The Columbia University - New York University Abu Dhabi SIGMORPHON 2016 Morphological Reinflection Shared Task Submission

Dima Taji[†], Ramy Eskander[‡], Nizar Habash[†], Owen Rambow[‡]

[†]Computational Approaches to Modeling Language Lab, New York University Abu Dhabi
{dima.taji,nizar.habash}@nyu.edu
[‡]Center for Computational Learning Systems, Columbia University
{reskander,rambow}@ccls.columbia.edu

Abstract

We present a high-level description and error analysis of the Columbia-NYUAD system for morphological reinflection, which builds on previous work on supervised morphological paradigm completion. Our system improved over the shared task baseline on some of the languages, reaching up to 30% absolute increase. Our ranking on average was 5th in Track 1, 8th in Track 2, and 3rd in Track 3.

1 Introduction

In this paper, we present a high-level description and error analysis of the Columbia University - New York University Abu Dhabi system for morphological reinflection, which was submitted to the SIGMORPHON 2016 shared task on morphological reinflection (Cotterell et al., 2016). The system builds on previous work on supervised morphological paradigm completion (Eskander et al., 2013). Although the core system is the same, additional efforts were needed to preprocess the shared task data in order to make it compatible with our existing approach, as well as to create a new interface that targets morphological reinflection specifically. Our system improved over the baseline on some of the languages, reaching up to 30% absolute increase. Our ranking on average was 5th in Track 1, 8th in Track 2, and 3rd in Track 3.

The rest of this paper is structured as follows. We present some related work in computational morphology in Section 2. We then discuss our approach to the shared task in Section 3. We discuss our performance and insights in Section 4. Finally, we give an outlook on our approach in Section 5.

2 Related Work

The area of computational morphology includes a rich and varied continuum of approaches and techniques. Within it, we find, on one end, systems painstakingly designed by hand (Koskenniemi, 1983; Buckwalter, 2004; Habash and Rambow, 2006; Détrez and Ranta, 2012); and on the other end, unsupervised methods that learn morphology models from unannotated data (Creutz and Lagus, 2007; Dreyer and Eisner, 2011; Rasooli et al., 2014; Monson et al., 2008; Hammarström and Borin, 2011). Closer to the former side of the continuum, we find work on minimally supervised methods for morphology learning that make use of available resources such as parallel data, dictionaries or some additional morphological annotations (Yarowsky and Wicentowski, 2000; Snyder and Barzilay, 2008; Cucerzan and Yarowsky, 2002). Closer to the other end, we find work that focuses on defining morphological models with limited lexicons that are then extended using raw text (Clément et al., 2004; Forsberg et al., 2006). The setting of the shared task on morphological reinflection (Cotterell et al., 2016), which provides a rich partly annotated training data set, encourages methods that are supervised.

Our shared task submission builds on our previously published work on paradigm completion (Eskander et al., 2013), which falls somewhere in the middle of the continuum outlined above. Our approach learns complete morphological models using rich morphological annotations. The match of the requirements for our approach and those for the shared task was far from perfect, but it was interesting to participate since we have always wanted to explore ways to reduce some of the expected input annotations – in particular word segmentations, which are not provided in the shared task.

Proceedings of the 14th Annual SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 71–75,
Berlin, Germany, August 11, 2016.

3 Approach

Our basic approach is supervised paradigm completion as presented in (Eskander et al., 2013). It was designed to learn a morphological analyzer and generator from data that has been manually segmented and clustered by lexeme. To adapt our approach to the shared task, we added two phases that sandwich the basic paradigm completion in a pipeline architecture: an initial *segmentation* phase to preprocess the shared task data to match our approach's expectations; and a phase to perform reinflection as specified in the various sub-tasks in the shared task.

3.1 Word Segmentation

In the segmentation phase, we segment every word in the training dataset (TRAIN) into prefix, stem, and suffix. For example, the Arabic word *al-muhandisatu* is ideally segmented into *al-+muhandis+atu*. This phase has three steps.

In the first step, we estimate the probability of every possible *affix-feature* and *stem-POS* (Part-of-Speech).[1] This is accomplished by summing over the partial counts of all possible segmentations of every word in TRAIN, constrained only by a minimal stem length parameter, s.[2]

In the second step, we cluster the words in *lemma clusters*, which are groups of words that share the same lemma and only vary in terms of inflectional morphology. For Task 1, lemmas were given in TRAIN, and we clustered all the words that appear with each given lemma. For Task 2 and Task 3, we used cooccurrence evidence to determine the clusters, where if two words appeared together in the same training instance, we assigned them to the same lemma cluster. This may result in under-clustering due to words not appearing together, as well as over-clustering, when the entries consist of derivational rather than inflectional morphology. To normalize the treatment of the different tasks, lemmas in Task 1 were included as words with the feature *feat=lemma*; and words appearing with no features in Task 3 were given the POS of the word they appeared with and the feature *feat=UNK*.

Finally, we decide on the optimal segmentation for each word in the context of its lemma cluster in the following manner. For every word in the cluster, we produce a ranking of top b segmentations[3] as an initial filter. This rank is based on *P(stem-POS)*P(affix-feature)*, which were computed in the first step. We select for each word the segmentation that minimizes the overall number of stems in the lemma cluster, with a bias towards the stem with the highest *prominence score*. A stem prominence score is computed as the probability of the stem-POS multiplied by the number of occurrences it has in the top b segmentation choices (for all words in the cluster). This ensures that, for all the words that have a segmentation including the top stem, this segmentation is selected. For the words that do not have the top stem among their b segmentations, we go for the next stem in the prominence score ranking, and so on.

Two problematic cases are handled exceptionally: words with *feat=UNK*, because it appears very frequently, and features that are infrequent (below a threshold x[4]). Those words are not used as part of determining the prominence score, but the stem is forced upon them.

At the end of this process, we should have a specific segmentation for every word. To assess the algorithm's performance, we ran it on an external data set of Egyptian Arabic (Eskander et al., 2013), for which we had a manually annotated gold standard. Our segmentation algorithm achieves a word segmentation accuracy of 84.7% when tested on this dataset.

3.2 Paradigm Completion

The core of our work is paradigm completion, in which we build complete inflectional classes (ICs) based on corpus annotations. The construction of the ICs follows the technique we presented in (Eskander et al., 2013), where the ICs have all the possible morphosyntactic feature combinations for every lemma in TRAIN.

The paradigm completion step works as follows. First, the entries in TRAIN are converted into paradigms, where each paradigm lists all the inflections of all morphosyntactic feature combinations for a specific lemma seen in the training data. The paradigms are then converted into inflectional classes (ICs), where stem entries are abstracted as templates by extracting out the root letters. We de-

[1] For the Arabic word *al-muhandisatu*, the ideal *affix-feature* pairs the affix *al-+ _ +atu*, with the feature set that appears with this word, *pos=ADJ, def=DEF, case=NOM, gen=FEM, num=SG*. The ideal stem-POS pairs the stem *muhandis* with the POS, *pos=ADJ*.

[2] $s = 3$, determined empirically.

[3] $b = 10$, determined empirically.

[4] x = 5, determined empirically.

termine which letters should be considered pattern (non-root) letters for each language. These are letters that change between stems in the same IC. For example, in English *sing, sang, sung*, we observe that the vowel *i* can change to *a* or *u*. So these three letters change in this IC. For each letter, we count the number of ICs in which the letter undergoes a change in the IC stems; and we order the letters by this frequency. We then use a small tuning corpus to determine what subset of letters is optimal for the specific language: we repeatedly perform paradigm completion with all initial segments of the ordered list of letters. We choose the subset which results in the best performance on the task. [5] Finally, the generated ICs are merged together into a smaller number of more condensed ICs, where two ICs merge if they share the same inflectional behavior. The ICs are then completed by sharing affix and stem information with each other. We apply the above process to the different POS types in the shared task, independently.

3.3 Morphological Reinflection

The set of ICs created in paradigm completion are used to create a morphological analyzer and generator. In cases in which the input is a previously seen (i.e., in TRAIN) lemma (Task 1) or an inflected form with a tag (Task 2) or without a tag (Task 3), we can match the input against the IC which was created from that item, and then we can just generate the requested form. In cases of unseen lemmas or forms, we run the ICs as a morphological analyzer, which matches it against an IC, and we again proceed to generate the inflected form.

4 Results

4.1 Shared Task

We participated in all the tracks and tasks of the shared task, with a total of nine submissions for almost all languages. Our ranking, on average over all languages and tasks, was 5th (5.4) in Track 1, 8th (7.6) in Track 2, and 3rd (2.6) in Track 3. The best performing systems used neural network models, which have improved performance in many areas of NLP (Manning, 2016).

[5] We do this for Task 1 (in which the lemma is available as input), and then use the same set in all task/track combinations. For some task/track combinations we should have used different data, but did not do this for lack of time. We acknowledge the methodological flaw, but we suspect that doing it differently would not have changed our results much.

In Table 1, we present the results for our system (Track 3 Task 3) and baseline together with a number of features of the different language sets and their correlations with the final scores. We do not include the results for Finnish and Russian as our system had many problems and we believe the results are not meaningful.[6] We also do not present any results on the other tasks and tracks, although we participated in all of them, because of limited space. The results across tasks and tracks are comparable to the task we selected here. For some languages, our approach did well, increasing quality over the official baseline for six languages by up to 30% absolute (Turkish).

Given our previous work on Arabic (Eskander et al., 2013), we were dismayed to see the lowish results on Arabic; although this was not unexpected given that the Arabic used in the shared task was not in standard Arabic orthography. Arabic has a morphophonemic orthography that hides some of the allomorphic distinctions made explicit in the shared task, and some of which we do not do well on. For instance, the Arabic definite article morpheme *al-* has a number of allomorphs that assimilate to the consonant that follows the article when the consonant is one of the 14 *sun letters* (see (Habash, 2010) for a discussion), e.g., *al-daftaru* → *ad-daftaru* 'the notebook', and *al-numūru* → *an-numūru* 'the tigers', as opposed to the default *al-'uk̲ti* → *al-'uk̲ti* 'the sister'. In Arabic's morphophonemic orthography, the different allomorphs are written using one form *Al*: *Ald∼aftaru* الدفتر, *Aln∼umuwru* النمور, and *AlÂuxti* الأخت.[7]

4.2 Correlation Insights

We present next some insights gained by considering correlations between specific features of the data sets for different languages and the languages' performance in the baseline and our system. We expect some of these insights to be general indicators of the complexity of modeling different languages morphologically.

The first two columns in Table 1, $\textbf{Stem}_{allo}$ and $\textbf{Affix}_{allo}$, are the stem allomorphy rate and the affix allomorphy rate, respectively. $\textbf{Stem}_{allo}$ is computed as the ratio of unique stems divided by the number of lemmas in our training data. The

[6] Our official system's performance on Track 3 Task 3 for Finnish and Russian are 20% (40% below baseline) and 62% (19% below baseline), respectively.

[7] Arabic transliteration is presented in the Habash-Soudi-Buckwalter scheme (Habash et al., 2007).

Language	$\textbf{Stem}_{allo}$	$\textbf{Affix}_{allo}$	Features	Lemmas	Examples	log(F)	log(A/S)	log(L/F)	log(E/F)	System	Baseline
Arabic	2.8	8.0	232	5,357	47,298	2.37	0.5	1.36	2.3	59%	51%
Georgian	1.1	5.9	96	9,483	49,813	1.98	0.7	1.99	2.7	90%	87%
German	1.2	6.7	105	10,834	49,633	2.02	0.8	2.01	2.7	79%	82%
Hungarian	2.0	14.3	91	6,936	82,244	1.96	0.9	1.88	3.0	89%	78%
Maltese	4.5	2.6	3,825	3,186	65,597	3.58	-0.2	-0.08	1.2	29%	25%
Navajo	5.9	22.8	58	2,235	41,401	1.76	0.6	1.59	2.9	84%	60%
Spanish	1.2	13.5	90	9,374	48,217	1.95	1.0	2.02	2.7	79%	89%
Turkish	1.8	9.9	195	5,620	46,693	2.29	0.7	1.46	2.4	84%	54%
$\textbf{System}_{correl}$	-0.43	0.54	-0.90	0.45	-0.16	-0.93	0.89	0.91	0.94		
$\textbf{Baseline}_{correl}$	-0.67	0.26	-0.76	0.81	-0.08	-0.82	0.88	0.93	0.85		

Table 1: Results for our system (Track 3 Task 3) and baseline together with a number of features of the different language sets and their correlations with the final scores. In columns 7 through 10: F=Features, L=Lemmas, E=Examples, A=Affixes, and S=Stems.

$\textbf{Affix}_{allo}$ is computed as the ratio of unique affixes divided by the number of features in our training data. Ideally, these two measures would reflect the complexity of the language, as well as how well we model it. The numbers may be larger than or smaller than the ideal model depending on the quality of our segmentation step and the amount of training data. Next in Table 1 are the counts of features, lemmas, and training examples. Finally, we present four metrics that are derived from the previously shown values, e.g. **logE/F** is the log of the ratio of *training examples to features*. In the last two rows, we show the correlation coefficient between each presented value and the final score over all languages.

The languages vary widely in performance in both the baseline and our system. It is immediately evident that among the basic aspects of the training data, the number of features has a very high negative correlation with the results – see Features, and log(F) in Table 1. The number of training examples is not an indicator of performance ranking in this set. However, the log of the ratio of training examples to features is a very strong indicator with very high correlation. This makes sense since we expect languages with richer morphology to require relatively more examples than languages with poorer morphology. We computed the $\textbf{Stem}_{allo}$ and $\textbf{Affix}_{allo}$ because we thought they may give us insights into how our approach handles different languages and perhaps reflect errors in the segmentation process: we expected segmentation errors to inflate these two values. This hypothesis was not possible to confirm since the number of variant forms is not only dependent on segmentation and number of features in a language, but ultimately the number of training examples in relation to the number of features. As such, these two values are not well correlated with the results; although, interestingly, the ratio of $\textbf{Affix}_{allo}$ to $\textbf{Stem}_{allo}$ is. This may simply reflect that languages with less variable stems and more content-heavy affixes may be easier to model. The log of the ratio of lemmas to features, is another interesting measure with very high correlation (particularly in the baseline). This measure reflects that it is harder to model languages with very rich features without providing lots of collections of examples (lemma sets).

4.3 Error Analysis

When analyzing the errors in our output, we observed a number of overlapping phenomena that made the identification of the specific sources of the errors difficult. The following are the three main phenomena we noted.

(1) Stemming errors, where the segmentation process added letters to the stem or ignored letters in the stem. For example, the Arabic word *al-ʻajalatu* was stemmed to *l-ʻajala* instead of *ʻajala*, and the system ended up generating *l-ʻajalatay* as the dual possessive noun instead of *ʻajalatay*. Similarly, in Maltese the stem of the word *nqtilthomx* was determined to be the word itself, thus generating *nqtilthomxthulhomx* instead of *nqtilthulhomx*.

(2) Paradigm errors, where the system optimized for the wrong pattern because of possible previous stemming errors. In Spanish, for example, the word *curta* was reinflected as *curtéis* instead of *curtáis*, and in Turkish *ortağa* generated *ortağlara* instead of *ortaklara*.

(3) Allomorphy modeling errors, which can be present in affixes or in stems. When one morpheme can have multiple allomorphs, the system

might prefer the wrong allomorph. This can be observed in the cases of the Arabic definite article's so-called *sun letters*, where the system generated *al-dafī'i* instead of *ad-dafī'i*, as well as the Turkish vowel harmony issues, where *uğratmiyorsunuz* was generated instead of *uğratmıyorsunuz*. Stem allomorphy happens when the stem slightly changes depending on specific features, for example, in Arabic *qāmūs* and *qawāmīs* are stems for the same noun (one singular and one broken plural), which caused the generation of *al-qāmūsi* instead of *al-qawāmīsi*.

5 Outlook

We built our morphological paradigm completion system for a specific purpose: we are annotating texts by hand in order to develop morphological analyzers and taggers for Arabic dialects. The question now arises whether we can reduce the human annotation work (say, by not requiring human segmentation of the input data) while still maintaining the same quality of morphological resources.

We intend to explore methods to improve the segmentation step in order to reduce the errors produced in it. In particular, we will explore joint segmentation and paradigm completion; better allomorphy modeling, perhaps by introducing new models for allomorphy detection, and for phonological and morphological rewrites; and application of deep learning.

References

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. LDC catalog number LDC2004L02, ISBN 1-58563-324-0.

Lionel Clément, Bernard Lang, Benoît Sagot, et al. 2004. Morphology based automatic acquisition of large-coverage lexica. In *LREC 04*, pages 1841–1844.

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.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing (TSLP)*, 4(1).

Silviu Cucerzan and David Yarowsky. 2002. Bootstrapping a multilingual part-of-speech tagger in one person-day. In *proceedings of the 6th conference on Natural language learning-Volume 20*, pages 1–7. Association for Computational Linguistics.

Grégoire Détrez and Aarne Ranta. 2012. Smart paradigms and the predictability and complexity of inflectional morphology. *EACL 2012*, page 645.

Markus Dreyer and Jason Eisner. 2011. Discovering morphological paradigms from plain text using a dirichlet process mixture model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 616–627. Association for Computational Linguistics.

Ramy Eskander, Nizar Habash, and Owen Rambow. 2013. Automatic Extraction of Morphological Lexicons from Morphologically Annotated Corpora. In *Proceedings of tenth Conference on Empirical Methods in Natural Language Processing*.

Markus Forsberg, Harald Hammarström, and Aarne Ranta. 2006. Morphological lexicon extraction from raw text data. *Advances in Natural Language Processing*, pages 488–499.

Nizar Habash and Owen Rambow. 2006. MAGEAD: A Morphological Analyzer and Generator for the Arabic Dialects. In *Proceedings of ACL*, pages 681–688, Sydney, Australia.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*. Springer.

Nizar Habash. 2010. *Introduction to Arabic Natural Language Processing*. Morgan & Claypool Publishers.

Harald Hammarström and Lars Borin. 2011. Unsupervised learning of morphology. *Computational Linguistics*, 37(2):309–350.

Kimmo Koskenniemi. 1983. Two-Level Model for Morphological Analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685.

Christopher D Manning. 2016. Computational linguistics and deep learning. *Computational Linguistics*.

Christian Monson, Jaime Carbonell, Alon Lavie, and Lori Levin. 2008. Paramor: Finding paradigms across morphology. *Advances in Multilingual and Multimodal Information Retrieval*, pages 900–907.

Mohammad Sadegh Rasooli, Thomas Lippincott, Nizar Habash, and Owen Rambow. 2014. Unsupervised morphology-based vocabulary expansion. In *ACL (1)*, pages 1349–1359.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio, June.

David Yarowsky and Richard Wicentowski. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 207–216. Association for Computational Linguistics.