface Design for Automated Directory Assistance", Proc. InterSpeech, 2509-2512, 2005

[3] G. Adda, M. Adda-Decker, J. Gauvain, and L. Lamel, "Text Normalization and Speech Recognition in French", Proc. EuroSpeech, 2711-2714, 1997.

[4] DD. Paul, and J. Baker, "The Design for the Wall Street Journal-based CSR Corpus", Proc. ICSLP, 899-902, 1992.

[5] A. Acero, N. Bernstein, R. Chambers, Y. Ju, J. Odell, P. Nguyen, O. Scholz, and G. Zweig, "Live Search for Mobile: Web Services by Voice on the Cellphone", Proc. ICASSP-2008.

[6] D. Yu, Y. Ju, Y. Wang, G. Zweig, and A. Acero, "Automated Directory Assistance System – from Theory to Practice", Proc. InterSpeech- 2007.

[7] Y. Ju, Y. Wang, and A. Acero, "Call analysis with classification using speech and non-speech features", Proc. InterSpeech-2006.

[8] M. Bacchiani, F. Beaufays, J. Schalkwyk, M. Schuster, and B. Strope, "Deploying GOOG-411: Early Lessons in Data, Measurement, and testing", Proc. ICASSP- 2008.

[9] S. Chang, S. Boyce, K. Hayati, I. Alphonso, and B. Buntschuh, "Modalities and Demographics in voice search: learnings from three case studies", Proc. ICASSP- 2008.

[10] X. Li, Y. Ju, G. Zweig, and A. Acero, "Language Modeling for Voice Search: a Machine Translation Approach", Proc. ICASSP- 2008.