Morphological Reinflection with Conditional Random Fields and Unsupervised Features

Ling Liu
Department of Linguistics
University of Colorado
ling.liu@colorado.edu

Lingshuang Jack Mao
Department of Linguistics
University of Colorado
lima4664@colorado.edu

Abstract

This paper describes our participation in the SIGMORPHON 2016 shared task on morphological reinflection. In the task, we use a linear-chain conditional random field model to learn to map sequences of input characters to sequences of output characters and focus on developing features that are useful for predicting inflectional behavior. Since the training data in the task is limited, we also generalize the training data by extracting, in an unsupervised fashion, the types of consonant-vowel sequences that trigger inflectional behavior, and by extending the available training data through inference of unlabeled morphosyntactic descriptions.

1 Introduction

Our approach to the shared task focuses on expanding well-known methods to learning inflections. As our starting point, we assume a discriminative model akin to Durrett and DeNero (2013), Nicolai et al. (2015), and the baseline system provided by the organizers of the shared task, all very similar systems at the core. To improve performance and to address the more difficult reinflection tasks introduced in the shared task, we explore methods of expanding the training data, performing better alignment on the training data for our discriminative sequence classifier, feature development, and using unsupervised features for better generalization from training data.

In what follows, we describe a baseline system we developed, the system we actually participated with, and present the results, together with some analysis.

2 Exploratory experiments: a suffix-based baseline

To assess the difficulty of the task and the variation of inflectional behavior in the data sets, we ran a preliminary test with the data using a simple, suffix-based inflection strategy to complement the SIGMORPHON baseline. The method simply learns to transform input word form suffixes to suffixes of inflected forms. It works as follows: from each Levenshtein-aligned example pair **x** → **y** belonging to some morphosyntactic description (MSD) $\mathbf{m_{source}} \rightarrow \mathbf{m_{target}}$, we extract all the possible suffix-based string-to-string mapping rules that describe this mapping. In task 1, where the source MSD is not known, we assume that the source mapping is the lemma form. For example, if we have seen the example Finnish inflection **rakko** → **rakoitta**, going from lemma to **pos=N,case=PRIV,num=PL**, we extract the following alignment, with extra start-of-word and end-of-word markers

```
< r a k k o _ _ _ _ >
< r a k _ o i t t a >
```

This allows us to extract rules like the following for inflecting from the lemma form to **pos=N,case=PRIV,num=PL**:

>	→	**itta>**
o>	→	**oitta>**
ko>	→	**oitta>**
kko>	→	**koitta>**
akko>	→	**akoitta>**
rakko>	→	**rakoitta>**

From this, we devise a simple inflection strategy at test time where we always pick the longest matching such rule extracted from all word pairs that pertains to the MSD of the source and the target. The rationale for this baseline is that many

Proceedings of the 14th Annual SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 36–40,
Berlin, Germany, August 11, 2016.

	Suff baseline	SIGMORPHON baseline
Arabic	48.02 (45.97)	**70.30**
Finnish	**88.36** (88.21)	68.27
Georgian	**94.09** (92.75)	89.83
German	**92.24** (91.99)	90.36
Hungarian	**91.47** (87.76)	74.10
Maltese	**37.69** (36.59)	36.56
Navajo	35.47 (11.33)	**71.90**
Russian	88.94 (88.18)	**90.38**
Spanish	**98.31** (98.25)	96.93
Turkish	**77.65** (76.24)	59.17

Table 1: Results of a simple suffix-based baseline on task 1. Results are on the dev-set, and results in parentheses describe performance on the dev-set duplicates from the training-set removed.

hand-written models of morphology for various languages focus on suffixes to predict morphological behavior (Détrez and Ranta, 2012). As is seen in table 1, this yields comparably strong results for those languages that have largely suffixing inflections in the shared task (Finnish, Georgian, German, Hungarian, Spanish). It also identifies the difficult languages of the task for both—-Arabic, Maltese, and Navajo. These are languages that exhibit significant stem-internal alternations and prefixation processes that thus lie outside the scope of this simple method.

3 Sequence labeling

To address the shortcomings of the two baselines tested—that the discriminative classifier-based baseline works well with stem-internal changes but weakly with predominantly suffixing processes, and that the suffix strategy works only with suffixing languages—we develop a discriminative conditional random field (CRF) model and focus on improving the initial alignment of the input and output to better and more consistently capture prefixation and suffixation.

3.1 Alignment

We use the alignment procedure in the baseline provided by the organizers (Cotterell et al., 2016). This is a one-to-one aligner that learns globally optimal costs for aligning a set of word pairs. We first ran all the word pairs as a batch through this aligner, obtaining a one-to-one alignment of each pair in the entire training data. We also experimented with variants on alignment using Levenshtein distance with a bias toward aligning vowels with vowels and consonants with consonants, with consistently worse results.

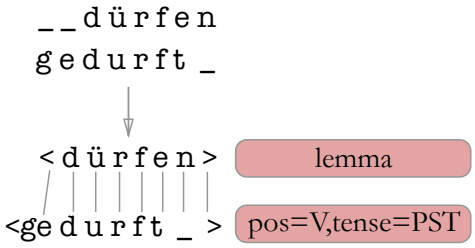

Figure 1: Example of the enforced one-to-many alignment after first aligning input-output pairs one-to-one.

After initial alignment of the input-output pairs, we additionally force a one-to-many alignment of the pairs, with added beginning and end markers $<$ and $>$. The markers are treated as actual symbols that serve to allow the stems to be entirely aligned on both sides despite possible prefixation and suffixation. In performing the alignment we enforce that the input side of the relation always comes in single characters, each of which alternatively map to the empty string, or a sequence. We bias this alignment in such a way that any initial input side zeroes are collapsed with the $<$-marker and any final output side zeroes are collapsed together with the $>$-marker. Stem-internal insertion sequences $x:y\ 0:z$ are always greedily associated with the leftmost change and become $x:yz$. This alignment simplifies the labeling process since each input letter is now assigned a label; furthermore, associating prefixes and suffixes with the alignment markers in a predetermined way allows for a consistent model of suffixing and prefixing in the label sequence learning process. This is illustrated in figure 1.

3.2 Labeling

We treat inflection generation as a labeling problem of converting an input sequence $\mathbf{x} = (x_1, \ldots, x_n)$ to an output sequence $\mathbf{y} = (y_1, \ldots, y_n)$. After the forced one-to-many alignment process, we convert the output side to a sequence of decisions $(y_1, \ldots, y_n)$ for use in a sequential labeling process. By default, the output strings, usually single characters, become the labels. However, we do not record a repetition (where the output equals the input) as a unique decision; rather, all repetitions are marked with a special symbol in the label sequence $\mathbf{y}$, i.e. all repetitions are marked alike in the output. Whenever the output differs from the

input, however, the output string itself becomes the label. In figure 1, the output sequence $\mathbf{y}$ would be **<ge-repeat-u-repeat-repeat-t-∅-repeat**. Decision sequences thus reflect the possible choices we have for each input symbol (including the boundary markers < and >)—we may repeat the symbol, delete the symbol, or output some other sequence of symbols.

Given input words of the form $\mathbf{x} = (x_1, \ldots, x_n)$ and the corresponding decision sequences $\mathbf{y} = (y_1, \ldots, y_n)$ we train a linear-chain CRF (Lafferty et al., 2001) by L-BFGS (Liu and Nocedal, 1989) using *CRFsuite* (Okazaki, 2007).

We model the conditional distribution of the output sequence in the standard way as

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z}\exp\Big(\sum_{i}^{n}\phi(y_{i-1}, y_i, \mathbf{x}, i)\Big) \quad (1)$$

where ϕ is a feature function which breaks down into k component functions

$$\phi(y_{i-1}, y_i, \mathbf{x}, i) = \sum_k w_k f_k(y_{i-1}, y_i, \mathbf{x}, i) \quad (2)$$

and where Z is the partition function which normalizes the expression to a proper distribution.

4 Features

We use a number of contextual features that look at variable amounts of context at each x_i point. Apart from standard local contextual features, we also employ features that refer to contexts as sequences of consonants and vowels (C/V).[1] In addition to local contextual C/V-features we also employ non-local features such as the types of vowels seen so far in the word and the last vowel seen at the current position, to better capture harmonic processes and Semitic root-and-pattern morphology. An overview of the most important features retained after ablation analysis is given in table 2.

5 Evaluation

5.1 Outside data

We separately test the feasibility of our approach against the data set published by Durrett and DeNero (2013), five data sets over three languages. That work used a similar approach (a semi-Markov CRF), albeit without the unsupervised features, and we improve upon their results that use a factored model, predicting each inflected word separately, as in the shared task, on three out of five data sets. We expect that with sparser, gappier training data—Durrett and DeNero (2013) used full inflection tables for training—our richer, more generic features will allow for better generalization.

5.2 MSD classification (task 3)

For task 3, where we are asked to inflect a word from an unknown source MSD, we first train a multi-class support vector machine (SVM) classifier (using LIBSVM (Chang and Lin, 2011)) to map the source form to an MSD. Each combination of MSDs is taken to represent a separate class—i.e. we treat each unique MSD-string as a class. As features, we use all substrings starting from the left and right edges of the word form in question, a method used successfully in e.g. morphological paradigm classification (Ahlberg et al., 2015). In track 2 (where only task 3 data is used), we train the classifier on only the given output forms and MSDs in the training data. In track 1, we feed the classifier all seen word forms and MSDs from any task whose data can be used.

5.3 Training method

In track 1, we inflect task 1 forms as described above whereas task 2 (arbitrary form to arbitrary form) is addressed by pivoting in two steps via the lemma form by first mapping the input form to the lemma form, and then mapping that form to the target form. We treat task 3 as a more difficult version of task 2; we first identify the unknown MSD of the task 3 input form, after which the procedure reduces to task 2. In the track 2 tasks 2 and 3, where only task-specific training data can be used, we are unable to pivot since form-to-lemma data is not available, and we train a separate CRF for each MSD to MSD mapping. In track 2 task 3, we first train the SVM classifier to identify MSDs, then classify the unknown MSDs of the input form in the training data, producing training data of the same format as in task 2.

We also experimented with training a single CRF model for each part of speech, using the feature/value pairs of the source/target forms as features. Somewhat surprisingly, this consistently yielded worse results on the development sets compared with training a separate model for each

[1] We used an off-the-shelf algorithm for this purpose (Hulden, in prep.); there are many highly reliable unsupervised methods for extracting vowels and consonants given a corpus of words in an alphabetic writing system (Guy, 1991; Kim and Snyder, 2013; Moler and Morrison, 1983; Sukhotin, 1962).

Feature	Description
`frombeg`	Position counting from left edge
`fromend`	Position counting from right edge
`insymbol`	The current input symbol
`prevsymbol`	The previous input symbol
`prevsymbol2`	The input symbol two to the left
`prevsymbol3`	The input symbol three to the left
`previoustwo`	The previous two input symbols
`nextsymbol`	The next input symbol
`nextsymbol2`	The input symbol two to the right
`nexttwo`	The next two input symbols
`nextgeminate`	1 if the next input equals the current input
`geminate`	1 if the current input equals the previous input
`isC`	Is the current input symbol a consonant
`isV`	Is the current input symbol a vowel
`prevC`	Is the previous input symbol a consonant
`prevV`	Is the previous input symbol a vowel
`nextC`	Is the next input symbol a consonant
`nextV`	Is the next input symbol a vowel
`lastvowel`	What is the last vowel seen to the left of the current position
`allvowels`	The set of vowels in the word
`trigram`	The trigram $x_{i-1}\ x_i\ x_{i+i}$
`trigramCV`	The trigram mapped to C/V symbols

Table 2: The main feature templates used.

	CRF	D&DN13	Suffix-rules
DE-V	**96.14**	94.76	91.29
DE-N	83.75	**88.31**	86.18
ES-V	**99.62**	99.61	63.95
FI-V	97.18	**97.23**	72.00
FI-N	92.30	92.14	**92.62**

Table 3: Our approach on the Durrett and DeNero (2013) dataset, comparing our model with that work (D&DN13) and the simple suffix-replacing model introduced earlier.

lemma-to-MSD (track 1) or MSD-to-MSD (track 2), and we settled for using separate models.

6 Results

The main results on the development data for task 1 are given in tables 4, 5, and 6. We separately list figures with and without the C/V-features, which resulted in an average increase in accuracy of 1.02% (task 1), 1.58% (task 2), and 1.18% (task 3). As the development data includes instances also found in the training data, we separately report the accuracy without such duplicates, given in parentheses, as these results better reflect the performance on the final test data.

7 Discussion

The approach we have used clearly outperforms the baselines provided by the task and our own

	dev		test
	no CV	CV	
Arabic	74.00 (72.13)	**74.63** (72.81)	72.42
Finnish	88.86 (88.71)	**90.05** (89.92)	88.65
Georgian	**94.79** (93.46)	94.59 (93.22)	93.86
German	92.42 (92.05)	**92.61** (92.25)	92.64
Hungarian	91.04 (88.74)	**93.94** (91.28)	91.05
Maltese	**42.03** (40.81)	41.49 (40.22)	43.49
Navajo	88.01 (65.23)	**92.01** (63.67)	53.28
Russian	**90.44** (89.79)	90.13 (89.45)	89.13
Spanish	98.68 (98.63)	**98.74** (98.70)	98.28
Turkish	85.34 (84.15)	**88.91** (88.01)	87.39

Table 4: Main results for track 1, task 1.

	dev		test
	no CV	CV	
Arabic	63.93 (63.93)	**65.62** (65.62)	62.74
Finnish	79.87 (79.87)	**82.00** (82.00)	80.19
Georgian	**92.37** (92.37)	92.25 (92.25)	90.87
German	89.31 (89.31)	**89.43** (89.43)	88.44
Hungarian	87.50 (87.50)	**90.20** (90.20)	87.49
Maltese	**22.66** (22.66)	21.29 (21.79)	22.54
Navajo	70.54 (70.48)	**76.67** (76.62)	46.13
Russian	**87.06** (87.06)	86.93 (86.93)	86.71
Spanish	**97.43** (97.43)	97.12 (97.12)	97.18
Turkish	67.12 (67.12)	**70.37** (70.37)	67.50

Table 5: Main results for track 1, task 2.

	dev		test
	no CV	CV	
Arabic	61.75 (61.75)	**62.62** (62.62)	58.83
Finnish	79.43 (79.43)	**81.68** (81.68)	79.45
Georgian	**91.86** (91.85)	91.80 (91.79)	90.43
German	**87.68** (87.71)	87.62 (87.39)	86.59
Hungarian	87.33 (87.32)	**89.95** (89.94)	87.04
Maltese	**20.54** (20.54)	19.58 (19.58)	20.58
Navajo	71.71 (71.45)	**80.66** (77.54)	47.30
Russian	86.31 (86.29)	**86.37** (86.35)	85.34
Spanish	**96.43** (96.43)	96.18 (96.18)	96.26
Turkish	64.43 (64.41)	**67.50** (67.47)	65.63

Table 6: Main results for track 1, task 3.

baseline. There is room for improvement, however. We attribute the weak performance on the difficult languages of the task (Arabic, Maltese, and Navajo, in particular) to limitations on the linear-chain CRF model. Because of the immediately local dependency on the previous label, the model is unable to accurately capture multiple disjoint changes in going from word form to word form—something that is present in the Semitic languages of the data sets and Navajo. In the future, we want to experiment with more general CRF models to address this shortcoming (Sutton and McCallum, 2011). We also want to explore techniques for training a single model per part-of-speech instead of a separate model for each inflection type. In our experiments of training single models, this produced no improvement, but it seems that such an approach is indispensable in order to be able to generalize beyond the specific training data given. Consider, for example, seeing the Finnish word **talo** ('house') in its *singular* and *plural inessives* **talossa/taloissa** and the *singular abessive*, **talotta**. In a single model, we should be able to infer, without ever seeing an inflection of that type, that the *plural abessive* form is **taloitta**, isolating the plural **i**-morpheme. However, in a model where each complex inflection is learned separately, this cannot be learned without actually seeing an example of the combination *abessive* and *plural*.[2]

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Proceedings of NAACL-HLT*, pages 1024–1029, Denver, Colorado, May–June. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *Proceedings of the 2016 Meeting of SIGMORPHON*, Berlin, Germany, August. Association for Computational Linguistics.

Grégoire Détrez and Aarne Ranta. 2012. Smart paradigms and the predictability and complexity of inflectional morphology. In *Proceedings of the 13th EACL*, pages 645–653. Association for Computational Linguistics.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of NAACL-HLT*, pages 1185–1195.

Jacques B.M. Guy. 1991. Vowel identification: an old (but good) algorithm. *Cryptologia*, 15(3):258–262.

Young-Bum Kim and Benjamin Snyder. 2013. Unsupervised consonant-vowel prediction over hundreds of languages. In *Proceedings of ACL*, pages 1527–1536, Sofia, Bulgaria, August. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528.

Cleve Moler and Donald Morrison. 1983. Singular value analysis of cryptograms. *American Mathematical Monthly*, pages 78–87.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Inflection generation as discriminative string transduction. In *Proceedings of NAACL-HLT*, pages 922–931, Denver, Colorado, May–June. Association for Computational Linguistics.

Naoaki Okazaki. 2007. CRFsuite: a fast implementation of conditional random fields (CRFs).

Boris V. Sukhotin. 1962. Eksperimental'noe vydelenie klassov bukv s pomoshch'ju EVM. *Problemy strukturnoj lingvistiki*, pages 198–206.

Charles Sutton and Andrew McCallum. 2011. An introduction to conditional random fields. *Machine Learning*, 4(4):267–373.

[2]This work has been partly sponsored by DARPA I20 in the program Low Resource Languages for Emergent Incidents (LORELEI) issued by DARPA/I20 under Contract No. HR0011-15-C-0113.