Phoneme-Lattice to Phoneme-Sequence Matching Algorithm based on Dynamic Programming

Ciro Gracia[1,2], Xavier Anguera[1], jordi Luque[1], and Ittai Artzi[3]

[1] Telefonica Research, Edificio Telefonica-Diagonal 00, 08019, Barcelona, Spain,
[2] Universitat Pompeu Fabra, Department of Information and Communications Technologies, Barcelona, Spain,
[3] Tel Aviv University, Tel Aviv, Israel,
ciro.gracia@upf.edu, {xanguera, jls}@tid.es

Abstract. A novel phoneme-lattice to phoneme-sequence matching algorithm based on dynamic programming is presented in this paper. Phoneme lattices have been shown to be a good choice to encode in a compact way alternative decoding hypotheses from a speech recognition system. These are typically used for the spoken term detection and keyword-spotting tasks, where a phoneme sequence query is matched to a reference lattice. Most current approaches suffer from a lack of flexibility whenever a match allowing phoneme insertions, deletions and substitutions is to be found. We introduce a matching approach based on dynamic programming, originally proposed for Minimum Bayes decoding on speech recognition systems. The original algorithm is extended in several ways. First, a self-trained phoneme confusion matrix for phoneme comparison is applied as phoneme penalties. Also, posterior probabilities are computed per arc, instead of likelihoods and an acoustic matching distance is combined with the edit distance at every arc. Finally, total matching scores are normalized based on the length of the optimum alignment path. The resulting algorithm is compared to a state-of-the-art phoneme-lattice-to-string matching algorithm showing relative precision improvements over 20% relative on an isolated word retrieval task.

Keywords: phoneme lattice search, keyword search, speech recognition, information retrieval

1 Introduction

The use of Phoneme Lattice-based Search (PLS) for efficient keyword search and retrieval has captured the attention of researchers during the last few years. For instance, the use of phoneme lattices for efficient representation of speech utterances has become popular in tasks like Spoken term detection (STD) [1] or Keyword spotting (KWS) [2]. Such tasks are oriented towards applications related to information retrieval which require the processing of large amounts of speech data. Some example of application include mining of the opinion and complains of customers in call center data [3], indexing of meetings or lectures [4] and browsing and retrieval of voice/video mail messages [5, 6].

Phoneme lattice-based search is based on a lattice representation of the set of phoneme alternatives decoded from the audio signal, used to compactly encode the acoustic signals as the N best decoding hypotheses. Although in 1-best decoding transcriptions the search for a keyword is a simple "grep" operation, it is found that errors on such transcript affect too negatively the retrieval performance. Encoding more than a single decoding hypothesis solves this problem in most part, although a graph or network structure is usually required for optimal encoding, making search and matching algorithms quite complex whenever we want to consider insertions and deletions. In this paper we propose an alternative indexing and search method targeted to phoneme lattices that simplifies prior approaches.

In general, PLS-based approaches [2, 7] are a good alternative to previous proposed strategies [8, 9] thanks to their flexibility, even where no prior knowledge about the searched words is provided, and thanks to their capacity to represent in a compact form multiple phoneme-sequences hypotheses between two time stamps in the utterance. In [7] the authors used a phoneme lattice-based approach for keyword search in a subset of the DARPA Resource Management (RM) task [10]. Their approach matches the keyword pronunciation against each lattice through dynamic programming, which takes into account phoneme recogniser insertions, deletions and substitutions by heuristically imposing a penalty on them. The authors also compared the lattice search with previous strategies like Keyword HMM [8], which makes use of the knowledge on the searched keywords to constrain a recognition network consisting on pre-trained HMM models per each word and a "garbage" model for the rest. PLS-based approaches can achieve significantly higher query speeds than HMM-based systems by first indexing the lattices for subsequent search.

In [11] the authors incorporate the benefits from Dynamic Programming sequence matching techniques aiming to decrease miss rates due to phoneme recognizer errors. Dynamic programming matches the keyword pronunciation against each lattice through a improvement on the Minimum Edit Distance (MED) where penalty substitutions are derived from a phoneme confusion matrix. The authors also employ HTK toolkit [12] for lattice generation, showing considerable improvements in terms of both keyword search accuracy and searching speed while maintaining low miss rate on a conversational telephone speech database.

In [13] the authors propose to derive substitution phoneme penalties for the MED distance from a phone confusion matrix. This leads to a considerable improvement in the accuracy of the keyword search algorithm. In order to build the confusion matrix, the authors used the ground truth transcription from a database to compare against an automatic transcription from a speech recognition system. In [14], spoken terms are searched in pruned phoneme lattices. A "term-dependent discriminative decision" technique is introduced which integrates multiple decisions into a classification posterior probability. Furthermore, the work addresses OOV detection through estimation of a posterior acoustic confidence.

In this paper we propose a novel PLS approach that integrates a dynamic programming matching algorithm first proposed in [15] for Minimum Bayes decoding on speech recognition systems. It is augmented with several improvements to perform a phoneme sequence to phoneme lattice matching in a fast and effective way. First, a self-trained phoneme confusion matrix for phoneme comparison is applied as phoneme penalties. Also, posterior probabilities are computed per arc, instead of using likelihoods and an acoustic matching distance is combined with the edit distance at every arc. Finally, combined matching scores are normalized based on the length of the optimum alignment path. We test our algorithm using a simple test scenario comparing multiple instances of words, designed to rapidly evaluate the capabilities of matching/search algorithms working with lattice representation. We show improvements of over 20% relative in a subset of the Switchboard word corpus [16] compared to results on the same task obtained by a state-of-the-art phoneme-lattice-to-string matching algorithm [1, 15] .

2 Edit Distance Computation on Lattices

In this section we review the algorithm proposed in [15] to obtain an upper bound on the edit distance calculation between a phoneme string and a phoneme lattice, which we will later improve in this paper. This algorithm compares a sequence of phonemes $R = \{r_1, ..., r_Q\}$ (i.e. a phoneme string or sequence) against a phoneme lattice L in the context of the minimum Bayes risk decoding for the optimization of speech recognition systems. The lattice L is composed by nodes $n_i \in \{n_1, ..., n_N\}$ that encode time stamps in the acoustic signals. Each node n_i has a set of incoming $pre(n_i)$ and a set of outgoing $post(n_i)$ arcs. An arc a_j represents a decoding hypothesis for a certain time interval and is represented by: a starting node $s(a_j)$, an ending node $e(a_j)$, a hypothesis class label $w(a_j)$ and, typically, a pair of log-likelihoods from the acoustic model $l_a(a_j)$ and the language model $l_l(a_j)$. The algorithm uses the Levenshtein edit distance[17] to compare a sequence of phonemes R and a sequence of phonemes $W(S)$ obtained from the sequence of arc labels in the lattice path S.

As a lattice can be seen as set of decoded strings, each of the decoded strings can be compared with the reference string generating a corresponding edit distance. In addition, each of the decoded strings has an associated likelihood, provided by the recognition system. Given $S \in L$ a path in the lattice, the log-probability of the path $(log(p(S)))$ is defined from the total likelihood obtained from the arcs in S as shown in equation 1.

$$log(p(S)) = \sum_{n=1}^{|S|} l_a(S_n) + l_l(S_n) \tag{1}$$

Where $|S|$ is the total number of arcs in the path, $l_a(S_n), l_l(S_n)$ are the acoustic and language model log-likelihoods of phoneme S_n in the path. The exact edit distance between the phoneme lattice L and the phoneme string R, Edit-Distance(L, R), can be computed by equation 2.

$$\text{Edit-Distance}(L, R) = \frac{\sum_{S \in L} p(S) ED(W(S), R)}{\sum_{S \in L} p(S)} \quad (2)$$

where $ED(W(S), R)$ is the standard edit distance between two strings, in this case the sequence of phonemes of path S in the lattice and the phoneme string (note the difference with Edit-Distance(L, R) which indicates the distance between the whole lattice and a string). Given that computing Eq. 2 is very expensive, in [15] they propose an efficient way to obtain an upper bound to Equation 2 through a dynamic programming procedure shown in Algorithm 1.

Algorithm 1 $\hat{L}$ = Edit-Distance'(L,R)

```
1:  $N \leftarrow |L|, Q \leftarrow |R|$
2:  Initialize array $\alpha(1...N), \alpha'(1...N, 0...Q), \alpha'_{arc}(0..Q)$
3:  $\alpha(1) \leftarrow 1, \alpha' \leftarrow 0$
4:  for $q \leftarrow 1..N$ do
5:      $\alpha'(1, q) \leftarrow \alpha(1, q-1) + loss(\epsilon, r_q)$
6:  end for
7:  for $n \leftarrow 2..N$ do
8:      $\alpha(n) \leftarrow \sum_{a \in pre(n)} \alpha(s(a) p(a)$
9:      $\forall q, \alpha'(n, q) \leftarrow 0$
10:     for $a \in pre(n)$ do
11:         $\lambda(a) \leftarrow \frac{\alpha(s(a)) p(a)}{\alpha(n)}$
12:         for $q \leftarrow 0..Q$ do
13:             if $q = 0$ then
14:                 $\alpha'_{arc}(q) \leftarrow \alpha'(s(a), q) + loss(w(a), \epsilon) + \delta$
15:             else
16:                 $\alpha'_{arc}(q) \leftarrow \min \begin{cases} \alpha'(s(a), q-1) + loss(w(a), r_q) \\ \alpha'(s(a), q) + loss(w(a), \epsilon) + \delta \\ \alpha'_{arc}(q-1) + loss(\epsilon, r_q) \end{cases}$
17:             end if
18:             $\alpha'(n, q) \leftarrow \alpha'(n, q) + \lambda(a) \alpha'_{arc}(q)$
19:         end for
20:     end for
21: end for
22: $\hat{L} \leftarrow \alpha'(N, Q)$
```

3 Adaptation to Phoneme Lattice Search

In this section we describe four proposed improvements to Algorithm 1.

3.1 Single-best-path Alignment

The use of a phoneme lattices instead of the 1-best phoneme decoding in Algorithm 1 increases the probability of finding correct matches with the phonetic sequence given by the word transcription. Still, Algorithm 1 computes at each node the weighted average of all arcs reaching that node, making the final distance strongly dependent on the structure of the lattice. Different parameter configurations in the lattice pruning may produce different lattice densities, which can translate into very different resulting distances, driven by the least probable lattice paths. For this reason we avoid the definition of a final score as the average edit distance with respect to the whole lattice and, instead, we define it as the average edit distance between the query phoneme sequence and the closest phoneme path within the phoneme lattice. To do so, we redefine the α' term (Line 18 in Algorithm 1) as $\alpha'(n,q) \leftarrow \min(\alpha'(n,q), \alpha'_{arc}(q))$ to propagate only the best scoring path at each node. This modification allows us to retrieve the single best path with respect to the query instead of the single best sequence of nodes.

3.2 Combining Edit Distance With Acoustic Likelihoods

Algorithm 1 defines the distance between a phoneme lattice and a phoneme sequence as the average edit distance. The edit distance measures the dissimilarity, in terms of edit operations, between phonetic sequences but it does not take into account neither the acoustic nor the language model likelihoods associated to each phoneme in that path. This leads to situations in where phone sequences with little phonetic mismatch might be selected although they produced low global acoustic probability scores. This is a problem in very dense lattices where it is more probable that spurious matching paths exist.

In order to take into account the phoneme likelihood into the algorithm we define a new global score that combines both sources of information. To do so, we first define in Equations 3 and 4 the $\hat{\alpha}(n)$ and $\hat{\beta(n)}$ variables which represent the acoustic score of the best path starting and ending at a certain node in the lattice. Then, in Equation 5 we define $\lambda(a)$ as the ratio between the total acoustic score of the best path containing arc a and the score of the most likely path in the lattice. This score lies within the range $[0-1]$, becoming 1 when a belongs to the most likely path. Finally, Equation 6 combines the edit distance and acoustic scores and is used in replacement of line 14 in Algorithm 1.

$$\hat{\alpha}(n) = \begin{cases} 0, & \text{if } n = 1 \\ max_{a \in pre(n)}\{\hat{\alpha}(s(a)) + l_a(a) + l_l(a)\}, & \text{otherwise} \end{cases} \tag{3}$$

$$\hat{\beta}(n) = \begin{cases} 0, & \text{if } n = N \\ max_{a \in post(n)}\{\hat{\beta}(e(a)) + l_a(a) + l_l(a)\}, & \text{otherwise} \end{cases} \tag{4}$$

$$\lambda(a) \leftarrow exp(\hat{\alpha}(s(a)) + l_a(a) + l_l(a) + \hat{\beta}(e(a)) - \hat{\alpha}(N)) \tag{5}$$

$$\alpha'_{arc}(q) \leftarrow \alpha'(s(a),q) + \theta(loss(w(a), r_q) + \delta) + (1-\theta)(1-\lambda(a)) \tag{6}$$

Where the factor θ is used to weight the contribution between the edit distance and the acoustic score. We have experimentally set it to $\theta = 0.85$.

3.3 Phonetic Loss Function Estimation

Algorithm 1 uses a loss function $loss(a,b) \leftarrow (1 \leftarrow a = b, 0 \leftarrow a \neq b)$ based on the edit distance and driven by binary (i.e $\{0,1\}$) decisions. Despite it being adequate for comparing general sequences of symbols its does not take into account any available domain knowledge. Similarly to [13], we improve the loss function by incorporating specific penalties for possible phoneme substitutions, insertions and deletions.

In order to define an appropriate loss function we apply a data driven approach. First we compute, for all words in the development set, all possible phoneme-level alignments between queries and lattices representing instances of the same word. Similarly, we also compute all possible phoneme-level alignments between instances of different words. Alignments are obtained through a back-tracking of the $\alpha'(n,q)$ matrix once Algorithm 1 has finished aligning all development data. With the alignment results we build two confusion matrices by accumulating the number of times the recognizer substituted the ith phoneme by the jth phoneme. If the value (i,j) in the matrix is greater than the value in (i,k), it means that the ith phoneme is more likely to be substituted by the jth phoneme than by the kth phoneme. The information in these matrices is used to specify lower penalty for phonemes that have been shown as frequently interchangeable due to variations in the pronunciation of the patterns and higher penalties for those representing mismatches. Formally, we define $C \in R^{|ph|\cdot|ph|}$ and $NC \in R^{|ph|\cdot|ph|}$ as square and symmetric matrices corresponding to the same-word and different-word confusion matrices,where $|ph|$ is the size of the phonetic alphabet. These matrices are normalized in order to obtain conditional probabilities. Equation 7 defines the final cost matrix: $COST \in R^{|ph|\cdot|ph|}$, containing the penalties associated to each of the possible substitutions,deletions, insertions and assignments of phoneme symbols.

$$\begin{gathered}
COST(a,b) = 1 - \left(\frac{C(a,b)}{C(a,b)+NC(a,b)}\right) \\
loss(a,b) \leftarrow \begin{cases} COST(a,b), & \text{if } a \neq b \\ 0, & \text{otherwise} \end{cases}
\end{gathered} \tag{7}$$

3.4 Matching Scores Normalization

The last improvement proposed over Algorithm 1 involves the normalization of the total matching score. While originally no normalization is proposed by [15], we observed that scores obtained from matching phoneme sequences of different lengths were not directly comparable with each other. For this reason we experimented with several normalization factors to reduce such mismatch. In this paper use the length of the 1-best path over the lattice in combination with the length of the phoneme sequence has been found the most successful.

4 Experimental Evaluation

We test the proposed algorithm using an isolated word matching test for a set of word lattices and word phoneme sequences. The task is focused into evaluating the algorithm's retrieval capabilities using the phoneme sequences as queries to be searched into the lattices. Performance is measured by the percentage of lattices corresponding to the same word as the query word that are contained in the top N retrieval results (P@N, N being total number lattices from the query word present in the database). We compare P@N obtained with our algorithm to a state of the art spoken term detection algorithm proposed in [1] and to the original algorithm proposed in [15] and reviewed in Section 2 above.

4.1 Dataset

Words	Development	Evaluation
anything	34	48
because	98	110
benefits	136	109
companies	42	40
company	66	64
everything	56	49
exactly	65	63
expensive	56	35
important	45	37
insurance	93	82
newspapers	57	45
plastic	57	63
probably	59	57
punishment	80	107
really	59	58
recycling	149	155
retirement	39	42
situation	50	35
something	56	71
vacation	39	61
Total	1336	1331

Table 1. Frequencies of the 20 words selected for the experiment

In order to evaluate the algorithm we use a subset of the Switchboard corpus as defined in [16]. In particular, we select all instances of 20 of the most repeated words both in the development and in the evaluation sets in [16]. Table 1 shows the selected words and the number of repetitions of each word we selected. Each word instance is extracted from the acoustic signal and decoded using the Kaldi Speech Recognition Toolkit [18], trained using the standard Switchboard recipe for English. For each instance we obtain a phoneme lattice and the first-best phoneme sequence, which will be used as the query in order to simulate an audio-to-audio search (alternatively one could use a grapheme-to-phoneme convertor on the text query for a standard spoken-term detection task).

4.2 Experiments

Table 2 shows average P@N results for the development and evaluation sets for the two baseline systems and for the improvements proposed here. The first improvement to [15] (SB) uses the single best scoring path in the lattice. The second improvement (SC) corresponds to the combination of Edit distance with the acoustic probability. Finally the third improvement (CM) uses the phoneme loss function estimated on the development set. The first values corresponds to the unweighted average p@N whereas the value between parentheses correspond to the weighted average P@N using the prior of each word class. We see how each proposed improvement gives an improvement in the P@N both in the development set as well as in the evaluation set. This achieves a 20% relative

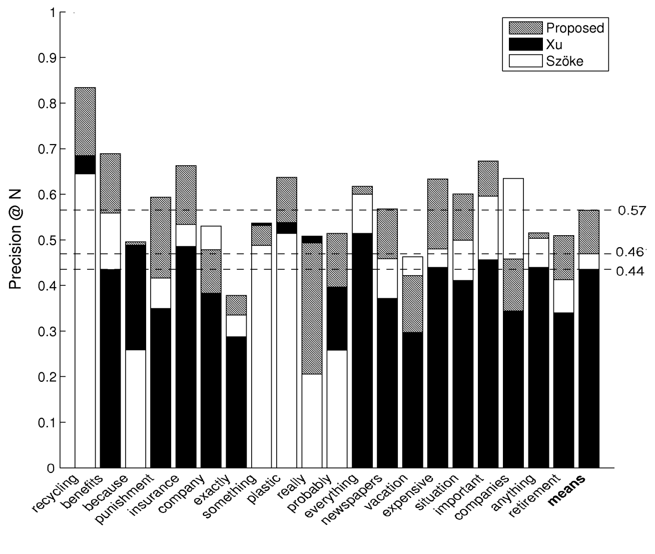

Fig. 1. Comparison of the methods for the development set

improvement in the development set and over a 28% improvement in the evaluation set in unweighted average P@N, and a 24% and 34% relative improvement in weighted average P@N.

In Figures 1 and 2 we evaluate in more detail the performance (unweighted P@N) of each word in the development and in the evaluation sets for some of the systems. We see that by itself, the algorithm in [15] usually obtains worse results than the algorithm in [1] (black bar is followed by the white region). On the contrary, by applying the proposed improvements an important gain is obtained over all baselines for most words (grey bars).

System	Development set	Evaluation Set
Szöke[1]	0.4697 (0.4773)	0.4752 (0.4737)
Xu[15]	0.4354 (0.4568)	0.4550 (0.4820)
Xu + SB	0.5432 (0.5771)	0.5713 (0.5956)
Xu + SB + SC	0.5653 (0.5938)	0.5883 (0.6108)
Xu + SB + SC + CM	0.5986 (0.6267)	0.6117 (0.6349)

Table 2. unweighted P@N (weighted P@N) results for both systems averaged over all words, for development and evaluation sets

5 Conclusions and future work

We present an algorithm based on dynamic programming for the task of phoneme sequence search on phoneme lattices. Our proposal extends an algorithm previously used in speech recognition by adapting it to the task of isolated word retrieval. The proposed improvements include modifying the optimization procedure to search for the single best scoring path in the lattice, adapting the

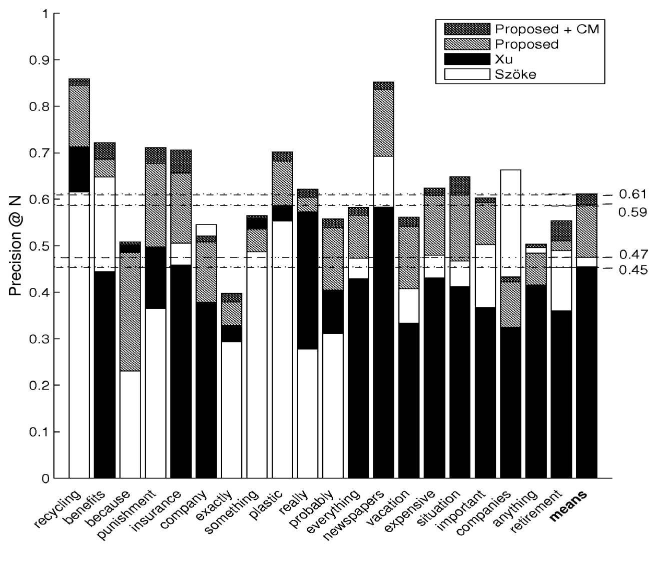

Fig. 2. Comparison of the methods for the evaluation set

scoring to balance it between path likelihood and phonetic mismatch and incorporating the estimation of a data driven loss function to exploit phoneme interactions present in the words pronunciations. The algorithm is evaluated on an isolated word retrieval task, obtaining very competitive results with previous state-of-the-art spoken-term detection methods obtaining over 20% relative improvement on a precision-at-N metric. Future work will focus on extending the algorithm to allow phonetic lattice to phonetic lattice search and retrieval. We plan to face the task of spoken term detection by adapting the algorithm to produce sub-sequence scores.

References

1. I. Szöke, P. Schwarz, P. Matjka, and M. Karafit, "Comparison of keyword spotting approaches for informal continuous speech," in *In Proceedings Eurospeech*, 2005.
2. S. J. Young, M. Brown, J. T. Foote, G. J. Jones, and K. Sparck Jones, "Acoustic indexing for multimedia retrieval and browsing," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*, vol. 1. IEEE, 1997, pp. 199–202.
3. U. Nambiar, T. Faruquie, L. V. Subramaniam, S. Negi, and G. Ramakrishnan, "Discovering customer intent in real-time for streamlining service desk conversations," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, ser. CIKM '11. New York, NY, USA: ACM, 2011, pp. 1383–1388. [Online]. Available: http://doi.acm.org/10.1145/2063576.2063776
4. C. Chelba and A. Acero, "Position specific posterior lattices for indexing speech," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ser. ACL '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 443–450. [Online]. Available: http://dx.doi.org/10.3115/1219840.1219895

5. J. T. Foote, J. T. Faate, M. Brown, G. Jones, S. Young, K. S. Jones, and J. S. J. Yaung, "Video mail retrieval by voice: Towards intelligent retrieval and browsing of multimedia documents," 1995.
6. J. T. Foote, G. J. F. Jones, K. S. Jones, and S. J. Young, "Talker-independent keyword spotting for information retrieval," 1995.
7. D. A. James and S. J. Young, "A fast lattice-based approach to vocabulary independent wordspotting," in *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94., 1994 IEEE International Conference on*, vol. 1. IEEE, 1994, pp. I–377.
8. J. Wilpon, L. Rabiner, C.-H. Lee, and E. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden markov models," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 11, pp. 1870–1878, Nov 1990.
9. K. M. Knill and S. Young, "Fast implementation methods for viterbi-based word-spotting," in *In Proc. ICASSP*, 1996, pp. 522–525.
10. P. Price, W. Fisher, J. Bernstein, and D. Pallett, "The darpa 1000-word resource management database for continuous speech recognition," in *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, Apr 1988, pp. 651–654 vol.1.
11. K. Thambiratnam and S. Sridharan, "Dynamic match phone-lattice searches for very fast and accurate unrestricted vocabulary keyword spotting," in *Proc. ICASSP*, vol. 5, 2005, pp. 465–468.
12. S. Young, W. P.C., and B. W.J., "Htk version 3.4.1: User, reference and programmer manual." September 1993.
13. K. Audhkhasi and A. Verma, "Keyword search using modified minimum edit distance measure," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4, April 2007, pp. IV–929–IV–932.
14. D. Wang, "Out of vocabulary spoken term detection." PhD thesis, Institute for Communicating and Collaborative Systems, School of Informatics, University of Edinburgh, 2010.
15. H. Xu, D. Povey, L. Mangu, and J. Zhu, "Minimum bayes risk decoding and system combination based on a recursion for edit distance," *Computer Speech and Language*, vol. 25, no. 4, pp. 802 – 828, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0885230811000192
16. M. A. Carlin, S. Thomas, A. Jansen, and H. Hermansky, "Rapid evaluation of speech representations for spoken term discovery." in *INTERSPEECH*, 2011, pp. 821–824.
17. V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals," in *Soviet physics doklady*, vol. 10, 1966, p. 707.
18. D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely, "The kaldi speech recognition toolkit," in *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*, vol. 2, no. 2. IEEE Signal Processing Society, Dec. 2011, pp. 2–3, iEEE Catalog No.: CFP11SRW-USB.