6-3-2014

Echolocation: Using Word-Burst Analysis to Rescore Keyword Search Candidates in Low-Resource Languages

Justin Richards
Graduate Center, City University of New York

Echolocation:

Using Word-Burst Analysis to Rescore Keyword Search Candidates in Low-Resource Languages

by

JUSTIN RICHARDS

A master's thesis submitted to the Graduate Faculty in Linguistics in partial fulfillment of the requirements for the degree of Master of Arts, The City University of New York

2014

This manuscript has been read and accepted for the Graduate Faculty in Linguistics in satisfaction of the dissertation requirement for the Master of Arts.

Andrew Rosenberg

Thesis Advisor

Date

Gita Martohardjono

Executive Officer of Linguistics

Date

Abstract

ECHOLOCATION: USING WORD-BURST ANALYSIS TO RESCORE KEYWORD SEARCH CANDIDATES IN LOW-RESOURCE LANGUAGES

by

Justin Richards

Advisor: Professor Andrew Rosenberg

State of the art technologies for speech recognition are very accurate for heavily studied languages like English. They perform poorly, though, for languages wherein the recorded archives of speech data available to researchers are relatively scant. In the context of these low-resource languages, the task of keyword search within recorded speech is formidable. We demonstrate a method that generates more accurate keyword search results on low-resource languages by studying a pattern not exploited by the speech recognizer. The word-burst, or burstiness, pattern is the tendency for word utterances to appear together in bursts as conversational topics fluctuate. We give evidence that the burstiness phenomenon exhibits itself across varied languages. Using burstiness features to train a machine-learning algorithm, we are able to assess the likelihood that a hypothesized keyword location is correct and adjust its confidence score accordingly, yielding improvements in the efficacy of keyword search in low-resource languages.

Contents

List of Figures

List of Tables

1

Introduction

The process of identifying keyword phrases in recorded speech requires a number of interacting components, among them language modeling, speech recognition, and keyword hypothesis scoring. The experiments described in this work operate on hypothesis scoring, a late-stage component, but the principle of discourse analysis informing them implies improvements in language modeling and speech recognition as well.

Locating orthographically presented keywords in speech data requires a competent transcription of that speech. Broadly speaking, the most common system of speech recognition in current use works by analyzing massive amounts of already-transcribed speech in order to learn a sophisticated correspondence from speech sounds to written words. Such a system contains three mappings: an acoustic model, which matches acoustic input to words in its vocabulary, a pronunciation model, which maps orthographic symbols to phonemes, and a language model, which tracks the likelihood of two- or three-word sequences (bigrams or trigrams) as observed in the aforementioned training data. In determining the likelihood of a putative transcription, the system will multiply the likelihood of that word's production from the given acoustics by the likelihood of that word's appearance after the words hypothesized to precede it. When large troves of training data are available, as they are for widely spoken and heavily studied languages like English and Mandarin Chinese, these two components suffice to produce near-optimal transcriptions of new speech data. Searching for keywords within accurately transcribed speech is essentially a solved problem. When a much smaller amount of data is available, however, speech recognition output and thus keyword search are impaired. The low-resource languages this work focuses on are Pashto, Tagalog, Turkish and

Vietnamese. In experiments with these languages, the recognizer is trained on only twenty hours of recorded speech. The acoustic, pronunciation and language models do not suffice as described above, and they must be altered or supplemented in order to improve results. Efforts here are focused on the language model, in particular on the limitations of the highly localized n-gram analyses that are traditionally used to estimate word likelihood.

N-gram probabilities, i.e. the likelihood that a particular word will occur given its status as the second, third, or nth member in some word sequence, are produced by counting the frequency of these sequences in training data, and they are presumed to apply universally across a language or at least within a certain domain for that language. N-gram probabilities calculated from English medical texts, for example, are expected to be equally useful for any medical discourse that occurs in English. Yet they are not. Consider a recognizer that has just detected the word body and for the next acoustic signal is weighing several transcriptions phonetically similar to the word "pass," among which "mass" and "cast" receive comparable scores from the acoustic model. Those scores will be multiplied with the language model scores, and "mass" will probably win out due to the higher frequency of the bigram "body mass" in the recognizer's training data. Now let us add the premise that the word "cast" has already occurred several times in the discourse. A human listener would tend toward choosing "cast" in the current context, but a typical speech recognizer would choose otherwise. A recognizer with the level of uncertainty described in this scenario e.g., one without enough training data to properly hone its model will make these sorts of errors if it relies entirely on universal n-gram probabilities and ignores the localized rise and fall of discourse topics. This fluctuation is often referred to as word burstiness, a term that will be used throughout this work.

Madsen et al [1], as well as Doyle and Elkan [2] have used word burstiness in topic modeling, but a similar pattern appears in other realms. The burstiness phenomenon seems to belong generally to processes wherein selections are made repeatedly from a very large reservoir of possibility. Airoldi et al [3] encountered bursty patterns in gene expression — i.e., if a gene has been recently transcribed in a cell, it is more likely to be transcribed again while Fei-Fei and Perona [4] found them in computer vision i.e., if a patch of visual data is observed in one part of an image, it is more likely to be observed elsewhere in the image.

In work closer to the speech recognition and keyword search tasks, the burstiness assumption

has been borne out by the relative success of adaptive language models, which take into account the likelihood of word repetition. In an analysis of the Brown Corpus, Church [5] found that the probability of a words repetition within a document is often several orders of magnitude greater than its marginal probability. In fact, Church argues that the probability of a words repetition has almost no dependence on word frequency. It has much more to do with that words lexical status i.e. content words are much more likely than function words to change their likelihood in this way. The distortion of a keywords marginal probability is thus especially pronounced for rare words probable candidates for search keywords. We might not expect lightning to strike twice, but it happens all the time, Church writes, especially for good keywords.

Burstiness modeling has been attempted in speech recognition through the use of a cache to track a local lexicon within the discourse. The term cache is borrowed from a similar strategy employed in computer architecture. It behooves a computers central processor to have especially quick access to the units of memory it is most likely to call upon, and it has been observed that computers access memory locations in repetitive or bursty patterns [6]. To store recently accessed units of memory for easy retrieval, designers built a small, high-speed memory holder close to the CPU. Likewise, cache-based language models keep a store of the n most recently used words. When the n+1th word is added, the least recently seen word is ejected from the cache. Kuhn and Mori developed a language model that included a trigram component and a 200-word cache component. They stored a separate cache for each of 19 parts of speech. Employing their method on text data and using perplexity as their objective function, they obtained the optimal weights on the trigram and cache components for every part of speech. They found that preference for the cache component was not restricted to content words, and in fact it was especially high for articles, pronouns and conjunctions. This provides some justification for our disregard, in this work, for a keywords part of speech. Burstiness is a general phenomenon of language the English language, at least not restricted to particular word categories. Kuhn and Mori use their results to draw an interesting conclusion about human language use. They suggest that people do not draw from their entire vocabulary in a consistent distribution when they speak or write, but move instead through sublanguages or language islands, which researchers can attempt to chart in order to model language effectively. Indeed, Jelinek et al used a similar combination of trigrams and caching in the context of speech recognition; after caching less than a thousand words, they achieved a 24 percent

decrease in the error rate of their recognizer.

If we presume now that burstiness is a real phenomenon which can be exploited to the benefit of language modeling and speech recognition, then there is an apparent drawback to the caching method. If the sublanguages described by Kuhn and Mori can be likened to islands, they are islands that extend in two directions. That is, the likelihood of a given word should be influenced not only by the vocabulary behind it but ahead of it as well. For the task of keyword search, different choices can be made. Chiu and Rudnicky [7] search within a window of surrounding discourse for a particular word when considering that words candidacy for transcription of an audio signal. They work in a context very similar to that of this research: improving keyword-search results on recorded speech in low-resource languages. Instead of incorporating word burstiness into the language model used in the original speech recognition system, they employ it, as we do, as a post hoc addition to the system, rescoring the speech recognition output. That output is, in this case, a so-called lattice consisting of nodes (word boundaries) connected by multiple arcs (hypothesized words), wherein each arc is assigned a score according to that words probability as determined by the acoustic and language models. Chiu and Rudnickys algorithm dictates that, for a given word not classified as a stopword (high-frequency word likely to be a function word), if no identical partner of that word is found in the lattice within a particular window, that words score should be decreased. This method yielded modest gains in the keyword search evaluation metric. One drawback of their approach is that three parameters must be fine-tuned: the percentage of the lexicon to reject as stopwords, the discourse window that defines the burstiness search, and the degree to which the target words score might be decreased. This costs time and computer memory, and the effectiveness of the language model is sensitive to the selection of parameter values.

This work differs from its predecessors by taking a machine-learning approach to burstiness modeling. In addition, it is entirely focused on the keyword search task, so it analyzes word-burst patterns only in keyword search results, not in the full speech transcript. For a target candidate in a result list, we generate a set of word-burst features which profile the strength and proximity of similar hypotheses about the same word in the discourse environment. We then train a classification algorithm to learn from these features whether the target candidates likelihood score should be boosted.

2

Corpora

The data for this project was 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 conversational speech data provided by the IARPA Babel Keyword Search project [8]. We develop and test our system on four different packages of recorded speech: Pashto, Turkish, Tagalog, Vietnamese and Zulu. Because the task in this work is to rescore keyword search results, we do not process the speech data directly. Instead, we work on results produced by other researchers. The keyword search result data we use is generated by the IBM Speaker-Adapted Deep Neural Network system for speech recognition, which outputs a pruned lattice of word hypotheses termed a consensus net [9]. Keyword hypothesis are plucked from that consensus net and used to populate a *posting list*. A posting list is a by-query listing of search results, where each putative hits entry includes a conversation identifier, a begin time and confidence score. Before normalization and rescoring, the confidence score for a hit entry in a posting list is equivalent to the weight on the consensus net arc that yielded that hit.

The speech recognizer was trained on forty hours of conversational speech for each language. Keyword search is performed on one of two additional sets of speech data: the development and evaluation partitions. What is referred to as evaluation data in this work is actually a small subset of the full evaluation partition released by NIST, but it is still larger than the development partition. A separate set of queries is searched for in each partition. The development queries number approximately 300, depending on the language, while the evaluation queries number approximately 3,000. Queries range in length from one to five tokens.

3

Statistical Analysis

We perform an empirical study of keyword appearances in Vietnamese and Turkish data to validate the assumption that a keyword phrase's likelihood will increase once that phrase has been introduced to a conversation. We assemble a random subset of 200 dev keywords for each language, then combine training and dev data and analyze the occurrences of keywords in all of those conversations. We find that within a conversation, the average probability of a keyword occurrence is orders of magnitude lower than the average probability of keyword repetition in both Vietnamese ($1.4 * 10^{-7}$ vs. $8.0 * 10^{-6}$) and Turkish data ($1.4*10^{-6}$ vs. $1.7*10^{-5}$). We choose these two languages because they stand at opposite ends of a spectrum that might affect word-burst patterns. Vietnamese words do not use morphological markings like suffixes and prefixes, while Turkish is morphologically rich. For morphologically rich language data, our system might miss tokens which should be considered word repetitions but which, because of morphological variation, are not identical to the target word. Our findings do not suggest that the morphological disparity between Vietnamese and Turkish has a significant effect on burstiness.

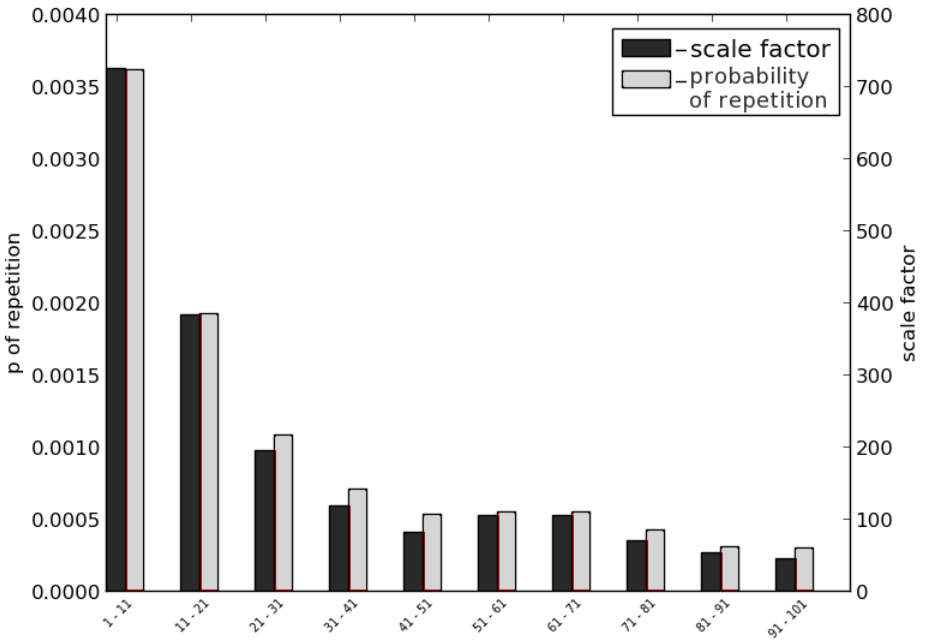
Furthermore, as depicted in Figure 3.1, we observe a rapid falloff in the elevated probabilities associated with burstiness as the conversation moves farther from the initial utterance of the keyword. That is, the burstiness effect decays with time. The results are averaged over 4,065 keywords in 1,042 10-minute conversation files. The horizontal axis lists the window regions tested. Here, unlike in our rescoring method, a window region is measured in word tokens, not in seconds. The left-hand vertical axis shows the marginal probability, within a particular window of word tokens, of keyword repetition; the right-hand vertical axis shows the scale factor by which that in-window

probability is greater than the keywords marginal probability. For example, a keyword might have a marginal probability in the data of .001; its marginal probability within a repetition window, meanwhile, might might be greater than that by a factor of 200, yielding a repetition probability of .2. This observation justifies our emphasis, in feature engineering, on the distance between a target hit and neighboring hits within a conversation.

The findings in this section show that the observations made by [1] and [2] are not restricted to English. Beyond that, they show that one can be completely agnostic about a language with no prior knowledge of the languages grammar or lexicon and still observe a word-burst effect. In fact, the methods of analysis used here and later in our experimental techniques treat the keyword hit candidates as generic data points, without regard to their meaning as pieces of language.

Figure 3.1: Keyword Burstiness by Window Size in Vietnamese Data

.



4

Methods

This section outlines the machine-learning technique used to classify hit hypotheses as targets or non-targets for rescoring and to adjust the scores of hits in a posting list according to those predicted classifications. For each hit, we assign a class label that indicates whether its score should be raised or not, then we extract features that describe its status per burstiness analysis. We train a logistic regression algorithm to learn a mapping between burstiness feature value and predicted class label. That algorithm then makes predictions on new data, and we use the class confidences associated with those predictions to adjust the scores in a posting list.

4.1 Evaluation Metric

I use term-weighted value, defined below, as the evaluation metric for a rescored result file. Metrics like accuracy, precision, and recall are not quite appropriate to the keyword search task, as there is not a fixed number of choices that can be made correctly or incorrectly as there is when, for example, distinguishing spam emails from non-spam. With search, the goal is to identify sparse important items within a sea of irrelevant data; in evaluation, we want to appropriately weigh the high importance of making a good find against the drawbacks of falsely identifying an item as a keyword. TWV allows us to do that. It 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 [10]. 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 decision threshold are not scored [10]. The TWV for each keyword is averaged to yield actual TWV (ATWV).

Once a threshold is has been drawn, the scores of hit hypotheses are not taken into account in calculating $P(FA)$ and $P(MISS)$. Thus the absolute scores of hit hypotheses are essentially unimportant. What matters is the *relative* scoring of results. In this light our work can be conceptualized as reranking just as well as rescoring. An suitable way to assess this ranking is Maximum TWV (MTWV). MTWV is the score that results after a search over possible thresholds is conducted and an optimal value is chosen. In a posting list that yields a perfect MTWV, every correct hypothesis will have a higher confidence score than every incorrect hypothesis. Because we seek to approximate this ideal ranking, the target of our rescoring work is the MTWV metric.

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.

4.2 Normalization

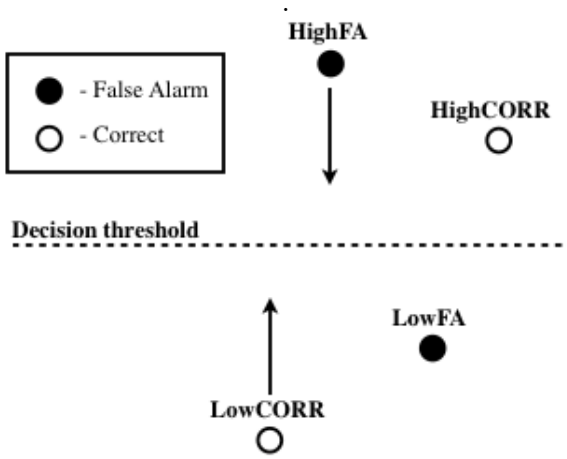
The distribution of scores among hit candidates can differ drastically from one keyword to the next, as the language model, acoustic model, and pronunciation model scores do not fit the same distribution for each keyword. Rare words, for example, are likely to be penalized. Meanwhile, TWV scoring requires that we apply the same threshold to each keyword query's hit candidates, so their distributions must be made comparable. Previous work has shown sum-to-one normalization by keyword to be an effective strategy[11]. The posting lists that we work with have been normalized in this way prior to rescoring, so we perform the same normalization on our rescore values. We normalize a third time after interpolating our rescore values with the original scores.

4.3 Classification

Recall that the ideal score ranking in a result file places each correct hypothesis above each incorrect hypothesis, or false alarm. If this arrangement is obtained, the decision threshold will be drawn

between the highest-scoring false alarms and the lowest-scoring correct hit. Presuming a fixed decision threshold, though, one can move toward this goal by pushing low-scoring correct hits above the threshold and pushing high-scoring false alarms below it. The first step is to identify these low-scoring correct hits and high-scoring false alarms, which can be achieved by parsing the hit candidates into four classes: Low False Alarm, Low Correct, High False Alarm, and High Correct. Figure 4.1 illustrates these class assignments visually. These four classes are the targets of our machine learning predictions. To assign class labels, a hits correct-or-incorrect status can be determined from the reference transcript, and its high-or-low status can be determined by its posting list score relative to the decision threshold selected in a control experiment.

Figure 4.1: Assignment of Class Labels



4.4 Feature Extraction

For each hit hypothesis, we generate a set of burstiness features that the machine-learning algorithm will use to assess whether that hit should be classified as a Low False Alarm, Low Correct, High False Alarm, or High Correct. 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 of neighbor hits
- The maximum, minimum and standard deviations of N
- $\frac{score(n)}{dist(n,t)}$

- $\sum_{i=1}^{N} \frac{score(n_i)}{|dist(n_i,t)|}$, where $dist(n,t)$ denotes the distance in seconds between a neighbor hit to a target hit

Variations of the last two formulae are computed using log- and square-root distance because, based on the observations presented in 4.1, the burstiness effect seems to decay with time in a non-linear way. Eleven of the features are computed twice: once for the speaker's side of the conversation only, and once across both sides of the conversation. Thus we consider two kinds of burstiness: within speaker and within discourse.

4.5 Algorithm

I use the logistic regression classification algorithm, implemented with the Weka machine-learning toolkit [12] for this machine learning task. A logistic regression classifier works by finding a mathematical function that separates the data points it believes to be in one class from the data points it believes to be in another. A data point (here, a hit candidate) with n features can be represented as a geometric point in n-dimensional space. All of the data points used to train the algorithm can thus be imagined as a constellation of points in that space. If the features are closely correlated with class assignments, data points will to some degree be clustered by class within that constellation. Logistic regression will then generate a mathematical function corresponding to an n-dimensional plane that separates one class from another. When called upon to generate class predictions on a new constellation of data, the algorithm will introduce that boundary into the space and use it to draw class distinctions. The algorithm will also assign a prediction confidence according to a data points proximity to the boundary, such that an instance very far from the boundary receives a high confidence score and an instance close to it receives a lower score.

4.6 Interpolation

After training a model, the classification algorithm outputs predictions on new data, with a confidence score assigned to each prediction. we make use of all four confidence scores by combining them in a linear interpolation, or weighted average. In rescoring a posting list, the combined score is interpolated with the original speech recognition score. This is done so that when the prediction

Table 4.1: *Results of Parameter Tuning on Multiple Languages.*

Language	Low FA	Low Correct	High FA	High Correct	η
Pashto	0.0	0.4	0.6	0.6	1.0
Turkish	0.0	0.5	0.5	0.5	0.9
Tagalog	0.0	0.3	0.7	0.7	0.9
Vietnamese	0.0	0.7	0.3	0.3	0.2

score differs greatly from the original score, some trust will be placed in the original score and the changes will be mitigated. As determined by these interpolation conditions, the formula for rescoring is as follows: $R(t) = (1-\eta) * s(t) + \eta * \sum_{k \in C}^{4} w_k * c_k$, where $R(t)$ is the rescore value, $s(t)$ is the original score of a target hypothesis t, C is the label set {Low Correct, Low False Alarm, High Correct, High False Alarm} of class confidences, and W is the set of co-indexed weights for those confidences. In practice, we replace $s(t)$ only if the new score is higher, so that we don't risk bringing correct hypotheses below the decision threshold.

4.7 Parameter Tuning

The classifier outputs predictions on both the training data and one on the testing data. The predictions on the training data are used to tune the parameters described above, i.e. weights on all four class confidences as well as the final interpolation parameter η. Note that the four class weights require only one actual parameter to be tuned. Low False Alarm is held at zero, Low Correct must be tuned, High False Alarm is equal to one minus Low Correct, and High Correct is set to be equal to High False Alarm, for reasons described below. Incrementing their values by .1 at each iteration, we perform a grid search to explore all possible parameter values.[1] At each iteration of the search, a training posting list is rescored and evaluated. As described in 4.1, a slightly different evaluation metric is used for tuning as will be used in testing. This is done in order to save time. Results of parameter tuning are shown in Table 4.1.

In earlier experiments, we found that setting non-zero weights for all high-scoring hits improves the 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.

[1] In earlier trials, we tuned parameters using optimization techniques such as the Simplex and Powell's Method algorithms, but a simple grid search yielded much better results.

Then we perform sum-to-one normalization over a keyword's hits, and by boosting high hits before normalization, the damage done by boosting false alarms seems to be mitigated 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.

5

Results

In this section, we present two sets of results. First, we show the outcome of the process described in Section 4: train a model on a language's data, tune parameters on the development queries and development data for that language, then test on the evaluation queries and evaluation data for that language. In the following subsection, we perform cross-language evaluation, using the model and parameters from one language to test on another language's data.

5.1 In-Domain Rescoring Results

The results in Table 5.1 support our primary hypotheses: that a machine learning algorithm is able to learn the likelihood of a true hit based on burstiness features, and that those likelihoods can be used to improve the ranking of keyword search hypotheses. These results also suggest that the more training instances used to train the classifier, the stronger the effect on MTWV. Although the amount of speech data for each language is the same, the number of hypothesized hits varies widely among languages. The Pashto results contain more than ten times as many results as the Vietnamese results, while the Turkish and Tagalog numbers fall in the middle. MTWV results,

Table 5.1: *Rescoring results on evaluation data.*

Language	Baseline	Rescore	% change
Pashto	0.3923	0.4006	+ 2.1
Turkish	0.4492	0.4577	+ 1.9
Tagalog	0.4899	0.4993	+ 1.9
Vietnamese	0.2980	0.3026	+ 1.5

Table 5.2: *Cross-language word burst rescoring.*
MTWV by Target Language

Training Language	Pashto	Turkish	Tagalog	Vietnamese	Avg. Improvement
Baseline	.3923	.4492	.4899	.2980	-
Pashto	*.4006*	.4574	**.5006**	.3013	1.8 %
Turkish	.3978	*.4577*	.4991	.3022	1.6 %
Tagalog	.3978	.4564	*.4993*	.3004	1.4 %
Vietnamese	.3990	.4537	.4986	*.3026*	1.5 %

correlated with these data sizes, bear out the notion that more training data can increase the efficacy of our system.

5.2 Cross-Language Evaluation

Intuition and historical findings suggest that burstiness is not a broad term for a variety of disparate phenomena across languages, but is instead a general feature of human conversation. We test this additional hypothesis by using the model and parameters from one language and testing them on another. The results shown in Table 5.2 validate this hypothesis. The model and parameters of each language are tested across the other three. The diagonal row, shown in italics, contains the same results shown in Table 5.1, i.e. those of a system trained and tested on the same language. Looking down a column, one can compare the result in italics with the results produced by cross-language systems and see that the difference is often nominal. In fact, the improvement produced by the Pashto model on Tagalog data is better than the result produced by that langauge's own model. We also see that the ranking of languages by effectiveness is the same is it is with the in-domain results. This suggests that the languages that perform the best do so not just because the burstiness effect is especially pronounced for that language, but because we were able to train an especially effective model.

Furthermore, these results indicate that word-burst rescoring of keyword search results, while improved by tuning parameters to a specific language domain, is not dependent on that particular parameter setting. Burstiness patterns may differ in some ways from one language to the next, but in other ways they clearly hold constant. Our system is optimized by parameter tuning and yet is not highly parameter-sensitive.

6

Conclusion

In this work, we sought not only to build support for theories of word-burst patterns in language but also to utilize those theories to improve on state-of-the-art language processing technologies. We have achieved both, identifying burstiness in our data and using it to strengthen the accuracy of keyword search in data.

Previous researchers have tested the word-burst effect in English, perhaps the most heavily studied language in the world, but little has been done to explore the phenomenon in other languages. We do so in four languages that are especially poorly studied by computational linguists, poorly enough that the U.S. government made them targets for a challenge carried out on data from "low-resource languages." We find that the burstiness effect is quite strong in these languages, providing evidence that the pattern generalizes well across domains.

From an engineering perspective, our work shows how a language model might be bolstered. In a high-resource language, a bigram or trigram model might suffice to achieve accurate speech recognition and keyword search. But when less training data is available, those systems have room for improvement, and the use of a word-burst rescoring system is likely to be useful.

Our rescoring experiments yield improvement over the baseline keyword search results for every language tested. Even without returning to the full speech recognition output or the speech itself, we were able to extract a useful word-burst signal from the keyword search results alone. Our cross-language experiments were also successful in every case — in every combination of training language and target language. These results show that, if confronted with a language unfamiliar to the algorithm trained, a researcher can still use that algorithm to rescore keyword search results

in the new language without training a new model.

The cross-language findings point to an avenue for future work. To make the classification model more robust, reducing the risk of overfitting, a word-burst rescoring system could be trained on data from multiple languages. This would also prepare the system well for unfamiliar languages. Another way to advance these findings would be apply burstiness analysis to word stems, so that "spelunking" is not missed as an echo for "spelunker." Burstiness rescoring can also be used in efforts to improve speech recognition, not just keyword search, in low-resource languages. This linguistic phenomenon, scarcely employed in current speech processing tasks, has proven capable of improving the state of the art in keyword search and shows promise in aiding speech recognition as well.

7

Bibliography

[1] R. E. Madsen, D. Kauchak, and C. Elkan, "Modeling word burstiness using the dirichlet distribution," in *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, (New York, NY, USA), pp. 545–552, ACM, 2005.

[2] G. Doyle and C. Elkan, "Accounting for burstiness in topic models," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, (New York, NY, USA), pp. 281–288, ACM, 2009.

[3] E. Airoldi, S. Fienberg, and E. Xing, "Mixed membership analysis of genome-wide expression data," in *Arxiv preprint at arXiv:0711.2520*, Arxiv, 2007.

[4] L. Fei-fei, "A bayesian hierarchical model for learning natural scene categories," in *In CVPR*, pp. 524–531, 2005.

[5] K. 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*, pp. 180–186, Association for Computational Linguistics, 2000.

[6] R. Kuhn and R. De Mori, "A cache-based natural language model for speech recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, pp. 570–583, June 1990.

[7] J. Chiu and A. Rudnicky, "Using conversational word bursts in spoken term detection," in *Interspeech*, 2013.

[8] N. I. for Standards in Technology, "Draft kws14 keyword search evaluation plan," *Nist.gov*, 2014.

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

[10] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddingtion, "Results of the 2006 spoken term detection evaluation," in *Proceedings of ACM SIGIR Workshop on Searching Spontaneous Conversational*, pp. 51–55, Citeseer, 2007.

[11] D. Karakos, R. Schwartz, S. Tsakalidis, L. Zhang, S. Ranjan, T. Ng, R. Hsiao, G. Saikumar, I. Bulyko, L. Nguyen, *et al.*, "Score normalization and system combination for improved keyword spotting," in *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pp. 210–215, IEEE, 2013.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.