Available online at www.sciencedirect.com

ScienceDirect

Computer Speech and Language 42 (2017) 122–142

COMPUTER SPEECH AND LANGUAGE

www.elsevier.com/locate/csl

# SVitchboard-II and FiSVer-I: Crafting high quality and low complexity conversational english speech corpora using submodular function optimization

Yuzong Liu *, Rishabh Iyer, Katrin Kirchhoff, Jeff Bilmes

*Department of Electrical Engineering, University of Washington, Seattle, USA*

Received 24 June 2016; accepted 7 October 2016

**Abstract**

We introduce a set of benchmark corpora of conversational English speech derived from the Switchboard-I and Fisher datasets. Traditional automatic speech recognition (ASR) research requires considerable computational resources and has slow experimental turnaround times. Our goal is to introduce these new datasets to researchers in the ASR and machine learning communities in order to facilitate the development of novel speech recognition techniques on smaller but still acoustically rich, diverse, and hence interesting corpora. We select these corpora to maximize an acoustic quality criterion while limiting the vocabulary size (from 10 words up to 10,000 words), where both "acoustic quality" and vocabulary size are adeptly measured via various submodular functions. We also survey numerous submodular functions that could be useful to measure both "acoustic quality" and "corpus complexity" and offer guidelines on when and why a scientist may wish use to one vs. another. The corpora selection process itself is naturally performed using various state-of-the-art submodular function optimization procedures, including submodular level-set constrained submodular optimization (SCSC/SCSK), difference-of-submodular (DS) optimization, and unconstrained submodular minimization (SFM), all of which are fully defined herein. While the focus of this paper is on the resultant speech corpora, and the survey of possible objectives, a consequence of the paper is a thorough empirical comparison of the relative merits of these modern submodular optimization procedures. We provide baseline word recognition results on all of the resultant speech corpora for both Gaussian mixture model (GMM) and deep neural network (DNN)-based systems, and we have released all of the corpora definitions and Kaldi training recipes for free in the public domain.

*Keywords:* Submodular function optimization; Automatic speech recognition; Speech corpus

## 1. Introduction

Speech recognition is one of the most challenging tasks in applied machine learning, and one that requires large amounts of diverse training data. Among different aspects of a speech recognition system, training acoustic models for in-situ conversational speech recognition is one of the most challenging tasks. First, the acoustic characteristics of conversational speech are more diverse than those of carefully read speech due to increased variability in pronunciation, speaker, and acoustic environment. Second, conversational speech recognition involves very large vocabularies. Thus, enormous amounts of training data is required to train a conversational speech recognition system even to

* Corresponding author.
*E-mail address:* yzliu@uw.edu (Y. Liu).

http://dx.doi.org/10.1016/j.csl.2016.10.002
0885-2308/© 2016 Elsevier Ltd. All rights reserved.

a reasonable, let alone a good, level of performance. Finally, recently developed acoustic modeling techniques using deep architectures Hinton et al. (2012); Dahl et al. (2012); Sainath et al. (2013); Graves et al. (2013); Sak et al. (2014a,b); Saon et al. (2015), that require very long training and, thus, system development times, and large, often GPU-based, computational devices.

While conversational speech recognition is challenging, it is even more difficult for researchers who have limited computational resources compared to the amount of data that is available. A typical speech recognition experiment starts with a scientific decision about the form of a model, something that can include the style of an HMM, the way to algorithmically construct a Gaussian mixture during training, or the number and width of layers and style of non-linearity in a DNN. Once such decisions are made, the model is trained on a training set, and then tested on a test set. This process of (1) selecting a model, (2) training it, and (3) testing it, repeats as many times as is either necessary or feasible. A speedup either in training or testing time, or ideally both, would allow more cycles of this process to occur during any fixed amount of time, and hence is a desirable goal.

To define some notation, we have a large set $V$ of speech utterances, where each $v \in V$ consists of a pair $v = (x, y)$ where $x = (x_1, x_2, \cdots, \mathrm{x}_{\mathrm{T_x}})$ is a an audio sample consisting of $T_x$ acoustic feature vectors (e.g., each $x_t \in \mathbb{R}^p$ for some $p$, and with MFCCs we may have $p = 39$), and $y = (y_1, y_2, \cdots, \mathrm{y}_{\mathrm{N_y}})$ is a word transcription of the audio where $y_i \in \mathcal{W}$ is a set of word types. The transcription $y$ therefore consists of a sequence of *tokens* (i.e., words, some of which might be repeated). We make the standard distinction between “*tokens*” and “*types*” — a “type” is the identity of the word, and a “token” is an instance of a type. A corpus has one or more tokens for every type. Hence, a corpus has size values in terms of number of samples $n = |V|$ (i.e., the number of utterance samples, or training pairs); also number of tokens $n_o = \sum_{v \in V} N_{y(v)}$, and number of types $n_t = |\mathcal{W}|$. The *vocabulary size* of a corpus is $n_t$. We always have $n_o \geq n_t$ and $n_o \geq n$ although typically $n_o > n_t$ and $n_o \gg n$.

The complexity of acoustic model training is usually at least linear, $O(n)$, in the number of training samples (utterances).[1] For massive training sets and when using computationally demanding models like deep neural networks (DNNs), it can take weeks to train just one system, even using multiple GPUs in parallel. Hence, producing a representative subset of the training set where the resultant $n$ is much smaller will make the model discovery process proceed more quickly.

During the testing phase of a model, when the sequence of tokens corresponding to an utterance is hidden and hence inferred, complexity for decoding is often $O(n_t^\ell)$ where $\ell$ is typically the order in, say, an $\ell$-gram language model. Historically, we would have $\ell \geq 3$, but in modern systems $\ell$ can be much bigger. The exponentially complexity growth in $\ell$ results in decoding being infeasible for single-pass strategies with $\ell$ as large as desired, not to mention the curse of dimensionality that results in estimating a model with large $\ell$. Hence, multi-pass strategies are often used, where an initial pass of decoding is performed with only moderate $\ell$ which then produces a list or lattice of decoding hypotheses to be re-scored using a latter stage and more complex system. This also contributes to long training/testing turnaround times. Hence, when one’s primary focus is on acoustic modeling, one may wish to test a system out on a corpus having a smaller vocabulary $n_t$ or order $\ell$.

In both cases, such long experimental turnaround times can make large-scale speech recognition difficult or impractical, particularly in academia where most researchers and students have limited computational resources relative to the available training sets. Even outside of academia, this problem limits the evaluation of many diverse models since fewer models can be evaluated given a fixed time and computational budget. Also, in either academia or industry, as available training data gets larger (thanks to data collection methods such as Apple’s SIRI, Google NOW, and Microsoft’s Cortana), this problem with continue for the foreseeable future.

Historically, two of the most commonly used and readily available conversational speech corpora are the Switchboard (Godfrey et al., 1992) and Fisher (Cieri et al., 2004) datasets, both of which are large in terms of vocabulary size and number of training samples, and both of which we utilize in the present work.

Our paper addresses these problems. Specifically, this paper contributes the following: (a) develop and evaluate flexible methodology that, given a large speech corpus, can efficiently subselect from it to produce large and acoustically rich sub-corpus while at the same time limiting the complexity (e.g., the vocabulary size $n_t$) of the corpus; (b) use this methodology to produce useful and acoustically rich subsets of both the Switchboard and Fisher corpora; (c) to establish baselines performance numbers for the resultant corpora; and (d) to release the corpora definitions for free to the community. We also (e) produce an empirical evaluation of three modern constrained and unconstrained submodular

[1] We say “at least” here since, in general, it is not in general known how the number of training epochs needed to train a system varies as a function of the training set size.

optimization methods on large-scale real-world problems. We refer to our resulting corpora as SVitchboard-II (SVB-II), and FiSVer-I, where in each case "SV" stands for "small vocabulary." By doing so, we hope to provide researchers with smaller but still challenging speech corpora, thus facilitating faster experimental throughput for testing novel acoustic modeling and machine learning methods.

A brief outline of the paper is as follows. First, Section 2 describes the contributions of the paper, including a bit of background material and notation, the goals of corpus creation (including the general notion of quality and complexity), and previous work. Next, Section 3 describes submodular functions, why they are appropriate models both for quality and complexity, and describes the various submodular optimization frameworks (namely, submodular level-set constrained submodular optimization (SCSC/SCSK), difference-of-submodular (DS) optimization, and unconstrained submodular minimization (SFM)) that are be used to produce our resulting corpora. Section 4 describes a wide variety of possible functions that can be used for quality and complexity functions and the relative merits of each kind. Section 5 describes the process of how we selected appropriate quality and complexity functions from those described in Section 4 and uses them via the optimization methods described in Section 3 to produce the resulting corpora. This section also describes baseline recognition results using both Gaussian mixture model (GMM) and deep neural network (DNN)-based ASR systems. Lastly, Section 6 concludes.[2]

## 2. Background, goals, and previous work

A basic goal of our paper is to develop, test, and deploy methods for the selection of high-quality limited-complexity speech corpora — this means choosing a large and/or an acoustically rich subset $X$ of a *ground set* $V$ of speech utterances (e.g., $V$ might be the entire 309-h Switchboard-I dataset) but that has limited complexity (e.g., the subset should have small $n_t$, but there are other ways of measuring complexity as described in Section 4, all of which are compatible with our framework). More specifically, we wish to choose a subset $X \subseteq V$ that simultaneously has the following two properties:

1. *High quality*: The subset $X$ being high quality might mean the utterances $X$ constitute a large amount of speech, a large number of tokens, or be acoustically rich, diverse, and/or confusible (and hence useful to stress test an ASR system) in some way. We construct a function $g(X)$ that measures the quality of $X$, and we choose $X$ such that $g(X)$ is maximized. We argue in Section 4.1 that there are many submodular functions that naturally model quality.
2. *Low complexity*: Complexity may correspond to computational cost of running an ASR system, so an obvious complexity measure might be the vocabulary size $n_t$ of the subset $X$ (i.e., the number of distinct types in $X$). We define a function $f(X)$ that measures the complexity of $X$, and choose $X$ such that $f(X)$ is minimized. Section 4.2 argues that there are many submodular functions that naturally model complexity.

In fact, there are three goals of this paper. First, we wish to develop methods to produce such corpora, as described above. Second, we wish to produce the corpora themselves, establish state-of-the-art for both Gaussian mixture and deep neural network baselines for these corpora, and release the corpora definitions and baselines for free online. Third, we wish to evaluate a set of modern combinatorial optimization methods (namely submodular optimization) as a tool for producing these corpora and, in particular, empirically evaluate three variants of submodular optimization on large-scale real-world problems.

We next define more notation and terminology that will be useful throughout this document. We describe this section here to gain intuition, even though we defer the formal definition of submodularity and supermodularity to Section 3. We form a bipartite graph $G = (V, U, E)$, where $V$ (the left set of vertices) is the set of utterances in a corpus, and $U$ (the right vertices) is the set of types (distinct words) within that corpus as shown in Fig. 1. For $X \subseteq V$, we define the function $\gamma$: $2^V \to 2^U$ (so $\gamma(X) \subseteq U$) as the set of neighbors of $X$ in $U$. That is for any $X \subseteq V$,

$$\gamma(X) \triangleq \{u \in U : \exists v \in X \quad \text{s.t.} \quad (v, u) \in E\}. \tag{1}$$

[2] This paper is a significantly expanded version of Liu et al. (2015). This paper adds: (1) the full details of the three submodular algorithms used; (2) a large assortment of objectives and strategies that one can use to produce sub-corpora, and explanations regarding why a scientist may wish to use one over the other; (3) the details of the baseline ASR systems (both GMM- and DNN-based) for the resulting corpora; and (4) offers many more results in the form of the plethora of corpora, and statistics thereof, that were produced by the optimization methods.

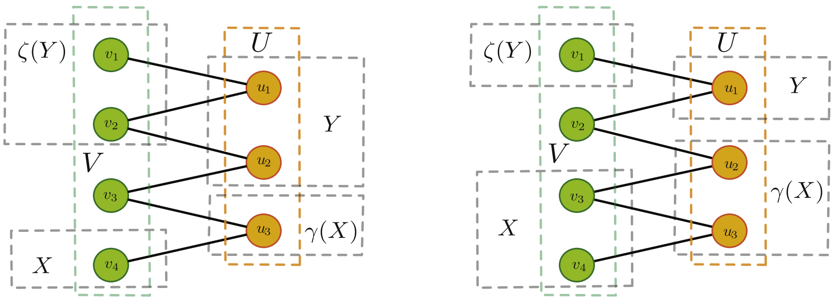

Fig. 1. Bipartite graph description. $V$ is set of utterances, and $U$ is set of types. The left and right examples show two different instances of $X \subseteq V$ and $Y \subseteq U$ and the corresponding values of $\gamma(X)$ and $\zeta(Y)$.

The function $\gamma(X)$ corresponds to the vocabulary words (types) corresponding to the utterances $X$. Also, for $Y \subseteq U$, we define the function $\zeta$: $2^U \rightarrow 2^V$ where $\zeta(Y)$ is the set of neighbors of $Y$ in $V$ and that themselves do not have a neighbors outside of the set $Y$. That is

$$\zeta(Y) \triangleq \{v \in V : \forall u \in U, (v,u) \in E \Rightarrow u \in Y\}. \tag{2}$$

Given a set of types $Y$, the function $\zeta(Y)$ gives the set of utterances that contain no vocabulary words (types) outside of $Y$. These functions are depicted in Fig. 1 for two examples of $X$ and $Y$. Let $w_V : 2^V \rightarrow \mathbb{R}_+$ be a function that gives a non-negative value $w_V(X)$ to every set of utterances $X \subseteq V$ as follows: $w_V(X) = \sum_{x \in X} w_V(x)$. As we will see in Section 3, such a function is known as a modular function. Also, let $w_U : 2^F \rightarrow \mathbb{R}_+$ be a modular function that gives a non-negative cost $w_U(Y) = \sum_{y \in Y} w_U(y)$ to a set of types $Y \subseteq U$, and using this we can valuate the cost of a set $X \subseteq V$ of utterances indirectly via $\gamma(\cdot)$ via $w_U(\gamma(X))$. As we will see in Section 3, $w_U(\gamma(X))$ is a submodular function in $X$ and $w_V(\zeta(Y))$ (Narayanan, 1997) is a supermodular function in $Y$. One very simple example uses $w_U(Y) = |Y|$ so that $|\gamma(X)|$ is the number of distinct types that are contained in the set of utterances $X$, and $w_V(X) = |X|$ so that $w_V(X)$ is the number of utterances in $X$. Also, $|\zeta(Y)|$ is the total number of utterances that do not contain types outside of the set $Y$. Hence, one way to maximize "quality" while minimizing "complexity" is to find a set $X \subseteq V$ that has large $w_V(X)$ value but small cost $w_U(\gamma(X))$, and Section 3 describes three algorithmic methods that can do this.

In an initial study (King et al., 2005), a heuristic was proposed to select different subsets of Switchboard (with vocabulary size of 10, 25, 50, 100, 250, and 500 words). The resulting corpora, named "SVitchboard I", are available online at http://tinyurl.com/svitchboardI. The heuristic greedily selected the most frequent word in the transcripts until the vocabulary size constraints were met. Specifically, starting with a boot vocabulary $Y \leftarrow Y_0$ where $Y_0 \subseteq U$ is the most frequent vocabulary words of a given size, we select, in each greedy step, an out-of-vocabulary (OOV) word to add to the vocabulary if the amount of data containing only and nothing other than this new vocabulary is maximized. That is, given $Y$, we next choose $y \in U \backslash Y$ that maximizes $w_V(\zeta(Y+y))$ where $w_V(v)$ gives the number of word tokens in utterance $v \in V$.[3] The algorithm stops when the desired vocabulary size is reached.

The greedy algorithm, although conceptually simple, can perform, as described in Lin and Bilmes (2011b), arbitrarily poorly for the corpus subset selection problem as solved by the above greedy algorithm. To see a simple example, let $U = \{a, b, c\}$ be a set with only three vocabulary types. Define a function $h$ as $h(\{a\}) = 1$, $h(\{b\}) = h(\{c\}) = 0$, $h(\{a, b\}) = h(\{a, c\}) = 1$, and $h(\{b, c\}) = p > 1$ and the goal is to choose two words (the cardinality constraint). The function $h$ can be verified as being supermodular (Section 3). Therefore, the greedy algorithm above attempts to solve the problem of maximizing a supermodular function subject to a cardinality constraint. Greedily maximizing $h$ leads to a solution $\{a, b\}$ with objective function value 1, while the true optimal objective function value is $p$ on set $\{b, c\}$. Since $p$ is an arbitrarily large non-negative value, the approximation factor for this example (the ratio of the resulting valuation and the optimal valuation, or $1/p$) can be made unboundedly poor. This is unsurprising, as $w_V(\zeta(Y)$ is a supermodular function in $Y$, and it is well known that the greedy algorithm works near-optimally when maximizing a *submodular* but not a *supermodular function* function subject to a cardinality constraint (Nemhauser et al., 1978).

A second approach to this problem was given in Lin and Bilmes (2011b). Here, given a quality function $g(X) = |X|$ and a vocabulary-size complexity function $f(.) = |\gamma(X)|$, a surrogate function is formed as $|\gamma(V \setminus X)| - |X|$ which

[3] We define $Y + y$ as the set $Y \cup \{y\}$ for notational convenience.

happens not only to be submodular but also “graph representable”, meaning we can minimize this function exactly in low-order polynomial time using max-flow/min-cut algorithms. This work produced results that were seen to be better than King et al. (2005), but no corpus was released, and it did not consider richer quality $g(\cdot)$ and complexity $f(\cdot)$ functions but that might not both be graph representable.

In the present work, we investigate several principled approaches to data selection using submodular function (Fujishige, 2005) optimization for an arbitrary submodular quality $g$ and complexity $f$ function. Our approach, in fact, builds upon Lin and Bilmes (2011b) where a subclass of the algorithms we present here was considered in Lin and Bilmes (2011b) for subselecting data. In fact, when $f$ and $g$ are graph representable, then the optimization method in Lin and Bilmes (2011b) corresponds to a min-cut/max-flow solution to the approach given in Section 3.1. In this work, Section 3.1, which addresses the problem as a form of unconstrained submodular minimization (SFM), not only allows $f$ and $g$ to be arbitrary submodular functions (and there are many that could be useful, as described in Section 4), but we also introduce and test two new algorithmic approaches to finding a set that has high quality $g$ and low complexity $f$, namely the submodular level-set constrained submodular optimization (SCSC/SCSK) approach in Section 3.2 and the difference-of-submodular (DS) approach in Section 3.3. Moreover, Lin and Bilmes (2011b) only proposed and tested subselection algorithms for this problem, but it did not produce experimental speech recognition results, provide resulting corpora definitions, nor publicly release the resulting corpora definitions. Here, we consider a more general class of algorithms (that includes those proposed in Lin and Bilmes (2011b)) and use them to find the very best possible set of *corpora* in terms of a variety of useful statistics. Furthermore, we select training, development, and test sets based on the resulting corpora and then produce baseline GMM-HMM and DNN based ASR systems, and Kaldi recipes, on these corpora that can be useful as comparisons for future research using these corpora.

## 3. Submodular functions

Submodular set functions are those that have the “diminishing returns” property. Given a finite set $V$, a set function $f: 2^V \rightarrow \mathbb{R}$ is said to be submodular if $f(A \cup \{v\}) - f(A) \geq f(B \cup \{v\}) - f(B)$ holds $\forall A \subseteq B \subseteq V$ and $v \notin B$. This means that the incremental value (or “gain” of element $v$), defined as $f(v|A) \triangleq f(A \cup \{v\}) - f(A)$, decreases as the context in which $v$ is considered grows from $A$ to $B$. Hence, $f$ is submodular if $f(v|A) \geq f(v|B)$ for all $A \subseteq B \subseteq V \setminus \{v\}$. Submodularity can equivalently be defined as those functions having $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for all $A, B \subseteq V$. *Supermodular* functions are those that have the inequality reversed, i.e., $h$ is supermodular if $h(v|A) \leq h(v|B)$ for all $A \subseteq B \subseteq V \setminus \{v\}$. We also define *modular* functions as those that are both submodular and supermodular, i.e., they satisfy the above inequality everywhere with equality. All modular functions $m: 2^V \rightarrow \mathbb{R}$ that are also normalized $m(\phi) = 0$ correspond to a $|V|$-dimensional vector, and for any $X \subseteq V$, a modular function valuates a set $X$ as $m(X) = \sum_{x \in X} m(x)$.

Submodular functions have shown good performance in several real world applications such as feature selection Krause et al. (2008b); Das and Kempe (2011); Liu et al. (2013); Kirchhoff et al. (2013), clustering (Narasimhan et al., 2005), structure learning (Narasimhan and Bilmes, 2004), document summarization (Lin and Bilmes, 2011a; 2012), image collection summarization (Tschiatschek et al., 2014), speech training data selection (Lin and Bilmes, 2009; Wei et al., 2014), sensor placement (Krause et al., 2008a), and many others. In this paper (in Section 4), we in fact show that there are many appropriate and natural submodular instantiations of the quality $g$ and complexity function $f$ that could be used. This, along with algorithms that, in one way or another, simultaneously minimize a submodular complexity function $f$ while maximizing another submodular quality function $g$ provide a more principled approach to corpus selection than just by using heuristics.

We consider and evaluate three distinct optimization strategies. The first is based on simple unconstrained submodular function minimization (SFM), which involves producing an approximate objective that can be solved exactly in polytime, although it can be slow in practice depending on the function. The second is based on constrained submodular optimization (SCSC/SCSK), in particular given two submodular functions, we optimize one under the constraint that the other must be bounded by a constant. Here, the objective is exact since we can use the exact quality and complexity functions, but an exact solution is not possible. Fortunately, approximation solutions (having mathematical guarantees) can be produced in this case in polynomial time. The third approach is based on optimizing a difference between two submodular (DS) functions—this is also an approximate objective, but good and fast heuristics exist that can produce solutions that typically are high quality. Hence, we have the age-old problem of solving a problem either by: (1) solving an approximate objective exactly, (2) solving an exact objective approximately, or (3) solving an approximate objective approximately. What is missing is (4) solving the exact objective exactly, which is also something we can do if we restrict the submodular functions to be piece-wise-linear concave over modular, or if we restrict the class of submodular functions to be only graph representable, as was done in Lin and Bilmes (2011b). Both of these cases, however, can again be seen as producing an approximate objective, and therefore optimizing the exact objective exactly, in general, is intractable.

## 3.1. Unconstrained submodular function minimization

The first approach we consider is general unconstrained submodular function minimization (SFM).

$$\text{Problem 1 (SFM)}: \quad \min_{X \subseteq V} h(X) \tag{3}$$

where $h(X) = g(V \setminus X) + \lambda f(X)$ is a submodular function. Here, we minimize $g(V \setminus X)$, the quality of $X$'s complement $V \setminus X$, rather than maximizing the quality of $X$. Since $g(\cdot)$ is submodular monotone non-decreasing and normalized (i.e., $g(\phi) = 0$), we have $g(V \setminus X) \geq g(V) - g(X)$. Hence, we have the relationship:

$$h(X) = g(V \setminus X) + \lambda f(X) \geq g(V) - g(X) + \lambda f(X) \tag{4}$$

Since $g(V)$ is a constant, minimizing $g(V) - g(X) + \lambda f(X)$ over $X$ is the same as minimizing $\lambda f(X) - g(X)$ which exactly corresponds to maximizing $g$ while minimizing a scaled $f$. Thus, minimizing $h$ means we minimize an upper bound of an objective that is desirable but intractable directly to minimize.

If $g(\cdot)$ is a modular function (as it is when $g(X) = w_V(X)$, see below), then we have that $g(V \setminus X) = g(V) - g(X)$ and the SFM approach exactly maximizes the objective $h(X) = g(X) - \lambda f(X)$. Hence, if we sacrifice some flexibility on the quality function (namely to use a modular rather than a submodular function) then we can get an exact solution. Indeed, this is exactly the approach used in Lin and Bilmes (2011b) where they used a modular quality function and a graph representable submodular complexity function $f(.) = |\gamma(X)|$ as described in Section 2. Hence the SFM approach directly generalizes that of Lin and Bilmes (2011b) and allows more general quality functions.

Another advantage of this approach is that, even when $g$ is submodular, solutions for *all* possible values of $\lambda$ and finding solutions for a particular $\lambda$ have the same complexity, thanks to the *principle partition* of submodular systems (see Fujishige, 2005; Lin and Bilmes, 2010).

On the other hand, as mentioned above, a potential disadvantage with this approach for our problem is that for submodular $g(\cdot)$, we minimize only an upper bound of our goal. Another disadvantage is that we have to accept the solutions that we get for different values of $\lambda$, and in general there is only a small set of critical values of $\lambda$ that matter—values of lambda other than a set of critical values will produce solutions identical to one of the critical values of $\lambda$. Hence, it is harder to precisely control the complexity (i.e., if we want a corpus that has an exact vocabulary size upper bound, there might not be a critical value of $\lambda$ that exactly achieves this particular constraint with equality). Finally, for submodular $g$ and $f$, general purpose submodular function minimization, while theoretically requiring polynomial time, can be slow in practice. However, there are relatively fast and practical algorithms for general purpose SFM, such as the minimum-norm point algorithm (Fujishige, 2005; Fujishige et al., 2006) and that we use below.

## 3.2. Constrained submodular optimization

Iyer and Bilmes (2013) defined a number of algorithms to solve the following two constrained submodular optimization problems, referred to as "Submodular Cost Submodular Cover (SCSC)", and "Submodular Cost Submodular Knapsack" (SCSK), respectively:

$$\text{Problem 2 (SCSC)}: \quad \min\{f(X) | g(X) \geq c\}, \text{and} \tag{5}$$

$$\text{Problem 3 (SCSK)}: \quad \max\{g(X) | f(X) \leq b\}, \tag{6}$$

where both $g: 2^V \to \mathbb{R}_+$ and $f: 2^V \to \mathbb{R}_+$ are polymatroid (non-negative monotone-nondecreasing submodular) functions.

This approach exactly addresses the problem we wish to solve. In particular, we can use the formulation of Problem 2 and directly enforce constraints on the vocabulary size while maximizing the quality. Unlike submodular function minimization, however, this problem is NP-hard (Iyer and Bilmes, 2013). Several of the algorithms proposed in Iyer and Bilmes (2013), however, are scalable and admit bounded approximation guarantees. In this paper, we use the iterative submodular knapsack algorithm, outlined in Section 4.2 in Iyer and Bilmes (2013).

## 3.3. *Difference of submodular functions optimization*

The third approach we consider minimizes the difference between submodular functions (Iyer and Bilmes, 2012; Narasimhan and Bilmes, 2005):

$$\text{Problem 3 (DS)}: \quad \min_{X \subseteq V} v(X) \tag{7}$$

where $v(X) = \lambda f(X) - g(X)$ is a difference of two submodular functions. Similar to SCSC/SCSK, this method addresses the underlying problem; different values of $\lambda$ will amount to different vocabulary sizes. Unfortunately, unlike SCSC and SCSK, we do not have explicit control over a constraint on $f$ (and thus the vocabulary size, when we choose $f(X) = |\gamma(X)|$) and we instead need to tune $\lambda$ to obtain the best solution. Like SCSC/SCSK, exact DS optimization is intractable, and in fact inapproximable, but the algorithms proposed in Iyer and Bilmes (2012) and Narasimhan and Bilmes (2005) are scalable and work well in practice on typical real-world functions (Iyer and Bilmes, 2012).

## 3.4. *Further discussion of the three approaches and our goals*

With these new algorithms, therefore, we can directly address the problem of high quality limited complexity corpus selection. Considering Problem II, for example, we can set a given budget $b$ to be, say a particular vocabulary size (e.g., $b \in \{10, 50, 100, 500, 1000, 5000, 10{,}000\}$) which would work if we choose $f(X) = |\gamma(X)|$. Any feasible solution $X$ s.t. $f(X) \leq b$, therefore, will naturally satisfy a vocabulary size constraint.

The advantages of Eq. (6) over Eq. (3) is that feasible solutions are the norm rather than the hope, i.e., if we want, say a 5000 word corpus, there might be no value of $\lambda$ in Eq. (3) that gets us a vocabulary size close to this. In Eq. (6), however, we are (in theory) never considering corpora that violate the constraint. The other advantages of Eq. (6) is that the algorithms defined in Iyer and Bilmes (2013), while approximate, should be fast and scalable, and will probably run much faster than general SFM (or the principle partition).

The potential problem, however, is that the algorithms in Iyer and Bilmes (2013) are only bi-approximate, and the resulting solutions might not always be exactly feasible. In any event, one of our goals in this work is to observe which of SFM, SCSC/SCSK, and DS yields superior corpora. To do this, we choose a particular $g(\cdot)$ and $f(\cdot)$ function, try them with SFM, SCSC/SCSK, and DS, run statistics on the resulting corpus, and then see which resulting corpus is better via these statistics. The best resulting corpora are those that we then establish baselines for, and then release. In this sense, we use the tools of submodular optimization to give us a low-dimensional parameter space to search for good corpora over (which includes the choice of algorithm, and the hyperparameter $\lambda$)—this is an inherently feasible task, unlike the combinatorial explosion that would be require to search over all possible corpora subsets.

## 4. Corpus creation via various $g(\cdot)$ and $f(\cdot)$

There are a plethora of different $g(\cdot)$ and $f(\cdot)$ functions we can choose from for the problem statement. In this section, we will describe different function instantiations for $g(\cdot)$ and $f(\cdot)$. We then choose several of them to test in our experiments. While we do not test all of the $g(\cdot)$ and $f(\cdot)$ functions we list in this section, we still include this compilation in order both to show the flexibility of our framework and also to offer ideas for future research in engineering good corpora.

### 4.1. *Quality functions*

We start with four different modular functions as quality functions $g$. All are normalized so that $g(\phi) = 0$ and $g(V) = 1$. The followings are the instantiations of $g(\cdot)$:

1. *Utterance count*: $g_1(X) = |X|/|V|$. This instantiation defines high quality as containing large percentage of utterances in $V$. Each utterance (short or long) is given equal weight.
2. *Amount of speech*: $g_2(X) = w_V(X)/w_V(V)$ where $w_V(v)$ measures how much speech (excluding silence) is in the acoustic signal $v$. Thus, longer utterances are preferred over shorter ones.
3. *Number of tokens*: $g_3(X) = w_V(X)/w_V(V)$ where $w_V(v)$ measures how many tokens are contained in the transcription of utterance $v$.
4. *Intra-utterance acoustic dispersion*. $g_4(X) = w_V(X)/w_V(V)$ where $w_V(v)$ measures the "acoustic dispersion" within utterance $v$. If $x^v = (x_1^v, x_2^v, \cdots, x_T^v)$ is a sequence of, say, MFCC or other acoustic vectors for utterance $v$, then we can measure acoustic dispersion via

$$w_V(v) = \frac{1}{T^2}\left|\sum_{i=1}^{T}\sum_{j=1}^{T}(x_i^v - x_j^v)^{\mathsf{T}}(x_i^v - x_j^v)\right| \tag{8}$$

Here, we prefer an utterance if it is internally acoustically diverse.

5. *Utterance error rate*: $g(X) = w_V(X)/w_V(V)$ where $w_V(v)$ measures the relative number of errors that utterance $v$ has in the corpus for a speech recognition system, i.e., we might wish to choose utterances that are inherently hard so that results on the subselected corpus act as a representative of worst-case performance for the full corpus. This could be useful to produce a difficult small corpus, or for use in boosting-like procedures.One very simple possibility might be to have $w_V(v) = 1$ iff $v$ had an error. Another possibility is to do relative number of errors (i.e., relative to the number of tokens in $v$).Note that we might also want to consider this in the context of long vs. short utterances, i.e., a long utterance with an error rate of 50% might be more valuable than 80% error rate on a short utterance. While we can always mix this function together with *speech amount*, it may make more sense to have a function where *speech amount* and *utterance error rate* interact more intimately.

All the above functions are modular, i.e., the score of an utterance does not interact with the score of another. Thus, there is a high chance of choosing a set $X$ that has high quality but that is also redundant. As an extreme example, if a corpus had duplicate entries which are very high quality, both would be chosen even though the corpus diversity would not improve.

To address this, we can utilize a strictly submodular function for $g(\cdot)$ in order to express desire for not only a high quality but also a diverse set of utterances. Our guide will be the functions that were used for speech subset selection (Wei et al., 2014; 2013) which is a different application and set of goals (since in that work, $f(\cdot)$ is always modular and we can often solve the problem of maximizing $g(X) - \lambda f(X)$ with submodular maximization procedures). The quality submodular functions used in that research still work quite naturally for our quality functions as well. We draw directly from Wei et al. (2013),Wei et al. (2014) to outline some possible candidates.

The first instantiation of a submodular $g(\cdot)$ is called a feature-based function, and is defined as follows:

6. *Feature-based*. Here, $g(X) = g_5(X)/g_5(V)$, where

$$g_5(X) = \sum_{u \in \mathcal{U}} \phi(m_u(X)) \tag{9}$$

where $\varphi(\cdot)$ is a non-negative monotone non-decreasing concave function, $\mathcal{U}$ is a set of features, and $m_u(S) = \sum_{j \in S} m_u(j)$ is a non-negative score for feature $u$ in set $S$, with $m_u(j)$ measuring the degree to which utterance $j \in S$ possesses feature $u$. Maximizing this objective naturally encourages diversity and coverage of the features within the chosen set $X$ of elements. We note that Eq. (9) is a sum of concave functions over modular functions, and is hence submodular.

Each term in the sum be based on a "feature" of the objects being scored. The feature based submodular functions are convenient for applications in speech processing since speech objects can often be described by a variety of phonetic or prosodic feature labels (e.g. phonemes, triphones, words, syllables, tones, etc.). Feature-based submodular functions, therefore, have the ability to leverage much of the important work on both knowledge- and data-driven feature engineering that has been available in speech processing. This class of submodular functions moreover avoids the use of a pair-wise similarity graph over utterances, as in Eq. (10), and hence, also unlike Eq. (10) avoids an expensive $O(|V|^2)$ function evaluation cost.

In Wei et al. (2014), $\mathcal{U}$ is the set of triphones over frame labels that are derived from the word transcriptions via a forced Viterbi alignment of a trained system. The function $\varphi()$ is the square root function. The score $m_u(s)$ is the count of feature $u$ in element $s$, normalized by term frequency-inverse document frequency (TF-IDF), i.e., $m_u(s) = \text{TF}_u(s) \times \text{IDF}_u(s)$, where $\text{TF}_u(s)$ is the count of feature $u$ in $s$, and $\text{IDF}_u = \log\left(\frac{|V|}{d(u)}\right)$ is the inverse document count of the feature $u$ with $d(u)$ being the number of utterances that contain the feature $u$ (each utterance is considered a "document").

We also introduce two additional (and more expensive) possible function instantiations for $g(\cdot)$.

7. *Facility location*: Here, $g(X) = g_{\text{fac}}(X)/g_{\text{fac}}(V)$ where $g_{\text{fac}}(X)$ is defined as

$$g_{\text{fac}}(X) = \sum_{i \in V} \max_{j \in X} w_{ij}, \tag{10}$$

where $w_{ij} \geq 0$ indicates the similarity (or affinity) between utterance $i$ and $j$. The similarity measure $w_{ij}$ may be computed by kernels derived from discrete representations of the acoustic utterance $i$ and $j$, as we do below. More specifically, a tokenizer is run over the acoustic signal that converts it into a sequence of discrete labels. Then a TF-IDF kernel or string kernel is used to compute the pair-wise similarity between the sequences of discrete labels of two speech utterances. In general, however, any non-negative similarity between utterances may be used.
8. *Saturated coverage*: Here, $g(X) = g_{\text{sat}}(X)/g_{\text{sat}}(V)$ where $g_{\text{sat}}(X)$ is defined as follows:

$$g_{\text{sat}}(X) = \sum_{i \in V} \min\{C_i(X), \alpha C_i(V)\}, \tag{11}$$

where $C_i(X) = \sum_{j \in X} w_{ij}$ and $0 \leq \alpha \leq 1$ is a saturation coefficient. Like the above, $w_{ij}$ is a non-negative similarity score between the corresponding two utterances $i$ and $j$.

We refer to both *facility location* and *saturated coverage* as graph-based submodular functions since a pair-wise similarity graph is required, i.e., $w_{ij}$ needs to be computed for all $i \in V$ and $j \in V$. This is an expensive objective function because of a time complexity of $O(|V|^2)$ and a memory complexity of $O(|V|^2)$ for the graph construction. In our task of large-scale speech data subset selection, the whole speech corpus is segmented into about 1.3 million individual segments; thus $|V| \approx 1.3e7$. Even with highly optimized data structures, efficient computation of similarity measures, and nearest neighbor graph approximation, graph construction presents a computational challenge for the application of such graph-based submodular objectives on large-scale speech data subset selection. Thus, in large-scale data selection, we tend to prefer feature-based functions to the graph-based functions.

## 4.2. Complexity functions

One obvious candidate of the complexity functions $f(\cdot)$ is the vocabulary size of $X$ (i.e., $f(.) = |\gamma(X)|$) since it allows us to directly control the vocabulary size. When $f(\cdot)$ is not the vocabulary size, then any given budget would be based not in terms of vocabulary size but something else. One of our primary goals in this work is to produce, define baselines over, and then publicly release a set of rich and realistic corpora that are useful for rapidly prototyping novel speech recognition algorithms. Our approach is to test a variety of $f(\cdot)$ functions, study the statistics of the resulting corpora, and repeat until we obtain an attractive result. In other words, varying and then optimizing with $f(\cdot)$ functions may be as tool to achieve one of our ultimate goals. Some $f(\cdot)$ functions might produce better corpora than others. For example, if we measure only vocabulary size with $f(\cdot)$, we might end up with a corpus consisting primarily of stop words in its lexicon. When $f(\cdot)$ is not vocabulary size, but we still wish to place a bound on vocabulary size (since that is an important limiter of the complexity of testing novel ASR methods), we would need to run optimization with various different budgets and hope that there are solutions that have vocabulary sizes close to the $\{10, 50, 100, 500, 1000, 5000, 10{,}000\}$ sizes we desire. In the worst of cases, we might not have exactly those desired sizes. If, say, the resulting vocabulary sizes were 9, 52, 103, 505, ..., rather than 10, 50, 100, 500, ..., these numbers are not so far off as to appreciably change the complexity requirements for studying novel ASR systems. These sizes, therefore, would suffice as long as the other properties of the corpus are still good. This, it turns out, is indeed the case as we show below.

To guide the design of good complexity functions, we must consider what makes a "good" corpus. In general, we wish the corpus to have many of the properties of a big corpus (e.g., acoustically interesting and diverse, rich set of hard vocabulary words, representative of and a good surrogate for the large corpus) but at the same time we wish it to be small (e.g., a small vocabulary size, no OOV issues) in order to allow for rapid prototyping of extremely novel ASR methods. As is the case the quality functions, the complexity function $f(\cdot)$ can be either modular or submodular. However, since in all cases complexity is based either directly or indirectly on the number of types in a resultant corpus, all complexity functions we consider below are submodular. These functions, however, can be sub-divided based on if they are either modular or submodular on the set of types $U$, as we next show.

### 4.2.1. *Functions submodular on* V *but modular on* U.

In this first case, $f(X) = w_U(\gamma(X)) = \sum_{u \in \gamma(X)} w_U(u)$, where $\gamma(X)$ is the vocabulary associated with $X$, and $w_U(u)$ is a function that indicates the undesirability of word $u$. A larger $w_U(u)$ states that word type $u$ is less important. This allows certain desirable properties of the vocabulary of the resultant corpus to be expressed.

There are a number of ways of defining the modular function $w_U : 2^U \to \mathbb{R}_+$ on subsets $Y \subseteq U$.

1. *Vocabulary size.* Here $w_U(Y) = |Y|$. This represents the collective vocabulary size of utterances in set $X$ and expresses desire for small vocabulary size. All words are equally and uniformly undesirable.
2. *Intra-word number of phones.* Here, $w_U(Y) = \sum_{y \in Y} \frac{C}{q_y}$ where $q_y$ is the number of phonemes in the pronunciation of word $y$ and $C$ is a non-negative constant. That is, types that have a larger number of phonemes are preferred (they are less undesirable).
3. *Number of pronunciations.* Here, $w_U(Y) = \sum_{y \in Y} \frac{C}{q_y}$ where $q_y$ is the number of pronunciations in a dictionary that word $y$ has. This makes a word more desirable when it has many pronunciations (meaning it is a harder word, making the resulting corpus more confusible since there are likely to be distinct types that might have similar pronunciations).A variant of this approach would exclude the initial and final phone of a word when measuring the count—the reason is that words with the same internals but only boundary differences might still be easy to distinguish.
4. *Intra-word phonetic diversity via lattice density of word pronunciations.* A variant of *number of pronunciations* would produce a phone lattice from the dictionary entries for all the pronunciations of a word, and compute the lattice density, which is often a good measure of intra-word phonetic diversity, i.e., we would take $w_U(y) = 1/(\text{phone lattice density for word } y)$ to express preference for high lattice density.A further variant of this idea would compute the max-cut within the lattice rather than the overall lattice density. Still another variant of this approach would be lattice-based diversity but that uses acoustic information as well. For example, let $y$ be a given type, and let $\{x^{y,i}\}_i$ be a set of acoustic segments corresponding to this type, i.e., consider in a forced Viterbi alignment of the training data all segments within utterances that correspond to type $y$. For each such segment, it is possible to produce a lattice representation of an $n$-best list state output of a decoding of the segment. In other words, for each acoustic segment $x^{y,i}$ we would produce an ASR-system state lattice. Once we collect all of the state lattices for every segment corresponding to a given type, we can measure the overall lattice density (or max cut) in a lattice that combines all of the segment-specific lattices. Hence, this is a hybrid dispersion of combined state lattices of segments for a given type.We note that the approach above requires an initial working baseline ASR system to produce these lattices and statistics thereon. The implicit assumption above is that a corpus developed based on the statistics computed from this baseline ASR system would still be attractive to other, quite different, ASR systems. This may or may not be true, depending on how different the baseline ASR system is from the novel ASR system. To the extent that it is not true, however, it should be possible to develop a staged approach, whereby once a new system methodology has been prototyped based on an initial corpus, and once that system in its final form has been trained on a full training set, that latter system can be then used to generate corpus statistics for a second stage selection process. This procedure can also iterate to produce a sequence of more refined corpora.
5. *Intra-word phonetic diversity via entropic acoustic dispersion.* Here, $w_U(y)$ would be based on an HMM-like model divergence measure. For example, given an HMM $p(x|y)$ for word $y$, we could compute $w_U(y) = 1/H(p(x|y))$ where $H$ is the entropy function. This would prefer words that have high intra-word variability (or high acoustic dispersion)

as measured by an acoustic measure.There are many different ways to produce the HMM. For example, it would be whole-word unsupervised, whole word from a trained ASR system, or phone based from an trained LVCSR system (this last instance is probably the one that will work best). It would also be straightforward to use more recent DNN- or RNN-based acoustic models to compute acoustic dispersion.

6. *Intra-word phonetic diversity via alignment acoustic dispersion.* Here, $w_U(y)$ would be based on pairwise acoustic alignments of various segments corresponding to type $y$, i.e., gathering all acoustic segments $S_y$ that have been aligned (in a forced alignment) with type $y$, we can then subsequently do a set of pairwise alignments for all pairs within $S_y$ (using, say, a 2-norm or Mahalanobis distance for frame distance), and then average these alignment costs together, $w_U(y)$ would be inversely proportional to that (so again, it prefers types that have diverse instances).A variant of this is to, rather than the acoustic segments being aligned to each other, each segment is aligned to a model. i.e., the forced Viterbi path could be used for this purpose. Then the Viterbi alignments corresponding to the acoustic segments could be used to produce a dispersion measure for type $y$.In any case, we see that whatever the measure of dispersion, we would have $w_U(y) = C/\text{dispersion}(y)$ for some constant $C$.
7. *Intra-word error rate.* Some words are inherently more confusible than other words are. Another option for is to take $w_U(y)$ as the accuracy of word $y$. Hence, we express desire for a corpus of words that have high error, and are inherently difficult. This will lead to a more difficult subselected corpus but one that is more likely to be indicative of results on the full corpus (since the hard words, the ones that are often confused are given special treatment).

### 4.2.2. *Functions submodular on* V *and also submodular on* $U$

The functions above were modular on $U$ in the sense that they all involved indirectly selecting a subset, say $S \subseteq U$ based on a subset $X \subset V$, and then valuating $X$ based on a weighted sum over $S$. For this reason, we consider them to be modular over $U$. Complexity functions $f(\cdot)$ that are also submodular over subsets of the set of $U$ offer additional capability and nuance for selecting good corpora. The functions we outline in this section are submodular over $U$ in the sense that they involve applying additional concave functions over weighted subsets of $U$, an operation that is guaranteed to preserver submodularity (Lin and Bilmes, 2011a). The following constitutes a set of candidate functions.

1. *Cooperation within blocks of a partition.* Here, we partition the types $U = U_1 \cup U_2 \cup \cdots \cup U_K$ into $K$ (necessarily disjoint) blocks and form a function of the form:

$$f(X) = \sum_{k=1}^{K} \phi[w_U(\gamma(X) \cap U_k)], \tag{12}$$

where $\varphi$ is a non-negative non-decreasing concave function. In general, such a function says that to minimize $f(\cdot)$ we wish to choose words that live in the same block of the partition rather than spreading out words over many different blocks.

2. *Confusibility functions*. In order to produce a corpus that is challenging, we may wish to select vocabulary words that are (acoustically) confusible with each other. Two words $y_1, y_2 \in U$ are confusible if it is easy, when one gets an instance of $y_1$, to confuse it with $y_2$ (and perhaps vice versa). Confusible groups are typically a problem for speech recognizers. If we, for example, chose only words in a sub-corpus that were very non-confusible with each other, building a recognizer for the corpus would be relatively easy. A corpus with many confusible words would be much harder and building a system for it would be more indicative of the performance on the full data.The partition function in Eq. (12) suits this goal perfectly. Here, each $U_k$ would be a confusible group, and to minimize complexity we would try to choose as many mutually confusible words as possible.A variant of this would be to utilize and then represent a form of directionality. For example, perhaps $y_1$ is easily confused with $y_2$ but not vice versa. More precisely, this can happen if, given a probabilistic model over words $y$ given acoustics $x$, $p(y|x)$, and given two typical samples $x_{y_1}$ and $x_{y_2}$, respectively, of words $y_1$ and $y_2$, we may have that $p(y_2|x_{y_1}) \approx p(y_1|x_{y_1})$ but that $p(y_2|x_{y_2}) \gg p(y_1|x_{y_2})$. From the perspective of a submodular complexity function $f(\cdot)$, this would require that $f(y_2|y_1) < f(y_1|y_2)$ which implies that $f(y_2) < f(y_1)$. This idea cannot be extended to more than a pair, however, since if $y_1$ is confusible with $y_2$, and $y_2$ is confusible with $y_3$, and, say, $y_3$ is confusible with $y_1$ (an intransitivity), we cannot adjust $f(y_1)$, $f(y_2)$, $f(y_3)$ to express this. However, if the confusibility graph is transitive (meaning, there are no directed cycles), we can find single values $f(y_i)$ that will express it (very similar to the case of utilities in game theory).

## 5. Experimental results

As described in Section 2, we have three main goals of this paper: develop methods to produce corpora, produce and then release the corpora, and evaluate the utility of three submodular optimization strategies for this purpose. Having established a set of essential tools and options in the preceding sections, we proceed with these goals in this section.

### 5.1. Comparison of different algorithms

In the previous section, we proposed three different algorithms with different $f(\cdot)$ and $g(\cdot)$ instantiations. Our goal here is to create subsets using different submodular optimization algorithms as well as function instantiations.

To produce the initial ground set of utterances, for Switchboard I and Fisher we first remove utterances containing the disfluencies and fillers. For example, we remove utterances that contain only word fragments (e.g. sim[ilar]-), *uh*, [noise], *yeah*, [laughter], *huh, hm*, [laughter-*], *uh-huh, um-hum hum, huh-uh, um*. The size of the resulting ground sets $V$ for Switchboard I and Fisher is then 93,312, and 1.7 million, respectively. For Switchboard I, we test all three algorithms and function instantiations for a given target vocabulary size. Because of its large ground set size, for Fisher, we use the SCSK algorithm with $g_5$ as the quality function because of the scalability and the computational efficiency of the semigradient-based SCSK algorithms (Iyer and Bilmes, 2013).

To choose the best resulting corpus for a particular target vocabulary size, we run each of the optimization methods (Sections 3.1–3.3) that gives us a relatively small number of corpora to choose from. We then compute a set of statistics on each of the resulting corpora such as the actual vocabulary size, average number of phonemes per word, the number of utterances and tokens, the speech durations, etc. We also compute the value $g_5$ for each subset; note that the $g_5$ function measures the representativeness of the subsets. We believe this value is a good indicator of corpus diversity. To show a corpus's phonetic balance, we compute the normalized entropy of the phoneme distribution $\frac{H(p)}{\log(43)}$ and the normalized entropy of the non-silence phoneme distribution $\frac{H(p)}{\log(42)}$, where we use 43 phones in the lexicon with 42 non-silence phones. $H(p)$ is the entropy of the probability distribution over phonemes in the selected subset.

In order to have the best final corpus for the current vocabulary size, we make the final corpus selection by visual inspection of these statistics (i.e., by hand). Table 1 shows the statistics of our chosen corpora, comprising both SVitchboard II and FiSVer I. Table 3 shows the specific algorithm and function instantiations we used to create each subset in Svitchboard II. The twelve tables (Tables 7–18) list the statistics of all of the corpora that resulted from the algorithms of Sections 3.1–3.3. We establish baseline systems for only our chosen sets listed in Table 3.

### 5.2. Comparison to random selection and SVB-I

We also compare our Svitchboard-II datasets to Svitchboard-I dataset (King et al., 2005) and random selected subsets. The random selection is done by repeatedly randomly picking an utterance and adding it to current corpus, and the selection is terminated once the target vocabulary is met. We do 100 random selections for each target vocabulary task. Table 2 shows the comparison of random selected subsets, Svitchboard-I and Svitchboard-II. For each target vocabulary size (50, 100 and 500), Svitchboard-II provides much more acoustic data. Svitchboard-I datasets provide subsets with vocabulary sizes only of 10, 25, 50, 100, 250, and 500; while Svitchboard-II provides corpora with vocabulary sizes up to 10,000 and having more than 60 h of speech. A 10,000 word acoustically rich subset of switchboard is a useful size even in today's big-data environment.

In addition, we compare the vocabularies in Svitchboard-I and the new Svitchboard-II datasets — Figs. 2–4 show the vocabulary difference between Svitchboard-I and Svitchboard-II dataset. Clearly, the newer methods produce a more interesting, richer, and phonetically diverse set of vocabulary words, while keeping the vocabulary size the same, but also while choosing more acoustic data. This shows a clear advantage of Svitchboard-II over Svitchboard-I.

### 5.3. Data partition for cross-validation

For SVitchboard-II, our baselines define a cross-validation procedure. The conversation sides in each subsets are split into 5 non-overlapping folds; each conversation only exists in one fold. Similar to King et al. (2005), we denote

Table 1
Statistics of SVitchboard-II (top table) and FiSVer-I (bottom table) datasets. Vocab size: actual vocabulary size; Avg. Phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 50 | 3.32 | 24,033 | 38,154 | 4.01 | 4.13054e9 | 4491 | 0.4688 | 0.3916 |
| 100 | 100 | 3.28 | 27,228 | 51,254 | 4.93 | 4.8425e9 | 4571 | 0.4998 | 0.4203 |
| 500 | 500 | 3.95 | 39,694 | 131,815 | 10.30 | 7.70767e9 | 4749 | 0.6122 | 0.5243 |
| 1000 | 1001 | 4.50 | 48,445 | 230,876 | 16.81 | 9.70981e9 | 4801 | 0.6831 | 0.5911 |
| 5000 | 5003 | 5.55 | 74,162 | 668,261 | 46.28 | 1.49496e10 | 4867 | 0.78340 | 0.6911 |
| 10000 | 9983 | 5.97 | 84,636 | 883,710 | 61.19 | 1.68402e10 | 4871 | 0.8059 | 0.7152 |
| All | 30021 | 6.22 | 262,473 | 3,109,768 | 224.11 | 1.0244404e11 | 4876 | 0.8016 | 0.7108 |
| (310 total) | | | | | | | | | |
| FiSVer-I dataset | | | | | | | | | |
| Task | Vocab size | Avg. phone | # Utts | # Tokens | Speech (h) | $g$-value | # Conv. | Norm. ent 1 | Norm. ent 2 |
| 10 | 10 | 4.6 | 64,998 | 73,650 | 9.96 | 3.21993e10 | 15561 | 0.4740 | 0.3815 |
| 50 | 50 | 5.92 | 115,512 | 144,906 | 17.95 | 6.91023e10 | 21052 | 0.5609 | 0.4678 |
| 100 | 100 | 5.6 | 138,722 | 191,156 | 22.01 | 9.56028e10 | 22062 | 0.5891 | 0.4958 |
| 500 | 500 | 5.34 | 258,307 | 653,847 | 55.32 | 2.25651e11 | 23111 | 0.7129 | 0.6175 |
| 1000 | 1000 | 5.43 | 352,261 | 1,299,566 | 99.06 | 3.23534e11 | 23214 | 0.7744 | 0.5911 |
| All | 42154 | 6.36 | 1.7 M | 17 M | 1242.5 (1593 total) | 1.30739e12 | 23300 | 0.8596 | 0.7737 |

Table 2
A comparison to random selected subsets and Svitchboard-I dataset. The statistics for the random selected subsets are averaged among 100 random trials.

| | Ave. Phoneme | # Utts | # Tokens | Speech (h) |
|---|---|---|---|---|
| Vocab Size | 50 | | | |
| Random | 3.47 | 10,628 | 13,624 | 1.58 |
| SVB-I | 2.78 | 12,442 | 20,914 | 1.93 |
| SVB-II | 3.32 | 24,033 | 38,154 | 4.01 |
| Vocab Size | 100 | | | |
| Random | 3.68 | 15,230 | 22,310 | 2.37 |
| SVB-I | 3.12 | 14,602 | 28,611 | 2.48 |
| SVB-II | 3.28 | 27,228 | 51,254 | 4.93 |
| Vocab Size | 500 | | | |
| Random | 4.39 | 30,216 | 74,390 | 6.32 |
| SVB-I | 3.93 | 23,670 | 89,420 | 6.44 |
| SVB-II | 3.95 | 39,694 | 131,815 | 10.30 |

these five folds as sets A, B, C, D, and E (with no conversation side overlaps). For each vocabulary task, we create 5 subtasks: we use 4 out of the 5 folds as training data. For development data, we use the first half of the remaining fold; for evaluation data, we use the second half of the remaining fold. Table 4 shows the cross-validation schemes of SVitchboard-II. For the FiSVer-I 10-vocabulary and 50-vocabulary subsets, we split the first 90% utterances as training data, and the remaining 5% and 5% utterances as used as development and evaluation set, respectively. For other vocabulary sizes we split the data into 98%, 1% and 1% as training, development and evaluation sets, respectively. A trigram language model is built for each experiment. For each subtask in SVitchboard-II, the language models must be trained *only* on the training data as shown in Table 4, *not* on the entire data before the splitting.

## 5.4. Baseline experiments

For each task, we establish two baseline systems, one with a triphone GMM-HMM system, and the other with a triphone DNN-HMM system, both of which are trained using the Kaldi open-source toolkit (Povey et al., 2011). For

Table 3
Selected datasets for Svitchboard-II and the corresponding algorithms and functions.

| Task | Algorithm and function |
|---|---|
| 50 | DS, $g_5$ |
| 100 | SFM, $g_2$ |
| 500 | DS, $g_5$ |
| 1000 | DS, $g_5$ |
| 5000 | SFM, $g_2$ |
| 10,000 | SFM, $g_2$ |

Table 4
Five-fold cross-validation schemes of SVitchboard-II. A–E correspond the five non-overlapping folds of the original dataset. The numbers in subscripts denote the first half or second half of the block.

| Subtask | Train | Dev | Eval |
|---|---|---|---|
| 1 | *ABCD* | $E_1$ | $E_2$ |
| 2 | *BCDE* | $A_1$ | $A_2$ |
| 3 | *CDEA* | $B_1$ | $B_2$ |
| 4 | *DEAB* | $C_1$ | $C_2$ |
| 5 | *EABC* | $D_1$ | $D_2$ |

the GMM-HMM system, we first flat-start a monophone GMM-HMM system, with 13 MFCCs and their deltas and delta-deltas (MFCC + $\Delta + \Delta\Delta$). Cepstral mean normalization is performed for each conversation side. After the monophone system has been trained, we use it to train a context-dependent GMM triphone model with MFCC + $\Delta + \Delta\Delta$ features. The total number of Gaussians for each task is around 25k. For the DNN-HMM system, we use alignments from the GMM-HMM system to bootstrap a triphone DNN/HMM system. Kaldi supports two different DNN training schemes: the first one is based on Veselỳ et al. (2013), which includes standard Restricted Boltzmann Machines (RBM), pre-training and stochastic gradient descent (SGD) training with GPUs; the second one supports parallel training on multiple CPUs and GPUs, and uses greedy layer-wise supervised training or layer-wise backpropagation (Bengio et al., 2007; Seide et al., 2011) and is described in detail in Povey et al. (2014). We use the second DNN training recipe with GPUs to create our baselines: for each task, we create a 4-layer network, with 1024 nodes in each network. The input features are spliced MFCCs (with a context window size of 4), followed by an LDA transformation (without dimensionality reduction) which is used to decorrelate the input features. The resulting feature vector has 117 dimensions in total. We use 20 epochs to train the DNN, with a mini-batch size of 256. For the first 15 epochs, we decrease the learning rate from 0.01 to 0.001 and fix the learning rate at 0.001 for the last 5 epochs. The number of Gaussian components is around 25k for each system. The numbers of parameters in the DNN systems are around 3.8 millions and 4.0 millions for the 50/100-vocabulary tasks, and around 5.2 millions for others. The baselines results for SVitchboard-II and FiSVer-I are shown in Table 5 and Table 6, respectively. We also run the same system on the 109-h Switchboard and obtained a WER of 46.4% and 31.6% on the Hub-5 Eval 2000 dataset, which is comparable to previous work (Lu and Renals, 2014) with a similar setup.

## 6. Conclusions

We have introduce a new set of benchmark corpora derived from the Switchboard-I and Fisher datasets. Our goal is to provide to the ASR and machine learning communities high-quality limited-complexity corpora of conversational English speech. The resulting SVitchboard-II and FiSVer-I datasets (obtained via various state-of-the art submodular optimization algorithms) will hopefully enable researchers to conduct experiments on rapidly prototyping novel machine learning algorithms and acoustic modeling methods without an inordinate turnaround time. The corpora definitions, and baseline system recipes, can be downloaded for free from https://bitbucket.org/melodi/hqlc-speechcorpora.

Table 5
Baseline results (word error rates) on SVitchboard-II using GMM-HMM systems and DNN-HMM systems.

| Task | Subtask | GMM-HMM | | DNN-HMM | |
|---|---|---|---|---|---|
| | | Dev (%) | Eval (%) | Dev (%) | Eval (%) |
| 50 | 1 | 35.37 | 36.98 | 26.89 | 29.93 |
| | 2 | 34.90 | 31.97 | 28.32 | 26.78 |
| | 3 | 32.39 | 35.63 | 27.61 | 30.18 |
| | 4 | 35.14 | 31.29 | 28.61 | 25.45 |
| | 5 | 32.05 | 34.16 | 26.34 | 26.49 |
| 100 | 1 | 39.13 | 41.53 | 29.74 | 33.18 |
| | 2 | 38.85 | 36.31 | 32.26 | 30.74 |
| | 3 | 36.05 | 39.09 | 31.06 | 32.50 |
| | 4 | 38.47 | 35.11 | 31.04 | 27.98 |
| | 5 | 35.51 | 36.77 | 28.68 | 28.24 |
| 500 | 1 | 44.85 | 43.15 | 37.86 | 35.00 |
| | 2 | 41.75 | 40.74 | 33.85 | 32.72 |
| | 3 | 43.18 | 44.48 | 35.64 | 36.96 |
| | 4 | 42.27 | 40.96 | 34.29 | 33.00 |
| | 5 | 42.58 | 40.44 | 33.62 | 32.52 |
| 1000 | 1 | 44.54 | 43.77 | 36.14 | 34.21 |
| | 2 | 42.11 | 41.97 | 33.22 | 32.10 |
| | 3 | 44.31 | 46.75 | 35.59 | 37.08 |
| | 4 | 42.89 | 41.29 | 33.18 | 31.69 |
| | 5 | 43.84 | 40.99 | 34.17 | 32.17 |
| 5000 | 1 | 44.52 | 44.23 | 33.69 | 33.49 |
| | 2 | 40.52 | 40.68 | 30.19 | 30.34 |
| | 3 | 45.10 | 45.79 | 35.74 | 34.63 |
| | 4 | 42.86 | 40.61 | 31.96 | 29.30 |
| | 5 | 43.53 | 41.56 | 32.84 | 31.28 |
| 10,000 | 1 | 45.22 | 45.26 | 32.04 | 32.17 |
| | 2 | 41.28 | 41.50 | 29.01 | 29.20 |
| | 3 | 45.16 | 46.74 | 31.25 | 33.14 |
| | 4 | 43.03 | 40.31 | 30.05 | 27.21 |
| | 5 | 44.53 | 43.00 | 31.47 | 30.43 |

Table 6
Baseline results (word error rates) on FiSVer-I using GMM-HMM systems and DNN-HMM systems.

| Task | GMM-HMM | | DNN-HMM | |
|---|---|---|---|---|
| | Dev (%) | Eval (%) | Dev (%) | Eval (%) |
| 10 | 3.76 | 8.55 | 2.25 | 5.43 |
| 50 | 11.37 | 15.69 | 8.66 | 13.30 |
| 100 | 25.09 | 20.71 | 19.03 | 17.89 |
| 500 | 36.97 | 32.69 | 27.87 | 23.80 |
| 1000 | 41.91 | 39.95 | 31.14 | 29.11 |

Table 7
Statistics of SVitchboard-II, SCSK, $g_1$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; # Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 50 | 3.48 | 21,304 | 30,419 | 3.42 | 3.64840e9 | 4410 | 0.4491 | 0.3729 |
| 100 | 100 | 3.63 | 24,634 | 39,243 | 4.11 | 4.27443e9 | 4522 | 0.4709 | 0.3937 |
| 500 | 500 | 4.52 | 33,987 | 83,135 | 7.18 | 6.36853e9 | 4689 | 0.5539 | 0.4704 |
| 1000 | 1000 | 4.78 | 40,655 | 135,330 | 10.61 | 7.87398e9 | 4759 | 0.6145 | 0.5265 |
| 5000 | 5000 | 5.63 | 70,010 | 551,571 | 38.58 | 1.38755E+10 | 4858 | 0.7691 | 0.6763 |
| 10,000 | 10,000 | 5.92 | 85,226 | 845,471 | 58.90 | 1.66003E+10 | 4871 | 0.8008 | 0.7096 |

Table 8
Statistics of SVitchboard-II, SCSK, $g_2$ . Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 50 | 3.54 | 21,268 | 30,307 | 3.41 | 3.67988E+09 | 4399 | 0.4526 | 0.3764 |
| 100 | 100 | 3.81 | 25,002 | 41,283 | 4.27 | 4.402E+09 | 4522 | 0.4806 | 0.4024 |
| 500 | 500 | 4.46 | 34,619 | 87,623 | 7.47 | 6.5059E+09 | 4707 | 0.5600 | 0.4761 |
| 1000 | 1000 | 4.86 | 41,715 | 147,024 | 11.38 | 8.13507E+09 | 4767 | 0.6258 | 0.5371 |
| 5000 | 5000 | 5.68 | 59,581 | 548,201 | 38.38 | 1.38414E+10 | 4864 | 0.7695 | 0.6758 |
| 10,000 | 10,000 | 5.98 | 82,808 | 800,569 | 55.79 | 1.62230E+10 | 4868 | 0.797 | 0.7061 |

Table 9
Statistics of SVitchboard-II, SCSK, $g_3$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 50 | 3.44 | 21,266 | 30,460 | 3.41 | 3.67407E+09 | 4406 | 0.4504 | 0.3742 |
| 100 | 100 | 3.84 | 24,923 | 39,966 | 4.23 | 4.33117E+09 | 4531 | 0.4773 | 0.3996 |
| 500 | 500 | 4.61 | 34,303 | 85,512 | 7.36 | 6.45428E+09 | 4698 | 0.5582 | 0.4744 |
| 1000 | 1000 | 4.85 | 41,488 | 144,420 | 11.23 | 8.08742E+09 | 4762 | 0.6238 | 0.5352 |
| 5000 | 5000 | 5.70 | 69,806 | 551,455 | 38.61 | 1.38791E+10 | 4863 | 0.7699 | 0.6770 |
| 10,000 | 10,000 | 6.00 | 82,985 | 802,078 | 55.89 | 1.62363E+10 | 4868 | 0.7974 | 0.7060 |

Table 10
Statistics of SVitchboard-II, SCSK, $g_4$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 50 | 3.32 | 22,165 | 32,192 | 3.57 | 3.75882E+09 | 4431 | 0.4529 | 0.3767 |
| 100 | 100 | 3.6 | 24,346 | 38,488 | 4.08 | 4.23205E+09 | 4505 | 0.4709 | 0.3935 |
| 500 | 500 | 4.44 | 34,330 | 86,807 | 7.39 | 6.4832E+09 | 4687 | 0.5592 | 0.4754 |
| 1000 | 1000 | 4.83 | 41,298 | 143,012 | 11.10 | 8.03212E+09 | 4763 | 0.6218 | 0.5333 |
| 5000 | 5000 | 5.68 | 69,685 | 546,945 | 38.26 | 1.38294E+10 | 4856 | 0.7686 | 0.6757 |
| 10,000 | 10,000 | 5.99 | 83,510 | 811,270 | 56.52 | 1.63138E+10 | 4868 | 0.7982 | 0.7068 |

Table 11
Statistics of SVitchboard-II, SFM, $g_1$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 51 | 3.14 | 23,769 | 37,076 | 3.92 | 4.03928E+09 | 4487 | 0.4641 | 0.3872 |
| 100 | 101 | 3.30 | 27,503 | 50,361 | 4.93 | 4.83285E+09 | 4589 | 0.4975 | 0.4184 |
| 500 | 504 | 3.92 | 39,717 | 131,683 | 10.27 | 7.68731E+09 | 4747 | 0.6113 | 0.5234 |
| 1000 | 1000 | 4.44 | 48,499 | 230,884 | 16.79 | 9.70056E+09 | 4800 | 0.6826 | 0.5907 |
| 5000 | 5009 | 5.52 | 75,006 | 654,515 | 45.53 | 1.48715E+10 | 4867 | 0.7849 | 0.6927 |
| 10,000 | 10,015 | 5.92 | 85,918 | 861,453 | 60.03 | 1.67336E+10 | 4871 | 0.8022 | 0.7111 |

Table 12
Statistics of SVitchboard-II, SFM, $g_2$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 50 | 2.96 | 23,204 | 36,360 | 3.83 | 3.98456E+09 | 4459 | 0.4620 | 0.3851 |
| 100 | 100 | 3.28 | 27,228 | 51,254 | 4.93 | 4.8425E+09 | 4571 | 0.4998 | 0.4203 |
| 500 | 509 | 3.91 | 35,617 | 111,525 | 8.79 | 7.07925E+09 | 4715 | 0.5963 | 0.5093 |
| 1000 | 991 | 4.44 | 47,603 | 232,546 | 16.81 | 9.679E+09 | 4799 | 0.6865 | 0.5944 |
| 5000 | 5003 | 5.55 | 74,162 | 668,261 | 46.28 | 1.49496E+10 | 4867 | 0.7834 | 0.6911 |
| 10,000 | 9983 | 5.97 | 84,636 | 883,710 | 61.19 | 1.68402E+10 | 4871 | 0.8059 | 0.7152 |

Table 13
Statistics of SVitchboard-II, SFM, $g_3$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 49 | 2.86 | 23,433 | 36,408 | 3.85 | 3.98938E+09 | 4473 | 0.4618 | 0.3850 |
| 100 | 103 | 3.33 | 27,256 | 49,134 | 4.84 | 4.7831E+09 | 4583 | 0.4947 | 0.4158 |
| 500 | 502 | 3.91 | 39,583 | 130,122 | 10.16 | 7.65191E+09 | 4745 | 0.6099 | 0.5222 |
| 1000 | 997 | 4.46 | 48,173 | 226,003 | 16.47 | 9.622E+09 | 4798 | 0.6799 | 0.5880 |
| 5000 | 5003 | 5.56 | 74,016 | 635,065 | 44.23 | 1.46911E+10 | 4868 | 0.7869 | 0.6948 |
| 10,000 | 9999 | 5.94 | 84,636 | 834,995 | 58.19 | 1.65136E+10 | 4869 | 0.8001 | 0.7090 |

Table 14
Statistics of SVitchboard-II, SFM, $g_4$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 52 | 3.13 | 23,865 | 37,399 | 3.95 | 4.05923E+09 | 4490 | 0.4650 | 0.3880 |
| 100 | 97 | 3.30 | 27,326 | 49,554 | 4.87 | 4.79539E+09 | 4584 | 0.4955 | 0.4165 |
| 500 | 496 | 3.90 | 39,665 | 131,150 | 10.23 | 7.67524E+09 | 4747 | 0.6108 | 0.5230 |
| 1000 | 998 | 4.42 | 48,417 | 230,370 | 16.75 | 9.692E+09 | 4797 | 0.6824 | 0.5905 |
| 5000 | 5001 | 5.52 | 74,866 | 651,536 | 45.33 | 1.48458E+10 | 4866 | 0.7830 | 0.6907 |
| 10,000 | 10,005 | 5.92 | 85,758 | 857,911 | 59.78 | 1.67048E+10 | 4871 | 0.8019 | 0.7108 |

Table 15
Statistics of SVitchboard-II, SCSK, $g_5$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 50 | 4.62 | 19,744 | 26,585 | 3.06 | 3.50021E+09 | 4347 | 0.4354 | 0.3604 |
| 100 | 100 | 4.79 | 22,231 | 32,930 | 3.61 | 3.99513E+09 | 4431 | 0.4548 | 0.3795 |
| 500 | 500 | 5.07 | 32,283 | 77,770 | 6.83 | 6.26751E+09 | 4677 | 0.5530 | 0.4695 |
| 1000 | 1000 | 5.19 | 40,120 | 141,234 | 11.07 | 8.05175E+09 | 4756 | 0.6274 | 0.5380 |
| 5000 | 5000 | 5.79 | 69,657 | 588,912 | 41.23 | 1.42644E+10 | 4860 | 0.7780 | 0.6855 |
| 10,000 | 10,000 | 6.06 | 83,457 | 863,729 | 60.18 | 1.67464E+10 | 4870 | 0.8048 | 0.7139 |

Table 16
Statistics of SVitchboard-II, DS-modmod, $g_5$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | # Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 48 | 4.06 | 19,478 | 25,506 | 3.03 | 3.41941E+09 | 4338 | 0.4409 | 0.3657 |
| 100 | 101 | 4.50 | 21,163 | 29,671 | 3.38 | 3.80866E+09 | 4425 | 0.4565 | 0.3806 |
| 500 | 500 | 4.99 | 32,707 | 82,269 | 7.10 | 6.39389E+09 | 4692 | 0.5594 | 0.4755 |
| 1000 | 996 | 5.15 | 39,955 | 145,408 | 11.31 | 8.12734E+09 | 4764 | 0.6315 | 0.5422 |
| 5000 | 4996 | 5.85 | 69,110 | 589,477 | 41.20 | 1.42532E+10 | 4862 | 0.7789 | 0.6865 |
| 10,000 | 10,000 | 6.10 | 83,099 | 841,695 | 58.69 | 1.66771E+10 | 4870 | 0.8036 | 0.7130 |

Table 17
Statistics of SVitchboard-II, DS-supsub, $g_5$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 50 | 4.28 | 19,341 | 24,745 | 2.98 | 3.47514E+09 | 4337 | 0.4394 | 0.3645 |
| 100 | 100 | 4.57 | 23,179 | 35,200 | 3.82 | 4.08583E+09 | 4473 | 0.4640 | 0.3872 |
| 500 | 502 | 5.07 | 32,272 | 76,775 | 6.74 | 6.1869E+09 | 4670 | 0.5492 | 0.4674 |
| 1000 | 1003 | 5.19 | 39,049 | 130,234 | 10.30 | 7.771E+09 | 4753 | 0.6157 | 0.5275 |
| 5000 | 4995 | 5.86 | 68,183 | 560,815 | 39.29 | 1.39679E+10 | 4863 | 0.7742 | 0.6815 |
| 10000 | 10002 | 6.12 | 82,604 | 841,695 | 58.69 | 1.6567E+10 | 4871 | 0.8029 | 0.7119 |

Table 18
Statistics of SVitchboard-II, DS-subsup, $g_5$. Vocab size: actual vocabulary size; Avg. phone: average number of phonemes per word; #Utts: number of utterances; #Tokens: number of tokens; Speech: hours of speech (excluding the silence parts); $g$-value: the function value of $g_5(X)$; #Conv.: number of conversation sides; Norm. ent 1: normalized entropy of phoneme distribution; Norm. ent 2: normalized entropy of non-silence phoneme distribution.

| SVitchboard-II dataset | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Task | Vocab size | Avg. phone | #Utts | #Tokens | Speech (h) | $g$-value | #Conv. | Norm. ent 1 | Norm. ent 2 |
| 50 | 50 | 3.32 | 24,033 | 38,154 | 4.01 | 4.13054E+09 | 4491 | 0.4688 | 0.3916 |
| 100 | 100 | 3.29 | 27,583 | 50,610 | 4.94 | 4.8585E+09 | 4588 | 0.4980 | 0.4188 |
| 500 | 500 | 3.95 | 39,694 | 131,815 | 10.30 | 7.70767E+09 | 4749 | 0.6122 | 0.5243 |
| 1000 | 1001 | 4.50 | 48,445 | 230,876 | 16.81 | 9.70981E+09 | 4801 | 0.6831 | 0.5911 |
| 5000 | 5002 | 5.57 | 74,739 | 660,250 | 45.99 | 1.49366E+10 | 4867 | 0.7849 | 0.6927 |
| 10,000 | 10,005 | 5.98 | 85,295 | 874,444 | 60.98 | 1.68511E+10 | 4871 | 0.8043 | 0.7134 |

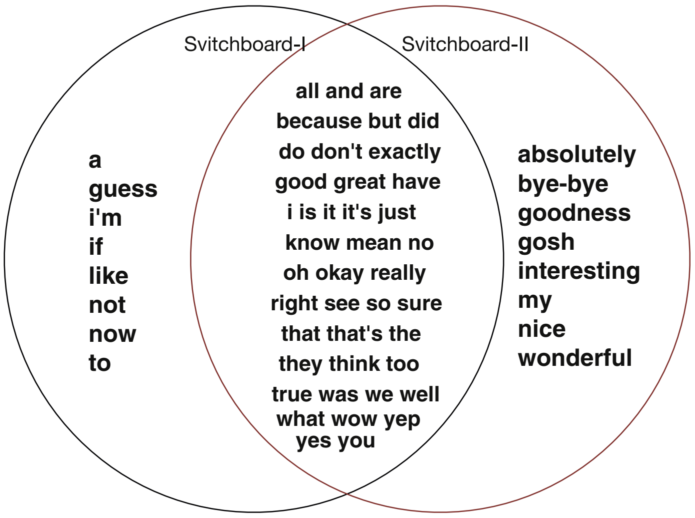

Fig. 2. Venn diagram showing the vocabulary difference between Svitchboard-I and Svitchboard-II (50-word task).

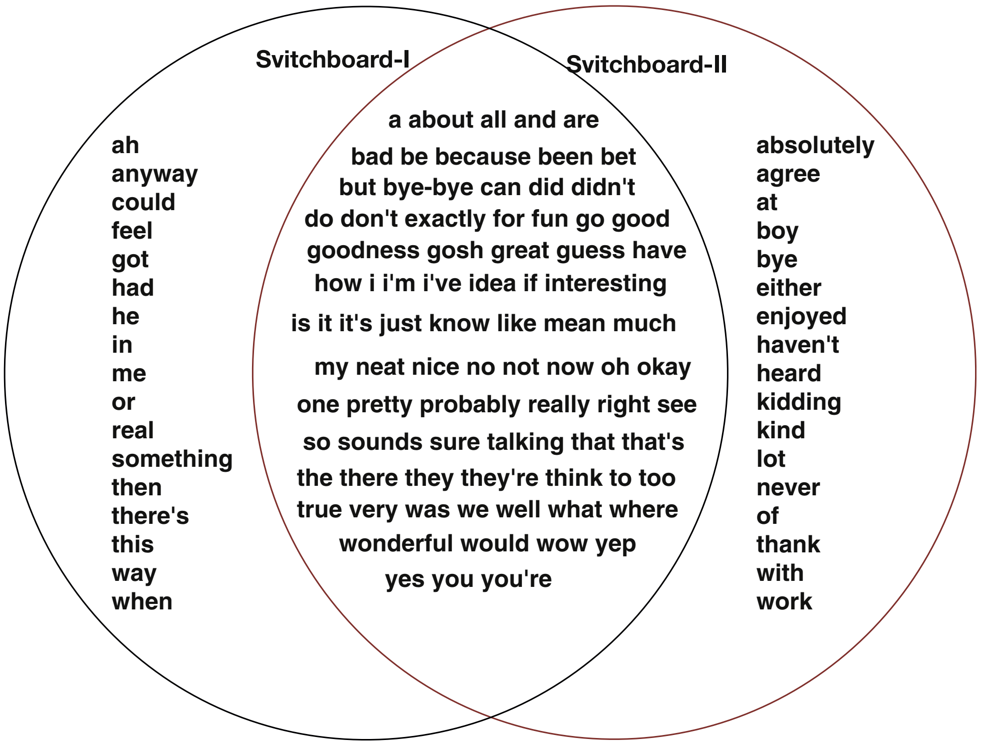
Svitchboard-I
Svitchboard-II
ah
anyway
could
feel
got
had
he
in
me
or
real
something
then
there's
this
way
when
a about all and are
bad be because been bet
but bye-bye can did didn't
do don't exactly for fun go good
goodness gosh great guess have
how i i'm i've idea if interesting
is it it's just know like mean much
my neat nice no not now oh okay
one pretty probably really right see
so sounds sure talking that that's
the there they they're think to too
true very was we well what where
wonderful would wow yep
yes you you're
absolutely
agree
at
boy
bye
either
enjoyed
haven't
heard
kidding
kind
lot
never
of
thank
with
work

Fig. 3. Venn diagram showing the vocabulary difference between Svitchboard-I and Svitchboard-II (100-word task).

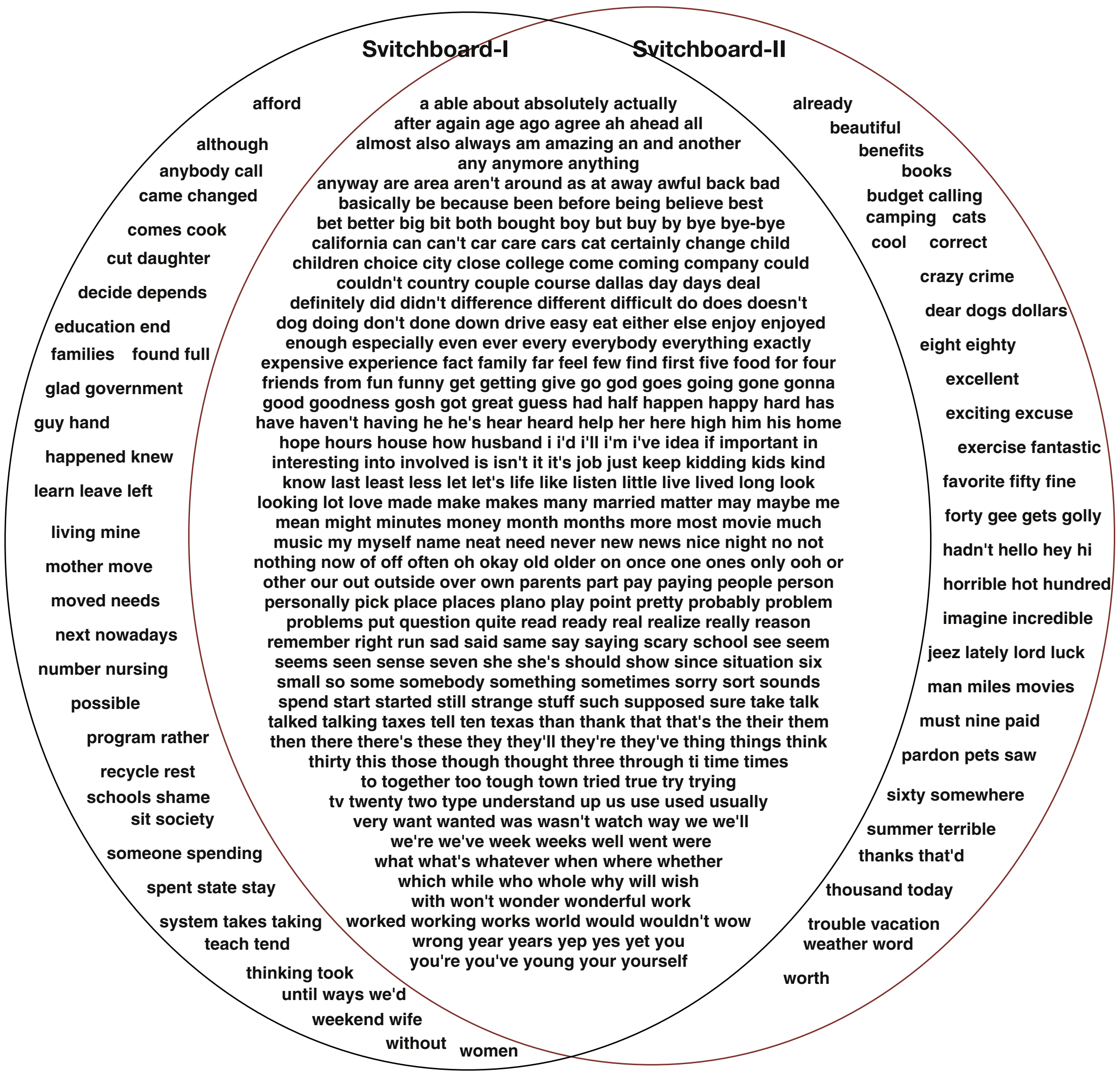
Svitchboard-I
Svitchboard-II
afford
although
anybody call
came changed
comes cook
cut daughter
decide depends
education end
families found full
glad government
guy hand
happened knew
learn leave left
living mine
mother move
moved needs
next nowadays
number nursing
possible
program rather
recycle rest
schools shame
sit society
someone spending
spent state stay
system takes taking
teach tend
thinking took
until ways we'd
weekend wife
without
women
a able about absolutely actually
after again age ago agree ah ahead all
almost also always am amazing an and another
any anymore anything
anyway are area aren't around as at away awful back bad
basically be because been before being believe best
bet better big bit both bought boy but buy by bye-bye
california can can't car care cars cat certainly change child
children choice city close college come coming company could
couldn't country couple course dallas day days deal
definitely did didn't difference different difficult do does doesn't
dog doing don't done down drive easy eat either else enjoy enjoyed
enough especially even ever every everybody everything exactly
expensive experience fact family far feel few find first five food for four
friends from fun funny get getting give go god goes going gone gonna
good goodness gosh got great guess had half happen happy hard has
have haven't having he he's hear heard help her here high him his home
hope hours house how husband i i'd i'll i'm i've idea if important in
interesting into involved is isn't it it's job just keep kidding kids kind
know last least less let let's life like listen little live lived long look
looking lot love made make makes many married matter may maybe me
mean might minutes money month months more most movie much
music my myself name neat need never new news nice night no not
nothing now of off often oh okay old older on once one ones only ooh or
other our out outside over own parents part pay paying people person
personally pick place places plano play point pretty probably problem
problems put question quite read ready real realize really reason
remember right run sad said same say saying scary school see seem
seems seen sense seven she she's should show since situation six
small so some somebody something sometimes sorry sort sounds
spend start started still strange stuff such supposed sure take talk
talked talking taxes tell ten texas than thank that that's the their them
then there there's these they they'll they're they've thing things think
thirty this those though thought three through ti time times
to together too tough town tried true try trying
tv twenty two type understand up us use used usually
very want wanted was wasn't watch way we we'll
we're we've week weeks well went were
what what's whatever when where whether
which while who whole why will wish
with won't wonder wonderful work
worked working works world would wouldn't wow
wrong year years yep yes yet you
you're you've young your yourself
already
beautiful
benefits
books
budget calling
camping cats
cool correct
crazy crime
dear dogs dollars
eight eighty
excellent
exciting excuse
exercise fantastic
favorite fifty fine
forty gee gets golly
hadn't hello hey hi
horrible hot hundred
imagine incredible
jeez lately lord luck
man miles movies
must nine paid
pardon pets saw
sixty somewhere
summer terrible
thanks that'd
thousand today
trouble vacation
weather word
worth

Fig. 4. Venn diagram showing the vocabulary difference between Svitchboard-I and Svitchboard-II (500-word task).

## References

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., et al., 2007. Greedy layer-wise training of deep networks. Adv. Neural Inf. Process. Syst. 19, 153.

Cieri, C., Miller, D., Walker, K., 2004. The Fisher corpus: a resource for the next-generation speech-to-text. In: Proceedings of Conference on Language Resources and Evaluation.

Dahl, G.E., Yu, D., Deng, L., Acero, A., 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. IEEE Trans. Audio Speech Lang. Process. 20 (1), 30–42.

Das, A., Kempe, D., 2011. Submodular meets spectral: Greedy algorithms for subset selection, sparse approximation and dictionary selection. In: Proceedings of International Conference on Machine Learning.

Fujishige, S., 2005. Submodular Functions and Optimization. Elsevier Science Ltd.

Fujishige, S., Hayashi, T., Isotani, S., 2006. The minimum-norm-point algorithm applied to submodular function minimization and linear programming, Preprint RIMS-1571. Research Institute for Mathematical Sciences Kyoto University, Kyoto, Japan.

Godfrey, J., Holliman, E., McDaniel, J., 1992. Switchboard: telephone speech corpus for research and development. In: Proceedings of International Conference on Acoustics, Speech and Signal Processing, pp. 517–520.

Graves, A., Mohamed, A.-r., Hinton, G., 2013. Speech recognition with deep recurrent neural networks. In: Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 6645–6649.

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al., 2012. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. 29 (6), 82–97.

Iyer, R., Bilmes, J., 2012. Algorithms for approximate minimization of the difference between submodular functions, with applications. In: Proceedings of Conference on Uncertainty in Artificial Intelligence.

Iyer, R., Bilmes, J., 2013. Submodular optimization with submodular cover and submodular knapsack constraints. In: Proceedings of Neural Information Processing Society (NIPS), Lake Tahoe, CA.

King, S., Bartels, C., Bilmes, J., 2005. SVitchboard 1: Small Vocabulary Tasks from Switchboard. In: Proceedings of the Ninth European Conference on Speech Communication and Technology. ISCA.

Kirchhoff, K., Liu, Y., Bilmes, J., 2013. Classification of developmental disorders from speech signals using submodular feature selection. In: Proceedings of Annual Conference of the International Speech Communication Association (INTERSPEECH), Lyon, France.

Krause, A., Leskovec, J., Guestrin, C., VanBriesen, J., Faloutsos, C., 2008a. Efficient sensor placement optimization for securing large water distribution networks. J. Water Resour. Plan. Manag. 134 (6), 516–526.

Krause, A., McMahan, B., Guestrin, C., Gupta, A., 2008b. Robust submodular observation selection. J. Mach. Learn. Res. 9, 2761–2801.

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

Lin, H., Bilmes, J., 2011a. A class of submodular functions for document summarization. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT). Portland, OR.

Lin, H., Bilmes, J., 2012. Learning mixtures of submodular shells with application to document summarization. Uncertainty in Artificial Intelligence (UAI). AUAI, Catalina Island, USA.

Lin, H., Bilmes, J.A., 2009. How to select a good training-data subset for transcription: Submodular active selection for sequences. In: Proceedings of Annual Conference of the International Speech Communication Association (INTERSPEECH), Brighton, UK.

Lin, H., Bilmes, J.A., 2011b. Optimal selection of limited vocabulary speech corpora. In: Proceedings of Annual Conference of the International Speech Communication Association (INTERSPEECH), Florence, Italy.

Liu, Y., Iyer, R., Kirchhoff, K., Bilmes, J., 2015. SVitchboard II and Fisver I: High-quality limited-complexity corpora of conversational English speech. In: Proceedings of Annual Conference of the International Speech Communication Association (INTERSPEECH), Dresden, Germany.

Liu, Y., Wei, K., Kirchhoff, K., Song, Y., Bilmes, J., 2013. Submodular feature selection for high-dimensional acoustic score space. In: Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP). IEEE.

Lu, L., Renals, S., 2014. Probabilistic linear discriminant analysis with bottleneck features for speech recognition. In: Proceedings of Conference of the International Speech Communication Association (INTERSPEECH).

Narasimhan, M., Bilmes, J., 2004. PAC-learning bounded tree-width graphical models. In: Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI-2004). Morgan Kaufmann Publishers.

Narasimhan, M., Bilmes, J., 2005. A submodular-supermodular procedure with applications to discriminative structure learning. In: Proceedings of Conference on Uncertainty in Artificial Intelligence (UAI).

Narasimhan, M., Jojic, N., Bilmes, J., 2005. Q-clustering. In: Proceedings of Conference on Neural Information Processing Systems (NIPS).

Narayanan, H., 1997. Submodular Functions and Electrical Networks. North-Holland, North-Holland, Amsterdam.

Nemhauser, G., Wolsey, L., Fisher, M., 1978. An analysis of approximations for maximizing submodular set functions I. Math. Program. 14 (1), 265–294.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al., 2011. The Kaldi speech recognition toolkit. In: Proceedings of Conference on Automatic Speech Recognition and Understanding (ASRU), pp. 1–4.

Povey, D., Zhang, X., Khudanpur, S., 2014. Parallel training of deep neural networks with natural gradient and parameter averaging. In: Proceedings of International Conference on Learning Representation (ICLR) Workshop 2015.

Sainath, T.N., Mohamed, A.-r., Kingsbury, B., Ramabhadran, B., 2013. Deep convolutional neural networks for lvcsr. In: Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 8614–8618.

Sak, H., Senior, A., Beaufays, F., 2014a. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: Proceedings of Annual Conference of the International Speech Communication Association (INTERSPEECH).

Sak, H., Vinyals, O., Heigold, G., Senior, A., McDermott, E., Monga, R., Mao, M., 2014b. Sequence Discriminative Distributed Training of Long Short-Term Memory Recurrent Neural Networks. In: Proceedings of Annual Conference of the International Speech Communication Association (INTERSPEECH), 2014.

Saon, G., Kuo, H.-K. J., Rennie, S., Picheny, M., 2015. The ibm 2015 english conversational telephone speech recognition system. arXiv preprint arXiv:1505.05899.

Seide, F., Li, G., Yu, D., 2011. Conversational speech transcription using context-dependent deep neural networks. In: Proceedings of Conference of International Speech Communication Association, pp. 437–440.

Tschiatschek, S., Iyer, R., Wei, H., Bilmes, J., 2014. Learning Mixtures of Submodular Functions for Image Collection Summarization. In: Proceedings of Neural Information Processing Society (NIPS), Montreal, CA.

Veselỳ, K., Ghoshal, A., Burget, L., Povey, D., 2013. Sequence-discriminative training of deep neural networks. In: Proceedings of Conference of International Speech Communication Association, pp. 2345–2349.

Wei, K., Liu, Y., Kirchhoff, K., Bartels, C., Bilmes, J., 2014. Submodular subset selection for large-scale speech training data. In: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, Florence, Italy.

Wei, K., Liu, Y., Kirchhoff, K., Bilmes, J., 2013. Using document summarization techniques for speech data subset selection. In: North American Chapter of the Association for Computational Linguistics/Human Language Technology Conference (NAACL/HLT-2013), Atlanta, GA.