Vocabulary Independent Spoken Query: a Case for Subword Units

Evandro Gouvêa, Tony Ezzat

Mitsubishi Electric Research Labs, Cambridge, MA, USA

egouvea@gmail.com, tonebone@mit.edu

Abstract

In this work, we describe a subword unit approach for information retrieval of items by voice. An algorithm based on the minimum description length (MDL) principle converts an index written in terms of words into an index written in terms of phonetic subword units. A speech recognition engine that uses a language model and pronunciation dictionary built from such an inventory of subword units is completely independent from the information retrieval task. The recognition engine can remain fixed, making this approach ideal for resource constrained systems. In addition, we demonstrate that recall results at higher out of vocabulary (OOV) rates are much superior for the subword unit system. On a music lyrics task at 80% OOV, the subword-based recall is 75.2%, compared to 47.4% for a word system.

Index Terms: information retrieval by voice, subword units, minimum description length

1. Introduction

Information retrieval by voice is becoming increasingly important. With the proliferation of smart-phones, speech becomes the preferred input modality for making queries to search engines, particularly when the queries are long, complex, and require a lot of typing.

A prototypical system for spoken query retrieval is shown in Figure 1. The system contains two main components: an automatic speech recognition (ASR) front-end and an information retrieval (IR) back-end. The ASR front-end decodes an input spoken query into an N-best list of word hypotheses. The N-best list is then submitted to the IR back-end, which retrieves the top-K relevant documents for that query.

Early attempts at building such systems [1] focused mainly on pointing out the robustness these systems exhibited to ASR word error rates. Typically, the language model (LM) used by the ASR is built from the entries in the database to be indexed. If the set of documents in this database changes, the LM has to change. Moreover, new databases may have words not present before. It is therefore necessary to re-prune or re-compress the LMs *whenever the databases to be indexed change*. This is because the novel words introduced by a new database need to be re-inserted into the LMs.

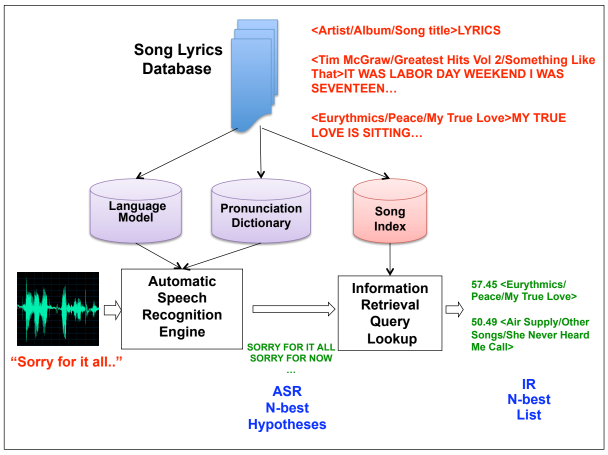
Song Lyrics Database
<Artist/Album/Song title>LYRICS
<Tim McGraw/Greatest Hits Vol 2/Something Like That>IT WAS LABOR DAY WEEKEND I WAS SEVENTEEN…
<Eurythmics/Peace/My True Love>MY TRUE LOVE IS SITTING…
Language Model
Pronunciation Dictionary
Song Index
Automatic Speech Recognition Engine
Information Retrieval Query Lookup
"Sorry for it all.."
SORRY FOR IT ALL
SORRY FOR NOW
…
ASR N-best Hypotheses
57.45 <Eurythmics/ Peace/My True Love>
50.49 <Air Supply/Other Songs/She Never Heard Me Call>
IR N-best List

Figure 1: Overview of an Information Retrieval by Voice for a Song Lyric Task

In our previous work [2], we presented an alternative where we dissociated the text we use to build the pronunciation and language model and the database containing the documents to be indexed. An algorithm, inspired by the Morfessor [3] algorithm and based on the *minimum description length* (MDL) principle, converts a database written in terms of words into a database written in terms of phonetic *subword* units. As a result, *once a subword unit LM is built, it does not need to be recompiled*. Rather, novel databases are simply rewritten in terms of the subword unit inventory. These phonetic subword units are vocabulary independent: if we change the set of documents we want to retrieve, the set of units used by the ASR engine remains the same.

Recent work on subword unit inventory creation methods [4][5][6] have focused primarily on the use of subwords for ASR, not retrieval, and in particular on their ability to handle out-of-vocabulary words (OOVs). In IR tasks, such as spoken term detection [7] and question answering [8], subword units do not have to be reconverted to a word to give it a more human friendly appearance. The IR engine can use the subword units directly.

Here, we extend our previous work by studying the effects of out of vocabulary (OOV) words in the information retrieval task. As a platform for our experiments, the song retrieval task was chosen. In this task, a user retrieves songs by speaking, not singing, portions of a song's lyrics.

HOURGLASS	AW_R + G_L_AE_S
HOUSE	HH_AW_S
HOUSES	HH_AW_S + IH_Z
HOUSES(2)	HH_AW + Z + AH + Z

Table 1: Examples of words rewritten in terms of subwords. Note that some words with alternate pronunciations have multiple subword representations.

In Section 2 we summarize the main points of the MDL algorithm, introduced in [2]. In Section 3 we describe the experimental setup, and in Section 4 we present and discuss the results, concluding in Section 5.

2. MDL Subword Unit Inventory

Our definition of a subword unit may be gleaned from Table 1. A word, *e.g.* HOURGLASS, is rewritten as a sequence of subword units AW_R and G_L_AE_S, where the subword units are sequences of *phonemes*. A subword unit may also span an entire word, as with HOUSE. The subword unit inventory is thus a *flat hybrid* [5] collection of subword units that span portions of words, or entire words. Our algorithm rewrites a database I in terms of a subword unit inventory U given the set of pronunciations Q of words found in I.

The subword unit inventory algorithm utilizes the Minimum Description Length (MDL) principle [3] to search for an inventory of units U which minimize the sum of two terms, $L(Q|U)$ and $L(U)$:

$$\arg\min_{U} \lambda L(Q|U) + (1-\lambda)L(U) \quad (1)$$

where $0 \leq \lambda \leq 1$ is chosen by the user to achieve the desired number of subwords M. $L(Q|U)$, the *Model Prediction Cost*, measures the number of bits needed to represent Q with the current inventory U. $L(U)$, the *Model Representation Cost*, measures the number of bits needed to store the inventory U itself. The MDL principle finds the *smallest model* which also *predicts the training data well.* Smaller models generalize better to unseen data.

The Model Representation Cost is computed over all the units in U from the probability $p(phoneme)$, estimated from the frequency counts of each phoneme in Q:

$$L(U) = \sum_{u \in U} \sum_{phoneme \in u} -\log p(phoneme) \quad (2)$$

The Model Prediction Cost measures the bits needed to represent Q with the current subword segmentation:

$$L(Q|U) = \sum_{q \in Q} \sum_{u \in tokens(q)} -\log p_u \quad (3)$$

Here $tokens(q)$ is a function that maps a pronunciation onto a sequence of subword units. It partitions phones in the pronunciation of a word into subword units in U.

To find the optimal subword inventory U and segmentation $tokens(q)$, we utilize a greedy, top-down, depth-first search algorithm, shown in Figure 2 as pseudocode.

```
Algorithm splitsubwords(node)
Require: node corresponds to an entire word or subword unit
Note: $L(U)$ is the model representation cost, $L(Q|U)$ is the model prediction cost

// FIRST, TRY THE NODE AS A SUBWORD UNIT//
evaluate $L(Q|U)$ using node
evaluate $L(U)$ using node
$bestSolution \leftarrow [L(Q|U) + L(U), node]$

// THEN TRY TWO-WAY SPLITS OF THE NODE //
for all substrings $pre$ and $suf$ such that $pre \circ suf$ = node do
  for subnode in $[pre, suf]$ do
    if subnode is present in the data structure then
      for all nodes $m$ in the subtree rooted at subnode do
        increase count of $m$ count by count of $node$
        increase $L(Q|U)$ if $m$ is a leaf node
    else
      add subnode into the data structure, same count as $node$
      increase $L(Q|U)$
      add contribution of subnode to $L(U)$
  if $L(Q|U) + L(U) <$ score stored in $bestSolution$ then
    $bestSolution \leftarrow [L(Q|U) + L(U), pre, suf]$

// SELECT THE BEST SPLIT OR NO SPLIT //
select the split (or no split) yielding $bestSolution$
update the data structure, $L(Q|U)$, and $L(U)$ accordingly

// PROCEED BY SPLITTING RECURSIVELY //
splitsubwords(pre)
splitsubwords(suf)
```

Figure 2: splitsubwords, a recursive, top-down, greedy, algorithm for inducing the subword unit inventory based on the MDL principle.

A random word is chosen and scanned left-to-right, yielding different prefix-suffix subword splits. For each split candidate, the cumulative cost is computed. The candidate with the lowest cost is selected. Splitting continues recursively until no more gains in overall cost are obtained by splitting a node into smaller parts. After all words have been processed, they are shuffled randomly, and each word is reprocessed. This procedure is repeated until the inventory size M is achieved and a subword unit inventory U is induced, where each unit u has an associated probability p_u.

2.1. Rewriting a Database and LM

Given a novel set of pronunciations Q' from a pronunciation dictionary W', the Viterbi algorithm is used to segment each novel pronunciation into subword units from the inventory U, with smallest cost $\sum_{i=1}^{n} -\log p_{u_i}$.

To rewrite a database I in terms of subword units, the words are scanned sequentially. Each word is mapped to subword unit sequence. If a word has multiple pronunciations, one mapping is chosen randomly. Once a database has been rewritten in terms of subword units, the LM is trained on the rewritten database.

3. Experimental Design

3.1. Dataset Description

The dataset used in this work is the same as the one used by [4]. The song collection consists of 35,868 songs. Each song consists of a song title, artist name, album name, and the song lyrics. A unique ID is created for each song by merging the song title, artist name, and album name. Figure 1 shows examples for several songs.

The test set originates from 1000 songs that were selected randomly from the song database, and divided into groups of 50. Twenty subjects (13 males and 7 females) were instructed to listen to 30-second snippets of 50 songs each, and to utter any portion of the lyrics that they heard. Subjects were also prompted to transcribe their recording, which served as reference transcripts (for calculating phone error rates). The song title was also kept.

The ground truth for the IR experiments is the set of songs with the same title as the query song. The song title as a key addresses the retrieval of covers, as well as songs re-recorded by the same artist. An exception table is used, however, to handle cases when songs have different lyrics but similar titles, *e.g. Angel* by *Jimi Hendrix* or *Dave Matthews Band*. This exception table was built by hand.

In these experiments, we worked with two subsets of the database. The smallest lyric set, `ls2000`, contains 1989 songs that serve as ground truth to the test set utterances. The largest set, `ls36000`, contains all the songs.

3.2. ASR

The prototypical system, shown in Figure 1, comprising of an ASR front-end and an IR back-end, forms the core architecture for experiments.

In this work, the CMU Sphinx-3 ASR system is used to generate the 7-best hypotheses for each spoken query, which are then submitted to the IR back-end for retrieval. The input spoken query is converted into standard MFCC. The acoustic models used by the decoder are triphone HMM, trained from Wall Street Journal data resampled to 8kHz. The word pronunciations are obtained from the CMU dictionary when available, or NIST's addttp (G2P tool) when not. Finally, the LMs are trigrams with Witten-Bell smoothing, built using the CMU SLM toolkit. All of these components are available as open source.

The ASR is evaluated based on the *Phone Error Rate (PER)*, the sum of substitutions, insertions, and deletions made by the ASR engine at the phone level. We used PER because we do not have the references for subwords.

3.3. Information Retrieval

The IR back-end uses a vector space model approach for retrieval. Each song document forms a multidimensional feature vector v. The query also forms a vector q in same feature space. A score $Score(q, v)$ measures the similarity between q and v. The songs with the top 7 scores are submitted for our recall analysis.

After evaluating several different feature spaces and scoring methods, the features used were counts of the unique unigrams, bigrams, and trigrams present in documents and query, which we call *terms*. The scoring method used was $Score(q, v) = \sum_{\forall t} \delta(t)\text{IDF}(t)$, where $t \in \{terms(q) \bigcup terms(v)\}$, $\delta(t)$ is 1 if term t appears in both query and document, 0 otherwise, and $\text{IDF}(t)$ is the inverse document frequency of term t. No document length normalization was performed. Similarly to question answering tasks [9], here the documents are too short to accurately estimate the probability distributions of words. Direct matches between words in the query and in the songs are therefore better measure of similarity than query likelihood.

The baseline system is a *word* system, in which the LM and index are comprised of words as base units. This architecture is compared with a *subword* system, where the LM and the index base units are subwords. The IR accuracy metric is the *k-call-at-n*, where the information need is considered satisfied if *at least k* correct retrievals appear in the top *n*. The *1-call-at-7* measures the percentage of test utterances for which the IR back-end retrieves *at least one* of the ground truth songs in the top 7 results.

3.4. Out of Vocabulary Rates

We simulated a range of OOV rates by pruning the dictionary and language model used by the recognizer or by the MDL algorithm. In the case of words, we built the LM from the set of songs we wanted to index. We simulated an OOV rate by pruning the dictionary based on word frequency computed in the index data. For an OOV rate of $N\%$, we pruned the dictionary so that $N\%$ of the words in the test set are removed, as well as all words less frequent than these. The minimum OOV rate is 5%.

In the case of subwords, we used the pruned dictionary as described above for building the subword unit inventory. We mapped `ls2000` (*cf.* Section 3.5) from words to subwords using this inventory. The mapping from words to subwords is induced by the Viterbi algorithm, as in Section 2.1. `ls2000`, mapped to subwords, was used to create an LM. The subword dictionary trivially maps a subword unit to its constituent phones. The LM and dictionary remained fixed for all recognition experiments regardless of the set of songs to index.

3.5. Subword Unit Inventory Sizes

In our previous work [2], we studied the effect of building the inventory of subword units from different datasets. We concluded that building the inventory from the smallest set was better than from the largest one, even generalizing better. Here, we use the smallest set, `ls2000`, to build inventories of sizes 300, 600, 1200, 2400, and 4800 units. For a given size and OOV rate, we ran recall experiments using indices of different sizes. We built each index by inducing a mapping from words in the songs to subword units. We assumed that it is much less expensive to generate pronunciations than to build an LM for each index. Therefore, at index-build time, we used a full pronunciation dictionary. All words used to build the IR are induced from the inventory built from `ls2000`.

4. Results and Discussion

Figure 3 shows recognition accuracy (in PER) as a function of OOV rate. We show two word-based systems built

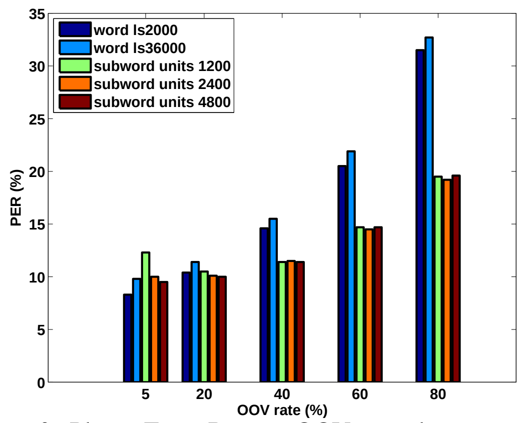

Figure 3: Phone Error Rate as OOV rate changes. *Word* systems built from different subsets of the database. *Subword* systems with various inventory sizes.

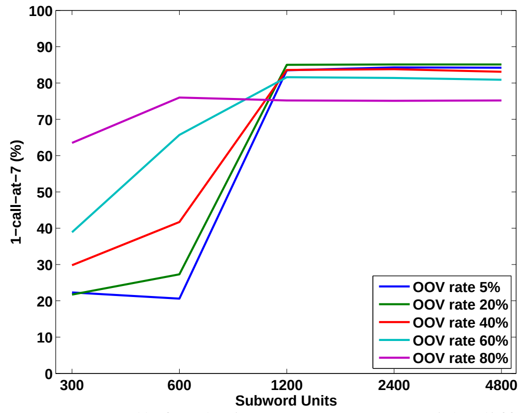

Figure 4: Recall for lyricset `ls36000` with different subword unit inventory sizes at different OOV rates.

from `ls2000` and `ls36000`, the smaller having a more constrained language model. We also show subword-based systems built with different number of units. As expected, the PER degrades much more gracefully for the subword systems as the OOV rates increases. The plot also shows that the PER is robust to the inventory size.

Figure 4 depicts the retrieval performance for a fixed lyricset, `ls36000`, as a function of subword inventory size. The dramatic performance drop as the number of units decreases can be explained by an analysis of the subword unit inventory. When its size is small, most of the pronunciations are mapped to sequences of phones instead of larger subword units. The index becomes mostly based on the distributions of phones in the documents. This distribution is not sufficiently discriminative, explaining the drop in recall. We used inventories of sizes larger than 1000 in the remaining experiments.

Figure 5 displays the retrieval performance as a function of OOV rates comparing the *word* and the *subword* systems. The figure shows results with the indices built from `ls2000` and `ls36000`. While the recall for the word system degrades as the OOV rate increases, as expected, the recall for the subword system remains at a reasonable level. This result was achieved by assuming that the LM, used by the ASR system, is fixed, but

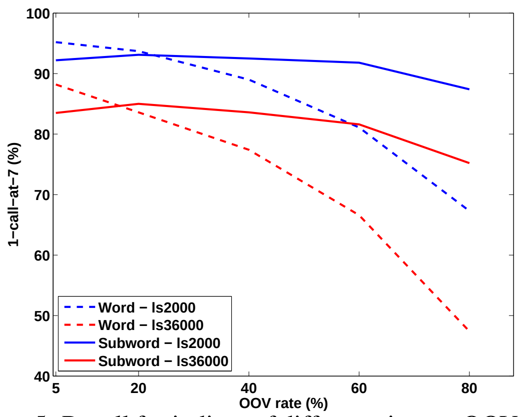

Figure 5: Recall for indices of different sizes as OOV rate changes. The subword unit inventory has 1200 units.

the pronunciation dictionary, used to induce a subword mapping, can change. This assumption is reasonable for embedded systems, where rebuilding an LM can be prohibitively costly, but using a G2P tool is still practical.

5. Conclusion

A subword based system isolates the ASR engine from the IR task. The ASR can use a fixed LM and dictionary, rather than an LM that has to be rebuilt whenever the IR index changes, possibly at a high computational cost.

We have demonstrated that a subword-based voice search system is much more robust to OOVs than its word-based counterpart. Novel words or unexpected spellings, common in applications such as lyrics search, can drive the OOV rate to high levels. This work shows that subword systems are fairly immune to this increase. Our results also indicate that, although within a limit, the recall rate is robust in a wide range of subword inventory sizes.

In future work, we would like to prove the generality of our results using other ASR and IR platforms. Future work also includes applying our algorithms to other types of datasets besides music lyrics.

6. References

[1] P. Wolf and B. Raj, "The MERL SpokenQuery information retrieval system a system for retrieving pertinent documents from a spoken query," in *Proc. ICME*, 2002.

[2] E. Gouvêa, T. Ezzat, and B. Raj, "Subword unit approaches for retrieval by voice," in *SpokenQuery Workshop on Voice Search*, 2010.

[3] M. Creutz and K. Lagus, "Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0." Helsinki University of Technology, Tech. Rep., Mar. 2005.

[4] G. Choueiter, "Linguistically-motivated sub-word modeling with applications to speech recognition," Ph.D. dissertation, MIT, 2009.

[5] M. Bisani and H. Ney, "Open vocabulary speech recognition with flat hybrid models," in *Proc. EUROSPEECH*, 2005, pp. 725–728.

[6] G. Zweig and P. Nguyen, "Maximum mutual information multi-phone units in direct modeling," in *Proc. Interspeech*, Sep. 2009.

[7] R. Rose *et al.*, "Subword-based spoken term detection in audio course lectures," in *Proc. ICASSP*, 2010.

[8] T. Mishra and S. Bangalore, "Speech-driven query retrieval for question-answering," in *Proc. ICASSP*, 2010.

[9] V. Murdock and W. B. Croft, "Simple translation models for sentence retrieval in factoid question answering," in *Proc. SIGIR 2004*.