Strategies for Rescoring Keyword Search Results Using Word-Burst and Acoustic Features

Min Ma[1], *Justin Richards*[2], *Victor Soto*[3], *Julia Hirschberg*[3], *Andrew Rosenberg*[1]

[1]Department of Computer Science, The CUNY Graduate Center, New York, USA
[2]Department of Linguistics, The CUNY Graduate Center, New York, USA
[3]Department of Computer Science, Columbia University, New York, USA

{mma,jrichards}@gc.cuny.edu, {vsoto,julia}@cs.columbia.edu, andrew@cs.qc.cuny.edu

10.21437/Interspeech.2014-529

Abstract

The identification of keyword queries in speech data from low-resources languages poses a challenge for current methods as speech recognition algorithms lack sufficient training data to produce high accuracy transcript. To compensate for these shortcomings, we extract signals from the data that are useful in keyword identification but are not being used by the speech recognizer. These signals take multiple forms — word burstiness, rescored confusion network posteriors and acoustic/prosodic qualities. The former denotes the tendency for keywords to occur in bursts within a conversational topic. We employ three different strategies to exploit this information: 1) a four-way classification of keyword hypotheses that targets low-scoring correct hits and high-scoring false alarms, 2) ranking algorithms, and 3) a direct adjustment of keyword hit scores based on hypothesized repetition. We find that interpolating the results of these three strategies in an ensemble provides a reliable way to improve the results of keyword search.

Index Terms: keyword search, word burst, burstiness, IARPA-BABEL, ensemble, rank learning

1. Introduction

In this work, we improve on current methods of keyword search (KWS) for conversational speech in low-resource languages. These languages present a special challenge for KWS because the performance of automatic speech recognition (ASR) is poor due to a paucity of training data. We elaborate on standard ASR-based approaches by extracting new signals from KWS hypotheses to rescore hypotheses.

The first signal this work depends on is so-called *word burst* information. The assumption here is that word mentions occur in bursts as conversational topics emerge and shift. Thus, when analyzing a hypothesized keyword hit, we examine the temporal neighborhood of that hit, assess the number and strength of similar hypotheses in the neighborhood, and then rescore the candidate accordingly. If a hit candidate is surrounded by other hypotheses for the same keyword, its confidence score should be higher than that of a candidate that stands alone.

Building on the success of previous work using this approach[1, 2], we combine several methods of word burst analysis in an ensemble. Two of them are machine-learning approaches: a classifier that seeks to distinguish correct from incorrect hypotheses and a ranker that seeks an ideal ordering of hypotheses. The third seeks to boost a keyword hypothesis directly based on observations of similar hypotheses within a temporal window. Though these methods exploit the same phenomenon, they utilize different strategies. By ensembling them we seek to maximize our inclusion of the word-burst signal. We use this ensemble method on data from five low-resource languages: Pashto, Turkish, Tagalog, Vietnamese, and Zulu.

For the fifth language, Zulu, we construct a rescoring system based a distinct feature set that includes, for each hit, a prosodic analysis, data about that hit's potential impact on our KWS evaluation metric, and structural information about the ASR output that yielded the hit. These features have proven effective in rescoring the entire ASR system for low-resource languages [3] and here we examine their efficacy in late-stage rescoring. Finally, we ensemble results from this method with the results from word-burst experiments. By combining these presumably unrelated and complementary approaches, we seek a comprehensive rescoring strategy that uses disparate signals not captured by ASR to improve the efficacy of keyword search.

2. Previous Work

Many information retrieval techniques have been borrowed from text-based information retrieval (IR) to improve the performance of KWS. A primary approach is to transcribe the speech into text first, then directly employ the text-based IR techinques [4]. Two significant flaws are: 1) limitations on the performance of the ASR system and 2) the loss, during transcription, of potentially useful information [5, 7]. An alternative solution is to directly make use of additional information carried by the speech signal, measuring similarities of acoustic/prosodic and other non-linguistic features [8, 9]. Church pointed out that the word-burst phenomenon is common to many languages [10]. Exploiting this word-burst pattern, Chiu et al. modified lattice and consensus network (CN) confidence scores to improve KWS performance [1]. This approach improved KWS performance derived from limited data packs in five low-resource langauges. However, the performance gains on full language pack results were more modest. Richards et al. proposed burst-based machine learning models to bolster KWS performance on full language pack results [2]. Also in the context of rescoring KWS results, Soto et al. introduced a two-pass scheme to rescore KWS results for low-resource languages [3]. In the first pass, based on a set of lexical, phonetic, and structural features, they rescore ASR lattices to improve word error rate. In the second pass, they adjust result scores based on the rescored lattices. We use features from this system in our experiments (cf. Section 3.3.2). The investigation of various KWS methods by Mamou et al. demonstrated that an ensemble of diverse ASR systems can outperform the best single system [6]. Karakos et al. combined different rescoring systems by interpolating their respective scores for keyword hits [11]. Their work employed

Powell's method to optimize the interpolation coefficients. Both of these approaches show that KWS performance benefits from system combination as well as score normalization, a technique also used in this work.

3. Methods

3.1. Data

All data for this project is disseminated by the National Institute of Standards and Technology (NIST) on behalf of the Intelligence Advanced Research Projects Activity (IARPA). Our analyses here are based on conversation speech data provided by the IARPA Babel Keyword Search project[14]. The language packs are named IARPA-babel104b-v0.4bY, IARPA-babel105b-v0.4, IARPA-babel106b-v0.2g, IARPA-babel107b-v0.7, and IARPA-babel206b-v0.1e, for Pashto, Turkish, Tagalog, Vietnamese and Zulu, respectively. We use results from an ASR system trained on the full language pack of 40 speech hours for the first four languages and on the limited language pack of 10 speech hours for Zulu. An additional set of development and evaluation data, with a distinct list of keyword queries for each data pack, is released by IARPA. Development queries number approximately 300; evaluation queries number approximately 3,000. A keyword query may comprise one or more words.

The KWS result data we use is generated by the IBM Speaker-Adapted Deep Neural Network (SA-DNN) system for speech recognition, which outputs a pruned lattice of word hypotheses termed a consensus net (CN) [15]. In a lattice, multiple hypotheses of word boundaries are entertained, but a CN forces consensus on word boundaries. The arcs betweeen boundary nodes are potential transcriptions, with each arc weighted according to that transcription's likelihood. A set of arcs is known as a"bin." From this output, lists of putative keyword search results, designated as *posting lists*, are generated on the development and evaluation data. Each hit candidate in a posting list is assigned a confidence value, and it is this score that we target with our rescoring strategies.

NIST has released a subset of evaluation data for certain languages, so that Babel teams can evaluate their own systems. When we report results for Pashto, Turkish, Tagalog and Vietnamese, we do so on this subset, dubbed evalpart1, which constitutes our "test" set. The development and test audio each contain 10-15 hours of speech, depending on the language. For Zulu, we report results on development (dev) data, but using a keyword set distinct from the set we tune and train on.

3.2. Evaluation Metric

Term-weighted value (TWV) is the evaluation metric for the Babel KWS task. TWV involves a linear combination of P_{Miss}, the probability that a true hit for a given keyword was missed, and P_{FA}, the probability that a one-second window of time in the conversation was incorrectly identified as a hit for the given keyword[16]. The formula for TWV is $TWV(\theta) = 1 - P_{Miss}(\theta) + \beta * P_{FA}(\theta)$, where β is a constant set to 999.9 and θ is a decision threshold. Hypotheses with confidence scores below the threshold will not be scored [16]. TWVs for each keyword are averaged to yield actual TWV (ATWV).

Maximum TWV (MTWV) is the score that results after a search over possible thresholds is conducted and an optimal θ is chosen. In a posting list that yields a perfect MTWV, every correct hypothesis will have a higher confidence score than every incorrect hypothesis. The target of our rescoring is MTWV.

For parameter tuning, however, we employ a slightly different tactic. Since it involves a search over all possible thresholds, the calculation of MTWV is costly. Calculating ATWV is approximately 30 times faster. We use the threshold yielded by a baseline MTWV calculation as input to an ATWV formula that becomes our objective function in parameter tuning. Experiments have proven that this method is sufficient to yield gains in MTWV scoring in the test phase.

The distribution of scores among hit candidates can differ drastically from one keyword to the next, due in part to the language-model scores for keywords. Yet NIST requires that the same threshold be used for scoring each keyword query's hit candidates, so these distributions must be made comparable. Previous work has shown sum-to-one normalization by keyword to be an effective strategy [11]. Because posting list scores have been normalized this way prior to rescoring, we perform the same normalization on our rescore values. For four-class experiments (cf. Section 3.4.2) we re-normalize after the interpolation stage, but for ranking experiments we do not, as this degrades results.

In a modified form of sum-to-one (STO) normalization, hits with scores above .8 are left untouched, so that high-scoring hits are not too intensely diminished when a query yields many hypotheses. This modified STO strategy is involved in some of the features described in 3.3.2, but it is not used in experiments.

3.3. Features

We extract three types of features, each useful for rescoring. For experiments on Pashto, Turkish, Tagalog and Vietamese, we use only word-burst features to train a model. For experiments on Zulu, we incorporate additional speech features based on CN structure and acoustics.

3.3.1. Word-burst Features

We extract 25 burst-related features from each target hit hypothesis t in a posting list. These features involve calculations regarding the number, strength, and proximity of neighbor hypotheses n within a conversation. They include the length of the set N_t of neighbor hits; the maximum, minimum and standard deviations of N_t; $\frac{\mathrm{score(n)}}{|\mathrm{dist(n,t)}|}$; and $\sum_{i=1}^{N_t} \frac{\mathrm{score(n_i)}}{|\mathrm{dist(n_i,t)}|}$, where $\mathrm{dist(n,t)}$ denotes the distance in seconds between a neighbor hit to a target hit. Variations of the latter two formulae are computed using log- and square-root distance. Eleven of these features are computed twice: once for the speaker's side of the conversation only, and once across both sides of the conversation. The full feature set is enumerated and described in [2].

3.3.2. Consensus Net Features

To extract these we revisit the consensus bins described in 3.1. For each posting list entry we take the original and rescored CN posterior and aggregate them using several functions (mean, standard deviation, geometric mean, product, max and min). We also include the number of CN bins matching that hit, the total number of arcs, the average arcs per bin, the average number of epsilon arcs, the number of tokens and the ratio between the number of matched bins and the number of tokens in the keyword term. Additional details can be found in [3]. From the posting list itself we include the posterior score, the duration of the entry and whether the keyword is out-of-vocabulary.

We compute rank-normalized probabilities of false alarm for each pair of keyword query kw and posterior score ps in the corpus, following [17] and global re-ranked posterior scores as described in [11]. The reranked posterior is computed by

first mapping the pair (kw, ps) to its rank r and then mapping back the rank to the average of the posterior scores with that rank. Also included are the exponential normalization posterior score, its keyword-dependent threshold [18], the STO posterior score and the 0.8-STO posterior score described in 3.2.

3.3.3. Acoustic Features

We extract the pitch contour of the of the posting list entry and compute its median, mean, standard deviation, maximum and minimum, the number of unvoiced cycles in the segment and its percentage, the harmonics to noise ratio (dB) and noise to harmonics ratio, and the autocorrelation of the pitch contour. We also extract the pulses and include the number of pulses, the number of periods and their mean and standard deviation, along with the number of voice breaks and their percentage. Finally we include jitter values (local, local in seconds, its relative average perturbation (RAP) and its 5-point period perturbation quotient) and shimmer values (local, local in dB, and its 3, 5, and 11- amplitude perturbation quotient) as computed in Praat. All acoustic features are normalized at the segment level. Intonational phrase boundaries and pitch accents are detected using prosodic event detectors trained in cross-language corpus [19].

To create discriminative duration features, we start by computing the average duration of a hit given it is correct $dur(kw|CORR)$ and given it is a false alarm $dur(kw|FA)$, in the train partition. The following features are then computed for the feature vector: the absolute value and square power of the difference of the duration of the posting list and the average duration of correct hits $|dur(hit, kw) - dur(kw|CORR)|$, $(dur(hit, kw) - dur(kw|CORR))^2$; the absolute value and the square power of the difference of the duration of the posting list and the average duration of false alarm hits $|dur(hit, kw) - dur(kw|FA)|$, $(dur(hit, kw) - dur(kw|FA))^2$; and the corresponding ratios and inverses.

3.4. Rescoring Strategies

Our classification and ranking methods are applied on all the feature sets described above. The rule-based method does not utilize any extracted features and is used only for word-burst experiments. All of them, meanwhile, depend on a set of parameters; in order to find an optimal parameter setting, we perform a grid search on the tuning data. In the case of machine-learning experiments, we tune on the same data used to train our models, dev query results found in dev audio.

Ultimately, we use a linear interpolation of rule-based, 4-class and Ranking rescoring approaches to ensemble results. We perform a grid search to identify optimal interpolation weights on tuning data, and we apply these to the eval results.

3.4.1. Rule-based

The rule-based method of rescoring is an algorithm for selectively boosting the scores of hits that depends on three parameters: a score threshold τ, an increment size ι, and window size ω, expressed in seconds. For a target hit t in a given conversation file, we perform the following steps:

1) Assemble a list N_t of neighbor hits for the same keyword within the time window.

2) Find max_{N_t}, the highest-scoring hit in N_t, and compare its confidence score to τ.

3) If max_{N_t} is higher than τ, then boost the score of t by adding to it ι * the score of max_{N_t}.

For tuning, we search a 0 to 1 range for the increment and threshold parameters, and we search a range of 0 to 600 seconds (the duration of a conversation file) for the window parameter.

3.4.2. Four-way Classification

In training a machine-learning model, we assign to each hit one of four class labels: low-scoring correct hits, low-scoring false alarms, high-scoring correct hits, and high-scoring false alarms, with an intention to target low-scoring correct hits for rescoring. To incorporate the KWS evaluation metric into our learning model, we append a weight to the hits for each keyword according to that keyword's term-weighted value (TWV) in a baseline experiment. Because Zulu contains 2,000 keyword queries, compared to a few hundred for other languages, we downsample this data down to 503 queries before training.

We then train a logistic regression model, implemented by the Weka machine-learning toolkit[20], using 10-fold cross-validation. We generate predictions on the training data, outputting for each hit a distribution of confidence scores over the four class labels. We take a weighted average of these four class confidences and interpolate that average with the score of a given hit using the coefficient η, yielding the formula $R(t) = (1 - \eta) * s(t) + \eta * \sum_{k \in C}^{4} w_k * c_k$, where $R(t)$ is the rescore value, $s(t)$ is the original score of a target hypothesis t, C is the label set {LowCORR, LowFA, HighCORR, HighFA} of class confidences, and W is the set of co-indexed weights for those confidences. However, we replace $s(t)$ only if the new score is higher, so that we don't risk bringing correct hypotheses below the decision threshold. The TWV formula dictates that when the number of true hits for a keyword is low, which is quite common, the cost of erroneously shrinking a good hit is much higher than the cost of erroneously boosting a false alarm.

In tuning, we seek optimal values for each of the four class weights as well as the interpolation coefficient η. We find that setting non-zero weights for all high-scoring hits improves our rescoring result. This improvement seems to follow from subtle changes in our normalization step. In boosting predicted low correct hits, we erroneously boost some low false alarms as well. Then we perform sum-to-one normalization over a keyword's hits, and by boosting high hits before normalization, we seem to mitigate the damage done by boosting false alarms by pushing enough of them back down below the decision threshold. Since what improves rescoring is to boost all high-scoring hits, we tie those two classes together with the same weight.

This method is an elaboration of a previously published approach. We find that each addition to that approach — tied HiFA and HiCORR weights, the allowance of score boosting only, and the use of TWV instance weighting in learning — incrementally improves results. Another distinction is the use of ATWV as a highly efficient proxy for MTWV in tuning.

3.4.3. Rank Learning

Here we seek to reorder the scores in a posting list not by classifying each hit individually, but by learning and predicting an ideal ranking of the entire list[21]. To rank our training data accordingly, we balance the input of reference labels and ASR confidence. For each keyword in a posting list, we order its correct hits by CN score, followed by the a simliar ordering of false alarm hits, such that the highest-scoring false alarm is ranked directly below the lowest-scoring correct hit. After assigning ranks to hits, we append the features described in 3.3.1, then shuffle the list of instances to reduce learning bias.

We then train several ranking models using the learning-to-rank algorithms provided by the RankLib toolkit[22]. These

Table 1: *Results of Parameter Tuning on Multiple Languages.*

	Vietnamese	Tagalog	Turkish	Pashto
RB:(ι, τ, ω)	(1.0, 0.4, 250)	(0.8, 0.1, 50)	(0.6, 0.2, 200)	(0.8, 0.1, 50)
$4C$:(LoFA, LoC, HiFA, HiC, η)	(0, 0.7, 0.3, 0.3, 0.2)	(0, 0.3, 0.7, 0.7, 0.9)	(0, 0.5, 0.5, 0.5, 0.9)	(0, 0.4, 0.6, 0.6, 1.0)
R:(η, ranker)	(0.1, RF)	(0.1, $LMART$)	(0.1, CA)	(0.1, $MART$)
Ensemble:(RB, $4C$, R)	(1.0, 0, 0)	(0.1, 0.9, 0)	(0, 1.0, 0.0)	(1.0, 0, 0)

comprise pointwise (Multiple Additive Regression Trees, or MART), pairwise (RankNet,RankBoost and LambdaRank), listwise (AdaRank, Coordinate Ascent, LambdaMart, and ListNet), and bipartite (Random Forest) rankers.

We treat the choice of ranker as a system parameter, and we tune this parameter on our development data. (For Zulu, in order to expedite training on this larger dataset, we fix the ranker parameter at Random Forest, the most reliable algorithm on average in other trials.)

In generating predictions, the ranker outputs a list of global rank scores. In the event that some of the scores are negative, we preprocess the list by shifting it into a positive range. This is necessary for the STO step that follows. After normalizing the rank scores we combine them with posting list scores with an interpolation coefficient η.

4. Results and Discussion

Table 1 shows the parameters that yielded the best results on development data for our word-burst experiments on Vietnamese, Tagalog, Turkish and Pashto. For ranking, $MART$, CA, $LMART$, and RF correspond to Multiple Additive Regression Trees, Coordinate Ascent, Lambda Multiple Additive Regression Trees, and Random Forest, respectively. For the ensemble, RB, $4C$ and R correspond to the rule-based, four-class, and ranking results, respectively. Table 2 shows MTWV results with the best result for each language is shown in bold.

Table 2: *MTWV Results for Different Methods.*

Method	Viet.	Tagalog	Turkish	Pashto	Avg. / Δ
Baseline	.2980	.4899	.4492	.3923	.4074/NA
Rule-based	.3013	**.5035**	.4489	.4004	.4135/+1.52%
4-Class	**.3026**	.4993	**.4577**	**.4006**	.4151/+1.89%
Ranking	.3008	.4961	.4558	.3947	.4119/+1.10%
Ensemble	.3013	.5024	**.4577**	.4004	.4155/+1.99%

Table 3: *Results of Parameter Tuning on Zulu Data*

	Speech	Word-Burst	Combined
(ι,τ,ω)	NA	(.1,.5,50)	NA
(LFA,LC,HFA,HC,η)	(0,.9,.1,.1,.9)	(0,.9,.1,.1,.9)	(0,.3,.7,.7,.2)
(RB,$4C$,R)	(NA,.6,.4)	(0,0,1.0)	(NA,.8,.2)

Table 4: *MTWV Results on Zulu*

	Speech	Word-Burst	Combined
Baseline	.2006	.2006	.2006
Rule-based	NA	.2023	NA
4-Class	.2120	.1973	.1965
Ranking	.2008	.2009	.2008
Ensemble	.2083	.2023	.1978

For nearly every strategy assessed and every language studied, we find improvement to MTWV. The ensemble method, though it does not improve over the highest-scoring strategy, proves a reliable way to *choose* a strategy. There is no strategy that consistently performs the highest regardless of the language studied, but on average, the combined or selected strategy yielded by the ensemble method generates a higher MTWV than would any single strategy.

Our Zulu case study was the exception. Although word-burst features produced MTWV gains on nearly every language, the four-class and ranking methods did not significantly improve MTWV on Zulu. Burst-based predictions that improved results on development data were included in ensemble fusions, only to drive the ensemble result beneath that of the best-performing member. In addition to the Zulu ensembles presented in Table 2, we combine the three highest-performing Zulu results. The setting that maximized dev results gave a weight of 0, 0.5, and 0.5 to ranking burst, rule-based burst, and four-class speech, but the test result of .2082 did not outperform four-class speech alone. This lack of robustness from tuning to testing further indicates that the burst signal is weak for Zulu, which agrees with our understanding of that language. Zulu is highly agglutinative, meaning that a word rarely appears without prefixes, infixes, or suffixes that depend on the word's grammatical context. This complicates our burstiness assumption, especially when keyword queries are multi-word phrases, which they often are.

5. Conclusion and Future work

We demonstrate several KWS rescoring strategies that each prove effective, as well as an ensemble tool which serves as a broker ensuring that a reliable strategy or combination of strategies will be employed. Our application of a learning-to-rank method is novel in the low-rescource KWS context. We also introduce a separate set of features, drawn from acoustic signals and from the structure of the ASR output, which proves effective on the data that was most challenging for the word-burst approaches. This suggests that where word-burst rescoring fails, a complementary model in the rescoring ensemble can provide support. The challege of word-burst rescoring on Zulu indicates an avenue for future work. The burst method yielded middling results on Zulu due to the agglutinative nature of that language. We hope to overcome this obstacle by decomposing Zulu keywords into morphological segments. Thus we will search the conversation for the stem of the target keyword, replacing our word-burst analysis with a "morph-burst" approach.

Likewise, the union of burst features with speech features did not yield improvements for Zulu. This should not discourage future work with this augmented feature set. The individual feature sets have each proven their effectiveness in isolation.

6. Acknowledgment

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD / ARL) contract number W911NF-12-C-0012. The data used in our experiments is provided by the IARPA Babel Program language collection released in 2013. Disclaimer: The paper should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

7. References

[1] Chiu, Justin, and Alexander Rudnicky. "Using Conversational Word Bursts in Spoken Term Detection." (2013).

[2] Justin Richards and Min Ma and Andrew Rosenberg. "Using Word Burst Analysis to Rescore Keyword Search Candidates on Low-resource Languages." Proceedings of the 39th annual international IEEE ICASSP conference. 2014.

[3] Victor Soto and Erica Cooper and Lidia Mangu and Andrew Rosenberg and Julia Hirschberg. "Rescoring Confusion Networks for Keyword Search" Proceedings of the 39th annual international IEEE ICASSP conference, 2014.

[4] Zhou, Bowen, and John H. Director-Hansen. "Audio parsing and rapid speaker adaptation in speech recognition for spoken document retrieval." (2003).

[5] Mamou, Jonathan, Bhuvana Ramabhadran, and Olivier Siohan. "Vocabulary independent spoken term detection." Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2007.

[6] Mamou, Jonathan, et al. "System combination and score normalization for spoken term detection." Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.

[7] Lee, H.-y., et al. "Improved open-vocabulary spoken content retrieval with word and subword lattices using acoustic feature similarity." Comput. Speech Lang. 2014.

[8] Tejedor, Javier, et al. "Query-by-Example Spoken Term Detection ALBAYZIN 2012 evaluation: overview, systems, results, and discussion." EURASIP Journal on Audio, Speech, and Music Processing 2013.1 (2013): 1-17.

[9] Chen, Chia-ping, et al. "Improved spoken term detection by feature space pseudo-relevance feedback." INTERSPEECH. 2010.

[10] Church, Kenneth W. "Empirical estimates of adaptation: the chance of two noriegas is closer to p/2 than p^2." Proceedings of the 18th conference on Computational linguistics-Volume 1. Association for Computational Linguistics, 2000.

[11] Karakos, Damianos, et al. "Score normalization and system combination for improved keyword spotting." Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on. IEEE, 2013.

[12] Kingsbury, Brian, et al. "A high-performance Cantonese keyword search system." Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.

[13] Cui, Jia, et al. "Developing speech recognition systems for corpus indexing under the IARPA Babel program." Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.

[14] Draft KWS14 Keyword Search Evaluation Plan(OpenKWS 2014).http://www.nist.gov/itl/iad/mig/upload/KWS14-evalplan-v11.pdf.

[15] Soltau, Hagen, George Saon, and Brian Kingsbury. "The IBM Attila speech recognition toolkit." Spoken Language Technology Workshop (SLT), 2010 IEEE. IEEE, 2010.

[16] Fiscus, Jonathan G., et al. "Results of the 2006 spoken term detection evaluation." Proc. SIGIR. Vol. 7. 2007.

[17] Bing Zhang, Richard M. Schwartz, Stavros Tsakalidis, Long Nguyen, Spyros Matsoukas: White Listing and Score Normalization for Keyword Spotting of Noisy Speech. INTERSPEECH 2012.

[18] Chen, Stanley F. "Performance prediction for exponential language models." Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, 2009.

[19] Rosenberg, Andrew. "AutoBI-a tool for automatic toBI annotation." INTERSPEECH. 2010.

[20] Hall, Mark, et al. "The WEKA data mining software: an update." ACM SIGKDD explorations newsletter 11.1 (2009): 10-18.

[21] Hang Li: Learning to Rank for Information Retrieval and Natural Language Processing. Synthesis Lectures on Human Language Technologies, Morgan and Claypool Publishers 2011.

[22] Ranklib. http://sourceforge.net/projects/lemur/files/lemur/RankLib-2.3/.