SUBWORD UNIT APPROACHES FOR RETRIEVAL BY VOICE

Evandro Gouvêa, Tony Ezzat

Mitsubishi Electric Research Labs
Cambridge, MA 02139
egouvea@merl.com, ezzat@merl.com

Bhiksha Raj

Carnegie Mellon University
Pittsburgh, PA 15213
bhiksha@cs.cmu.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 with vocabulary size V into an index written in terms of phonetic subword units of size $M << V$. We demonstrate that, with this highly reduced vocabulary of subword units, improvements in ASR decode speed and memory footprint can be achieved, at the expense of a small drop in recall performance. Results on a music lyrics retrieval task are demonstrated.

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

1. INTRODUCTION

Information retrieval by voice is becoming an increasingly important application [1]. With the proliferation of smartphones, speech becomes the preferred input modality for making queries to search engines, particularly when the queries are long, complex, and would require a lot of typing.

A prototypical system for information retrieval by voice 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.

Typically, the language model (LM) used by the ASR is formed from the entries in the databases to be indexed. As the sizes of these databases increase, however, the ASR LMs also grow larger as they incorporate all novel words from these databases. As a result, ASR decoding speed using these LMs increases measurably. Additionally, the ASR memory footprint also increases, making it harder to load an entire LM into a small memory footprint device. Finally, IR precision performance drops as the databases increase in size.

In this work, we seek to alleviate these scaling issues by using a *subword* unit approach for information retrieval. An algorithm based on the minimum description length (MDL) principle converts a database written in terms of words with vocabulary size V into a database written in terms of phonetic subword units of size $M << V$. We demonstrate that, with this highly reduced vocabulary of subword units, significant reductions in ASR decode speed and ASR memory footprint can be achieved, with little drop in IR performance.

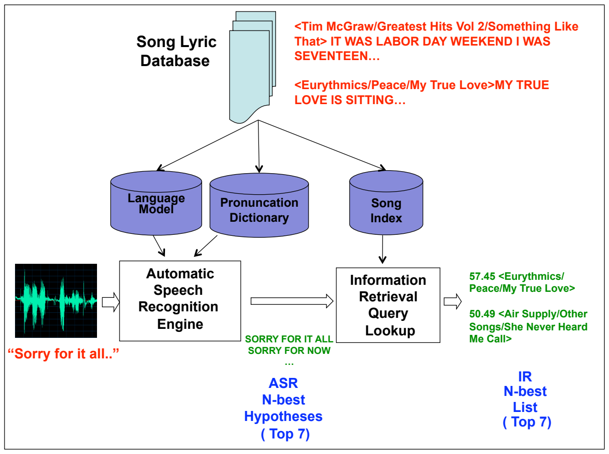
Song Lyric Database
<Tim McGraw/Greatest Hits Vol 2/Something Like That> IT WAS LABOR DAY WEEKEND I WAS SEVENTEEN…
<Eurythmics/Peace/My True Love>MY TRUE LOVE IS SITTING…
Language Model
Pronunciation Dictionary
Song Index
"Sorry for it all.."
Automatic Speech Recognition Engine
SORRY FOR IT ALL
SORRY FOR NOW
…
ASR N-best Hypotheses (Top 7)
Information Retrieval Query Lookup
57.45 <Eurythmics/Peace/My True Love>
50.49 <Air Supply/Other Songs/She Never Heard Me Call>
IR N-best List (Top 7)

Fig. 1. Overview of an Information Retrieval by Voice for a Song Lyric 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.

2. BACKGROUND

Early attempts at building systems for information retrieval by voice [2] [3] focused mainly on pointing out the robustness these systems exhibited to ASR word error rates. With the proliferation of massive internet-scale databases, however, recent systems [4] have begun to observe the need to address LM scaling issues as database size keeps growing.

Two recent approaches to handle LM scaling issues are *n-gram pruning* [5] and *language model compression* [6]. While these approaches do resolve the decode time and memory footprint issues, it is still necessary to re-prune or re-compress the LMs *whenever the databases to be indexed change*. This is because the novel words introduced by these databases need to be re-inserted into the LMs.

In this work, a different approach is taken, in which the novel database are themselves rewritten in terms of subword units. As a result, *once a subword unit LM is built, it does not need to be re-compiled.* Rather, novel databases are simply rewritten in terms of the subword unit inventory.

There have been many recent subword unit inventory creation methods in the recent literature[7] [8] [9] [10] [11]. All of them 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 this work, we focus our analysis on subword unit gains for ASR decode time, memory footprint, and retrieval precision, deferring OOV analysis to future work.

Our subword unit approach builds on our own earlier work [12], and is inspired by the Morfessor [13] algorithm, which relies on the *minimum description length* (MDL) principle to perform morphological analysis of text. We choose this approach because MDL incorporates an explicit *compression* criterion, enabling the creation of subword unit inventories with control over their total size. This property is desirable in light of our efforts to address LM scaling issues. In contrast, many of the earlier subword unit approaches *augment* a regular word-based LM with subword units, which actually leads to LM size *increase*.

3. FROM WORDS TO SUBWORDS

Our definition of a subword unit may be gleaned from Figure 2. As an example, the word `HOURGLASS` is rewritten as a sequence of two subword units `AW_R` and `G_L_AE_S`. Our subword units are composed of a sequence of *phonemes*. A subword unit may also span an entire word, as with `HOUSE`. The subword unit inventory is thus a *flat hybrid* [10] collection of subword units that span portions of words, or entire words. In the following sections, we describe how our algorithm rewrites a database I in terms of a subword unit inventory U.

3.1. Preliminary Definitions

An input database I is composed of a collection of terms t, where each term is itself a sequence of words. In our case, each term is a set of words comprising a song's lyrics.

The set W of size V is defined to be the set of unique words used by I. Using a pronunciation dictionary or a G2P tool, W can be mapped to Q, the set of *word pronunciations*. Since words can have more than one pronunciation, the size of Q usually larger than V. The algorithm retains this forward mapping $W \mapsto Q$ for future use. Each word has a count c_w which is the frequency of its occurrence in the database I. Correspondingly, each pronunciation has a count c_q, which is set equal to the corresponding word count. Given Q, algorithm segments each pronunciation q into a set of phonetic subword units u:

HOURGLASS	AW_R + G_L_AE_S
HOURLY	AW_R + L_IY
HOUSE	HH_AW_S
HOUSEHOLD	HH_AW_S + HH_OW_L_D
HOUSES	HH_AW_S + IH_Z
HOUSES(2)	HH_AW + Z + AH + Z

Fig. 2. Examples of words rewritten in terms of subwords. Note that some words have multiple subword representations due to the fact that they have different pronunciations.

$$q_i \rightarrow u_a + u_b + u_c \quad (1)$$

$$q_j \rightarrow u_d + u_a \quad (2)$$

Note that some units, such as u_a above, may be used in multiple word segmentations. The set of unique units $\bigcup_{m=1}^{M}\{u_m\} = \{u_a, u_b, u_c, \ldots\}$ is termed the subword unit inventory U. Each unit is associated with a count c_u which represents the frequency of times it occurs in the segmentation of the entire index. In the example above, $c_{u_a} = c_{q_i} + c_{q_j}$ since u_a is used to rewrite both q_i and q_j. $N = \sum_{m=1}^{M} c_m$ is defined to be the total unit count in the word corpus segmentation. Each unit count c_u may be converted to a probability p_u by normalizing each count by the sum of all the unit counts: $p_u = c_u / \sum_{m=1}^{M} c_m$.

3.2. Minimum Description Length Principle

The subword unit inventory algorithm utilizes the Minimum Description Length (MDL) principle [13] to search for an inventory of units U which minimize the sum of two terms, $L(Q|U)$ and $L(U)$:

$$\arg\min_{U} L(Q|U) + L(U) \quad (3)$$

The left term, the Model Prediction Cost, measures the number of bits needed to represent Q with the current subword unit inventory U. The right term, the Model Representation Cost, measures the number of bits needed to store the inventory U itself. As such, MDL principle finds the *smallest model* which also *predicts the training data well*. The motivation for doing this is that small models generalize better to unseen data.

The Model Representation Cost sums the bits needed to represent all the units in the inventory:

$$L(U) = \sum_{u \in U} \sum_{phoneme \in u} -\log p(phoneme) \quad (4)$$

Here $p(phoneme)$ is estimated from the frequency counts of each phoneme in Q.

Likewise, the Model Prediction Cost measures the bits needed to represent Q with the current subword unit segmentation:

$$L(Q|U) = \sum_{q \in Q} \sum_{u \in tokens(q)} -\log p_u \tag{5}$$

Here $tokens(q)$ is a function as in Equations 1 and 2 that lists the subword unit segmentation for each pronunciation q.

The unit inventory U which minimizes the cost above may not have the desired M elements, so it is important to be able to increase the weight of the Model Representation Cost $L(U)$ relative to the Model Prediction Cost $L(W|U)$. We do this by introducing a regularization parameter λ which performs this re-weighting:

$$\arg\min_U \lambda L(Q|U) + (1-\lambda) L(U) \tag{6}$$

where $0 \leq \lambda \leq 1$. Typically λ is chosen by the user interactively until the desired number M of subwords is achieved.

3.3. Subword Unit Inventory Search

To find the optimal subword inventory U and segmentation $tokens(q)$, a greedy, top-down, depth-first search algorithm is utilized. Shown in Figure 3 is pseudocode for the search algorithm.

Initially each word w is a subword unit u of its own. At every iteration, a table is maintained of the units in the current inventory U, along with the cumulative code length as defined by Equation 6. A random word is chosen and scanned in a left-to-right manner, yielding different prefix-suffix subword splits. For each split candidate, the cumulative code length is computed. The split (or no split) yielding the lowest code length is selected. In case of a split, splitting of the left and right parts continues recursively and stops when no more gains in overall code length can be obtained by splitting a node into smaller parts. After all words have been processed once, they are again shuffled randomly, and each word is reprocessed. This procedure is repeated until the desired number of subword units in the inventory M is achieved.

3.4. Viterbi Segmentation of Novel Words

At the conclusion of the subword unit search algorithm in Figure 3, a subword unit inventory U is induced, where each unit u has an associated probability p_u.

Given a novel set of pronunciations Q', the Viterbi algorithm is used to segment each novel pronunciation q into subword units $u_1, u_2, \ldots, u_n$ from the unit inventory U. with smallest total bit cost $\sum_{i=1}^{n} -\log p_i$. Shown in Figure 2 are example Viterbi segmentations of words from an database.

3.5. Rewriting a Database and LM

Effectively, the Viterbi algorithm described above provides the forward mapping from pronunciations to subword units $Q \mapsto U$. The forward mappings $W \mapsto Q$ and $Q \mapsto U$ can be composed to provide the mapping from words to subword units $W \mapsto U$.

```
**Algorithm** splitsubwords(node)
**Require**: node corresponds to an entire word or a substring of a word
**Note**:
  $L(U)$ below represents the model representation cost
  $L(Q|U)$ below represents the model prediction cost

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

// THEN TRY EVERY SPLIT OF THE NODE //
// INTO TWO SUBSTRINGS //
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 $c_m$ by $c_{node}$
        increase $L(Q|U)$ if $m$ is a leaf node
    else
        add subnode with $c_{node}$ into the data structure
        increase $L(Q|U)$
        add contribution of subnode to $L(U)$
  if $L(Q|U) + L(U) <$ code length 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)
```

Fig. 3. `splitsubwords`, a recursive, top-down, greedy, algorithm for inducing the subword unit inventory based on the MDL principle.

To rewrite a database I in terms of subword units, the words in each term are scanned sequentially. Each word is mapped to subword units using $W \mapsto U$. However, since some words have multiple pronunciations, the mapping is one-to-many. One mapping is chosen randomly. Once a database has been re-written in terms of subword units, the LM is trained on the re-written database.

4. EXPERIMENTAL DESIGN

In this section, we describe our experiments, elucidating particular axes of variation and performance metrics.

4.1. Dataset Description

The dataset used in this work is the same as the one used by [8]. 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. Shown in Figure 1 at the top example fields for several songs.

The test set consists of 1000 utterances collected in the following manner: 1000 songs 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 record 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 ground truth for the IR experiments is the set of songs with the same title as the query song. In the general case, using song title as a key addresses the retrieval of covers by other artists, as well as songs that appear in more than one album 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.

4.2. 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 convert spoken queries into text. The input spoken query is first converted into standard 39-dimensional MFCC feature vectors (cepstra and Δ and $\Delta\Delta$ features). Decoding is performed using triphone HMM acoustic models trained from Wall Street Journal data resampled to 8kHz. The word pronunciations are obtained from the CMU dictionary when available, or the NIST G2P tool [14] when not. Finally, the LM is a trigram LM with Witten-Bell smoothing, built using the CMU Statistical Language Modelling toolkit. All of these components are available as open source at [15]. In all our experiments, we used Sphinx-3 to generate the 7-best hypotheses for each spoken query, which are then submitted to the IR back-end for retrieval.

4.3. 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 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 the term t appears in both query and document, 0 otherwise, and IDF(t) is the inverse document frequency of term t. No document length normalization was performed.

4.4. Performance Metrics

The system architecture for our experiments is shown in Figure 1. The baseline system is *word-word* system, in which the LM base unit and index base unit are both comprised of words. A *subword-subword* system forms the main comparison architecture with the baseline system. For both system architectures, our metrics of performance are as follows:

Phone Error Rate (PER) : the number of insertions, deletions, and substitutions made by the ASR engine. Comparison is made with the reference transcript for each test utterance.

ASR Decode Time : The average time, in xRT units, taken to decode the test set.

"1-call-at-7" : IR accuracy metric based on the *k-call at n* [16], with $k = 1$ and $n = 7$. Since each test utterance is associated with a corresponding set of "groundtruth" songs, 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.

IR Query Time : The total time, in seconds, to perform the IR lookup query of the whole test set.

4.5. Lyric Database Sizes

An important axis of variation is database size. Five different song lyric databases, or *lyricsets*, were created: `ls2000` $\subset$ `ls4000` $\subset$ `ls8000` $\subset$ `ls17000` $\subset$ `ls36000`. The smallest lyric set, `ls2000` contains the 1989 songs that serve as groudtruth to all test set utterances. Each larger lyric set contains all the lyrics from the previous set, plus an additional set of random lyrics. The largest set `ls36000` contains all 35868 songs. The size of the lyric sets increases exponentially from smallest to largest.

4.6. Subword Unit Inventory Generalization

Another axis of analysis for subwords is *generalization*: given a subword unit inventory created from a certain lyricset, how is retrieval performance affected when that inventory is used to rewrite novel databases, *while keeping the subword recognition LM fixed.*

In this regard, two experiments are performed: In the first experiment, termed *largest-set-subwords*, a subword unit inventory is induced from the largest lyric set, `ls36000`, and used to rewrite all smaller lyricsets. Recognition for all lyric sets is performed using the fixed subword LM formed from `ls36000`. In the second experiment, termed *smallest-set-subwords*, a subword unit inventory is induced from the smallest lyric set, `ls2000`, and used to rewrite all larger lyricsets. Recognition for all lyric sets is performed using the fixed subword LM induced from `ls2000`.

4.7. Subword Unit Inventory Sizes

Finally, our last axis of analysis is the number of subword units induced in the subword unit inventory. Five different subword unit inventory sizes were chosen. For the *largest-set-subwords* experiment, these sets are `5k`, `10k`, `20k`, `30k`, and `45k`, each denoting the number of subwords induced.

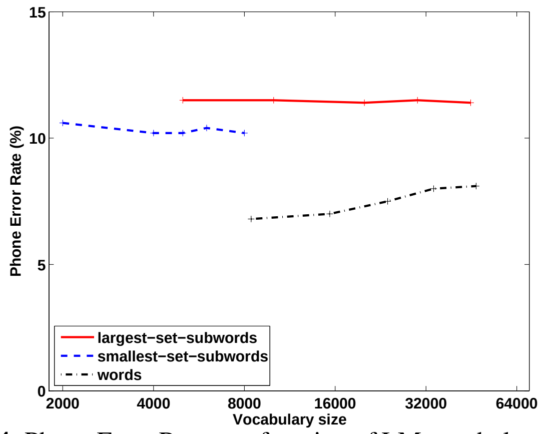

Fig. 4. Phone Error Rate as a function of LM vocabulary size.

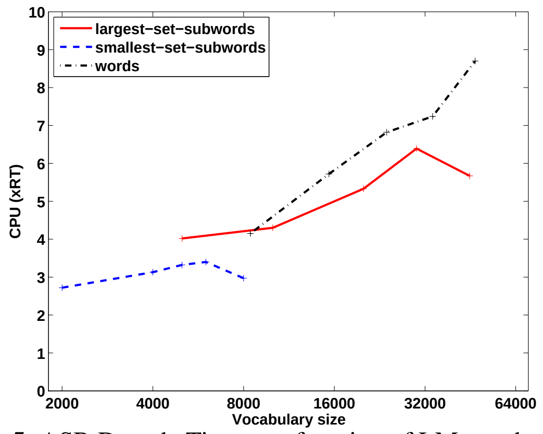

Fig. 5. ASR Decode Time as a function of LM vocab size.

The largest set comprises 45k subwords, just shy of the total number of unique words V in `ls36000`, 46939. For the *smallest-set-subwords* experiment, these sets are `2k`, `4k`, `5k`, `6k`, and `8k`. The largest set comprises 8k subwords, just shy of the total number of unique words V in `ls2000`, 8940.

5. RESULTS AND DISCUSSION

Figure 4 plots recognition accuracy (in PER) as a function of vocab size for the word baseline system, the *largest-set-subword* system, and the *smallest-set-subword* system. In the word experiments, vocabulary size varies with the lyricset used to build the word LM. In the subword experiments, vocabulary size varies with the desired inventory size for LMs induced from the largest (or smallest) lyric set. Clearly, PER is higher for subword unit LMs than for word-based LMs, which is expected given the increase in LM perplexity caused by subword unit induction.

Figure 5 depicts the ASR Decode time (in xRT) as a function of vocab size. The subword unit LMs seem to result in xRT gains over word-based LMs. For example, the 2k-subword LM presents an xRT gain of 3 over the largest word LM. Additionally, word-based LMs are slower than subword-based LMs *for a given vocabulary size*. More interestingly, word-based recognition time increases at a rate faster rate than subwords.

The number of N-grams in a LM is used as a proxy for ASR LM memory footprint. Figure 6 plots the number of N-grams in the LM as a function of vocab size. As shown, word N-gram count increases with vocabulary size. Subword N-gram counts remain fairly constant, however, and depend on whether the LMs were created from the largest lyric set or the smallest lyric set. The smallest-set subword LMs represent a size reduction of a factor of 10 compared with the largest word LMs or largest-set subword LMs.

Figure 7 depicts the retrieval performance for a fixed lyricset, `ls36000`, as a function of vocab size. Subword unit LMs built from the largest lyric sets cause a 4% drop in 1-call-at-7 performance relative to word-based LMs. Surprisingly, smallest-set subword LMs cause a smaller 2% drop in performance. Retrieval performance is flat across subword unit inventory size.

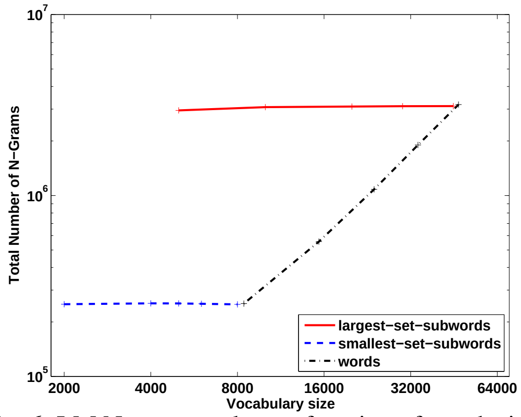

Fig. 6. LM N-gram totals as a function of vocab size.

Figure 8 displays the retrieval performance as a function of different lyricset sizes. The subword inventory size is fixed at 5000 subwords. Performance, as expected, decreases with larger lyricsets. As compared to the word system, the subword-based ones result in moderate degradation, less than 3% for the smallest-set-subword condition and less than 5% for the largest-set-subword condition, considerably better than what one could expect from the degradation in PER in Figure 4.

Figure 9 depicts the total IR lookup time for the whole test set of 1000 utterances. As expected, this increases with the lyricset size, but is negligible compared with the recognition time. Still, the recall times for both subword-based conditions are comparable to the word-based system, considering that document size is much larger after words are converted to subword units.

6. CONCLUSION

Our results indicate that it is indeed possible to build a subword system that performs comparably to a word-based sys-

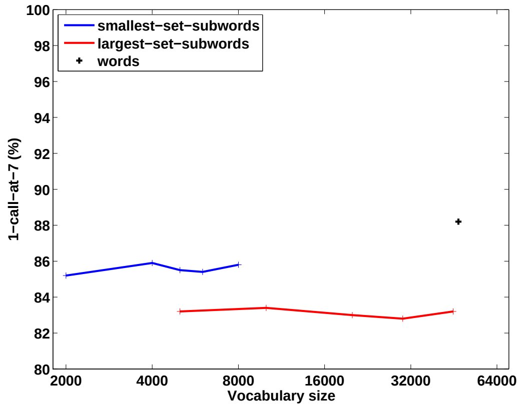

Fig. 7. Recall at fixed lyricset, `ls36000`. The number of words depends on the lyricset used, therefore, only one data point is shown for the *words* case.

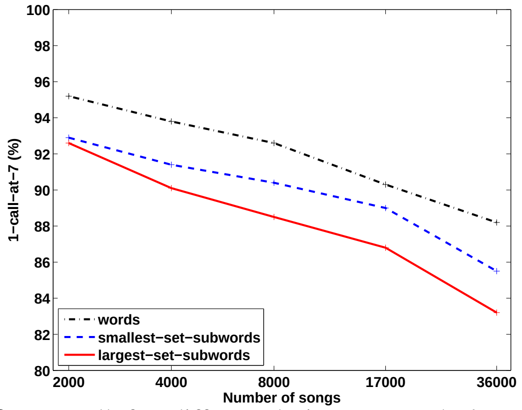

Fig. 8. Recall for different lyricsets. Both *largest-set-subwords* and *smallest-set-subwords* contain 5000 units.

tem. A gain in ASR decode speed and LM memory footprint comes at the price of a small drop in recall. As an example, the 2k-subword system in the smallest-set experiment exhibited a 2-3% drop in recall relative to a complex word-based system, but with a gain of 10 in LM memory footprint, and a gain of 3 in ASR decode speed. Furthermore, we have shown that recognition with fixed subword LM does exhibit *generalization* capacity: the ability to be used to rewrite and retrieve documents from a larger novel database.

It is worth pointing out that, surprisingly, smallest-set subword LMs outperformed largest-set subword LMs. This is probably due to the fact that the MDL subword algorithm is forced to break up largest-set words into smaller pieces in order to achieve the desired inventory size. Smallest-set subwords are thus longer and more word-like, which explains their superior performance.

While our results are encouraging, it is important to note that further experiments are needed on other ASR and IR platforms to verify the generality of our results. Additionally, future work includes exploring the benefit of using subword units when the test set has a significant percentage of OOV

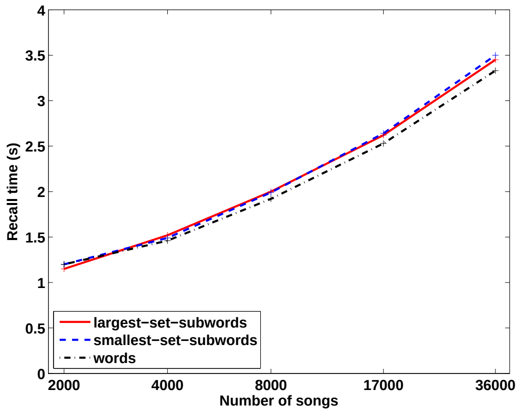

Fig. 9. Total IR Query Lookup time for different lyricsets. The inventories of both *largest-set-subwords* and *smallest-set-subwords* contain 5000 units.

terms, as well as applying our algorithms to other types of datasets besides music lyrics.

7. REFERENCES

[1] Y-Y. Wang, D. Yu, Y-C. Ju, and A. Acero, "An introduction to voice search," *IEEE Signal Processing Magazine*, vol. 25, May 2008.

[2] 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.

[3] F. Crestani, "Spoken query processing for interactive information retrieval," *Data Knowl. Eng.*, vol. 41, no. 1, pp. 105–124, 2002.

[4] M. Bacchiani, F. Beaufays, J. Schalkwyk, M. Schuster, and B. Strope, "Deploying GOOG-411: Early lessons in data, measurement, and testing," in *Proc. ICASSP*, 2008, pp. 5260–5263.

[5] A. Stolcke, "Entropy-based pruning of backoff language models," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, 1998, pp. 270–274.

[6] B. Harb, C. Chelba, J. Dean, and S. Ghemawat, "Back-off language model compression," in *Proc. Interspeech*, September 2009.

[7] E. W. D. Whittaker and P. C. Woodland, "Particle-based language modelling," in *Proc ICSLP*, 2000, vol. 1, pp. 170–173.

[8] G. Choueiter, *Linguistically-Motivated Sub-word Modeling with Applications to Speech Recognition*, Ph.D. thesis, MIT Department of Electrical Engineering and Computer Science, February 2009.

[9] A. Rastrow, A. Sethy, B. Ramabhadran, and F. Jelinek, "Towards using hybrid word and fragment units for vocabulary independent lvcsr systems," in *Proc. Interspeech*, September 2009.

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

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

[12] E. B. Gouvêa and B. Raj, "Word particles applied to information retrieval," in *Proc. European Conference on Information Retrieval (ECIR)*, April 2009.

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

[14] "NistG2P," http://www.itl.nist.gov/iad/mig/tools/addttp4-11tarZ.htm.

[15] "CMU Sphinx," `http://cmusphinx.org`.

[16] H. Chen and D. R. Karger, "Less is more: probabilistic models for retrieving fewer relevant documents," in *Proc. ACM SIGIR*, 2006.