Optimal Selection of Limited Vocabulary Speech Corpora

Hui Lin, Jeff Bilmes

Department of Electrical Engineering, University of Washington, Seattle
{hlin,bilmes}@ee.washington.edu

Abstract

We address the problem of finding a subset of a large speech data corpus that is useful for accurately and rapidly prototyping novel and computationally expensive speech recognition architectures. To solve this problem, we express it as an optimization problem over submodular functions. Quantities such as vocabulary size (or quality) of a set of utterances, or quality of a bundle of word types are submodular functions which make finding the optimal solutions possible. We, moreover, are able to express our approach using graph cuts leading to a very fast implementation even on large initial corpora. We show results on the Switchboard-I corpus, demonstrating improved results over previous techniques for this purpose. We also demonstrate the variety of the resulting corpora that may be produced using our method.
Index Terms: corpus subset selection, submodularity, LVCSR

1. Introduction

Large vocabulary spontaneous conversational speech recognition is one of the most challenging tasks in speech processing and is one of the most computationally demanding in all machine learning. In recent times, very large amounts of transcribed data, with both many tokens and many types, have become available. Some corpora have a vocabulary size as large as one million and as many as 230 billion tokens [1]! While having such a wealth of training data is useful from the perspective of producing better speech recognition systems (there is no data like more data), the data size itself presents a serious problem for novel speech recognition research.

Novel ASR systems are often not highly optimized or tuned, including at the implementation level (where low-level coding tricks and years of human effort can have a significant speed and memory benefit) and also at the algorithmic level (where different or new algorithms can later be discovered to more efficiently solve the same underlying problem). The more novel the idea, the more effort it takes to get it working on a large system since there is less chance of potential implementation reuse from a pre-existing system. In general, it is important to be able to test a novel idea quickly, without investing enormous amounts of time on the engineering effort to make the ideas perform well, and if a new idea ends up performing poorly, knowing this sooner rather than later will avoid futile work.

Novel speech recognition systems, moreover, deserve rich data on which to be evaluated. For example, novel systems might not show their benefit on data lacking the characteristics the novel system is designed to address. Now, large corpora are useful not simply because they are large, but because they contain information simply unavailable in typical smaller corpora. For example, a large corpus can contain not only rich phonetic variety but also a full representation of that variety. That is, a large corpus has many samples of high probability word pronunciations (certain pronunciations might even be over-represented, and less data is sufficient to produce for an accurate model). Even low probability pronunciations, however, might have a sufficient number of samples in a very large corpus to produce a good statistical pronunciation model.

On the other hand, recent large data sets are unkind to novel ASR systems simply because they are so large. ASR systems often have complexity that is linear in the number of tokens and polynomial in the number of types (e.g., decoding using a trigram language model with size-N vocabulary has, in the worst case, a complexity of at least $O(N^3)$). A challenge is determining how to quickly test a new system on large data sets.

One way to address this problem is to produce a smaller version of the corpus, and one way to do this is to draw a subset uniformly at random. Any such subset, however, might not possess the richness mentioned above. It should moreover be possible to produce a smaller corpus that removes over-representation while retaining proper representation for every speech unit. Ideally, we would like a process that can take a large corpus and produce a subset that satisfies a particular purpose. For example, when vocabulary size is the key attribute hindering the rapid evaluation of novel acoustic method, we might choose a limited vocabulary subset of data of maximal size. On the other hand, we may wish to correct for some other quality, such as imposing a bias against certain word forms. We moreover wish the results from the corpus to be an accurate reflection of results from the entire corpus.

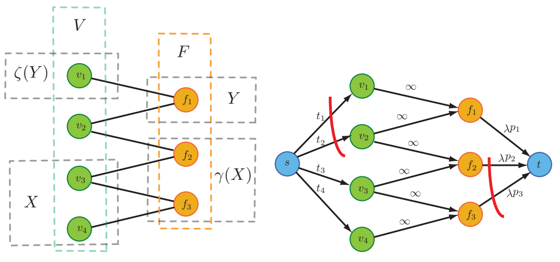
In this paper we address this problem by formulating it as an optimization problem via the use of submodular functions [2]. As we will see, this allows us to express the problem of corpus subset selection in a variety of flexible ways, suiting the needs of an individual novel ASR system and its designer, and allows us to find the optimal solution for our objective, improving on a previously proposed method for this purpose [3].[1]

2. Why Not Greedy

Our goal is to create a corpus of spontaneous conversational speech with limited (small) vocabulary. We do this by selecting utterances from a large vocabulary conversational speech corpus (e.g., Swithboard). The question is: how can we select as much and as rich acoustic data as possible while limiting the vocabulary size?

One straightforward and simple way to solve this corpus subset selection problem is to use a greedy algorithm: Here, we start with an empty vocabulary. In each greedy step, we add an out-of-vocabulary (OOV) word into the vocabulary if the amount of data containing **only and nothing other than** this new vocabulary is maximized (this is made formal below via function f_{svb}). The algorithm stops when the desired vocabulary

[1]We note, early aspects of our approach were presented in a previous unrefereed workshop paper [4], while in this paper we elaborate on speech applications and explanations.



(a) Bipartite graph (b) s-t graph

Figure 1: In subfigure (a), $V = \{v_1, v_2, v_3, v_4\}$ represents four utterances, which might be (from top to bottom) "yes", "oh yes", "oh right, right" and "right"; $F = \{f_1, f_2, f_3\}$ is the vocabulary, i.e., f_1, f_2, f_3 represent words "yes", "oh" and "right" respectively. For $X = \{v_3, v_4\}$, $\gamma(X) = \{f_2, f_3\}$. For $Y = \{f_1\}$, $\zeta(Y) = \{v_1\}$. In (b), the s-t graph corresponding to Eq. 4.

size is reached. This algorithm was used in [3], for instance.

The greedy algorithm, although conceptually simple, could perform arbitrarily poorly for the corpus subset selection problem. To see this, we first formalize the greedy approach as follows. Let F be the set of distinct words (i.e., vocabulary, or types) in the original (large) corpus. For $Y \subseteq F$, let the function $\zeta(Y)$ denote the set of utterances that contain *only* words in Y. Given an utterance v, let t_v represent the amount of data contained in x. For instance, in [3], t_v is the number of word tokens in utterance v. The greedy algorithm then attempts to select a set of words such that a set function $f_{\text{svb}} : 2^F \to \mathbb{R}$ is maximized, where $f_{\text{svb}}(Y) \triangleq \sum_{v \in \zeta(Y)} t_v$.

Now it can be shown that f_{svb} is a supermodular [5] set function.[2]Therefore, the greedy algorithm above attempts to solve the problem of maximizing a supermodular function subject to a cardinality constraint. Unfortunately, the greedy algorithm in this case has an unboundedly poor approximation factor. For instance, let $F = \{a, b, c\}, f(\{a\}) = 1, f(\{b\}) = f(\{c\}) = 0, f(\{a, b\}) = f(\{a, c\}) = 1, f(\{b, c\}) = p > 1$ and the cardinality is constrained to be at most 2. This function f is supermodular. Greedily maximizing f leads to a solution $\{a, b\}$ with objective function value 1, while the true optimal objective function value is p. Since p is arbitrary, the approximation factor for this example is unboundedly poor. This is unsurprising, as it is known that greedy algorithm works near-optimally only when maximizing a *submodular* function subject to cardinality (knapsack) constraint [6], and this nice property has been used in our previous work on selecting good unlabeled training data to transcribe [7] and for document summarization [8, 9].

3. Problem Setup

In this paper, we treat the corpus creation problem as finding a subset of utterances that simultaneously minimizes the vocabulary size and maximizes the total amount of data, measured as either the number of utterances, the number of tokens in the utterances, or duration of speech. The problem can be seen as a combinatorial optimization problem defined on a bipartite graph. Let V be a set of corpus utterances, and let F be the vocabulary (set of distinct words) contained collectively in these utterances. We define a bipartite graph $G = (V, F, E)$ where $E \subseteq V \times F$ are the set of edges. Each $(v, f) = e \in E$ is an edge between an utterance $v \in V$ and a word $f \in F$ if utterance v contains word f (see Figure 1(a) for example). We find $X \subseteq V$ that maximizes the following objective function

$$w(X) - \lambda\Gamma(X) \tag{1}$$

where $w(X)$ measures the amount of data contained in utterances X, $\Gamma(X)$ represents the vocabulary size associated with utterances X, and $\lambda \geq 0$ is a tradeoff coefficient.

Intuitively, maximizing Eq. 1 simultaneously maximizes the total amount of data ($w(X)$) and minimizes the vocabulary size ($\Gamma(X)$). Note that we *optimize* the vocabulary size instead of having hard constraints on it. By changing the value of λ, we can produce corpora with different vocabulary sizes, where each vocabulary size is determined during the optimization process and is optimal in terms of balancing with the amount of the associated data. In general, the larger λ is, the smaller the resulting vocabulary size will be.

There are at least two advantages of formulating the corpus subset selection problem as maximizing Eq. 1. First of all, this allows us to express the problem of corpus subset selection in a variety of flexible ways, suiting the needs of an individual novel ASR system its designer. Secondly, unlike the problem setup in [3] where only a sub-optimal solution is available, our optimization problem can be solved exactly and efficiently by leveraging techniques of submodular function minimization. We discuss these two aspects in more detail in the following two sections.

4. Objective Functions

When designing a corpus for novel ASR system development, certain properties of the data as well as certain word forms might be preferred, both of which can be easily modeled in our approach by using different forms of the objective function (Eq. 1). In particular, we may have different w functions depending on our needs. For instance, when w is the cardinality function, i.e. $w(X) = \sum_{v \in X} 1 = |X|$, it measures the number of the utterances. In this case, maximizing Eq. 1 will give us a corpus that favors a large number of utterances. We can also have a weight on each utterance where the weight indicates how important the corresponding utterance is, i.e., we can have $w(X) = \sum_{v \in X} s_v$, where s_v is the weight for utterance x. When s_v represents the speech time-length of utterance x, $\sum_{x \in X} s_v$ measures the total speech duration of the utterances X, in which by maximizing Eq. 1, we will have a bias on utterances that contain more speech. On the other hand, we can also have different forms of $\Gamma(X)$ to impose a bias against certain word forms. First, we define $\gamma(X)$ to be the distinct words that appear in utterances X. That is

$$\gamma(X) \triangleq \{f \in F : \exists v \in X \text{ s.t. } (v, f) \in E\}. \tag{2}$$

In other words, $\gamma(X)$ is the vocabulary of utterances (corpus) X, and the cardinality of $\gamma(X)$ (i.e., $|\gamma(X)|$) is the size of the vocabulary (see Figure 1(a)). Then we can have $\Gamma_1(X) = |\gamma(X)|$, which represents the collective vocabulary size of utterances in set X. We can also have

$$\Gamma_2(X) = \sum_{f \in \gamma(X)} p_f, \tag{3}$$

where p_f indicates the unimportance of word f. A larger p_f states that word f is less important. This allows certain desirable properties of the vocabulary of the resultant corpus to be expressed (e.g., words with more syllables might be preferred). Note if $p_f = 1, \forall f$, then $\Gamma_2 = \Gamma_1$.

[2] A set function $f : 2^V \to \mathbb{R}$ is submodular if for any $A \subseteq B \subseteq V$ and $k \notin B$, we have $f(A \cup \{k\}) - f(A) \geq f(B \cup \{k\}) - f(B)$. f is said to be supermodular if $-f$ is submodular.

By using different forms of objective function, corpora suiting different needs can be created. Table 1 illustrates several objective functions that we used in our experiments, where for instance, Corpus D was created by maximizing the total speech duration in the resultant corpus while limiting the vocabulary with preference for words with more syllables.

5. Algorithm

Note that maximizing Eq. (1) is identical to finding X that minimizes

$$L(\lambda, X) \triangleq w(V \setminus X) + \lambda\Gamma(X). \qquad (4)$$

For a given λ, if $L(\lambda, X)$ is a submodular function on X, then $\min_{X \subseteq V} L(\lambda, X)$ can solved exactly in polynomial time [2] Fortunately, all the aforementioned w functions are modular (both submodular and supermodular), and all Γ functions are submodular, making $L(\lambda, X)$ submodular in all our cases. Therefore we can solve our corpus creation problem optimally by leveraging submodular function minimization techniques.

To create a corpus with the desired property (e.g., a fixed upper limit on vocabulary size), different values of the trade-off coefficient λ must be tried, where multiple calls of the optimization algorithm are required. In other words, we do not have direct control over the vocabulary constraint, only indirect control via λ. In our case, however, *all* possible solutions for $\min_{X \subseteq V} L(\lambda, X)$ for *all* possible $\lambda \geq 0$ can be found in the same complexity as the complexity of solving $\min_{X \subseteq V} L(\lambda, X)$ for a *single* λ, and moreover there are only a finite (no more than $|V|$) number of distinct values of λ that makes a difference, all thanks to submodularity.

Finding all distinct λ values (and minimizing sets) can be done using parametric submodular function minimization. When using a push-relabel framework [10], finding solutions for all λ requires only the same asymptotic running time as a *single* submodular function minimization. Note that the push-relabel framework was firstly introduced in [11] for network flow (graph cut) problems.

Now, interestingly, the submodular functions used in this paper are all graph-representable. In other words, we can convert our submodular minimization problem to the problem of finding minimum s-t cuts in a graph, and therefore a fast parametric flow algorithm [11] can be used to find all the solutions for the corpus creation problem for all possible values trade-off coefficient λ, while only requiring the computational complexity of running a single minimum s-t cut, which can be solved very efficiently even on very large graphs.

We next describe how we convert our minimization problem into a minimum s-t graph cut problem. Take $\min_{X \subseteq V} w(V \setminus X) + \lambda\Gamma_2(X)$ for example. We build an s-t graph as follows. Add a source node s and connect it to every node $v \in V$ with weight t_v. Add a sink node t and connect to it every node in $f \in F$ with weight λp_f. Use an infinite weight for every original graph edge $(v, f) \in E$. Now in such a graph, due to the max-flow/min-cut theorem, any minimum cut cannot have any edge $(v, f) \in E$ since that edge has infinite capacity, and therefore, any minimum cut will consist of either:

1. all the edges $\{(s, v) : v \in V\}$ having a cut value of $\sum_{v \in V} t_v = w(V)$;
2. all of the edges $\{(f, t) : f \in F\}$ having a cut value of $\lambda \sum_{f \in F} p_f = \lambda\Gamma_2(V)$; or

Table 1: Objective functions used for Corpora A, B, C and D. t_v and s_v are the number of word tokens, and the duration of speech in utterance v respectively. q_f is the number of phonemes in the pronunciation of word f.

Corpus ID	Objective Function	
	$w(X)$	$\Gamma(X)$
A	$\|X\|$	$\|\gamma(X)\|$
B	$\sum_{v \in X} t_v$	$\|\gamma(X)\|$
C	$\sum_{v \in X} s_v$	$\|\gamma(X)\|$
D	$\sum_{v \in X} s_v$	$\sum_{f \in \gamma(X)} \frac{100}{q_f}$

3. the edges $\{(s, v) : v \in V \setminus X\} \cup \{(f, t) : f \in \gamma(X)\}$, with a cut value of $\sum_{v \in V \setminus X} t_v + \lambda \sum_{f \in \gamma(X)} p_f = w(V \setminus X) + \lambda\Gamma_2(X)$.

Note that the first case has $X = \emptyset$, and the second case $X = V$, both of which are special cases of the third case. The transformation is shown in Figure 1(b). Therefore, finding the minimum s-t cut will minimize our objective $w(V \setminus X) + \lambda\Gamma_2(X)$.

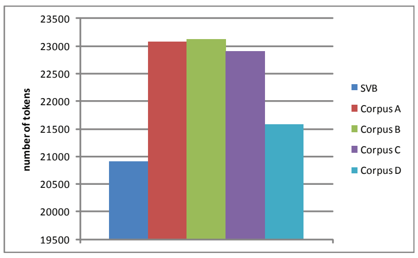
6. Experiments

We tested our corpus subset selection approach on Switchboard I. To be comparable with [3], we followed the same experimental setup. In particular, each side of the long conversations in the Switchboard-I corpus was divided into shorter segments. The initial cuttings used were based on the segmentations produced by Mississippi State University [12]. These segments were further divided into smaller utterances at every silence longer than 500ms. The resulting utterances were pruned by removing those containing disfluency and filler-model words: i.e., all word fragments (e.g. sim[ilar]-), words ending in a digit, uh, [noise], i-, yeah, [laughter], huh, hm, [laughter-*], uh-huh, um-hum hum, huh-uh, um.

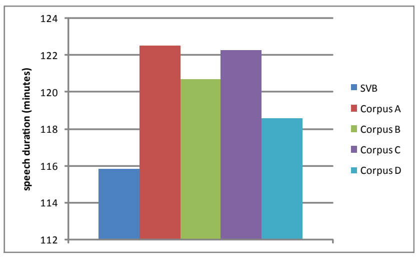
We investigated several types of w and Γ function, and produced corpora with different properties. In particular, we created four corpora (corpus A, B, C and D) using objective functions illustrated in Table 1, and compare them to SVitchboard (SVB, corpora created using the greedy algorithm [3].) Note that in our approach the vocabulary sizes of the resulting corpora are naturally determined by the objective rather than predefined, and we choose those that are as close as possible to the SVB vocabulary sizes (10, 25, 50, 100, and 500) for comparison.

Corpus A was produced with the objective function that maximizes the number of utterances while restricting the vocabulary size. It turns out that corpus A indeed includes more utterances compared to other corpora (created with objectives that are not maximizing the number of utterances) given the same vocabulary size. For instance, with vocabulary size 10, corpus A contains 7615 utterances while SVB has 6775. It even contains more utterances (26165 vs. 23670) with a smaller vocabulary (489 vs. 500), compared to SVB.

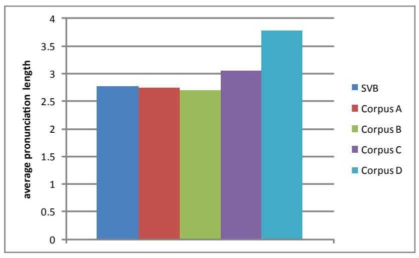
Corpora B and C were produced in a similar way as corpus A except that utterance weights were used. In particular, for corpus B, each utterance was weighted by the number of tokens it contains (i.e. $\sum_{x \in X} t_x$ measures the number of tokens in utterances X), while in corpus C, each utterance was weighted by the duration of the non-silence speech (i.e. $\sum_{x \in X} s_x$ measures the total speech duration of X). The resulting corpora B and C do have the desired property. As illustrated in Figure 2(a), with vocabulary size 50, there are 23124 tokens in corpus B, which is larger than those in SVB, or in corpora C and D; there are about 122 minutes of speech in corpus C, which



(a) toal number of tokens



(b) total speech duration (minutes)



(c) average number of phones per pronunciation

Figure 2: Corpora statistics on the total number of tokens in the utterances, the duration of speech, and the average number of phones in the pronunciations of the corpus vocabulary. The plots correspond to a vocabulary size of 50.

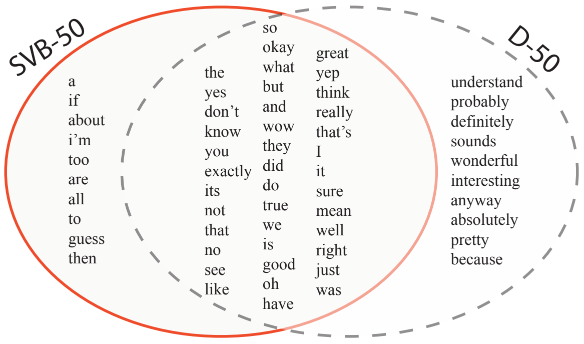


Figure 3: Venn diagram showing the vocabulary difference between SVB-50 [3] and our D-50.

is more than those in SVB, or in corpora B and D, as shown in Figure 2(b). Note that for corpus A, the vocabulary size closest to 50 is 51, but all the remaining corpora have a λ yielding a vocabulary size of exactly 50. While corpus C (at 50 types) is optimized for speech duration, corpus A is slightly larger in this measure, due to it having a 51 types. Corpus C, however, also had a λ corresponding to 51 types and in this case, a 51-type corpus C had a total speech duration of 123.17, beating the 51-type corpus A with its total speech duration of 122.49.

In some situations, we may be concerned not only with the amount of data and the vocabulary size, but also wish for the resulting vocabulary to possess certain properties. Our method can handle such scenarios naturally and efficiently. For instance, a corpus with a limited vocabulary but rich phonetic variety could be useful for research on novel pronunciation modeling, and this is how we produced corpus D. We approximated the phonetic richness by the number of phoneme tokens in the pronunciation of a word [12]. We used an objective function as shown in Table 1. Intuitively, by using this objective function, words with more phones will have a lower weight and therefore a higher chance of being selected. The resulting corpus D is well balanced in terms of corpus size and vocabulary variety. Compared to SVB at vocabulary size 50, corpus D not only has a vocabulary with longer average pronunciation length (Figure 2(c)) but also includes more tokens (Figure 2(a)) and more acoustic speech (Figure 2(b)). We compare the complete vocabulary of SVB-50 and D-50 using a Venn diagram in Figure 3, clearly showing that D-50 has a richer lexicon (with words like “absolutely” and “definitely”).

7. Conclusions

We have presented a framework for selecting a limited vocabulary subset of a large speech corpora. We show that a previous approach [3] for this purpose is theoretically unjustified, while the selection process in our new approach is optimal in the sense that formulating the subset selection problem as an optimization problem over submodular functions leads to an optimal solution. Our approach can be scaled to very large initial corpora, thanks to the efficiency of finding a minimum graph cut. Experimental results on Switchboard show that our approach indeed produces limited vocabulary corpora with both additional and richer acoustic data.

8. References

[1] C. Chelba, T. Brants, W. Neveitt, and P. Xu, “Study on interaction between entropy pruning and Kneser-Ney smoothing,” in *Proc. of Interspeech*, September 2010.

[2] S. Fujishige, *Submodular functions and optimization*, Elsevier Science Ltd, 2005.

[3] S. King, C. Bartels, and J. Bilmes, “SVitchboard 1: Small Vocabulary Tasks from Switchboard,” in *Ninth European Conference on Speech Communication and Technology*. ISCA, 2005.

[4] H. Lin and J. Bilmes, “An application of the submodular principal partition to training data subset selection,” in *NIPS Workshop on Discrete Optimization in Machine Learning: Submodularity, Sparsity & Polyhedra*, Vancouver, Canada, December 2010.

[5] H. Narayanan, *Submodular Functions and Electrical Networks*, North-Holland, Amsterdam, 1997.

[6] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher, “An analysis of approximations for maximizing submodular set functions I,” *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.

[7] H. Lin and J. Bilmes, “How to select a good training-data subset for transcription: Submodular active selection for sequences,” in *Proc. of Interspeech*, Brighton, UK, September 2009.

[8] H. Lin and J. Bilmes, “Multi-document summarization via budgeted maximization of submodular functions,” in *NAACL/HLT-2010*, Los Angeles, CA, June 2010.

[9] H. Lin and J. Bilmes, “A class of submodular functions for document summarization,” in *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, Portland, OR, June 2011.

[10] L. Fleischer and S. Iwata, “A push-relabel framework for submodular function minimization and applications to parametric optimization,” in *Symposium on Theory of Computing*, 2000.

[11] G. Gallo, M. D. Grigoriadis, and R. E. Tarjan, “A fast parametric maximum flow algorithm and applications,” *SIAM J. Comput.*, vol. 18, no. 1, pp. 30–55, 1989.

[12] N. Ganapathiraju, A. Deshmukh, A. Gleeson, A. Hamakera, and J. Picone, “Resegmentation of SWITCHBOARD,” in *Proc. ICSLP*, 1998.