Reducing the Annotation Effort for Letter-to-Phoneme Conversion

Kenneth Dwyer and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2E8
{dwyer,kondrak}@cs.ualberta.ca

Abstract

Letter-to-phoneme (L2P) conversion is the process of producing a correct phoneme sequence for a word, given its letters. It is often desirable to reduce the quantity of training data — and hence human annotation — that is needed to train an L2P classifier for a new language. In this paper, we confront the challenge of building an accurate L2P classifier with a minimal amount of training data by combining several diverse techniques: context ordering, letter clustering, active learning, and phonetic L2P alignment. Experiments on six languages show up to 75% reduction in annotation effort.

1 Introduction

The task of letter-to-phoneme (L2P) conversion is to produce a correct sequence of phonemes, given the letters that comprise a word. An accurate L2P converter is an important component of a text-to-speech system. In general, a lookup table does not suffice for L2P conversion, since out-of-vocabulary words (e.g., proper names) are inevitably encountered. This motivates the need for classification techniques that can predict the phonemes for an unseen word.

Numerous studies have contributed to the development of increasingly accurate L2P systems (Black et al., 1998; Kienappel and Kneser, 2001; Bisani and Ney, 2002; Demberg et al., 2007; Jiampojamarn et al., 2008). A common assumption made in these works is that ample amounts of labelled data are available for training a classifier. Yet, in practice, this is the case for only a small number of languages. In order to train an L2P classifier for a new language, we must first annotate words in that language with their correct phoneme sequences. As annotation is expensive, we would like to minimize the amount of effort that is required to build an adequate training set. The objective of this work is not necessarily to achieve state-of-the-art performance when presented with large amounts of training data, but to outperform other approaches when training data is limited.

This paper proposes a system for training an accurate L2P classifier while requiring as few annotated words as possible. We employ decision trees as our supervised learning method because of their transparency and flexibility. We incorporate context ordering into a decision tree learner that guides its tree-growing procedure towards generating more intuitive rules. A clustering over letters serves as a back-off model in cases where individual letter counts are unreliable. An active learning technique is employed to request the phonemes (labels) for the words that are expected to be the most informative. Finally, we apply a novel L2P alignment technique based on phonetic similarity, which results in impressive gains in accuracy without relying on any training data.

Our empirical evaluation on several L2P datasets demonstrates that significant reductions in annotation effort are indeed possible in this domain. Individually, all four enhancements improve the accuracy of our decision tree learner. The combined system yields savings of up to 75% in the number of words that have to be labelled, and reductions of at least 52% are observed on all the datasets. This is achieved without any additional tuning for the various languages.

The paper is organized as follows. Section 2 explains how supervised learning for L2P conversion is carried out with decision trees, our classifier of choice. Sections 3 through 6 describe our four main contributions towards reducing the annotation effort for L2P: context ordering (Section 3), clustering letters (Section 4), active learning (Section 5), and phonetic alignment (Section 6). Our experimental setup and results are discussed in

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 127–135,
Suntec, Singapore, 2-7 August 2009. ©2009 ACL and AFNLP

Sections 7 and 8, respectively. Finally, Section 9 offers some concluding remarks.

2 Decision tree learning of L2P classifiers

In this work, we employ a decision tree model to learn the mapping from words to phoneme sequences. Decision tree learners are attractive because they are relatively fast to train, require little or no parameter tuning, and the resulting classifier can be interpreted by the user. A number of prior studies have applied decision trees to L2P data and have reported good generalization accuracy (Andersen et al., 1996; Black et al., 1998; Kienappel and Kneser, 2001). Also, the widely-used Festival Speech Synthesis System (Taylor et al., 1998) relies on decision trees for L2P conversion.

We adopt the standard approach of using the letter context as features. The decision tree predicts the phoneme for the focus letter based on the m letters that appear before and after it in the word (including the focus letter itself, and beginning/end of word markers, where applicable). The model predicts a phoneme independently for each letter in a given word. In order to keep our model simple and transparent, we do not explore the possibility of conditioning on adjacent (predicted) phonemes. Any improvement in accuracy resulting from the inclusion of phoneme features would also be realized by the baseline that we compare against, and thus would not materially influence our findings.

We employ *binary* decision trees because they substantially outperformed *n-ary* trees in our preliminary experiments. In L2P, there are many unique values for each attribute, namely, the letters of a given alphabet. In a *n*-ary tree each decision node partitions the data into *n* subsets, one per letter, that are potentially sparse. By contrast, a binary tree creates one branch for the nominated letter, and one branch grouping the remaining letters into a single subset. In the forthcoming experiments, we use binary decision trees exclusively.

3 Context ordering

In the L2P task, context letters that are adjacent to the focus letter tend to be more important than context letters that are further away. For example, the English letter c is usually pronounced as [s] if the following letter is e or i. The general tree-growing algorithm has no notion of the letter distance, but instead chooses the letters on the basis of their estimated *information gain* (Manning and Schütze, 1999). As a result, it will sometimes query a letter at position $+3$ (denoted l_3), for example, before examining the letters that are closer to the center of the context window.

We propose to modify the tree-growing procedure to encourage the selection of letters near the focus letter before those at greater offsets are examined. In its strictest form, which resembles the "dynamically expanding context" search strategy of Davel and Barnard (2004), l_i can only be queried after $l_0, \ldots, l_{i-1}$ have been queried. However, this approach seems overly rigid for L2P. In English, for example, l_2 can directly influence the pronunciation of a vowel regardless of the value of l_1 (c.f., the difference between *rid* and *ride*).

Instead, we adopt a less intrusive strategy, which we refer to as "context ordering," that biases the decision tree toward letters that are closer to the focus, but permits gaps when the information gain for a distant letter is relatively high. Specifically, the ordering constraint described above is still applied, but only to letters that have above-average information gain (where the average is calculated across all letters/attributes). This means that a letter with above-average gain that is eligible with respect to the ordering will take precedence over an ineligible letter that has an even higher gain. However, if all the eligible letters have below-average gain, the ineligible letter with the highest gain is selected irrespective of its position. Our only strict requirement is that the focus letter must always be queried first, unless its information gain is zero.

Kienappel and Kneser (2001) also worked on improving decision tree performance for L2P, and devised tie-breaking rules in the event that the tree-growing procedure ranked two or more questions as being equally informative. In our experience with L2P datasets, exact ties are rare; our context ordering mechanism will have more opportunities to guide the tree-growing process. We expect this change to improve accuracy, especially when the amount of training data is very limited. By biasing the decision tree learner toward questions that are intuitively of greater utility, we make it less prone to overfitting on small data samples.

4 Clustering letters

A decision tree trained on L2P data bases its phonetic predictions on the surrounding letter context.

Yet, when making predictions for unseen words, contexts will inevitably be encountered that did not appear in the training data. Instead of relying solely on the particular letters that surround the focus letter, we postulate that the learner could achieve better generalization if it had access to information about the *types* of letters that appear before and after. That is, instead of treating letters as abstract symbols, we would like to encode knowledge of the similarity between certain letters as features. One way of achieving this goal is to group the letters into classes or clusters based on their contextual similarity. Then, when a prediction has to be made for an unseen (or low probability) letter sequence, the letter classes can provide additional information.

Kienappel and Kneser (2001) report accuracy gains when applying letter clustering to the L2P task. However, their decision tree learner incorporates neighboring phoneme predictions, and employs a variety of different pruning strategies; the portion of the gains attributable to letter clustering are not evident. In addition to exploring the effect of letter clustering on a wider range of languages, we are particularly concerned with the impact that clustering has on decision tree performance when the training set is small. The addition of letter class features to the data may enable the active learner to better evaluate candidate words in the pool, and therefore make more informed selections.

To group the letters into classes, we employ a hierarchical clustering algorithm (Brown et al., 1992). One advantage of inducing a hierarchy is that we need not commit to a particular level of granularity; in other words, we are not required to specify the number of classes beforehand, as is the case with some other clustering algorithms.[1]

The clustering algorithm is initialized by placing each letter in its own class, and then proceeds in a bottom-up manner. At each step, the pair of classes is merged that leads to the smallest loss in the average *mutual information* (Manning and Schütze, 1999) between adjacent classes. The merging process repeats until a single class remains that contains all the letters in the alphabet. Recall that in our problem setting we have access to a (presumably) large pool of unannotated words. The unigram and bigram frequencies required by the clustering algorithm are calculated from these words; hence, the letters can be grouped into classes prior to annotation. The letter classes only need to be computed once for a given language. We implemented a brute-force version of the algorithm that examines all the possible merges at each step, and generates a hierarchy within a few hours. However, when dealing with a larger number of unique tokens (e.g., when clustering words instead of letters), additional optimizations are needed in order to make the procedure tractable.

[1]This approach is inspired by the work of Miller et al. (2004), who clustered *words* for a named-entity tagging task.

Letter	Bit String	Letter	Bit String
a	01000	n	1111
b	10000000	o	01001
c	10100	p	10001
d	11000	q	1000001
e	0101	r	111010
f	100001	s	11010
g	11001	t	101010
h	10110	u	0111
i	0110	v	100110
j	10000001	w	100111
k	10111	x	111011
l	11100	y	11011
m	10010	z	101011
		#	00

Table 1: Hierarchical clustering of English letters

The resulting hierarchy takes the form of a binary tree, where the root node/cluster contains all the letters, and each leaf contains a single letter. Hence, each letter can be represented by a *bit string* that describes the path from the root to its leaf. As an illustration, the clustering in Table 1 was automatically generated from the words in the English CMU Pronouncing Dictionary (Carnegie Mellon University, 1998). It is interesting to note that the first bit distinguishes vowels from consonants, meaning that these were the last two groups that were merged by the clustering algorithm. Note also that the beginning/end of word marker (#) is included in the hierarchy, and is the last character to be absorbed into a larger cluster. This indicates that # carries more information than most letters, as is to be expected, in light of its distinct status. We also experimented with a manually-constructed letter hierarchy, but observed no significant differences in accuracy vis-à-vis the automatic clustering.

5 Active learning

Whereas a passive supervised learning algorithm is provided with a collection of training examples that are typically drawn at random, an active learner has control over the labelled data that it obtains (Cohn et al., 1992). The latter attempts to select its training set intelligently by requesting the labels of only those examples that are judged to be the most useful or informative. Numerous studies have demonstrated that active learners can make more efficient use of unlabelled data than do passive learners (Abe and Mamitsuka, 1998; Miller et al., 2004; Culotta and McCallum, 2005). However, relatively few researchers have applied active learning techniques to the L2P domain. This is despite the fact that annotated data for training an L2P classifier is not available in most languages. We briefly review two relevant studies before proceeding to describe our active learning strategy.

Maskey et al. (2004) propose a bootstrapping technique that iteratively requests the labels of the n most frequent words in a corpus. A classifier is trained on the words that have been annotated thus far, and then predicts the phonemes for each of the n words being considered. Words for which the prediction confidence is above a certain threshold are immediately added to the lexicon, while the remaining words must be verified (and corrected, if necessary) by a human annotator. The main drawback of such an approach lies in the risk of adding erroneous entries to the lexicon when the classifier is overly confident in a prediction.

Kominek and Black (2006) devise a word selection strategy based on letter n-gram coverage and word length. Their method slightly outperforms random selection, thereby establishing passive learning as a strong baseline. However, only a single Italian dataset was used, and the results do not necessarily generalize to other languages.

In this paper, we propose to apply an active learning technique known as *Query-by-Bagging* (Abe and Mamitsuka, 1998). We consider a pool-based active learning setting, whereby the learner has access to a pool of unlabelled examples (words), and may obtain labels (phoneme sequences) at a cost. This is an iterative procedure in which the learner trains a classifier on the current set of labelled training data, then selects one or more new examples to label, according to the classifier's predictions on the pool data. Once labelled, these examples are added to the training set, the classifier is re-trained, and the process repeats until some stopping criterion is met (e.g., annotation resources are exhausted).

Query-by-Bagging (QBB) is an instance of the Query-by-Committee algorithm (Freund et al., 1997), which selects examples that have high classification variance. At each iteration, QBB employs the bagging procedure (Breiman, 1996) to create a committee of classifiers C. Given a training set T containing k examples (in our setting, k is the total number of letters that have been labelled), bagging creates each committee member by sampling k times from T (with replacement), and then training a classifier C_i on the resulting data. The example in the pool that maximizes the disagreement among the predictions of the committee members is selected.

A crucial question is how to calculate the disagreement among the predicted phoneme sequences for a word in the pool. In the L2P domain, we assume that a human annotator specifies the phonemes for an entire word, and that the active learner cannot query individual letters. We require a measure of confidence at the word level; yet, our classifiers make predictions at the letter level. This is analogous to the task of estimating record confidence using field confidence scores in information extraction (Culotta and McCallum, 2004).

Our solution is as follows. Let $\mathbf{w}$ be a word in the pool. Each classifier C_i predicts the phoneme for each letter $l \in \mathbf{w}$. These "votes" are aggregated to produce a vector $\mathbf{v}_l$ for letter l that indicates the distribution of the $|C|$ predictions over its possible phonemes. We then compute the *margin* for each letter: If $\{p, p'\} \in \mathbf{v}_l$ are the two highest vote totals, then the margin is $M(\mathbf{v}_l) = |p - p'|$. A small margin indicates disagreement among the constituent classifiers. We define the disagreement score for the entire word as the minimum margin:

$$score(\mathbf{w}) = \min_{l \in \mathbf{w}} \{M(\mathbf{v}_l)\} \qquad (1)$$

We also experimented with maximum vote entropy and average margin/entropy, where the average is taken over all the letters in a word. The minimum margin exhibited the best performance on our development data; hence, we do not provide a detailed evaluation of the other measures.

6 L2P alignment

Before supervised learning can take place, the letters in each word need to be aligned with phonemes. However, a lexicon typically provides just the letter and phoneme sequences for each word, without specifying the specific phoneme(s) that each letter elicits. The sub-task of L2P that pairs letters with phonemes in the training data is referred to as *alignment*. The L2P alignments that are specified in the training data can influence the accuracy of the resulting L2P classifier. In our setting, we are interested in mapping each letter to either a single phoneme or the "null" phoneme.

The standard approach to L2P alignment is described by Damper et al. (2005). It performs an Expectation-Maximization (EM) procedure that takes a (preferably large) collection of words as input and computes alignments for them simultaneously. However, since in our active learning setting the data is acquired incrementally, we cannot count on the initial availability of a substantial set of words accompanied by their phonemic transcriptions.

In this paper, we apply the ALINE algorithm to the task of L2P alignment (Kondrak, 2000; Inkpen et al., 2007). ALINE, which performs phonetically-informed alignment of two strings of phonemes, requires no training data, and so is ideal for our purposes. Since our task requires the alignment of phonemes with *letters*, we wish to replace every letter with a phoneme that is the most likely to be produced by that letter. On the other hand, we would like our approach to be language-independent. Our solution is to simply treat every letter as an IPA symbol (International Phonetic Association, 1999). The IPA is based on the Roman alphabet, but also includes a number of other symbols. The 26 IPA letter symbols tend to correspond to the usual phonetic value that the letter represents in the Latin script.[2] For example, the IPA symbol [m] denotes "voiced bilabial nasal," which is the phoneme represented by the letter m in most languages that utilize Latin script.

The alignments produced by ALINE are of high quality. The example below shows the alignment of the Italian word *scianchi* to its phonetic transcription [ʃaŋki]. ALINE correctly aligns not only identical IPA symbols (i:i), but also IPA symbols that represent similar sounds (s:ʃ, n:ŋ, c:k).

s	c	i	a	n	c	h	i
|			|	|	|		|
ʃ			a	ŋ	k		i

[2]ALINE can also be applied to non-Latin scripts by replacing every grapheme with the IPA symbol that is phonetically closest to it (Jiampojamarn et al., 2009).

7 Experimental setup

We performed experiments on six datasets, which were obtained from the PRONALSYL letter-to-phoneme conversion challenge.[3] They are: English CMUDict (Carnegie Mellon University, 1998); French BRULEX (Content et al., 1990), Dutch and German CELEX (Baayen et al., 1996), the Italian Festival dictionary (Cosi et al., 2000), and the Spanish lexicon. Duplicate words and words containing punctuation or numerals were removed, as were abbreviations and acronyms. The resulting datasets range in size from 31,491 to 111,897 words. The PRONALSYL datasets are already divided into 10 folds; we used the first fold as our test set, and the other folds were merged together to form the learning set. In our preliminary experiments, we randomly set aside 10 percent of this learning set to serve as our development set.

Since the focus of our work is on algorithmic enhancements, we simulate the annotator with an oracle and do not address the potential human interface factors. During an experiment, 100 words were drawn at random from the learning set; these constituted the data on which an initial classifier was trained. The rest of the words in the learning set formed the unlabelled pool for active learning; their phonemes were hidden, and a given word's phonemes were revealed if the word was selected for labelling. After training a classifier on the 100 annotated words, we performed 190 iterations of active learning. On each iteration, 10 words were selected according to Equation 1, labelled by an oracle, and added to the training set. In order to speed up the experiments, a random sample of 2000 words was drawn from the pool and presented to the active learner each time. Hence, QBB selected 10 words from the 2000 candidates. We set the QBB committee size $|C|$ to 10.

At each step, we measured word accuracy with respect to the holdout set as the percentage of test words that yielded no erroneous phoneme predictions. Henceforth, we use accuracy to refer to word accuracy. Note that although we query examples using a committee, we train a *single* tree on these examples in order to produce an intelligible model. Prior work has demonstrated that this configuration performs well in practice (Dwyer and Holte, 2007). Our results report the accuracy of the single tree grown on each iteration, averaged

[3]Available at http://pascallin.ecs.soton.ac.uk/Challenges/PRONALSYL/Datasets/

over 10 random draws of the initial training set.

For our decision tree learner, we utilized the J48 algorithm provided by Weka (Witten and Frank, 2005). We also experimented with Wagon (Taylor et al., 1998), an implementation of CART, but J48 performed better during preliminary trials. We ran J48 with default parameter settings, except that binary trees were grown (see Section 2), and subtree raising was disabled.[4]

Our feature template was established during development set experiments with the English CMU data; the data from the other five languages did not influence these choices. The letter context consisted of the focus letter and the 3 letters appearing before and after the focus (or beginning/end of word markers, where applicable). For letter class features, bit strings of length 1 through 6 were used for the focus letter and its immediate neighbors. Bit strings of length at most 3 were used at positions $+2$ and -2, and no such features were added at ± 3.[5] We experimented with other configurations, including using bit strings of up to length 6 at all positions, but they did not produce consistent improvements over the selected scheme.

8 Results

We first examine the contributions of the individual system components, and then compare our complete system to the baseline. The dashed curves in Figure 1 represent the baseline performance with no clustering, no context ordering, random sampling, and ALINE, unless otherwise noted. In all plots, the error bars show the 99% confidence interval for the mean. Because the average word length differs across languages, we report the number of *words* along the x-axis. We have verified that our system does not substantially alter the average number of letters per word in the training set for any of these languages. Hence, the number of words reported here is representative of the true annotation effort.

[4]Subtree raising is an expensive pruning operation that had a negligible impact on accuracy during preliminary experiments. Our pruning performs subtree *replacement* only.

[5]The idea of lowering the specificity of letter class questions as the context length increases is due to Kienappel and Kneser (2001), and is intended to avoid overfitting. However, their configuration differs from ours in that they use longer context lengths (4 for German and 5 for English) and ask letter class questions at every position. Essentially, the authors tuned the feature set in order to optimize performance on each problem, whereas we seek a more general representation that will perform well on a variety of languages.

8.1 Context ordering

Our context ordering strategy improved the accuracy of the decision tree learner on every language (see Figure 1a). Statistically significant improvements were realized on Dutch, French, and German. Our expectation was that context ordering would be particularly helpful during the early rounds of active learning, when there is a greater risk of overfitting on the small training sets. For some languages (notably, German and Spanish) this was indeed the case; yet, for Dutch, context ordering became more effective as the training set increased in size.

It should be noted that our context ordering strategy is sufficiently general that it can be implemented in other decision tree learners that grow binary trees, such as Wagon/CART (Taylor et al., 1998). An n-ary implementation is also feasible, although we have not tried this variation.

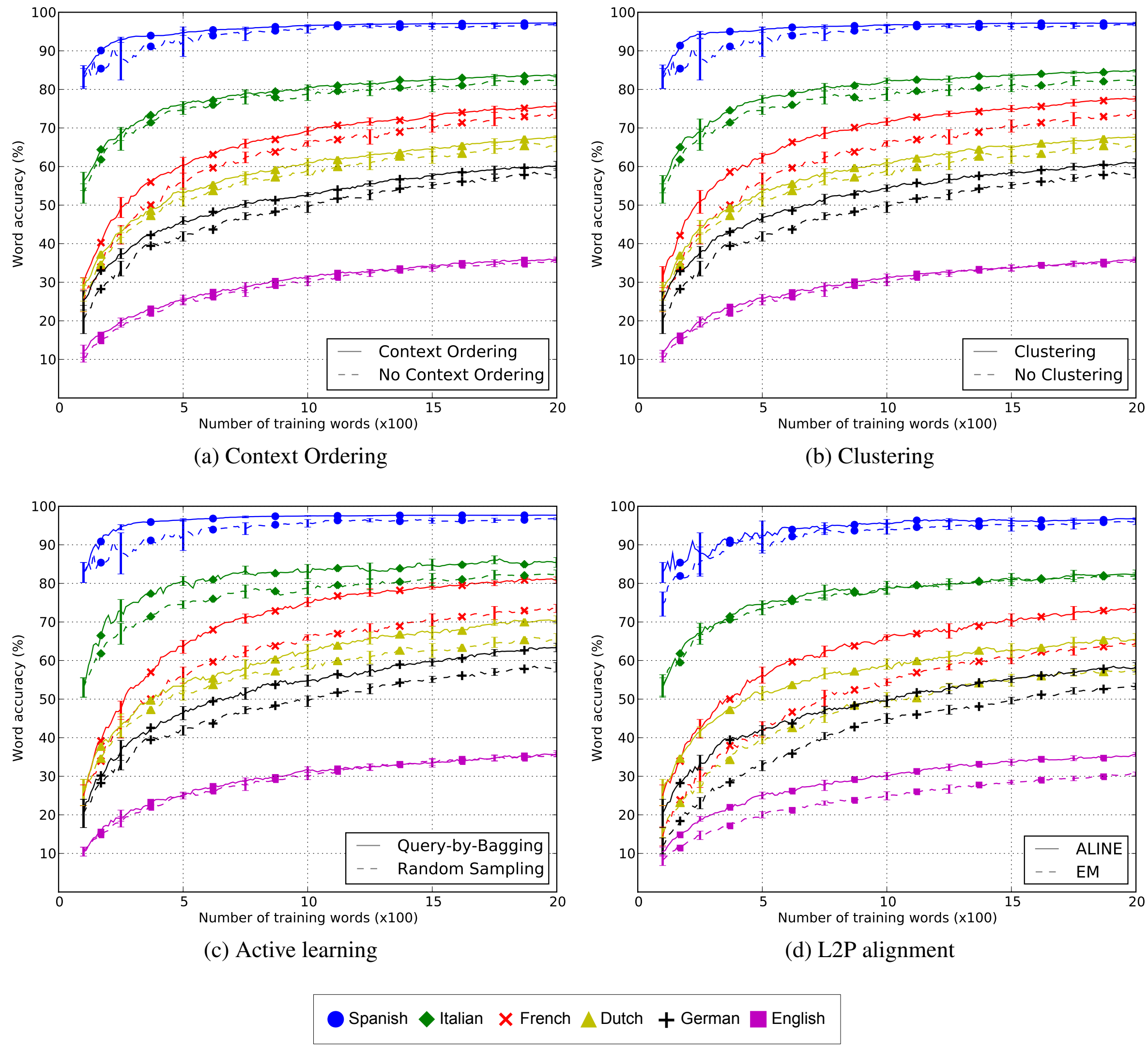
8.2 Clustering letters

As can be seen in Figure 1b, clustering letters into classes tended to produce a steady increase in accuracy. The only case where it had no statistically significant effect was on English. Another benefit of clustering is that it reduces variance. The confidence intervals are generally wider when clustering is disabled, meaning that the system's performance was less sensitive to changes in the initial training set when letter classes were used.

8.3 Active learning

On five of the six datasets, Query-by-Bagging required significantly fewer labelled examples to reach the maximum level of performance achieved by the passive learner (see Figure 1c). For instance, on the Spanish dataset, random sampling reached 97% word accuracy after 1420 words had been annotated, whereas QBB did so with only 510 words — a 64% reduction in labelling effort. Similarly, savings ranging from 30% to 63% were observed for the other languages, with the exception of English, where a statistically insignificant 4% reduction was recorded. Since English is highly irregular in comparison with the other five languages, the active learner tends to query examples that are difficult to classify, but which are unhelpful in terms of generalization.

It is important to note that empirical comparisons of different active learning techniques have shown that random sampling establishes a very



(a) Context Ordering (b) Clustering (c) Active learning (d) L2P alignment

Figure 1: Performance of the individual system components

strong baseline on some datasets (Schein and Ungar, 2007; Settles and Craven, 2008). It is rarely the case that a given active learning strategy is able to unanimously outperform random sampling across a range of datasets. From this perspective, to achieve statistically significant improvements on five of six L2P datasets (without ever being beaten by random) is an excellent result for QBB.

8.4 L2P alignment

The ALINE method for L2P alignment outperformed EM on all six datasets (see Figure 1d). As was mentioned in Section 6, the EM aligner depends on all the available training data, whereas ALINE processes words individually. Only on Spanish and Italian, languages which have highly regular spelling systems, was the EM aligner competitive with ALINE. The accuracy gains on the remaining four datasets are remarkable, considering that better alignments do not necessarily translate into improved classification.

We hypothesized that EM's inferior performance was due to the limited quantities of data that were available in the early stages of active learning. In a follow-up experiment, we allowed EM to align the entire learning set in advance, and these aligned entries were revealed when requested by the learner. We compared this with the usual procedure whereby EM is applied to the labelled training data at each iteration of learning. The learning curves (not shown) were virtually indistinguishable, and there were no statistically significant differences on any of the languages. EM appears to produce poor alignments regardless of the amount of available data.

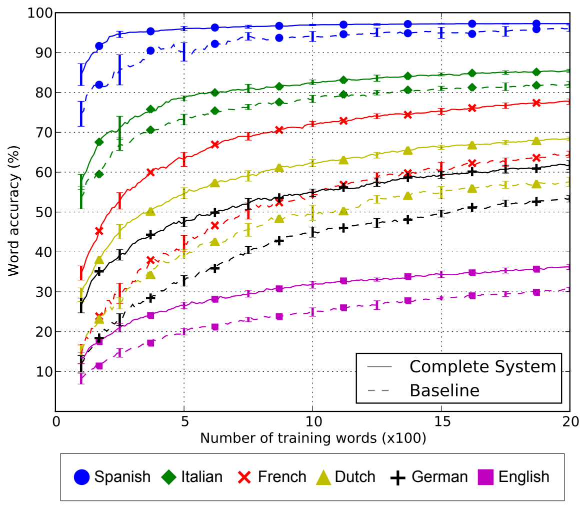


Figure 2: Performance of the complete system

8.5 Complete system

The complete system consists of context ordering, clustering, Query-by-Bagging, and ALINE; the baseline represents random sampling with EM alignment and no additional enhancements. Figure 2 plots the word accuracies for all six datasets.

Although the absolute word accuracies varied considerably across the different languages, our system significantly outperformed the baseline in every instance. On the French dataset, for example, the baseline labelled 1850 words before reaching its maximum accuracy of 64%, whereas the complete system required only 480 queries to reach 64% accuracy. This represents a reduction of 74% in the labelling effort. The savings for the other languages are: Spanish, 75%; Dutch, 68%; English, 59%; German, 59%; and Italian, 52%.[6] Interestingly, the savings are the highest on Spanish, even though the corresponding accuracy gains are the smallest. This demonstrates that our approach is also effective on languages with relatively transparent orthography.

At first glance, the performance of both systems appears to be rather poor on the English dataset. To put our results into perspective, Black et al. (1998) report 57.8% accuracy on this dataset with a similar alignment method and decision tree learner. Our baseline system achieves 57.3% accuracy when 90,000 words have been labelled. Hence, the low values in Figure 2 simply reflect the fact that many more examples are required to learn an accurate classifier for the English data.

[6]The average savings in the number of labelled words with respect to the entire learning curve are similar, ranging from 50% on Italian to 73% on Spanish.

9 Conclusions

We have presented a system for learning a letter-to-phoneme classifier that combines four distinct enhancements in order to minimize the amount of data that must be annotated. Our experiments involving datasets from several languages clearly demonstrate that unlabelled data can be used more efficiently, resulting in greater accuracy for a given training set size, without any additional tuning for the different languages. The experiments also show that a phonetically-based aligner may be preferable to the widely-used EM alignment technique, a discovery that could lead to the improvement of L2P accuracy in general.

While this work represents an important step in reducing the cost of constructing an L2P training set, we intend to explore other active learners and classification algorithms, including sequence labelling strategies (Settles and Craven, 2008). We also plan to incorporate user-centric enhancements (Davel and Barnard, 2004; Culotta and McCallum, 2005) with the aim of reducing both the effort and expertise that is required to annotate words with their phoneme sequences.

Acknowledgments

We would like to thank Sittichai Jiampojamarn for helpful discussions and for providing an implementation of the Expectation-Maximization alignment algorithm. This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Informatics Circle of Research Excellence (iCORE).

References

Naoki Abe and Hiroshi Mamitsuka. 1998. Query learning strategies using boosting and bagging. In *Proc. International Conference on Machine Learning*, pages 1–9.

Ove Andersen, Ronald Kuhn, Ariane Lazaridès, Paul Dalsgaard, Jürgen Haas, and Elmar Nöth. 1996. Comparison of two tree-structured approaches for grapheme-to-phoneme conversion. In *Proc. International Conference on Spoken Language Processing*, volume 3, pages 1700–1703.

R. Harald Baayen, Richard Piepenbrock, and Leon Gulikers, 1996. *The CELEX2 lexical database*. Linguistic Data Consortium, Univ. of Pennsylvania.

Maximilian Bisani and Hermann Ney. 2002. Investigations on joint-multigram models for grapheme-to-phoneme conversion. In *Proc. International Conference on Spoken Language Processing*, pages 105–108.

Alan W. Black, Kevin Lenzo, and Vincent Pagel. 1998. Issues in building general letter to sound rules. In *ESCA Workshop on Speech Synthesis*, pages 77–80.

Leo Breiman. 1996. Bagging predictors. *Machine Learning*, 24(2):123–140.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Carnegie Mellon University. 1998. The Carnegie Mellon pronouncing dictionary.

David A. Cohn, Les E. Atlas, and Richard E. Ladner. 1992. Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

Alain Content, Phillppe Mousty, and Monique Radeau. 1990. Brulex: Une base de données lexicales informatisée pour le français écrit et parlé. *L'année Psychologique*, 90:551–566.

Piero Cosi, Roberto Gretter, and Fabio Tesser. 2000. Festival parla Italiano. In *Proc. Giornate del Gruppo di Fonetica Sperimentale*.

Aron Culotta and Andrew McCallum. 2004. Confidence estimation for information extraction. In *Proc. HLT-NAACL*, pages 109–114.

Aron Culotta and Andrew McCallum. 2005. Reducing labeling effort for structured prediction tasks. In *Proc. National Conference on Artificial Intelligence*, pages 746–751.

Robert I. Damper, Yannick Marchand, John-David S. Marsters, and Alexander I. Bazin. 2005. Aligning text and phonemes for speech technology applications using an EM-like algorithm. *International Journal of Speech Technology*, 8(2):147–160.

Marelie Davel and Etienne Barnard. 2004. The efficient generation of pronunciation dictionaries: Human factors during bootstrapping. In *Proc. International Conference on Spoken Language Processing*, pages 2797–2800.

Vera Demberg, Helmut Schmid, and Gregor Möhler. 2007. Phonological constraints and morphological preprocessing for grapheme-to-phoneme conversion. In *Proc. ACL*, pages 96–103.

Kenneth Dwyer and Robert Holte. 2007. Decision tree instability and active learning. In *Proc. European Conference on Machine Learning*, pages 128–139.

Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168.

Diana Inkpen, Raphaëlle Martin, and Alain Desrochers. 2007. Graphon: un outil pour la transcription phonétique des mots français. Unpublished manuscript.

International Phonetic Association. 1999. *Handbook of the International Phonetic Association: A Guide to the Use of the International Phonetic Alphabet*. Cambridge University Press.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2008. Joint processing and discriminative training for letter-to-phoneme conversion. In *Proc. ACL*, pages 905–913.

Sittichai Jiampojamarn, Aditya Bhargava, Qing Dou, Kenneth Dwyer, and Grzegorz Kondrak. 2009. DirecTL: a language-independent approach to transliteration. In *Named Entities Workshop (NEWS): Shared Task on Transliteration*. Submitted.

Anne K. Kienappel and Reinhard Kneser. 2001. Designing very compact decision trees for grapheme-to-phoneme transcription. In *Proc. European Conference on Speech Communication and Technology*, pages 1911–1914.

John Kominek and Alan W. Black. 2006. Learning pronunciation dictionaries: Language complexity and word selection strategies. In *Proc. HLT-NAACL*, pages 232–239.

Grzegorz Kondrak. 2000. A new algorithm for the alignment of phonetic sequences. In *Proc. NAACL*, pages 288–295.

Christopher D. Manning and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press.

Sameer R. Maskey, Alan W. Black, and Laura M. Tomokiya. 2004. Boostrapping phonetic lexicons for new languages. In *Proc. International Conference on Spoken Language Processing*, pages 69–72.

Scott Miller, Jethran Guinness, and Alex Zamanian. 2004. Name tagging with word clusters and discriminative training. In *Proc. HLT-NAACL*, pages 337–342.

Andrew I. Schein and Lyle H. Ungar. 2007. Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proc. Conference on Empirical Methods in Natural Language Processing*, pages 1069–1078.

Paul A. Taylor, Alan Black, and Richard Caley. 1998. The architecture of the Festival Speech Synthesis System. In *ESCA Workshop on Speech Synthesis*, pages 147–151.

Ian H. Witten and Eibe Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2nd edition.