The Automatic Characterization of Grammars from Small Wordlists

Jordan Kodner[1], Spencer Caplan[1], Hongzhi Xu[2], Mitchell P. Marcus[2], Charles Yang[1]
University of Pennsylvania
[1] Department of Linguistics
[2] Department of Computer and Information Science
{jkodner, spcaplan}@sas.upenn.edu
{xh, mitch}@cis.upenn.edu
charles.yang@ling.upenn.edu

Abstract

We present two novel examples of simple algorithms which characterize the grammars of low-resource languages: a tool for the characterization of vowel harmony, and a framework for unsupervised morphological segmentation which achieves state-of-the-art performance. Accurate characterization of grammars jump starts the process of description by a trained linguist. Furthermore, morphological segmentation provides gains in machine translation as well, a perennial challenge for low-resource undocumented and endangered languages.

1 Introduction

A persistent difficulty in describing a language's grammar is the time consuming and error prone task of pouring over collected data searching for patterns. While the idea of automation is appealing, existing natural language processing applications are often non-functional on the kind of data which may be collected in the field. They may rely on supervised algorithms which require labeled or annotated data, but that defeats the purpose here. A successful tool for characterizing an unknown grammar should not require the user to have described the grammar beforehand. So called unsupervised algorithms, which are are trained on unlabeled data, exist, but they tend to require very large digital corpora to perform well. This is not a problem for tools meant to run on English or French, but collecting enough material on an endangered or undocumented language is often extremely difficult or impossible.

Standard NLP applications are not up to the task. Useful tools in this field must run on unusually small corpora with high enough accuracy if they are to be a help rather than a hindrance to a trained linguist. So, what is needed is a class of tools designed specifically with small data in mind and which will work for a diverse variety of languages. We present two such tools that operate on different grammatical domains. The first is a method for automatically searching for, identifying, and characterizing vowel harmony. Designed with small raw wordlists in mind, it leverages basic typological and theoretical principles. Tested on families ranging from Turkic (Turkish, Uyghur) to Pama-Nyungan (Warlpiri), it accurately describes diverse harmony systems. Our approach is novel in that it does not require the researcher to specify what kind of harmony system, if any, to expect. The algorithm discovers the patterns on its own.

Additionally, we present an algorithm which achieves state-of-the-art results in unsupervised morphological segmentation. Prior work in unsupervised segmentation has taken advantage of very large corpora, on the order of billions of words (Goldwater et al., 2006; Narasimhan et al., 2015). Tested on English and Turkish, our algorithm produces superior results not on billions, but on only hundreds of thousands of words. An obvious issue here is the lack of languages to test on. While anyone can run the algorithm on any language, we can only test the quality of the segmentation on languages for which a gold-standard segmentation already exists. To remedy this, we are currently developing annotation standards for automatic morphological segmentation, with Korean and Faroese as test languages. We hope this will prove useful for other researchers working on their own languages of interest.

2 Discovering and Describing Vowel Harmony

We begin with an algorithm for discovering vowel harmony in unannotated wordlists. It is designed to produce some kind of useful output no matter how small its input data, but the more that is provided to it, the more information it will extract. In all, it answers a number of important questions about a language's vowel harmony system beginning with the obvious: *does this language have vowel harmony?*. If so, then what are the harmonizing sets, and how do vowels map between them? Which vowels, if any, are neutral? Are neutral vowels transparent or opaque? Is secondary harmony present? And so on. These questions constitute the major part of what defines a harmony system. Answering them gives a trained linguist a leg up before working out the finer details, or serves to inform other NLP tools in a pipeline.

While prior work exists on the automatic discovery or analysis of vowel harmony systems, our method is, to our knowledge, the first end-to-end pipeline for describing both primary and secondary vowel harmony on arbitrary languages. The Vowel Harmony Calculator (Harrison et al., 2004) calculates metrics over a language given a specification of its harmony system. This is not useful for the present purpose because it requires a description of the system beforehand. Sanders and Harrison (2012) improves on this by not requiring a full description as input. Nevertheless, it only provides a metric for how "harmonic" languages are and not a description of the vowel harmony system (Sanders and Harrison, 2012). Baker (2009) describes a two-state Hidden Markov Model (HMM) which characterizes English and Italian, and may describe Turkish and Finnish correctly depending on random initial parameter settings (Baker, 2009). The model is restricting in that it requires the user to decide upfront how many levels of harmony to find. It only accounts for secondary harmony by setting up the HMM with four initial states at the outset. The paper also shows that a models using mutual information (MI) and Boltzmann fields can accurately identify vowel-to-vowel interactions but does not provide a means for describing vowel harmony given the results.

2.1 Algorithm Overview

In an approach broadly similar to (Baker, 2009)'s MI model, our algorithm leverages distributional patterns within the vowel system to extract information about vowel harmony. But due to clever transformations of the data and typologically motivated assumptions, it neatly extracts the detailed information about the system to produce a description.

The algorithm is flexible in its input. At a minimum, all it requires is a wordlist a few hundred words long. If the orthography corresponds more-or-less to a phonemic representation (like with Finnish, for example), no transcription is needed. This alone provides enough information for a basic rough analysis. A wordlist thousands long, with frequencies if possible, is better. For a complete analysis, the current version of the algorithm requires a short list of potentially relevant vowel features for each vowel. These basic features (e.g., ±rnd, ±hi, etc.) are not strongly tied to any particular phonological theory, and are customizable by the researcher.

The algorithm takes a wordlist (with frequencies if available) as input, a list of vowels in the same orthography as the input, and an optional table of proposed vowel features for each vowel. It outputs a partition of the vowel space into two primary harmonizing sets if present, and/or a neutral set. If provided with features, it outputs mappings between the two sets and a partition for secondary harmony if present.

Algorithm 1 describes the detection process. At a high level, it proceeds as follows. First, if frequencies are provided, the lowest frequency items are removed. These are more likely than high frequency items to contain names and loan words which fail to adhere to the harmony pattern. Then tier-adjacent vowel pairs are tabulated. For example, in the Finnish *Kalevala*, the tier-adjacent pairs are *a-e*, *e-a*, and *a-a*. These counts are converted to mutual information. The MI step is necessary to account for the uneven frequencies of individual vowels. For example, if *a* is very common in a language while *y* is rare, the counts for every vowel with *a* might be higher than with *y* in spite of harmony constraints. MI controls for these unconditioned frequency effects.

This process yields an MI co-occurrence vector for every vowel. In the absence of co-occurrence restrictions, each vector should correspond to a fairly level distribution. That is, each vowel is expected to co-occur with every other vowel at a more or less uniform rate if the frequencies of both vowels are accounted for. However, with vowel harmony restricting co-occurrence, the distributions should be skewed because vowels should rarely co-occur with members of the opposing harmony set. Any vowel with a sufficiently level distribution is probably neutral. If all or all but one of the language's vowels appears neutral, then the language does not have vowel harmony[1]. K-means clustering, a standard and simple unsupervised clustering algorithm, is used to find the optimal division of the non-neutral vowels into two harmonizing sets. If features are provided, they are used to match up which vowel maps to which between sets. Then the process is rerun with the harmonizing feature removed to find additional harmony patterns.

Algorithm 1 VH Characterization Algorithm

```
if frequencies provided then
    Trim tail off wordlist
while $True$ do
    Calculate tier-adjacent vowel-vowel co-occurrence matrix
    Calculate MI between each vowel pair
    Identify vowels whose MI distributions uniform within threshold.
    Assign these to the neutral vowel set and remove from consideration
    if number of non-neutral vowels $\leq 1$ then
        return
    Run k-means ($k = 2$) clustering on the remaining vowels' MI vectors
    if no features provided then
        return
    else
        Map vowels between harmonizing sets by finding pairs that share the most features in common.
        vowel list ← Collapse vowels along the harmonizing feature
        rerun for secondary harmony
return
```

The algorithm requires that a list of vowels be provided as input. This might be a problem if the user chooses to include marginal vowels from loanwords as input, for example ⟨y⟩ in German or ⟨å⟩ in Finnish. So as a fail-safe, the algorithm has facilities for automatically detecting and removing such vowels from the analysis. Marginal vowels tend to have much higher self-MI than MI with any other vowel, allowing them to be identified. This follows intuitively from the assumption that marginal vowels tend to appear only in loanwords.

[1]Vowel harmony requires a partitioning of the vowel system into at least two sets. Therefore, there need to be at least two vowels with skewed distributions to propose harmony

2.2 Results

This simple algorithm performs very well on the languages tested. We choose languages with easily accessible corpora from Indo-European, Turkic, Uralic, and Pama-Nyungan, but we have good reason to believe that the results port to other families as well. We begin by testing Turkish, Finnish, Hungarian, Uyghur, and Warlpiri with corpora ranging in size from roughly 28,000 to 400,000. The model achieves perfect results out-of-the-box on three of five languages tested. This includes successfully capturing secondary harmony in Turkish.

The Warlpiri result is encouraging because it demonstrates that the algorithm may be expected to perform on wordlists in only the tens of thousands. It is worth noting that the Turkish results are despite upwards of 30% of lexical items containing at least one harmony violation (due mostly to a set of bound morphemes with refuse to participate in harmony). This underlines the algorithm's robustness. The algorithm also appropriately maps vowels between harmonizing sets in all cases when features are provided as input.

In both Hungarian and Uyghur, all errors were of the same type: they misclassified harmonizing vowels as neutral. It is encouraging that the algorithm never places a vowel in the wrong harmonizing set. In that way, it does not lead the researcher astray as to the nature of the specific language's harmony system. A second test was performed on Hungarian in which vowel length (as acute accents) was removed from the orthography. After this change, Hungarian achieved a perfect score as well.

English and German were chosen as control cases. Neither language exhibits vowel harmony, and vowel harmony was discovered in neither language.

Finally, two experiments were conducted to test the limits of the algorithm's power. First, the algorithm was run on Estonian (Finnic), which crucially once had, but has since lost productive harmony (Harms, 1962). We predicted that such a language would still show remnants of harmony's distributional fingerprint. This turns out to be the case. Run on 87,586 types with no frequency information (Eesti Kirjandusmuuseumi Folkloristika Osakond, 2005), we discover remnant front-back harmony. ⟨a⟩, ⟨e⟩, ⟨i⟩, and ⟨u⟩ are found to be neutral which is unsurprising since

Language	# Types	Primary H?	V correct	Secondary H?	V correct
Turkish	303,013	✓	8/8	✓	4/4
Finnish	396,770	✓	8/8	–	–
Hungarian	53,839	✓	11/15	–	–
Uyghur	392,403	✓	7/8	–	–
Warlpiri	28,885	✓	3/3	–	–
German	225,327	–	5/5	–	–
English	101,438	–	6/6	–	–

Table 1: Vowel co-occurrences are taken from corpus orthographies. Marginal vowels (e.g. Finnish *å* and German *y*) are automatically detected and removed. Corpora are from MorphoChallenge (Kurimo et al., 2010) when available. Uyghur and Hungarian were provided for the DARPA LORELEI project. Warlpiri is from (Swartz, 1997).

we expect some erosion in the distributional asymmetries. The remaining vowels divide into two classes along frontness: ⟨ä⟩, ⟨ö⟩, ⟨ü⟩ vs. ⟨o⟩, ⟨õ⟩ (/ɤ/). This is system is reminiscent of productive harmony in Finnish today and provides interesting insight into diachronic study of the language.

Second, we test the limits of the input data. A run on the most frequent 500 types for Turkish successfully discovers primary but not secondary harmony. This seems to represent the lower bound on the algorithm's performance window. A second test on Buryat (Mongolic) confirmed this lower bound. Only spurious harmony was detected on a wordlist of 235 stems without frequencies (Ladefoged and Blankenship, 2007). We expect that wordlists containing inflected forms to perform better than lists of stems because harmony alternations are frequently observed as allomorphy.

Overall, these results are highly encouraging. On all but the smallest of wordlists, the algorithm produces sufficiently accurate results to be of use to linguists. It is even useful for diachronic study, uncovering lost harmony in Estonian. Future work will leverage morphological segmentation to achieve mapping without the linguist having to provide vowel features to the algorithm. We also expect to test on a variety of poorly documented languages once we acquire more wordlists.

3 Morphological Segmentation for Small Data

We now turn to our algorithm for unsupervised morphological segmentation. As with the vowel harmony algorithm, this is designed specifically with small corpora in mind. Nevertheless, it has achieved state-of-the-art performance on standard test languages. This algorithm is unique in leveraging the concept of paradigms in discovering segmentation. While the notion of paradigms is common in traditional morphology literature (Stump, 2001), it has not often been used in automatic segmentation tasks.

Automatic segmentation should be contrasted with automatic feature analysis. Segmentation is the division of whole word strings into morpheme subunits. It is a function only of form. Well-equipped segmentation processes such as the one presented here, permit transformations as well. These account for orthographic rules (e.g., the doubling on <n> in *running* → *run +n ing*) or theoretical concerns, such as allomorphy or stem alternations (e.g., Latin *paludis* from *palus* as *palus +d-s is*). Feature analysis, on the other hand, is a function of meaning. Words are annotated by the semantic contribution of each of their component morphemes rather than split into substrings. Feature analysis is not well suited for unsupervised tasks because it requires that the meaning of each morpheme be provided beforehand. It is not possible, from a wordlist alone at least, to deduce them. The combination of segmental and featural analysis yields enough information for a standard Leipzig gloss.

Segmentation

```
nd aka chi teng es a
```

Features

```
buy CAUS FV 1SM CL7OM PAST
```

Gloss

```
nd  -aka  -chi    -teng -es   -a
1SM -PAST -CL7OM -buy  -CAUS -FV
```

Figure 1: Gloss for the Shona (S-Bantu) verb *ndakachitengesa* 'I sold it' with accompanying segmental and featural analyses

While the paradigm approach is novel, other algorithms for morphological segmentation already exist. Probably the most famous is the Morfessor family of algorithms (Creutz and Lagus, 2005), which have come to be treated as a standard baseline. Also of note is the Morsel algorithm which achieved the best performance for English and Finnish at the most recent MorphoChallenge (Lignos, 2010).

The current published state-of-the-art, MorphoChain (Narasimhan et al., 2015) achieves top performance for English, Turkish, and Arabic. The algorithm gets its name from the way in which it conceives as words as derivational chains. Any word is composed of a stem plus some series of affixations and transformations. Then the probability of each step in a derivation is computed considering a number of features (in the machine learning sense, i.e., predictors). Most of these features are word-internal and can be computed directly from a wordlist. The would-be frequency of the proposed affix, whether or not the stem appears independently, correlations between affixes, and the scope of the required transformations are taken into account among other things.

Word-internal features alone push MorphoChain's performance above the competition. But to achieve its most impressive result, it falls back on word-external features as well. Cosine similarity between derivations in the chain is calculated via Word2Vec (Mikolov et al., 2013) feature vectors. The distributional similarities captures by Word2Vec approximate semantic and syntactic similarity. For example, *teach* and *teach-er* show high distributional similarity and are semantically related, while *corn* and *corn-er* are not distributionally similar and are not related. This information is useful for detecting spurious segmentations such as *corn-er* which would seem very plausible based on word-internal information alone. Word2Vec does not require labels and is thus unsupervised. However, corpora of *at least* hundreds of millions of words of running text are needed to calculate reliable vectors. Since these are on the order of a million times larger than the wordlists that we target, this subverts our goal.

The practical benefits of word segmentation models are more than simply theoretical. Our framework has been leveraged to improve the performance of the best systems for low-resource Uyghur machine translation as well as the automatic detection of named entities for the DARPA LORELEI project. Bridging the performance gap of NLP applications on languages without large annotated corpora, which is the vast majority, benefits the communities that speak them through greater accessibility of information.

3.1 Algorithm Overview

The morphological segmentation algorithm combines three processes: segmentation, paradigm construction, and pruning. An overview of each is provided. Iterative application of these three functions on an unannotated wordlist with frequencies yields segmentations as described in the introduction to this project.

Algorithm 2 Morphological Segmentation Algorithm Overview

```
if pre-segmentation then
    Split compounds
    Calculate initial $P(r)$ on attested frequencies
Perform initial segmentations
while iteration do
    Create paradigms
    Prune paradigms
return
```

3.1.1 Segmentation

Initial segmentation is achieved through a Bayesian model which estimates a probability $P(r, s, t)$ over candidate roots r, affixes s, and transformations t for each word. This process is itself iterative. 'Root' here refers to what remains after a single affix is removed from the word rather than its theoretically defined $\sqrt{ROOT}$. So after a single iteration, the Shona word *ndakachitengesa* with formal root $\sqrt{TENG}$ is segmented into root $ndakachitenges$ and affix a. The algorithm acknowledges that affixation often triggers additional phonological (or at least orthographical) processes. These are captured through the transformations *deletion*, *substitution*, and *duplication* at segmentation boundaries. Only considering transformations at segmentation boundaries drastically reduces the search space but prevents us from considering Semitic templatic morphology, for example. We are working to incorporate this functionality into future versions of the algorithm.

The likelihood $P(r, s, t|w)$ of a segmentation (r, s, t) out of all possible candidate segmentations (r', s', t') given a word w is 0 if that segmentation

- **Deletion** (DEL) of the end letter x of root r. E.g., the word 'using' as `(use, ing, DEL-e)`.
- **Substitution** (SUB) of the end letter x of root r with z. E.g., the word 'carries' as `(carry, es, SUB-y+i)`.
- **Duplication** (DUP) of the end letter x of root r. E.g., the word 'stopped' as `(stop, ed, DUP+p)`.

Figure 2: Description of transformations

cannot describe the word. Otherwise, it is defined as follows:

$$P(r, s, t|w) = \frac{P(r) \times P(s) \times P(t|f(r, s))}{\sum_{(r',s',t') \in w} P(r', s', t')}$$

This formula is based on the assumption that roots r and suffixes s are independent. Transformations, however, are conditioned on a function $f(r, s)$ defined over the entire corpus. For example, the -*ed* suffix in English deletes any immediately preceding e in the root. The f function allows these generalizations to be captured as morphological rules rather than individual cases.

Initially, each possible candidate segmentation is assumed to have equal probability $1/|(r, s, t) \in w|$. This generates spurious segmentations of atomic roots with high probability. Subsequent parameter estimation through paradigms and pruning removes spurious segmentations to yield a maximum likelihood segmentation.

$$(r, s, t|w) = \underset{(r',s',t')}{\text{argmax}} P(r', s', t')$$

As an additional step, the algorithm has the option of pre-segmenting compound words. If a word w contains two individually attested words w_1 and w_2, the word is split and both of its components are evaluated separately. This process is found to improve overall recall. As a final optional step, initial $P(r)$ can be weighted by frequency. In languages which attest bare roots, more frequent roots are more likely to be valid.

3.1.2 Paradigm Construction

The segmentation step yields a triple (r, s, t) for each word. Grouping triples by their root yields a set of suffixes, a *paradigm* enumerating attested affixes on the root. For example, the root *walk* surfaces with suffixes {-*s*,-*ing*,-*ed*, -*er*}. Note that many roots may share the same paradigm. For example, *stop*, and *bake* share a paradigm set with *walk*.

Paradigms are tabulated by the number of times each occurs in the proposed segmentations. We define a paradigm's *support* as its frequency in this calculation. We assume that more robustly supported paradigms are more likely to contain only valid affixes. Table 2 shows the most common paradigms discovered for English.

Paradigm	**Support**
(-ed, -ing, -s)	772
(-ed, -ing)	331
(-ed, -er, -ing, -s)	219
(-ly, -ness)	208
(-ed, -ing, -ion, -s)	154
(-ic, -s)	125

Table 2: Frequent English suffix paradigms contain valid affixes

Root	**Paradigm**
ticker	(-s, -tape)
piney	(-al, -apple, -a, -hill, -hurst, -ido, -iro, -wood)
corks	(-an, -screw, -well)
sidle	(-aw, -ed, -ee, -er, -ey, -in, -ine, -ing, -s)
lantz	(-anum, -ra, -ronix, -ry)
nadir	(-adze, -la, -ne, -s)
reith	(-a, -er, -ian)
bodin	(-ce, -es, -etz, -ford, -ly, -ne, -ng, -ngton)
musee	(-euw, -s, -um)
taiyo	(-iba, -uan)
bilge	(-er, -rami, -s)

Table 3: Ten suffix paradigms only supported by a single root. These contain many spurious affixes.

3.1.3 Paradigm Pruning

Not all proposed paradigms are real. Some are the result of segmentation errors. For example, if *closet* is segmented as (*close, t, NULL*), then it will yield a paradigm {-*er*, -*est*, -*ed*, -*ing*, -*s*, ***-t***}. Identifying such spurious paradigms directs the algorithm towards potentially spurious affixes.

To avoid such spurious paradigms, we perform pruning. First, paradigms with support $\geq$ 3 and at least two members are retained as well-supported. All other paradigms are considered suspect. We identify which component affixes of these paradigms receive the most support from other paradigms. The score of a set of affixes S is described as follows. It is the combined frequency of each member affix across all paradigms.

$$score(S) = \sum_{s \in S} freq(s)$$

For example, the {*-er*, *-est*, *-ed*, *-ing*, *-s*} subset of the spurious *closet* paradigm has the highest possible score of all subsets, and is a well-supported paradigm in and of itself. Therefore we can discard *-t* with reasonable confidence.

The probability $P(s)$ of each suffix is re-estimated over its attestation all the remaining paradigms. We then rerun the initial segmentation step. This time, spurious affixes probabilities have been penalized, so errors along the lines of (*close, t, NULL*) are less likely to reoccur. This improves algorithm precision by eliminating spurious affixation.

3.2 Experiments and Results

3.2.1 Data and Evaluation

To facilitate comparison between our algorithm and currently used competitors, we adopt the same data set used by (Narasimhan et al., 2015) and (Kurimo et al., 2010), the MorphoChallenge 2010 set for training and the combined MorphoChallenge 2005-2010 for testing. We test against Morfessor, as well as AGMorph (Sirts and Goldwater, 2013), MorphoChain-C (with only word-internal information), and the full MorphoChain model which is trained on (word-external) word embeddings from English Wikipedia and the BOUN corpus (Sak et al., 2008). MorphoChain-C presents the best direct comparison to our model since we do not consider word-external data. Nevertheless, our model outperforms both implementations.

The output of each algorithm is scored according to the MorphoChain metric. Precision, recall, and F1 are calculated across segmentation points. For example, *walking* segmented as *wal king* contains one incorrect segmentation point after *wal* and is missing a correct segmentation point in *king*. However, *walk ing* contains a correct segmentation point and no incorrect ones.

First, we report the contribution from each of our algorithm's component processes. Table 4 shows the results. *Base* is only the initial Bayesian segmentation step. *Suff* extends the Bayesian model with a final re-estimation based on suffix frequencies. *Trans* implements the above plus segmentation. *Comp* extends *Trans* with compound splitting. *Prune* implements paradigm pruning as well, and *Freq* considers root frequencies in initial segmentation.

	English			**Turkish**		
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**
Base	0.414	0.725	0.527	0.525	0.666	0.587
Suff	0.490	0.648	0.558	0.617	0.621	0.619
Tran	0.524	0.757	0.619	0.589	0.726	**0.650**
Comp	0.504	0.843	0.631	0.581	0.727	0.646
Prun	0.709	0.784	0.744	0.652	0.518	0.577
Freq	0.804	0.764	**0.784**	0.715	0.467	0.565

Table 4: Contribution of each algorithm component

Segmentation on English performs the best when all optional processes are enforced in the model. This is not the case for Turkish however, which achieves its peak performance at *Trans*. This discrepancy has to do with the relative availability of bare roots in the languages. English has limited affixation, and it can be assumed that many bare roots will appear in any reasonable English dataset. This is not the case for Turkish, however, in which nearly all noun and verb tokens have at least one affix. Therefore, processes which rely on the presence of unaffixed bare roots in the corpus cannot function.

Lang.	**Model**	**Prec.**	**Recall**	**F1**
English	Morfessor-Base	0.740	0.623	0.677
	AGMorph	0.696	0.604	0.647
	MorphChain-C	0.555	0.792	0.653
	MorphChain-All	0.807	0.722	0.762
	Our model	0.804	0.764	**0.784**
Turkish	Morfessor-Base	0.827	0.362	0.504
	AGMorph	0.878	0.466	0.609
	MorphChain-C	0.516	0.652	0.576
	MorphChain-All	0.743	0.520	0.612
	Our model	0.589	0.726	**0.650**

Table 5: All numbers except for ours are reported in (Narasimhan et al., 2015). All scoring was performed using the MorphoChain metric. Best results are reported for English and Turkish.

3.3 Testing on Other Languages

While our segmentation algorithm is unsupervised, scoring its output requires human-segmented data to compare against. This presents a challenge when testing the process

Paradigm	Support
(-al)	54
(-ly)	43
(-idad)	32
(-ismo)	29
(-ing, -s)	25
(-er)	23
(-ed)	21
(-ista)	19
(-ly, -s)	15
(-ed, -s)	13
(-er, -ing, -s)	11
(-er, -s)	11
(-ed, -ing)	11

Table 6: Foreign influence among the top 30 most frequent Tagalog suffix paradigms

on under-documented languages for which segmented wordlists are hard to come by. The DARPA LORELEI project had originally planned to produce human-annotated segmentations for a wide range of languages, but this has not yet come to fruition. In response, we are developing simple annotation standards which linguists and knowledgeable native speakers may use to produce segmented wordlists of their own.

We have run segmentation on Navajo, Tagalog, and Yoruba in additional to the languages reported above. Unfortunately, because no gold standards are available, the algorithm's performance cannot be assessed quantitatively. Additionally, the DARPA LORELEI and Wikipedia data which we tested on contain substantial English (and Spanish for Tagalog) vocabulary. Table 6 shows the effect this has on paradigms.

4 Discussion and Conclusion

The success of these unsupervised algorithms attests to what can be accomplished by gleaning information from small corpora. The vowel harmony detector describes harmony systems with high accuracy at as few as 500 types. The segmentation performs very well, on languages for which its accuracy could be assessed quantitatively, with corpora orders of magnitude smaller than what can be successfully processed by the next best method. Nevertheless, both algorithms are incomplete.

An obvious next step is to combine the two into a single pipeline. This is projected to have benefits for both. One inconvenience with the current iteration of the harmony detector is that it requires a linguist to provide features for each of the language's vowels if a harmony mapping is desired. It would be nice for this process to happen automatically instead. And here it is useful to leverage morphology. A language with affixal morphology and harmony should present two or more sets of paradigms which differ according to harmony. For example, in Turkish, the plural *-lar* occurs in paradigms with other back vowel allomorphs while *-ler* occurs with other front vowel allomorphs. There should exist a mapping between morphemes like *-lar* and *-ler* which differ only in their vowels. Combining the evidence across all of the morphemes will allow us to create harmony mappings without explicit reference to features.

There are improvements to be made in the other direction as well. Vowel harmony information will improve segmentation accuracy. As it is, the segmentation algorithm has no way of knowing that two paradigms which differ only due to harmony constraints are in fact two sets of phonologically determined allomorphs. This forces it to spread sparse evidence over an unnecessarily large number of paradigms. Understanding harmony will allow the algorithm to collapse these sets, calculate more accurate support, and therefore improve overall performance.

Harmony aside, a vital resource for estimating and improving the performance of the segmentation algorithm is the presence of morphologically segmented gold standards in a variety of languages. We have taken up this task and are working towards annotation conventions for this kind of work. So far, we are still in the early phases of the process but have already begun annotation on Faroese and Korean as test languages. These are useful languages for testing because they exhibit a variety of affixes and alternations that need to be accounted for. With an effective annotation standard, we can also proceed to develop a new scoring metric for segmentation. It will address concerns raised by the existing MorphoChallenge metric (Kurimo et al., 2007) due to the lack of a unified annotation standard.

Finally, we plan to extend the distributional approach to vowel harmony to other typological patterns. We are developing models to detect whether a given language exhibits stem alternations, tends to be agglutinative, has common prefixes, infixes, and/or suffixes, shows reduplication, etc. As with vowel harmony, these will aid linguists seeking to describe the grammars of undocumented languages. Additionally, they will prove useful as inputs to the segmentation algorithm.

Acknowledgments

We thank the rest of the University of Pennsylvania's LORELEI summer research team for their helpful insights. We also thank audiences at the 10th Northeast Computational Phonology Circle (NECPhon) and Penn's Computational Linguistics and Lunch series for their suggestions, as well as the anonymous reviewers for ComputEL-2 who gave us useful comments. This research was funded by the DARPA LORELEI program under Agreement No. HR0011-15-2-0023.

References

Adam C Baker. 2009. Two statistical approaches to finding vowel harmony. Technical report, Citeseer.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Eesti Kirjandusmuuseumi Folkloristika Osakond. 2005. Estnische sprichwörter.

Sharon Goldwater, Thomas L Griffiths, and Mark Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 673–680. Association for Computational Linguistics.

Robert Thomas Harms. 1962. *Estonian grammar*, volume 28. Indiana University.

David Harrison, Emily Thomforde, and Michael OKeefe. 2004. The vowel harmony calculator.

Mikko Kurimo, Mathias Creutz, and Matti Varjokallio. 2007. Morpho challenge evaluation using a linguistic gold standard. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 864–872. Springer.

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010. Morpho challenge competition 2005–2010: evaluations and results. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95. Association for Computational Linguistics.

Peter Ladefoged and B Blankenship. 2007. The ucla phonetics lab archive - buryat.

Constantine Lignos. 2010. Learning from unseen data. In *Proceedings of the Morpho Challenge 2010 Workshop*, pages 35–38.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *arXiv preprint arXiv:1503.02335*.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2008. Turkish language resources: Morphological parser, morphological disambiguator and web corpus. In *Advances in natural language processing*, pages 417–427. Springer.

Nathan Sanders and K David Harrison. 2012. Discovering new vowel harmony patterns using a pairwise statistical model. In *Poster presented at the 20th Manchester Phonology Meeting, University of Manchester*.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using adaptor grammars. *Transactions of the Association for Computational Linguistics*, 1:255–266.

Gregory T Stump. 2001. *Inflectional morphology: A theory of paradigm structure*, volume 93. Cambridge University Press.

Stephen Swartz. 1997. *Warlpiri yimi kuja karlipa wangka*. Summer Institute of Linguistics, Australian Aborigines and Islanders Branch, Warlpiri Translation Project.