Learning word-level dialectal variation as phonological replacement rules using a limited parallel corpus

Mans Hulden
University of Helsinki
Language Technology
mans.hulden@helsinki.fi

Iñaki Alegria
IXA taldea
UPV-EHU
i.alegria@ehu.es

Izaskun Etxeberria
IXA taldea
UPV-EHU
izaskun.etxeberria@ehu.es

Montse Maritxalar
IXA taldea
UPV-EHU
montse.maritxalar@ehu.es

Abstract

This paper explores two different methods of learning dialectal morphology from a small parallel corpus of standard and dialect-form text, given that a computational description of the standard morphology is available. The goal is to produce a model that translates individual lexical dialectal items to their standard dialect counterparts in order to facilitate dialectal use of available NLP tools that only assume standard-form input. The results show that a learning method based on inductive logic programming quickly converges to the correct model with respect to many phonological and morphological differences that are regular in nature.

1 Introduction

In our work with the Basque language, a morphological description and analyzer is available for the standard language, along with other tools for processing the language (Alegria et al., 2002). However, it would be convenient to be able to analyze variants and dialectal forms as well. As the dialectal differences within the Basque language are largely lexical and morphophonological, analyzing the dialectal forms would in effect require a separate morphological analyzer that is able to handle the unique lexical items in the dialect together with the differing affixes and phonological changes.

Morphological analyzers are traditionally handwritten by linguists, most commonly using some variant of the popular finite-state morphology approach (Beesley and Karttunen, 2002). This entails having an expert model a lexicon, inflectional and derivational paradigms as well as phonological alternations, and then producing a morphological analyzer/generator in the form of a finite-state transducer.

As the development of such wide-coverage morphological analyzers is labor-intesive, the hope is that an analyzer for a variant could be automatically learned from a limited parallel standard/dialect corpus, given that an analyzer already exists for the standard language. This is an interesting problem because a good solution to it could be applied to many other tasks as well: to enhancing access to digital libraries (containing diachronic and dialectal variants), for example, or to improving treatment of informal registers such as SMS messages and blogs, etc.

In this paper we evaluate two methods of learning a model from a standard/variant parallel corpus that translates a given word of the dialect to its standard-form equivalent. Both methods are based on finite-state phonology. The variant we use for experiments is Lapurdian,[1] 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 facing similar challenges.

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

[1] Sometimes also called Navarro-Labourdin or Labourdin.

Proceedings of EMNLP 2011, Conference on Empirical Methods in Natural Language Processing, pages 39–48,
Edinburgh, Scotland, UK, July 27–31, 2011.

others (Alegria et al., 2011). However, these tools do not work well in processing the different dialects of Basque where lexical items have a different orthographic representation owing to slight differences in phonology and morphology.

Here is a brief contrastive example of the kinds of differences found in the (a) Lapurdian dialect and standard Basque (b) parallel corpus:[2]

(a) Ez gero uste izan **nexkatxa guziek** tu egiten **dautatela**
(b) Ez gero uste izan **neskatxa guztiek** tu egiten **didatela**

As the example illustrates, the differences are minor overall—the word order and syntax are unaffected, and only a few lexical items differ. This reflects the makeup of our parallel corpus quite well—in it, slightly less than 20% of the word tokens are distinct. However, even such relatively small discrepancies cause great problems in the potential reuse of current tools designed for the standard forms only.

We have experimented with two approaches that attempt to improve on a simple baseline of memorizing word-pairs in the dialect and the standard. The first approach is based on work by Almeida et al. (2010) on contrasting orthography in Brazilian Portuguese and European Portuguese. In this approach differences between substrings in distinct word-pairs are memorized and these transformation patterns are then applied whenever novel words are encountered in the evaluation. To prevent overgeneration, the output of this learning process is later subject to a morphological filter where only actual standard-form outputs are retained. The second approach is an Inductive Logic Programming-style (ILP) (Muggleton and De Raedt, 1994) learning algorithm where phonological transformation rules are learned from word-pairs. The goal is to find a minimal set of transformation rules that is both necessary and sufficient to be compatible with the learning data, i.e. the word pairs seen in the training data.

The remainder of the paper is organized as follows. The characteristics of the corpus available to us are described in section 2. In sections 3, 4, and 5, we describe the steps and variations of the methods we have applied and how they are evaluated. Section 6 presents the experimental results, and finally, we discuss the results and present possibilities for potential future work in section 7.

[2]English translation of the example: *Don't think all girls spit on me*

1.1 Related work

The general problem of supervised learning of dialectal variants or morphological paradigms has been discussed in the literature with various connection to computational phonology, morphology, machine learning, and corpus-based work. For example, Kestemont et al. (2010) presents a language-independent system that can 'learn' intra-lemma spelling variation. The system is used to produce a consistent lemmatization of texts in Middle Dutch literature in a medieval corpus, Corpus-Gysseling, which contains manuscripts dated before 1300 AD. These texts have enormous spelling variation which makes a computational analysis difficult.

Koskenniemi (1991) provides a sketch of a discovery procedure for phonological two-level rules. The idea is to start from a limited number of paradigms (essentially pairs of input-output forms where the input is the surface form of a word and the output a lemmatization plus analysis). The problem of finding phonological rules to model morphological paradigms is essentially similar to the problem presented in this paper. An earlier paper, Johnson (1984), presents a 'discovery procedure' for learning phonological rules from data, something that can be seen as a precursor to the problem dealt with by our ILP algorithm.

Mann and Yarowsky (2001) present a method for inducing translation lexicons based on transduction models of cognate pairs via bridge languages. Bilingual lexicons within languages families are induced using probabilistic string edit distance models. Inspired by that paper, Scherrer (2007) uses a generate-and-filter approach quite similar to our first method. He compares different measures of graphemic similarity applied to the task of bilingual lexicon induction between Swiss German and Standard German. Stochastic transducers are trained with the EM algorithm and using handmade transduction rules. An improvement of 11% in F-score is reported over a baseline method using Levenshtein Distance.

	Full corpus	80% part.	20% part.
Sentences	2,117	1,694	423
Words	12,150	9,734	2,417
Unique words			
Standard Basque	3,553	3,080	1,192
Lapurdian	3,830	3,292	1,239
Filtered pairs	3,610	3,108	1,172
Identical pairs	2,532	2,200	871
Distinct pairs	1,078	908	301

Table 1: Characteristics of the parallel corpus used for experiments.

2 The corpus

The parallel corpus used in this research is part of "TSABL" project developed by the IKER group in Baiona (fr. Bayonne).[3] 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 contains running text in two variants: complete sentences of the Lapurdian dialect and equivalent sentences in standard Basque.

The details of the corpus are presented in table 1. The corpus consists of 2,117 parallel sentences, totaling 12,150 words (roughly 3,600 types). 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% (423 sentences) for evaluation of the learning process. As is seen, roughly 23% of the word-pairs are distinct. Another measure of the average deviation between the word pairs in the corpus is given by aligning all word-pairs by minimum edit distance (MED): aligning the 3,108 word-pairs in the learning corpus can be done at a total MED cost of 1,571. That is, roughly every 14th character in the dialect data is different from the standard form.

3 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 learning portion (80%) of the corpus. This list of correspondences contains all different word pairs in the variant vs. standard corpus. The baseline approach consists simply of memorizing all the distinct word pairs seen between the dialectal and standard forms, and subsequently applying this knowledge during the evaluation task. That is, if an input word during the evaluation has been seen in the training data, we provide the corresponding previously known output word as the answer. Otherwise, we assume that the output word is identical to the input word.

[3] *Towards a Syntactic Atlas of the Basque Language*, web site: *http://www.iker.cnrs.fr/-tsabl-towards-a-syntactic-atlas-of-.html*

4 Overview of methods

We have employed two different methods to produce an application that attempts to extract generalizations from the training corpus to ultimately be able to produce the equivalent standard word corresponding to a given dialectal input word. The first method is based on already existing work by Almeida et al. (2010) that extracts all substrings from lexical pairs that are different. From this knowledge we then produce a number of phonological replacement rules that model the differences between the input and output words. In the second method, we likewise produce a set of phonological replacement rules, using an ILP approach that directly induces the rules from the pairs of words in the training corpus.

The core difference between the two methods is that while both extract replacement patterns from the word-pairs, the first method does not consider negative evidence in formulating the replacement rules. Instead, the existing morphological analyzer is used as a filter after applying the rules to unknown text. The second method, however, uses negative evidence from the word-pairs in delineating the replacement rules as is standard in ILP-approaches, and the subsequent morphological filter for the output plays much less of a role. Evaluating and comparing both approaches is motivated because the first method may produce much higher recall by virtue of generating a large number of input-output candidates during application, and the question is whether the corresponding loss in precision may be mitigated by judicious application of post-processing filters.

4.1 Format of rules

Both of the methods we have evaluated involve learning a set of string-transformation rules to convert words, morphemes, or individual letters (graphemes) in the dialectal forms to the standard variant. The rules that are learned are in the format of so-called phonological replacement rules (Beesley and Karttunen, 2002) which we have later converted into equivalent finite-state transducers using the freely available *foma* toolkit (Hulden, 2009a). The reason for the ultimate conversion of the rule set to finite-state transducers is twofold: first, the transducers are easy to apply rapidly to input data using available tools, and secondly, the transducers can further be modified and combined with the standard morphology already available to us as a finite transducer.

In its simplest form, a replacement rule is of the format

$$A \rightarrow B \,||\, C \_ D \tag{1}$$

where the arguments A,B,C,D are all single symbols or strings. Such a rule dictates the transformation of a string A to B, whenever the A occurs between the strings C and D. Both C and D are optional arguments in such a rule, and there may be multiple conditioning environments for the same rule.

For example, the rule:

```
h -> 0 || p _ , t _ , l _ , _ a s o
```
(2)

would dictate a deletion of h in a number of contexts; when the h is preceded by a p, t, or l, or succeeded by the sequence *aso*, for instance transforming *ongiethorri* (Lapurdian) to *ongietorri* (Batua).

As we will be learning several rules that each target different input strings, we have a choice as to the mode of application of the rules in the evaluation phase. The learned rules could either be applied in some specific order (sequentially), or applied simultaneously without regard to order (in parallel).

For example, the rules:

```
u -> i || z a _
```
(3)

```
k -> g || z a u _
```
(4)

would together (in parallel) change *zaukun* into *zaigun*. Note that if we imposed some sort of ordering on the rules and the `u` → `i` rule in the set would apply first, for example, the conditioning environment for the second rule would no longer be met after transforming the word into *zaikun*. We have experimented with sequential as well as parallel processing, and the results are discussed below.

4.2 Method 1 (lexdiff) details

The first method is based on the idea of identifying sequences inside word pairs where the output differs from the input. This was done through the already available tool *lexdiff* which has been used in automatic migration of texts between different Portuguese orthographies (Almeida et al., 2010). 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` (stemming from pairs such as (*joaiten/joaten* and *dautzut/dizut*), indicating that `ait` has changed into `at` 76 times in the corpus, etc., thus directly providing suggestions as to phonologically regular changes between two texts, with frequency information included.

With such information about word pairs we generate a variety of replacement rules which are then compiled into finite transducers with the *foma* application. Even though the *lexdiff* program provides a direct string-to-string change in a format that is directly compilable into a phonological rule transducer, we have experimented with some possible variations of the specific type of phonological rule we want to output:

- 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 or more times.

- 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

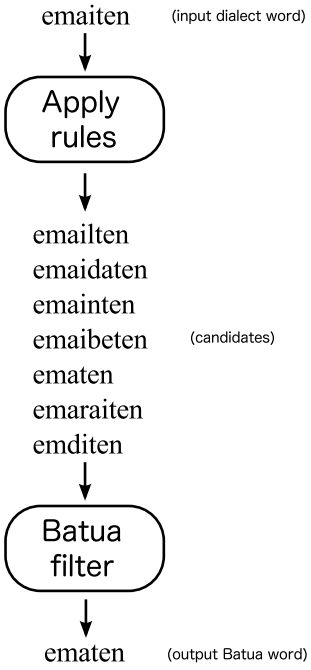

Figure 1: The role of the standard Basque (Batua) analyzer in filtering out unwanted output candidates created by the induced rule set produced by method 1.

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: sequential or parallel. 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, i.e. producing a rule:

$$\texttt{k u -> p e} \;||\; \texttt{r \_ n} \qquad (5)$$

This has a bearing on the previous point and will allow more rewritings within a single word in parallel replacement mode since there are fewer characters overlapping.

Once a set of rules is compiled with some instantiation of the various parameters discussed above and converted to a transducer, we modify the transducer in various ways to improve on the output.

First, since we already have access to a large-scale morphological transducer that models the standard Basque (Batua), we restrict the output from the conversion transducer to only allow those words as output that are legitimate words in standard Basque. Figure 1 illustrates this idea. In that figure, we see an input word in the dialect (*emaiten*) produce a number of candidates using the rules induced. However, after adding a morphological filter that models the Batua, we retain only one output.

Secondly, in the case that even after applying the Batua filter we retain multiple outputs, we simply choose the most frequent word (these unigram counts are gathered from a separate newspaper corpus of standard Basque).

4.3 Method 2 (ILP) details

The second method we have employed works directly from a collection of word-pairs (dialect/standard in this case). We have developed an algorithm that from a collection of such pairs seeks a minimal hypothesis in the form of a set of replacement rules that is consistent with all the changes found in the training data. This approach is generally in line with ILP-based machine learning methods (Muggleton and De Raedt, 1994). However, in contrast to the standard ILP, we do not learn statements of first-order logic that fit a collection of data, but rather, string-to-string replacement rules.[4]

[4]Phonological string-to-string replacement rules can be defined as collections of statements in first-order logic and compiled into transducers through such logical statements as well;

The two parameters to be induced are (1) the collection of string replacements $X \rightarrow Y$ needed to characterize the training data, and (2) the minimal conditioning environments for each rule, such that the collection of rules model the string transformations found in the training data.

The procedure employed for the learning task is as follows:

(1) Align all word pairs (using minimum edit distance by default).

(2) Extract a collection of phonological rewrite rules.

(3) For each rule, find counterexamples.

(4) For each rule, find the shortest conditioning environment such that the rule applies to all positive examples, and none of the negative examples. Restrict rule to be triggered only in this environment.

The following simple example should illustrate the method. Assuming we have a corpus of only two word pairs:

emaiten	ematen
igorri	igorri

in step (1) we would perform the alignment and produce the output

e	m	a	i	t	e	n		i	g	o	r	r	i
e	m	a	∅	t	e	n		i	g	o	r	r	i

From this data we would in step (2) gather that the only active phonological rule is $i \rightarrow \emptyset$, since all other symbols are unchanged in the data. However, we find two counterexamples to this rule (step 3), namely two i-symbols in *igorri* which do not alternate with $\emptyset$. The shortest conditioning environment that accurately models the data and produces no overgeneration (does not apply to any of the is in *igorri*) is therefore:

```
i -> ∅ || a _        (6)
```

see e.g. Hulden (2009b) for details. In other words, in this work, we skip the intermediate step of defining our observations as logical statements and directly convert our observations into phonological replacement rules.

the length of the conditioning environment being 1 (1 symbol needs to be seen to the left plus zero symbols to the right). Naturally, in this example we have two competing alternatives to the shortest generalization: we could also have chosen to condition the i-deletion rule by the t that follows the i. Both conditioning environments are exactly one symbol long. To resolve such cases, we a priori choose to favor conditioning environments that extend farther to the left. This is an arbitrary decision—albeit one that does have some support from phonology as most phonological assimilation rules are conditioned by previously heard segments—and very similar results are obtained regardless of left/right bias in the learning. Also, all the rules learned with this method are applied simultaneously (in parallel) in the evaluation phase.

4.3.1 String-to-string vs. single-symbol rules

In some cases several consecutive input symbols fail to correspond to the output in the learning data, as in for example the pairing

d	a	u	t
d	i	∅	t

corresponding to the dialect-standard pair *daut/dit*. Since there is no requirement in our formalism of rewrite rules that they be restricted to single-symbol rewrites only, there are two ways to handle this: either one can create a string-to-string rewriting rule:

au $\rightarrow i$ / CONTEXT

or create two separate rules

a $\rightarrow i$ / CONTEXT , u $\rightarrow \emptyset$ / CONTEXT

where CONTEXT refers to the minimal conditioning environment determined by the rest of the data. We have evaluated both choices, and there is no notable difference between them in the final results.

5 Evaluation

We have measured the quality of different approaches by the usual parameters of precision, recall and the harmonic combination of them, the F_1-score, and analyzed how the different options in the two approaches affect the results of these three parameters. Given that we, especially in method 1, 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. As mentioned above, 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 by a unigram count from the separate newspaper corpus. This latter filter turns out to serve a much more prominent role in improving the results of method 1, while it is almost completely negligible for method 2.

6 Results

As mentioned above, the learning process has made use of 80% of the corpus, leaving 20% of the corpus for evaluation of the above-mentioned 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 the 301 words that differ between the dialect and the standard in the evaluation part of the corpus.

The results for the baseline—i.e. simple memorization of word-word correspondences—are (in %): P = 95.62, R = 43.52 and F_1 = 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 low, as is expected: slightly less than half the words in the evaluation corpus have been encountered before.[5]

6.1 Results with the lexdiff method

Table 2 shows the initial experiment of method 1 with different variations on the frequency threshold—this is the limit on the number of times we must see a string-change to learn it. The results clearly show that the more examples we extract (frequency 1), the better results we obtain for recall while at the same time the precision suffers since many spurious outputs are given—even many different ones that each legitimately correspond to a word in the standard dialect. The F_1-score doesn't vary very much and it maintains similar values throughout. The problem with this approach is one which we have noted before: the rules produce a large number of outputs for any given input word and the consequence is that the precision suffers, even though only those output words are retained that correspond to actual standard Basque.

	P	**R**	**F_1**
$f \geq 1$	38.95	66.78	49.20
$f \geq 2$	46.99	57.14	51.57
$f \geq 3$	49.39	53.82	51.51

Table 2: Values obtained for Precision, Recall and F-scores with method 1 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 within a word.

	P	**R**	**F_1**
$f \geq 1$	70.28	58.13	63.64
$f \geq 2$	70.18	53.16	60.49
$f \geq 3$	71.76	51.50	59.96

Table 3: Values obtained for Precision, Recall and F-score with method 1 by changing the threshold frequency of the correspondences and applying a post-filter.

With the additional unigram filter in place, the results improve markedly. The unigram-filtered results are given in table 3.

We have also varied the maximum number of possible rule applications within a single word as well as applying the rules in parallel or sequentially, and compacting the rules to provide more context-sensitivity. We shall here limit ourselves to presenting the best results of all these options in terms of the F_1-score in table 4.

In general, we may note that applying more than

[5]The reason the baseline does not show 100% precision is that the corpus contains minor inconsistencies or accepted alternative spellings, and our method of measuring the precision suffers from such examples by providing both learned alternatives to a dialectal word, while only one is counted as being correct.

	P	R	F_1
Exp1	72.20	57.81	64.21
Exp2	72.13	58.47	64.59
Exp3	75.10	60.13	66.79

Table 4: Method 1. **Exp1**: frequency 2; 2 rules applied; in parallel; without contextual conditioning. **Exp2**: frequency 1; 1 rule applied; with contextual conditioning. **Exp3**: frequency 2; 2 rules applied; in parallel; with contextual conditioning.

one rule within a word has a negative effect on the precision while not substantially improving the recall. Applying the unigram filter—choosing the most frequent candidate—yields a significant improvement: much better precision but also slightly worse recall. Choosing either parallel or sequential application of rules (when more than one rule is applied to a word) does not change the results significantly. Finally, compacting the rules and producing context-sensitive ones is clearly the best option.

In all cases the F_1-score improves if the unigram 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 unigram filter was applied.

Figure 2 shows how precision and recall values change in some of the experiments done with method 1. There are two different groups of points depending on if the unigram filter is applied, illustrating the tradeoff in precision and recall.

6.2 Results with the ILP method

The ILP-based results are clearly better overall, and it appears that the gain in recall by using method 1 does not produce F_1-scores above those produced with the ILP-method, irrespective of the frequency filters applied. Crucially, the negative evidence and subsequent narrowness of the replacement rules learned with the ILP method is responsible for the higher accuracy. Also, the results from the ILP-based method rely very little on the post-processing filters, as will be seen.

The only variable parameter with the ILP method concerns how many times a word-pair must be seen to be used as learning evidence for creating a replacement rule. As expected, the strongest result

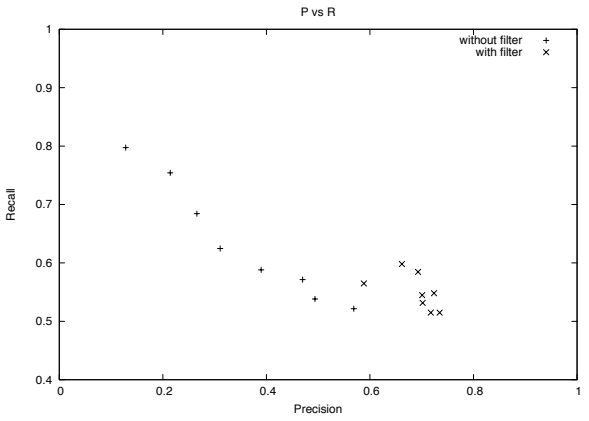

Figure 2: Tradeoffs of precision and recall values in the experiments with method 1 using various different parameters. When the unigram filter is applied the precision is much better, but the recall drops.

	P	R	F_1
$n = 1$	85.02 (86.13)	58.47 (57.80)	69.29 (69.18)
$n = 2$	82.33 (83.42)	54.15 (53.49)	65.33 (65.18)
$n = 3$	80.53 (82.07)	50.83 (50.17)	62.32 (62.26)
$n = 4$	81.19 (82.32)	50.17 (49.50)	62.01 (61.83)

Table 5: Experiments with the ILP method using a threshold of 1–4 (times a word-pair is seen) to trigger rule learning. The figures in parentheses are the same results with the added postprocessing unigram filter that, given several output candidates of the standard dialect, chooses the most frequent one.

is obtained by using all word-pairs, i.e. setting the threshold to 1. Table 5 shows the degradation of performance resulting from using higher thresholds.

Interestingly, adding the unigram filter that improved results markedly in method 1 to the output of the ILP method slightly worsens the results in most cases, and gives no discernible advantage in others. In other words, in those cases where the method provides multiple outputs, choosing the most frequent one on a unigram frequency basis gives no improvement over not doing so.

Additionally, there is comparatively little advantage with this method in adding the morphological filter to the output of the words in method 2 (this is the filter that rules out non-standard words). The results in table 5 include the morphological filter, but omitting it altogether brings down the best F_1

	P	R	F_1
Baseline	95.62	43.52	59.82
Method 1 (lexdiff)	75.10	60.13	66.79
Method 2 (ILP)	85.02	58.47	69.29

Table 6: The best results (per F_1-score of the two methods). The parameters of method 1 included using only those string transformations that occur at least 2 times in the training data, and limiting rule application to a maximum of 2 times within a word, and including a unigram post-filter. Rules were contextually conditioned. For method 2, all the examples (threshold 1) in the training data were used as positive and negative evidence, without a unigram filter.

to 56.14 from 69.29. By contrast, method 1 depends heavily on it and omitting the filter brings down the F_1-score from 66.79 to 11.53 with the otherwise strongest result of method 1 seen in table 6. The most prominent difference between the two approaches is that while method 1 can be fine-tuned using frequency information and various filters to yield results close to method 2, the ILP approach provides equally robust results without any additional information—in particular, frequency information of the target language. We also find a much lower rate of errors of commission with the ILP method; this is somewhat obvious as it takes advantage of negative evidence directly while the first method only does so indirectly through filters added later.

7 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. As background knowledge, we have a complete standard Basque morphological analyzer and a small parallel corpus of dialect and standard text. The approach has been based on the idea of extracting string-to-string transformation rules from the parallel corpus, and applying these rules to unseen words. We have been able to improve on the results of a naive baseline using two methods to infer phonological rules of the information extracted from the corpus and applying them with finite state transducers. In particular, the second method, inferring minimal phonological rewrite rules using an Inductive Logic Programming-style approach, seems promising as regards inferring phonological and morphological differences that are quite regular in nature between the two language variants. We expect that a larger parallel corpus in conjunction with this method could potentially improve the results substantially—with a larger set of data, thresholds could be set so that morphophonological generalizations are triggered only after a sufficient number of training examples (avoiding overgeneration), and, naturally, many more unique, non-regular, lexical correspondences could be learned.

During the current work, we have also accumulated a small but valuable training and test corpus which may serve as a future resource for evaluation of phonological and morphological rule induction algorithms.

In order to improve the results, we plan to research the combination of the previous methods with other ones which infer dialectal paradigms and relations between lemmas and morphemes for the dialect and the standard. These inferred relations could be contrasted with the information of a larger corpus of the dialect without using an additional parallel corpus.

Acknowledgments

We are grateful for the insightful comments provided by the anonymous reviewers. This research has been partially funded by the Spanish Science and Innovation Ministry via the OpenMT2 project (TIN2009-14675-C03-01) and the European Commission's 7th Framework Program under grant agreement no. 238405 (CLARA).

References

Alegria, I., Aranzabe, M., Arregi, X., Artola, X., Díaz de Ilarraza, A., Mayor, A., and Sarasola, K. (2011). Valuable language resources and applications supporting the use of Basque. In Vetulani, Z., editor, *Lecture Notes in Artifitial Intelligence*, volume 6562, pages 327–338. Springer.

Alegria, I., Aranzabe, M., Ezeiza, N., Ezeiza, A., and Urizar, R. (2002). Using finite state technology in natural language processing of basque.

In *LNCS: Implementation and Application of Automata*, volume 2494, pages 1–12. Springer.

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*.

Hulden, M. (2009a). 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.

Hulden, M. (2009b). Regular expressions and predicate logic in finite-state language processing. In Piskorski, J., Watson, B., and Yli-Jyrä, A., editors, *Finite-State Methods and Natural Language Processing—Post-proceedings of the 7th International Workshop FSMNLP 2008*, volume 191 of *Frontiers in Artificial Intelligence and Applications*, pages 82–97. IOS Press.

Johnson, M. (1984). A discovery procedure for certain phonological rules. In *Proceedings of the 10th international conference on Computational linguistics*, COLING '84, pages 344–347. Association for Computational Linguistics.

Kestemont, M., Daelemans, W., and Pauw, G. D. (2010). Weigh your words—memory-based lemmatization for Middle Dutch. *Literary and Linguistic Computing*, 25(3):287–301.

Koskenniemi, K. (1991). A discovery procedure for two-level phonology. *Computational Lexicology and Lexicography: A Special Issue Dedicated to Bernard Quemada*, pages 451–446.

Mann, G. S. and Yarowsky, D. (2001). Multipath translation lexicon induction via bridge languages. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, NAACL '01, pages 1–8.

Muggleton, S. and De Raedt, L. (1994). Inductive Logic Programming: theory and methods. *The Journal of Logic Programming*, 19:629–679.

Scherrer, Y. (2007). Adaptive string distance measures for bilingual dialect lexicon induction. In *Proceedings of the 45th Annual Meeting of the ACL: Student Research Workshop*, ACL '07, pages 55–60. Association for Computational Linguistics.