First experiments with developing an unsupervised method for learning morphology of variants

Mans Hulden
University of Helsinki, Language Technology
mans.hulden@helsinki.fi
Iñaki Alegria, Izaskun Etxeberria, Montse Maritxalar
IXA taldea, UPV-EHU
{i.alegria izaskun.etxeberria montse.maritxalar}@ehu.es

Abstract. A long-standing open question in computational morphology is how to combine linguistic and machine-learning approaches. In our work with the Basque language we try to infer a morphological description of variant using the standards description and small standard/variant parallel corpus. The key task is the inference of phonological rules. Although the results obtained in the experiments are encouraging, it seems necessary to improve upon them if we want to use the application for real tools.

1 Introduction

Computational morphology has traditionally been carried out in two ways:

- The linguist approach: experts model a lexicon, paradigms and phonological alternations producing a morphological analyzer/generator. Technology based on finite-state machines (Beesley and Karttunen, 2002) have been the most successful practical implementation of this approach.
- The machine learning approach: from a corpus of the language, sometimes using additional information about paradigms, a program learns a segmentation of word-forms in morphemes. Goldsmith (2001), for example, has proposed a popular method based on this idea.

While the first approach often produces better results and is considered the standard method of building morphological tools, the second approach is sometimes used when development time is at a premium or when experts and linguists are not available for such in-depth work.

An long-standing open question is how to combine both approaches. In our work with the Basque language, a morphological description is available for the standard language, but we want to learn to analyze variants and dialectal forms as well. The hope is that this second part—dealing with dialectal variant forms—could be automatically learned from a corpus given that we have tools to handle the standard language. This is an interesting problem because a good solution to this problem could be applied to many other tasks as well: to improve access to digital libraries (containing diachronic and dialectal variants) or to improve treatment of informal registers such as SMS messages and blogs, etc.

In this paper we assume that a small standard/variant parallel corpus is available (if not, it is possible prepare one) and we propose a method based on finite-state phonology to learn from the information of the corpus and translate a given word of the dialect to its standard-form equivalent. The variant we use for experiments is Lapurdian, a dialect of Basque spoken in the Lapurdi (fr. Labourd) region in the Basque Country.

Because Basque is an agglutinative, highly inflected language, we believe some of the results can be extrapolated to many other languages as well.

One of the motivations for the current work is that there are a large number of NLP tools available and in development for standard Basque (also called Batua): a morphological analyzer, a POS tagger, a dependency analyzer, an MT engine, among others. However, these tools do not work well with the different dialects of Basque and there is a desire to explore the possibility of reusing all or some of these for handling dialect-form input as well.

Here is a brief contrastive example of the kinds of differences found in the dialect (a, Lapurdian) and standard Basque (b) parallel corpus:

(a) Ez gero uste izan **nexkatxa guziek** tu egiten **dautatela**

(b) Ez gero uste izan **neskatxa guztiek** tu egiten **didatela**

As is clear, the differences are minor overall, but even such small discrepancies cause great problems in the potential reuse of current tools designed for the standard forms only.

We have experimented with an approach that attempts to improve on a simple baseline of learning word-pairs in the dialect and the standard. In our approach we have used the *lexdiff* command proposed by Almeida et al. (2010) in their related work on contrasting Brazilian Portuguese and the Portuguese in Portugal. We use *lexdiff* to extract information about the changes between the dialect and the standard, and then induce rules to apply these changes, given input forms in the dialect.

The remainder of the paper is organized as follows. The characteristics of the corpus available to us are described in section 2. In section 3 we describe the steps and variations of the methods we have applied. Section 4 and 5 present the parameters used in the evaluation and the experimental results. Finally, we discuss the results and present possibilities for potential future work in section 6.

2 The corpus

The corpus used in this research have been created as part of "TSABL" project (*Towards a Syntactic Atlas of the Basque Language*, web site: *http://www.iker.cnrs.fr/-tsabl-towards-a-syntactic-atlas-of-.html?lang=fr)*. Two groups are collaborating in this project, the IXA group at the University of the Basque Country and the IKER group in Baiona (fr. Bayonne). The main objective of the IKER group is to analyze and process dialects of Basque language and they have developed an application to analyze syntactic changes between dialects and the standard. This application works with examples they have collected. The researchers of the IKER project have provided us with examples of the Lapurdian dialect and their corresponding forms in standard Basque. Our parallel corpus then consists of running text in two variants: complete sentences of the Lapurdian dialect and equivalent sentences corresponding to standard Basque.

The characteristics of the corpus are presented in Table 1. Our corpus contains 2,117 sentences and 12,150 words for each variant (3,600 different words more or less). So, it can be considered like a parallel corpus of Lapurdian dialect and standard Basque.

In order to provide data for our learning algorithms and also to test their performance, we have divided the corpus into two parts: 80% of the corpus is used for the learning task (1,694 sentences) and the remaining 20% (409 sentences) for evaluation of the learning process.

	Full corpus	**80% part.**	**20% part.**
Sentences	2,117	1,694	423
Words	12,150	9,734	2,417
Unique words	3,610	3,108	1,243
Pairs of diff.	2,169	1,732	437
Unique pairs	1,078	908	301

Table 1. Characteristics of the parallel corpus used for experiments. Last two rows present the number of words that have different spellings in the dialect and in the standard.

3 Methods

We have used different methods to produce an application that will give us the equivalent standard word corresponding to an input word in the dialect.

To extract information from the corpus we use the *lexdiff* command, developed by Almeida et al. (2010). This program comes as part of a toolkit to aid in the automatic adjustment of orthography written in different Portuguese orthographies (Brazilian/European). Despite its focus on Portuguese, the toolkit is designed to be somewhat language-independent and relies produces a set of orthographic adjustment rules that are learned by comparing different corpora. The rule induction component (*lexdiff*) is what we have used in our research.

The overall process consists of three steps:

1. Apply *lexdiff* to the parallel corpus to obtain information about correspondences between the variant and the standard. With this pre-processing we can obtain a list of equivalent words or a list of equivalent n-grams with their frequency in the corpus.
2. Use previous information to “learn” phonological rules. There are many options to learn and we have to experiment with them to see how they modify the results.
3. Constrain the output of the rules learned by *lexdiff* to words in the Basque standard morphology.

3.1 The baseline

The baseline of our experiments is a simple method, based on a dictionary of equivalent words with the list of correspondences between words extracted from the 80% of the corpus with *lexdiff* command. This list of correspondences contains all aligned words in the variant vs. standard corpus, be they identical or not. For example, the output `112 eman = eman` indicates that the correspondence is between the same form and appears 112 times in the corpus while the entry `61 emaiten => ematen` indicates that the correspondence is between different forms and appears 61 times in the corpus.

In other words, the baseline approach is simply to memorize all the word pairs seen between the dialectal and standard forms, and subsequently use this knowledge in later conversion tasks.

3.2 Method 1

The second approximation is to infer phonological rules from the equivalences between words obtained with *lexdiff* and compile these rules into finite-state transducers (we use the freely available *foma* toolkit for this (Hulden, 2009)).

The *lexdiff* program tries to identify sequences of changes from seen word pairs and outputs string correspondences such as, for example: `76 ait => at ; 39 dautz => diz`, indicating that `ait` has changed into `at` 76 times in the corpus, etc.

With such information about word pairs we generate a variety of so-called replacement rules which can subsequently be compiled into finite transducers with the *foma* application. Even though the *lexdiff* program provides a direct string-to-string change as a rule, there are several ways to encapsulate its output as replacement rules and finite transducers, yielding variant approaches such as the following:

- We can restrict the rules by frequency and require that a certain type of change be seen at least n times in order to apply that rule. For example, if we set this threshold to 3, we will only apply a string-to-string changing rule that has been seen three times or more.
- We limit the number of rules that can be applied to the same word. Sometimes the *lexdiff* application divides the change between a pair of words into two separate rules. For example the word-word correspondence `agerkuntza => agerpena` is expressed by two rules: `rkun => rpen` and `ntza => na`. Now, given these two rules, we have to be able to apply both to produce the correct total change `agerkuntza => agerpena`. By limiting the number of rules that can apply to a single input word we can avoid creating many spurious outputs, but also at the same time we may sacrifice some ability to produce the desired output forms.
- We can also control the application mode of the rules: whether they be sequential or parallel. The rules in *foma* can be applied in parallel or sequentially and the results are different depending on the mode of application. The previous example can serve to illustrate the difference between two modes. If the previous two rules are applied in parallel, the form obtained from `agerkuntza` will not be correct since the *n* overlaps with the two rules. That is, when applying rules simultaneously

in parallel, the input characters for two rules may not overlap. However, if these two rules applied in sequence (the order in this example is irrelevant), the output will be the correct: we first change `rkun => rpen` and later `ntza => na`. We have not a priori chosen to use parallel or sequential rules and have decided to evaluate both approaches.

– We can also compact the rules output by *lexdiff* by eliminating redundancies and constructing context-sensitive rules. For example: given a rule such as `rkun => rpen`, we can convert this into a context-sensitive rule that only changes `ku` into `pe` when flanked by `r` and `n` to the left and right, respectively. This has a bearing on the previous point and will allow more rewritings within a single word in parallel replacement mode since there are less characters overlapping.

4 Evaluation

We have measured the quality of different approaches by the usual parameters of precision, recall and the harmonic combination of them, the F1-score. We have analyzed how the different options in the approaches affect the results of these three parameters. Given that we extract quite a large number of rules and that each input word generates a very large number of candidates if we use all the rules extracted, it is possible to produce a high recall on the conversion of unknown dialect words to the standard form. However, the downside is that this naturally leads to low precision as well, which we try to control by introducing a number of filters to remove some of the candidates output by the rules. More specifically we use two filters: (1) an obligatory filter which removes all candidate words that are not found in the standard Basque (by using an existing standard Basque morphological analyzer), and (2) using an optional filter which, given several candidates in the standard Basque, picks the most frequently occurring one.

5 Results

As mentioned, the learning process has been done using the 80% of the corpus, leaving 20% of the corpus for evaluation of the abovementioned approaches. In the evaluation, we have only tested those words in the dialect that *differ* from words in the standard (which are in the minority). In total, in the evaluation part, we have tested 301 words.

The results for the baseline—i.e. simple memorization of word-word correspondences—are (in %): P = 95.62, R = 43.52 and F1 = 59.82. As expected, the precision of the baseline is high: when the method gives an answer it is usually the correct one. But the recall of the baseline is naturally low: slightly less than half of the words in the evaluation corpus have been encountered before.

In the following, we give the results of a number of experiments using method 1.

5.1 Results depending on a frequency threshold

Varying the frequency threshold (see 3.2), we have tested with values of 1, 2, and 3. The values are in table 2. The results clearly show that the more information we extract

(frequency 1), the better results we obtain for recall while at the same time the precision suffers. The F-score doesn't vary very much and it maintains similar values throughout.

If we compare the results presented in Table 3 with the results of the baseline, it is obvious that the baseline is 9-10 points better with respect to the F-score: the precision of the baseline is very high compared to the precision of our first approach; on the other hand, the recall of the baseline is worse, but not significantly. The problem with this approach is one which we have mentioned before: the rules produce more than one answer for any given word and the consequence is that the precision suffers, even though only those output words are retained that correspond to actual standard Basque. With the frequency filter in place, the results improve somewhat[1]. The filtered results are given in table 3: with this addition, we can improve on the baseline, but not significantly.

	P	**R**	**F**
Baseline	95.62	43.52	59.82
Freq. 1	38.95	66.78	49.20
Freq. 2	46.99	57.14	51.57
Freq. 3	49.39	53.82	51.51

Table 2. Values obtained for Precision, Recall and F-scores by changing the minimum frequency of the correspondences to construct rules for *foma*. The rest of the options are the same in all three experiments: only one rule is applied in a word, and without context.

	P	**R**	**F**
Baseline	95.62	43.52	59.82
Freq. 1	70.28	58.13	63.64
Freq. 2	70.18	53.16	60.49
Freq. 3	71.76	51.50	59.96

Table 3. Values obtained for Precision, Recall and F-score by changing the threshold frequency of correspondences and applying a post-filter.

5.2 Results depending on other options

We have also varied the maximum number of possible rule applications within a single word by limiting it to 1 and 2 as well as applying the rules in parallel or sequentially, and

[1] These frequencies were obtained from a corpus of a Basque newspaper

compacting the rules to provide more context-sensitivity. We shall here limit ourselves to present the best results of all these options in terms of the F-score in table 4.

In general, we may note that applying more than one rule has a negative effect on the precision and does not help the recall very much either. Applying the post-filter—choosing the most frequent candidate—yields a limited improvement: mildly better precision but also slightly worse recall, and the F-score does not improve significantly. The parallel or sequential application of the rules (when they are more than one) doesn't change the results very much and if we analyze the F-score, it seems better to apply the rules in parallel. Finally, compacting the rules and producing context-sensitive ones is clearly the best option.

In all cases the F-score improves if the frequency filter is applied; sometimes significantly and sometimes only slightly. All the results of the table 4 which lists the best performing ones come from experiments where the frequency filter was applied.

	P	**R**	**F**
Baseline	95.62	43.52	59.82
EXP. 1	72.20	57.81	64.21
EXP. 2	72.13	58.47	64.59
EXP. 3	75.10	60.13	66.79

Table 4. EXP. 1: frequency 2; 2 rules applied; in parallel; without context. **EXP. 2**: frequency 1; 1 rule applied; with context. **EXP. 3**: frequency 2; 2 rules applied; in parallel; with context.

6 Conclusions and future work

We have presented a number of experiments to solve a very concrete task: given a word in the Lapurdian dialect of Basque, produce the equivalent standard Basque word. The approach has been based on the idea of extracting string-to-string changing rules from a parallel corpus of the two dialects, and to apply these rules to unseen words. We have been able to improve on the results of a naive baseline using a method which infers phonological rules of the information extracted from the corpus and applies them using finite state technology.

Although the results obtained in the experiments are encouraging, it seems necessary to improve upon them if we want to use the application for real tools. In particular, the overgeneration of the learned rules remains a problem and leads to low precision. We are working on other algorithms for string-to-string pattern induction based *Inductive Logic Learning*-type algorithms with the goal to limit this overgeneralization and still produce high recall. During the current work, however, we have accumulated a small but valuable training and test corpus which may serve as a future workbench for evaluation of phonological rule induction algorithms.

Bibliography

Almeida, J. J., Santos, A., and Simoes, A. (2010). Bigorna–a toolkit for orthography migration challenges. In *Seventh International Conference on Language Resources and Evaluation (LREC2010), Valletta, Malta*.

Beesley, K. R. and Karttunen, L. (2002). Finite-state morphology: Xerox tools and techniques. *Studies in Natural Language Processing. Cambridge University Press*.

Goldsmith, J. (2001). Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.

Hulden, M. (2009). Foma: a finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, pages 29–32, Athens, Greece. Association for Computational Linguistics.