USING WORD BURST ANALYSIS TO RESCORE KEYWORD SEARCH CANDIDATES ON LOW-RESOURCE LANGUAGES

Justin Richards[1] *Min Ma*[1] *Andrew Rosenberg*[1,2]

[1] CUNY Graduate Center {jrichards, mma}@gc.cuny.edu [2] Queens College/CUNY andrew@cs.qc.cuny.edu

ABSTRACT

For low-resource languages, keyword search (KWS) remains challenging due to the lack of training data. This work aims to bolster KWS performance in low-resource languages by incorporating word burst information into the decision process. We find that this information can improve performance when we focus analysis on particularly problematic KWS candidates: low-scoring correct hits, and high-scoring false alarms.

Index Terms— Keyword Search, Babel, Word Burst, Spoken Term Detection, Low-resource Languages

1. INTRODUCTION

The goal of this work is to improve the accuracy of keyword search (KWS) in conversational speech for low-resource languages. For languages that have adequate data for automatic speech recognition (ASR), many KWS systems have been developed with near-optimal performance [1]. However, their effectiveness depends heavily on the accuracy of transcription, which in turn depends on the amount of training data and language resources. Shortcomings emerge when dealing with low-resource languages[2]. Despite the abundance of such languages, their speech recognition output is rife with uncertainty. To account for this the search for a given term returns a large number of candidate locations with a range of assigned likelihoods, most of them very low. Thus, even in current state-of-the-art systems, transcription performance and confidence estimation remain fairly inaccurate. We seek to improve this output by extracting useful information from the data which is currently not incorporated into the ASR acoustic and language models, and using this information to re-rank the candidate hits. In this work we focus on so-called *word burst* information. The assumption of word burst, or burstiness, analysis is that once a word or phrase has been uttered in a conversation, the probability of that phrase's repetition is higher than its marginal probability.

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 release 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.

Our work with burstiness stems from an intuitive hypothesis. A word like "burstiness," which has a very low marginal probability in the language, should have a far higher probability in a discourse in which it has already been mentioned. We begin our work with the assumption that keywords are likely to be rare terms like "burstiness" and unlikely to be function words like "the", which are less affected by this burst phenomenon. In preliminary analyses, we validate these assumptions by counting keyword occurrences in reference data.

Our work integrates this burstiness information into the KWS system at the last stage, rescoring the decisions made by the system. We work with a KWS system that outputs hypothesized keyword locations in the form of a *posting list*. For each keyword, a posting list contains a series of candidate hits, each of which bears the properties beginning time, duration, source file, and confidence score. In an ideal system, every correct hit has a higher confidence score than every false alarm. Our work aims to reassign confidence scores in an attempt to move closer to this ideal. This means both lowering the scores of false alarms and raising the scores of correct hits. For this work, only the ranking of hits is important, not the absolute score. Therefore, the evaluation metric used is maximum term-weighted value (MTWV) — the efficacy of KWS regardless of optimal threshold detection.

We use machine learning to rescore candidate KWS hits based on word burst features extracted from the posting list. Unlike most burst models [3, 4], which define burst only in terms of word occurrence, we add additional burst based features associated with ASR confidence scores and distance proximity into our model. We classify the hit candidates according to predicted correctness and rank (above or below threshold). The confidence score of this class prediction parameterize a rescoring function which is then interpolated with that candidate's prior KWS score in order to generate new results. This approach effectively improves KWS performance on low-resource languages.

2. PREVIOUS WORK

An essential component of KWS is the decision maker, which examines each putative detection and determines whether it is a correct hit or a false alarm [5]. Typically, it maps the lattice-based features (e.g. ASR confidence) into a final KWS

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

score [6]. This kind of approach relies on the quality of ASR confidence scores. In the case of low-resource languages, we seek additional information that can improve the robustness of KWS to poor ASR performance and confidence estimates. One source of such information is word burst.

To take advantage of word burst features, several methods have been proposed. If a word tends to appear in bursts, its probability of occurrence will violate independent observation assumptions [7]. Therefore, various estimation methods of probabilities have been proposed. For estimating the probability of positive adaptation, Church and Gale [8] introduced a generalization of document frequency and estimated the probability of recurring words using: $P(+adapt) = P(k \geq 2|k \geq 1) \approx \frac{df_2}{df_1}$, where df_k referred to the number of documents that contain a word sequence of length $n > 0$ for k or more times. However, this method requires sufficient training data to reliably approximate $P(+adapt)$; where data is limited this representation will suffer. In 1997, Jelinek[9] proposed a cache language model to incorporate word burst into a language model by adapting the probabilities of a word after seeing it. In this way, he developed a dynamic language model which can model the actual documents more closely. The above approaches were designed for languages which have adequate training data. Few investigations have been devoted to KWS using word burst in low-resource languages. Recently, Chiu and Rudnicky [2] boosted the scores of words that recurred while penalizing words that occurred only once. Their work proved that the word burst-based method is particularly effective in limited resource languages. While their work rescored confusion networks (CNs), we make our improvements by rescoring the posting list alone. Additionally, their rescoring is based on direct adjustment of hit scores based on the proximity of repeated hits, while this work uses a larger set of burst-based features to optimize rescoring behavior. While effective on 10 hour (LimitedLP) data sets, this approach shows limited gains on 40 hours of data (FullLP).

3. METHODOLOGY

3.1. Data

The data for our experiments consists of conversational speech provided by the IARPA Babel project from two low-resource languages: Turkish and Vietnamese. The ASR output we work with was trained on the full language pack (FullLP) for each language, which contains approximately 80 hours of training data divided into files separated by speaker into conversational sides. (Further data, consisting of scripted speech, was also released but is not used in this work.) This choice is significant because previous work with burstiness rescoring saw nominal or no improvement on the FullLP[2]. Language packs for these languages include IARPA Babel Program language collections IARPA-babel105b-v0.4 (Turkish) and IARPA-babel107b-v0.7 (Vietnamese).

The data is divided into training, development (dev), and evaluation partitions. Reference data is provided for the development set so that posting lists can be scored for that data. Separate sets of dev and evaluation keywords are also released. 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). NIST safeguards the reference data for the evaluation set; however, in order to enable research on Babel data, NIST has released a subset, designated "part one," of the evaluation reference data so that researchers can perform local tests after tuning on development data. All evaluation results in this paper are generated from that "evalpart1" subset.

The speech recognizer used in our experiments was the IBM Speaker-Adapted DNN (SA DNN) system. The recognizer was trained using the FullLP training data and a deep neural network (DNN) acoustic model with the standard front-end pipeline [10].

3.2. Statistical Analysis

Empirical study of the Babel data validates the assumption that a phrase's likelihood will increase once that phrase has been introduced to a conversation. To investigate these patterns, we assemble a random subset of 200 dev keywords for each language, then combine training and dev data and analyze the occurences of keywords in all of those conversations. We find that the average marginal probability of a keyword's occurence in a conversation is orders of magnitude lower than the average probability of keyword repetition within a conversation in both Vietnamese ($1.4 * 10^{-7}$ vs. $8.0 * 10^{-6}$) and Turkish data ($1.4*10^{-6}$ vs. $1.7*10^{-5}$). In other such analyses, we have observed a rapid falloff in this elevated probability as the conversation moves farther from the initial utterance of the keyword. That is, the burstiness effect decays rapidly with time. This observation justifies our emphasis, in feature engineering, on the distance between a target hit and neighboring hits within a conversation.

3.3. Feature Extraction

We extract burst-related features for each target hit from the posting list. These features are enumerated below, where t represents the target hit, n is a neighboring hit candidate of the same keyword in the same conversation, N is the number of all such neighbor hits, $dist$ is distance in seconds between hits, $score$ is the confidence score of t from word lattices given by the ASR engine[11], and each feature description is followed by a bracketed feature count. Note that because each telephone conversation is split into two files, one for each speaker, the features described in 3 - 6 are computed twice: once for each speaker, and once for the entire dialogue.

1. keyword duration and start time (relative to the conversation) in seconds.[1]

2. ASR confidence score for t.[1]

3. the repetition count N. [2]

4. $\frac{score(n)}{dist(n,t)}$ for the neighbor n which is closest in time to t. This feature is also recalculated using log and square-root distance. [6]

5. $\sum_{i=1}^{N} \frac{score(n_i)}{|dist(n_i,t)|}$. Variations are computed as above.[6]

6. maximum, minimum, mean and std. dev. of ASR confidence scores over the set of all neighbor hits. [8]

3.4. Two-class rescoring

To generate the two class labels of correct hit (CORR) and false alarm (FA), we evaluate a development posting list using NIST's Framework For Detection Evaluations (F4DE) tool, which compares posting list candidates to reference data. F4DE's output includes an FA or CORR label for each hit in a posting list. Our goal is to train a classifier to predict which hits are CORR or FA based on word-burst features.

For all classification, we employ a logistic regression classifier implemented by Weka[12], which provides the best F-Measures on this data. We train our classifier on posting lists from the dev set, and we tune parameters on the dev set as well; testing is done on unseen evaluation data. Because only about one-tenth of posting list candidates are CORR, we address class imbalance problems by appending a higher classification weight to feature vectors with the CORR label. We search a range of possible values for these weights. Subject to the constraint that w_{FA} + w_{CORR} = 1, we search for an optimal weighting by searching $w_{CORR} \in [\frac{count_{total} - count_{CORR}}{count_{total}}, .5]$. That is, the highest weight is inversely proportional to the distribution of CORR and FA entries. This parameter is used in training the classifier, not in rescoring a posting list.

The classifier thus trained makes class predictions on the development posting list, with each prediction assigned a score on the interval of $[0, 1]$. We combine the original ASR confidence and the new classification confidence with mixing parameter η, as shown in the following formula:

$$R_b(t) = \begin{cases} s(t) * (1-\eta) + r(t) * \eta & \text{if } t \in CORR \\ s(t) * (1-\eta) + (1-r(t)) * \eta & \text{if } t \in FA \end{cases}$$

$R_b(t)$ is the final new score for hit candidate t, $s(t)$ is the base ASR score for t, $r(t)$ is the classifier confidence score for t's prediced class, and η indicates how much we assign to the predicted scores. After rescoring the hypotheses, we apply sum-to-one normalization to the set of hits for each keyword before doing an evaluation. This is an attempt to make scores compatible across keywords and is generally effective to improve KWS performance [11]. We use a grid search to find the optimal values of η and w_{CORR}. Table 1 shows the optimal parameters after tuning.

Language	w_{CORR}	η
Vietnamese	0.62	0.1
Turkish	0.54	0.1

Table 1. *Optimal two-class rescoring parameters.*

3.5. Four-class rescoring

In observing the results of two-class rescoring, we note that an excessive number of false alarms have their scores boosted. Moreover, the two-class model is forced to lump very low- and high-scoring correct hits together even though the optimal rescoring behavior for these two may be different.

Correct *relative* KWS scores are more important than correct absolute scores. Increasing the score of a correct hit will only improve KWS performance if its score is increased higher than that of a false alarm. The two-class approach attempts to boost the scores of all correct hits and reduce the scores of all false alarms. However, if false alarms already have low ASR-based scores, their scores do not need to be reduced, and if a correct hits have high ASR-based scores, their scores don't need to be boosted. The posting list entries that we care about are false alarms that have high ASR scores and hits that have low ASR scores. Inspired by the rescoring behavior necessary to improve KWS performance, we use a 4-class classification rescoring approach. Thus, we define

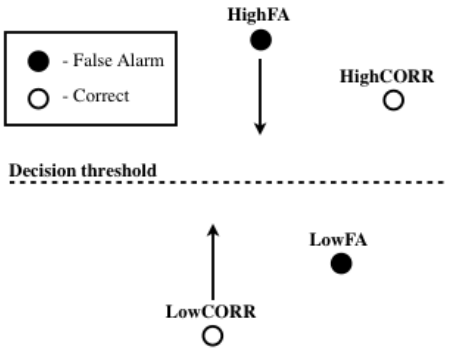

Fig. 1. Definition of four-class labels and rescoring direction.

four classes: HighCORR, LowCORR, HighFA and LowFA (cf. Figure 1.).As with the two-class method, we generate labels by running a F4DE evaluation on baseline data, which also returns the optimal decision threshold. We use this decision threshold to split FA labels into HighFA and LowFA, and we do likewise for correct hits.

Again, we use Weka's logistic regression classifier to predict classes on the development set, and we output the full distribution of all four class confidences. In rescoring, we assign a weight w_k to each of the four confidence scores. The new weighted rescoring formula is defined as follows, $R_f(t) = (1-\eta) * s(t) + \eta * \sum_{k \in C}^{4} w_k * c_k$ where $s(t)$ is the original score of a keyword hypothesis from ASR output and $C = \{\text{LowCORR, LowFA, HighCORR, HighFA}\}$. We use a

grid search to determine the optimal five-parameter tuple for each language.

Table 2 shows the optimal parameters for each language after a grid search. Boosting HighFAs, a strategy counter-intuitive at first appearance, seems to have a salutary effect: By raising high-scoring hits (of which HighFAs are the most common) before STO normalization, we effectively push down the scores of hits at the lower end of the distribution. Among those scores pushed down are actual LowFA hits which were erroneously boosted by the weight on LowCORR predictions. In this way boosting HighFA-predicted hits before normalization mitigates the errors made in boosting LowCORR-predicted hits.

Language	w_{LC}	w_{LF}	w_{HC}	w_{HF}	η
Vietnamese	0.6	0.0	0.0	0.4	0.1
Turkish	0.1	0.0	0.0	0.9	0.9

Table 2. *Optimal four-class rescoring parameters where Low/High are abbreviated L/H, and CORR/FA as C/F.*

4. RESULTS

4.1. Classification-based Rescoring

We measure KWS performance using the evaluation measurement defined by NIST for KWS evaluation, Term-Weighted Value (TWV). TWV generates a weighted linear combination of P_{Miss} and P_{FA} such that false alarms are far more costly than misses. A miss is defined as a correct hit that falls below the decision threshold. The formula for TWV is $TWV(\theta) = 1 - P_{Miss}(\theta) + \beta * P_{FA}(\theta)$ where θ is the decision threshold and β is a weight constant set to 999.9[1]. Maximum TWV (MTWV) is the score that results after a search through possible thresholds is conducted and an optimal θ is chosen. KWS is also evaluated using Actual TWV (ATWV) where a threshold is specified about which terms are considered "hits". ATWV calculation requires identification of an optimal KWS score threshold. Our focus in this work is on the optimal reordering of candidate hits rather than threshold optimization. Thus, Table 3 displays MTWV on eval data using the two- and four-class methods.

Language	Classes	Baseline	Rescore	% change
Vietnamese	2	0.2980	0.2942	-1.3
Vietnamese	4	0.2980	0.3002	+0.7
Turkish	2	0.4492	0.4532	+0.9
Turkish	4	0.4492	0.4560	+1.5

Table 3. *Rescoring results on evaluation data.*

Experimental results support two of our primary hypotheses. First, four-class models outperform two-class models by isolating the classes of hits for which rescoring is most imporant: high-scoring false alarms and low-scoring correct hits. Second, a machine learning algorithm is able to learn hit likelihood based on burstiness features.

The implications of these findings may apply to ASR as well as KWS. A low-resource ASR system which incorporates burstiness information into its language model may improve its accuracy. Indeed, one interpretation of our work is that it addresses a gap in the ASR language model. A language model typically calculates the probability of a word's occurrence using its position in short-range n-gram sequence. Our work suggests that by using some simple, long-range statistics about the word's prior or future occurence in the discourse, a language model can be made more accurate.

4.2. Cross-Language Evaluation

Burstiness may be a universal feature of human conversation. Here we investigate the cross-language generalization of four-class word burst rescoring. Results are shown in Table 4. Indeed, we find cross-language testing produced better results on Vietnamese than that language's own model and parameters did. However, cross-language modeling in the other direction reduces results on Turkish. We hesitate to suggest that the Turkish model represents a language universal model of word burst. It is possible that Turkish is more robust due to the increased amount of training data (14k vs. 6k hits). This does suggest that for low-resource languages, cross- or multi-language training of word burst models might have a salutary effect on keyword rankings.

Eval Language	Baseline	Rescore	% change
Vietnamese	0.2980	0.3013	+1.1
Turkish	0.4492	0.4027	-10.4

Table 4. *Cross-Language word burst rescoring.*

5. CONCLUSION

In this paper, we describe an effective classification-driven approach to rescoring KWS results in Turkish and Vietnamese using word burst information. We find that isolating hits that are more likely to benefit from rescoring (low scoring correct hits, and high scoring false alarms) allows us to rescore more effectively than classifying correct vs. false alarm candidates. In applying word burst rescoring models across languages, we find some degree of language independence in the context of these experiments. Determining the robustness of this independence is a direction for future work. We also plan on investigating the role of morphology in word burst. Turkish is morphologically rich, and burst analysis may be more effective on the morph rather than word level. Finally, we will explore direct optimization of TWV in rescoring rather than an exhaustive parameter search.

6. REFERENCES

[1] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddington, "Results of the 2006 spoken term detection evaluation," 2007, pp. 45–50.

[2] Justin Chiu and Alexander Rudnicky, "Using conversational word bursts in spoken term detection," in *INTERSPEECH*, 2013.

[3] Jon Kleinberg, "Bursty and hierarchical structure in streams," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 373–397, 2003.

[4] Rasmus E Madsen, David Kauchak, and Charles Elkan, "Modeling word burstiness using the dirichlet distribution," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 545–552.

[5] Javier Tejedor, Doroteo T Toledano, Miguel Bautista, Simon King, Dong Wang, and José Colás, "Augmented set of features for confidence estimation in spoken term detection," 2010.

[6] Frank Wessel, Klaus Macherey, and Ralf Schluter, "Using word probabilities as confidence measures," in *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*. IEEE, 1998, vol. 1, pp. 225–228.

[7] Kenneth Ward Church and William A Gale, "Poisson mixtures.," *Natural Language Engineering*, vol. 1, no. 2, pp. 163–190, 1995.

[8] Kenneth W Church, "Empirical estimates of adaptation: the chance of two noriegas is closer to p/2 than p 2," in *Proceedings of the 18th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, 2000, pp. 180–186.

[9] Frederick Jelinek, *Statistical Methods for Speech Recognition*, Language, Speech, & Communication: A Bradford Book. MIT Press, 1997.

[10] H. Soltau, G. Saon, and B. Kingsbury, "The IBM Attila speech recognition toolkit," in *Proc. SLT*, 2010, pp. 97–102.

[11] Damianos Karakos et al., "Score normalization and system combination for improved keyword spotting," in *to appear in IEEE Automatic Speech Recognition and Understanding Workshop*, 2013.

[12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.