Inferring Morphotactics from Interlinear Glossed Text: Combining Clustering and Precision Grammars

Olga Zamaraeva
University of Washington
Seattle, WA, USA
olzama@uw.edu

Abstract

In this paper I present a k-means clustering approach to inferring morphological position classes (morphotactics) from Interlinear Glossed Text (IGT), data collections available for some endangered and low-resource languages. While the experiment is not restricted to low-resource languages, they are meant to be the targeted domain. Specifically my approach is meant to be for field linguists who do not necessarily know how many position classes there are in the language they work with and what the position classes are, but have the expertise to evaluate different hypotheses. It builds on an existing approach (Wax, 2014), but replaces the core heuristic with a clustering algorithm. The results mainly illustrate two points. First, they are largely negative, which shows that the baseline algorithm (summarized in the paper) uses a very predictive feature to determine whether affixes belong to the same position class, namely edge overlap in the affix graph. At the same time, unlike the baseline method that relies entirely on a single feature, k-means clustering can account for different features and helps discover more morphological phenomena, e.g. circumfixation. I conclude that unsupervised learning algorithms such as k-means clustering can in principle be used for morphotactics inference, though the algorithm should probably weigh certain features more than others. Most importantly, I conclude that clustering is a promising approach for diverse morphotactics and as such it can facilitate linguistic analysis of field languages.

1 Introduction

Morphological analysis is a critical component in NLP systems for morphologically rich languages (Daille et al., 2002). Yet, while automatic morphological analysis may be well-developed for languages like English and Spanish, the list of these languages is rather short. There are at least two reasons for that. One is that high-resource languages offer big training corpora. This makes the use of various machine learning algorithms easier. Another reason is that many high-resource languages, most notably English, happen to feature fairly simple morphology. A morphological analyzer for a language like English does not need to model complex *morphotactics*, the constraints on the ordering of the morphemes types.

While there are many systems which are capable of segmenting words into morphemes (Creutz and Lagus, 2006; Johnson, 2008) and some systems which include more sophisticated morphological analyzers and use supervised machine learning for some tasks (Pasha et al., 2014), there do not seem to be many systems out there which can actually infer morphotactics in an unsupervised fashion. Yet many languages exhibit complex morphotactics. Furthermore, most of the world's languages are low-resource, meaning that there are few digitized resources that can be used in computational projects. Many are also understudied, meaning that the properties of the language including its morphotactics are not well-documented or well-understood.

Documenting morphological rules of understudied languages which often also have endangered status is of critical importance for the purposes of both linguistic research and cultural diversity conservation efforts (Krauss, 1992). At the same time, the scarcity of data makes many modern learning approaches that rely on big data inapplicable in this domain. However, field linguists who work on these languages have small sized but richly annotated data, Interlinear Glossed Text (IGT), and so the richness can be leveraged to compensate for the modest size of the corpora. An example of IGT from Chintang [ctn][1] is given below as (1):

(1) unisaŋa khatte
u-nisa-ŋa khatt-e
3sPOSS-younger.brother-ERG.A take-IND.PST
mo kosiʔ moba
mo kosi-i mo-pe
DEM.DOWN river-LOC DEM.DOWN-LOC
'The younger brother took it to the river.' [ctn] (Bickel et al., 2013)

[1] Spoken in Nepal.

Proceedings of the 14th Annual SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology, pages 141–150,
Berlin, Germany, August 11, 2016.

I take an existing approach to automatically extracting morphological rules from IGT as the baseline (Wax, 2014) and present a k-means clustering approach to the same problem. I evaluate the results by morphological parsing (analyzing a list of verbs by finding for each verb a sequence of morphological rule applications that would produce this form) on several languages from different language families, including some low-resource languages. I show that grammars obtained using k-means are generally worse than the baseline though they can be on par with it in a particularly noisy setting. K-means still strongly outperforms a grammar hand-built by language experts because automated processing ensures better recall. I notice that, unlike the baseline approach, k-means is capable of picking up non-canonical phenomena like circumfixation. I conclude that unsupervised classification methods like k-means clustering can help the field linguists come up with more complete hypotheses about morphotactics (accounting for more affixes and more relationships between them) and also discover non-canonical morphological phenomena in their data.

2 Background

This section briefly explains the theoretical assumptions about morphology that are used in this paper, looks at related work, presents the evaluation framework, and finally goes over the baseline system.

2.1 Canonical and Non-Canonical Morphotactics

Position classes, or *morphotactics*, are slots for groups of morphemes (affixes) that are in complementary distribution. The slots can have strict or variable ordering, and an affix that attaches to another affix is said to take the second affix as *input*. For example, Finnish [fin] is known to have the following order of position classes for finite verbs (Karlsson, 2008):

(2) *root* + *passive* + *tense/mood* + *person* + *particle*

Here, the root serves as input to the passive marker, the passive marker is input to the tense/mood marker, etc.

Canonical morphotactics, in Stump's (1993) terminology used also in works like Crysmann and Bonami (2015),[2] assume a strict ordering of position classes (for example, if the affix that means tense always follows the one that means aspect, as in Finnish above). Deviations from that which involve variable morpheme ordering can be called *non-canonical* morphotactics (Stump, 1993). Another type of non-canonical phenomena is circumfixation, when a prefix always comes together with a certain suffix, or in other words an affix can be said to split into two parts. For a more complete review of non-canonical phenomena, see Crysmann and Bonami (2015). Non-canonical morphotactics are found very often in the world's languages yet they are often overlooked in implemented systems which tend to be biased towards Indo-European and even just English characteristics (Bender, 2011).

2.2 Morphological Analysis in NLP

The big body of research about automatic morphological analysis that exists today is mostly not concerned with morphotactics. Automatic segmentation, which admittedly is a necessary step in any morphological analysis system, is probably the most developed area. In my study, I assume that the segmentation has already been done, and the goal is to capture relationships between groups of morphemes. There are approaches which advertise themselves as deep morphological analysis (Yarowsky and Wicentowski, 2000; Schone and Jurafsky, 2001), but they focus on well-studied and high-resource Indo-European languages, and mostly aim to learn a maximally broad-coverage table of mappings from stems and affix combinations to some linguistic tag (e.g. a Penn TreeBank POS tag). What they don't yield is a generative model of the language's morphology which would contain information about the position or inflectional classes.

Work that is most similar to mine in what it aims for is Oflazer and Gokhan (1996) and Oflazer, Nirenburg and McShane (2001). Oflazer and Gokhan (1996) use constraints to model morphotactics, but the constraints are hand-built and unsupervised learning is used only for segmentation. Oflazer, Nirenburg and McShane (2001) combine rule-based and machine learning techniques and include elicitation in the loop in order to build finite state automata for low-density languages.

[2] Cf. a broader term for canonical morphology as in Corbett (2009).

Their FSAs encode non-canonical morphotactic phenomena such as conditioning, and they induce morphological rules using transformation-based learning (Brill, 1995). Still, their approach focuses more on identifying affixes and roots than on paradigms and position classes, while the latter is necessary for the rules to become part of a morphological grammar.

2.3 Precision Grammars and Evaluation by Parsing

For evaluation, I use automatically generated precision grammars (Bender et al., 2008), a type of digitized language resource. A precision grammar consists of a lexicon and a hierarchy of lexical and phrasal rules written according to the HPSG theory of syntax (Pollard and Sag, 1994). The term 'precision' is meant to emphasize that any parse or generation by the grammar will comply with the rules and will in that sense be linguistically sound, or precise. The grammar is machine-readable. In combination with software such as the LKB system (Copestake, 2002), precision grammars can generate syntactic trees of complete feature structures[3] along with semantic representations. Lexical morphological rules apply first to construct words, and then phrasal rules apply to construct sentences. Such grammars are useful to evaluate the quality of linguistic analyses (Bender et al., 2008). In particular, I used precision grammars to evaluate my results by parsing.

I used the Grammar Matrix customization system (Bender et al., 2002; Bender et al., 2010) to compile precision grammars from the specifications which were output by either the baseline (Wax, 2014) or by my k-means system. In both cases, the morphotactics is represented internally as a directed acyclic graph (DAG) where nodes are affix types (position classes) and edges mean that one class serves as input to another. Cycles are not allowed mainly because of the internal Grammar Matrix restrictions, though iterating position classes are indeed rare.[4] The DAG implementation is provided entirely by the customization system, as are all the other functional parts of the grammar. The baseline and the k-means system supply only the specification for the DAG in form of nodes and edges. Below are a sample entry for a verb position class from a specification file (Figure 1) and the relevant snippet from the grammar itself, in HPSG-style (Pollard and Sag, 1994) type description language (Figure 2) (Copestake, 2000). The customization system reads in the specification file and, in this case, it would create a node in the DAG that corresponds to verb-slot1 (verb position class 1) and an edge to it from the stems node (called simply 'verb' in the figure).

```
verb-slot1_name=non-fin
verb-slot1_order=after
  verb-slot1_input1_type=verb
  verb-slot1_morph1_name=prp
  verb-slot1_morph1_orth=ing
    verb-slot1_morph1_feat1_name=aspect
    verb-slot1_morph1_feat1_value=prog
    verb-slot1_morph1_feat1_head=verb
    verb-slot1_morph1_feat2_name=form
    verb-slot1_morph1_feat2_value=prp
    verb-slot1_morph1_feat2_head=verb
```

Figure 1: Sample precision grammar specification for a file verb position class entry

```
prp-lex-rule := infl-lex-rule & non-fin-lex-rule-super &
  [ SYNSEM.LOCAL [ CONT.HOOK.INDEX.E.ASPECT prog,
                   CAT.HEAD.FORM prp ] ].
```

Figure 2: HPSG grammar snippet in type description language (Copestake, 2000)

For clarity, the examples are from a toy English grammar. The lexical rule which is illustrated will add a suffix *ing* to verbs to produce the participial form. This way a string like *walking* will be parsed and a feature structure will be produced which will capture the fact that this is a non-finite verb form, for example.[5]

2.4 Baseline: Inferring Position Classes DAG by Input Overlap

One approach to inferring the morphotactic DAG from IGT that has been tried is Wax (2014), and I use it as the baseline. The code for the baseline system was shared by its author. It was also used by Bender et al. (2014) in their set of experiments with automatically generated grammars of Chintang. Wax (2014) processes the input IGT (which already have segmentation and alignment between the language line and the gloss line) to identify the

[3] A feature structure is the form of linguistic unit description in HPSG. Feature structures can combine with each other by constraint unification to form phrase structures.

[4] Chintang [ctn] has them (Schikowski (2012) as cited in Bender et al. (2012)), but it may be one of very few languages with this feature.

[5] There would have to be a separate lexical rule for gerund, because the morphosyntactic constraints are distinct.

original affix types: affix instances which share the same orthography, gloss, and input relations. The original DAG is a function of these affix types, with affixes being nodes and input relations between them being directed edges. The system then takes a minimum input (edge) overlap value from the user (e.g. 80%, 50%, 20%) and compresses the graph according to this value, i.e. two nodes which share more than a certain percentage of edges will be considered the same position class. The principle is illustrated in the figures below on a toy Russian morphology graph which assumes an input of two verbs: *vy-kup-i-l* and *po-kup-a-et*.

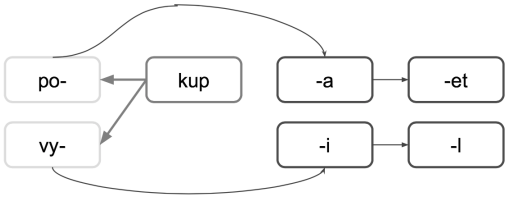

Figure 3: Two affix nodes (*po-* and *vy-*) are detected to have a 100% overlap (*kup*).

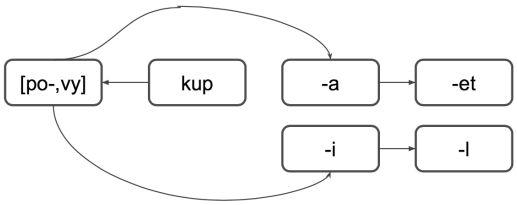

Figure 4: Two affix nodes are collapsed into one. Then the previous step is repeated with the next pair of affixes which share the minimum edge overlap.

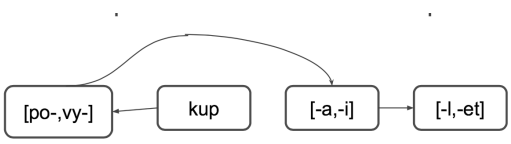

Figure 5: Eventually, the entire graph consists of classes of affixes, which can also be mapped from orthography to linguistic features through the IGT glossing.

Since the system will not allow a cycle in the graph, the compression is limited. If the system is trying to merge nodes A and B and one of B's edges would create a cycle if added to A's edges, such edge will not be added to A (it will there be lost). For example, the minimum number of nodes in the compressed graph of Chintang is 48 while the literature reports 13 position classes (Bickel et al., 2007). One advantage of the k-means approach is that it allows the user to pick any number of position classes directly, though a smaller number means more edges may be sacrificed.

Wax's (2014) system outputs a grammar specification where the lexicon and the morphology sections are filled out, and the rest of the settings are set to default. In particular, subject and object drop are allowed in all cases, and this makes it possible to parse stand-alone verbs. Then the specification is compiled into grammar rules by the Grammar Matrix (Bender et al., 2002; Bender et al., 2010) and this grammar can be used for parsing with software such as the LKB (Copestake, 2002) or ACE (Crysmann and Packard, 2012).

3 Data

Chintang

The most interesting results were obtained on the Chintang [ctn] data, possibly because it is the biggest and the highest quality IGT collection that I had. I used 8667 sentences for "training" (in this case to learn the morphological rules) and 708 verbs for testing by morphological parsing. The collection was used with permission from the field linguists who created it (Bickel et al., 2013). Chintang was shown to have free prefix ordering (Bickel et al., 2007) and is a morphologically rich agglutinative language. The position classes for Chintang are described in Bickel et al. (2007). Furthermore, Bender et al. (2012) hand-built an Oracle morphological precision grammar based on this description, accounting for certain phenomena such as position classes iteration. I used this grammar in evaluation.

Matsigenka

Another low-resource language that I used for this study was Matsigenka [mcb]. The IGT collection was again obtained from the field linguists who created it (Michael et al., 2013). I used a part of the collection which had English translations (376 sentences for training and 47 for testing, which results in 118 verbs for testing). Matsigenka is also an agglutinative, morphologically rich language with possibly variable morpheme ordering (Michael, p.c.).

ODIN: Turkish, Tagalog, and Russian

ODIN (Lewis and Xia, 2010) is a database of IGT obtained from crawling linguistic papers online. Though the particular languages which I used from ODIN are not low-resource, the datasets still represent noisy and scarce data. Because it comes from examples in linguistic papers, ODIN data is fairly homogeneous and not necessarily representative of natural language use, but it does provide a big selection of different languages IGT (currently 1497 datasets). For this experiment, I used three ODIN datasets: Turkish [tur], Tagalog [tgl], and Russian [rus]. Turkish and Tagalog can be seen as being on the opposite sides of the morphotactic spectrum: Turkish has many position classes but the morphotactics is mostly canonical, while Tagalog only basically has one prefix and one suffix position class but features infixes and reduplication.[6] In addition to Turkish and Tagalog, I used the Russian dataset from ODIN. Russian is a morphologically rich language with a few prefixal and a few suffixal position classes and a native speaker was available to perform qualitative error analysis,[7] so it was included for the diversity of the test set.

4 Method

The method and the evaluation process are illustrated in Figure 6 and described in the subsections below.

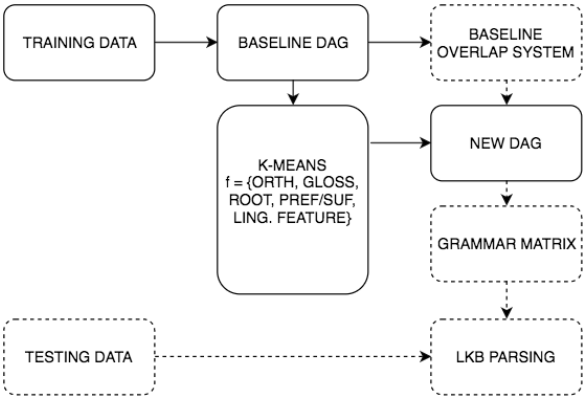

Figure 6: General method. The steps relevant only to evaluation are indicated by dotted lines.

[6]There is also a historical reason for using Turkish and Tagalog: Wax originally tested his system on them. However, he used the data in a different form and his original results are not directly comparable to mine.

[7]Results for Russian turned out to be uninteresting.

4.1 Training/Testing Split and the Effect of the Random Split on the Results

The Chintang and Matsigenka datasets were split randomly into training and testing for the purposes of other projects, and these were the splits that I used. The ODIN data I split myself. The split has a noticeable effect on the results. Namely, different splits result in a different number of position classes with the same minimum overlap value. Poor alignment between the language and the gloss line in the ODIN data leads to different items being discarded as the affix objects are added to the system, depending on which IGTs are originally in the training set.

There does not seem to be a strong correlation between the number of position classes that the baseline system comes up with and with either the number of IGT in the training set or the number of nodes in the original graph (Pearson coefficient between -0.08 and +0.13). The effect is probably just due to the randomness of the split. For all the three ODIN datasets, I report the numbers and analyze the results for the training-testing split which corresponds to a representative run. The representative run is one that resulted in the average value for the final number of position classes over 100 runs.

4.2 Affix Objects

The k-means system takes as input the original affix DAG created by Wax's (2014) system as described in section 2.4. The baseline system reads in the IGT and identifies affixes using the segmentation that is already in the IGT, the alignment with the gloss line, and a list of known glosses. The affixes then are stored in a DAG as nodes, and a directed edge between two nodes means that one affix can serve as input to the other. Stems are also part of the DAG, though they only have outgoing edges and no incoming edges. An affix instance is mapped to an existing node if it is associated with the same gloss, orthography, and inputs. Otherwise a new affix node is created. After the original DAG is built, instead of compressing the graph by using edge (input) overlap, I apply k-means clustering to merge the nodes based on a number of features described below.

4.3 Clustering

4.3.1 k-means

I used classical k-means clustering in form of a package for the Python programming language (Pedregosa et al., 2011)[8] with the following feature set where each feature is binary: affix's orthography, affix's gloss, linguistic feature type (e.g. tense, number), immediate context (previous morpheme, or the input, the only feature that the baseline also uses), the root of the word the affix occurred in, and whether the affix is a prefix (occurs to the left of the root). I run k-means on affix instances rather than on the DAG nodes, but each affix instance had been associated with a particular node in the DAG as described in the previous section. The nodes are then merged as described below.

4.3.2 Applying Clustering Labels to the Affix DAG

After a label is obtained for each affix instance, a new DAG is constructed as follows: I take a collection of new nodes, based on the clustering labels. Nodes from the old DAG for which the clustering label is the same are merged, with all inputs and outputs kept, regardless of cycles. If a certain node from the old DAG is associated with two different clusters, the node is split into two copy-nodes. Then a spanning tree is constructed by breadth-first search using the 'heaviest' outgoing edge for each node, where the weight is the outdegree of the node to which the edge is pointing. Then all other possible outgoing edges, sorted by weight and starting from the heaviest, are added for all nodes so long as they don't create a cycle in the graph. Some have to be sacrificed.

4.3.3 Choosing k

One goal was to evaluate the system using k equal to the number of position classes that the baseline system produces, so that they can be compared. Since the baseline system's result depends on the minimum overlap choice, that had to be fixed at a particular value. At the same time, for Chintang, there exists an Oracle precision grammar built by language experts (Bender et al., 2012). The baseline system is limited in how small of a graph it can produce. In particular, when run on the Chintang data, it produced a minimum of 48 position classes when input overlap is less or equal to 0.1, and 55 position classes with overlap = 0.2. Fifty-five is close to 54, the number of position classes in the Oracle grammar. Therefore I decided to use this 0.2 value for all languages to be able to compare the Oracle grammar to both the baseline system and the k-means system as well as to be consistent with respect to all other parts of the experiment. In addition, for Chintang I used k=13, the number which does not account for iterating affixes but is nonetheless the number that is hypothesized in the literature (Bickel et al., 2007).

[8] http://scikit-learn.org/stable/modules/clustering.html

5 Results and Discussion

5.1 Evaluation Method

It should be stated upfront that the results of this study seem most interesting if analyzed qualitatively, in terms of what kind of affixes get clustered together and whether this can be helpful to a filed linguist in any way. At the same time, it is appropriate to include quantitative results. For this, I use morphological parsing.

Morphological parsing is analyzing isolated words (e.g. extracted from a held-out test set) lexically, defaulting the phrase structure rules, in that each word (such as a verb) can be analyzed as a full sentence, provided there is a path in the morphotactic DAG that generates this word. This is an appropriate evaluation method given that labeled data for morphotactic inference virtually does not exist for most languages, be it high-resource or not. I am assuming that a grammar which achieves better coverage on a held out dataset may better represent the real grammar of the language, especially if k is kept modest.[9] The Chintang Oracle grammar I also use indirectly, looking at its performance in terms of morphological parsing and comparing to both the baseline and the k-means systems.

All grammars, including the Oracle, were normalized with respect to the lexicon and only differ in morphological rules. The test sets were filtered to just contain one instance of each verb. As such, the evaluation does not take into account how frequent the verbs are. The test sets for most languages are rather small (Chintang is the biggest with 708 unique verbs in the set). This is a realistic setting for low-resource language research.

[9] If k is very big, the grammar is likely to parse more but it cannot be easily mapped to the language's actual morphology.

Language	System	k/PC	% parsed
	Oracle	54	75.5
	Wax (2014)	55	**90.8**
ctn	k-means	55	86.1
	k-means	13	83.3
	Wax (2014)	23	**78.4**
mcb	k-means	23	56.8
	Wax (2014)	6	**67.9**
tgl	k-means	6	50.9
	Wax (2014)	21	53.4
tur	k-means	21	53.4
	Wax (2014)	5	**47.9**
rus	k-means	5	47.1

Table 1: Morphological parsing results.

5.2 Results

The results are summarized in Table 1. The results show that, in terms of morphological parsing, a k-means grammar is generally worse than the baseline system, though it can sometimes achieve similar coverage (in the noisy ODIN setting). However both the baseline and the k-means systems strongly outperform the hand-built Oracle grammar of Chintang. Furthermore, the resulting grammars can be examined by hand, not in terms of parsing but in terms of what they look like and how they compare to the languages' actual morphological rules. In case of Chintang at least, k-means clusters together affixes which constitute circumfixes, while the baseline grammar cannot possibly do that because it will never cluster together a prefix and a suffix.

5.3 Analysis and Discussion

Given largely negative results, the main points of this paper are given in qualitative linguistic analysis of concrete examples, mainly from the Chintang experiments. In most cases, the k-means algorithm and the baseline come up with different sets of morphological rules. While the baseline system clearly is better at parsing, Chintang and Matsigenka have examples which the k-means can parse and the baseline system cannot. That the baseline is usually better at parsing suggests that input overlap is an important feature and possibly the strongest predictor of whether two affixes belong to the same position graph. However, the k-means system is capable of picking up phenomena which the input overlap will never detect, because they are related to variable order and generally non-canonical phenomena. For such phenomena to be detected, the algorithm should consider features beyond the affix's immediate context. The clearest example of this is the Chintang circumfix *mai-/-yokt* which is consistently put in the same cluster by the k-means. Below I mostly talk about the Chintang results, as they provide the most insight into the difference between the baseline and the k-means.[10]

5.3.1 Chintang

Oracle Grammar versus Automatically Induced Grammars

In terms of morphological parsing, both the k-means morphological grammar and the baseline grammar clearly outperform the Oracle grammar. The main reason for this is that an automatic procedure which goes through the entire dataset in a consistent fashion picks up a lot more affixes than is described in the sources used by Bender et al. (2014). In part, that is because Chintang employs nominalization, compounding, and also features many phonological variations, but there are also indications that there are true verbal inflections that are missing in the Oracle. While the Oracle grammar cannot parse 158 items out of 708, the baseline only misses 65, and the k-means system misses 92. Examples of affixes which both automatic grammars pick up which the Oracle grammar misses include *-ʔ* (glossed EMPH), *-ko* (nominalizer), and, most interestingly, *-en*, which is glossed PST.NEG, so it is clearly an affix that has something to do with verb inflection and as such should probably have been included in the Oracle grammar but was missed for some reason. This suggests that either the description of the grammar in the literature is incomplete, or there are errors in the corpus which should be corrected. In either case, identifying verb inflections candidates automatically would be helpful for the field linguist who is working with the data.

Baseline Overlap=0.2 vs. k=55

The baseline system ended up compressing the original graph to a number of nodes similar to the Oracle number (55 instead of 54) when input overlap was set to 0.2. There are 7 items which the k-means system parses and the baseline grammar does not in this setting. A few of them, like *a-lis-a-hat-a-ce-e*, require that there be two nodes for

[10] Admittedly, more analysis could be done on Matsigenka. This remains part of future work.

the *-a* orthography, such that one takes the root (in this case *lis*) as input and the other takes the complementizer *hat*. The baseline grammar does not have an edge from the complementizer slot to the tense slot. There are 34 items which the baseline grammar parses and the k-means grammar does not. This is because the k-means ends up sacrificing more of the useful edges to avoid cycles in the graph. Neither grammar parses 58 items. Of these, some are due to unseen lexemes but most are due to discarded edges (since the baseline also discards edges when merging nodes).

The True Number of Position Classes: k=13

The most interesting (from the linguistic perspective) k-means result is the one with k=13. First of all, it is not possible to obtain this number using the baseline grammar, since the smallest number it produces is 48. Secondly, the resulting graph has some resemblance to Chintang morphotactics as described in the literature, and that can be seen more easily in a smaller graph. This means that the k-means system can be useful to a researcher who is trying to come up with hypotheses about the language's morphotactics and may have an idea about roughly how many position classes there are but not necessarily which affixes belong together and what the input relationships are. I evaluate this scenario with k=13, the number of position classes in Chintang suggested by Bickel et al. (2007).

There is some resemblance between the system's output and Chintang morphotactics as described by Bickel et al. (2007). An abridged version of the results is presented in Figure 7. Three of the clusters (not shown) are very heterogeneous and contain stems as well as different kinds of morphemes. These cannot be directly mapped to actual Chintang morphotactics, though they are useful in parsing compound verbs. There are a few clusters that k-means seems to get roughly right (all of them are in the figure), and some of the input edges (also in the figure) reflect actual Chintang morphotactics as well. One cluster, namely 9, has affixes that are clearly glossed as a verb inflection in the data (*3, 3s, 3p*) but are not accounted for in Bickel et al. (2007). One especially interesting cluster is the one presented in Figure 8. It captures the fact that *-yokt* and *mai-* behave as a circumfix, i.e. they tend to occur only together, one to the right and one to the left of the root. Clustering in this case is not necessarily helpful for parsing, but it is helpful for identifying morphotactic constraints generally.

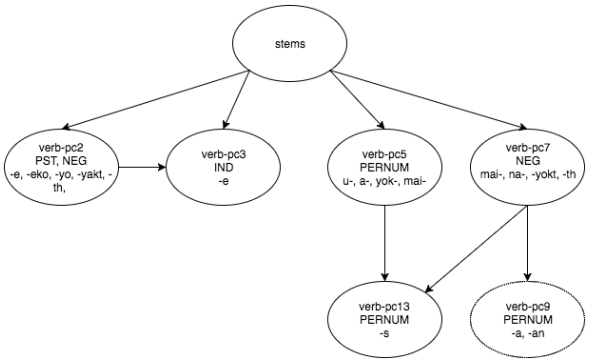

Figure 7: A part of the morphotactic graph output by k-means with k=13. Dotted ellipse (verb-pc9) shows a cluster which is not accounted for in the Chintang literature but seems plausible as a position class. All the rest included clusters at least roughly correspond to the position classes in the literature. Some clusters and edges are not shown.

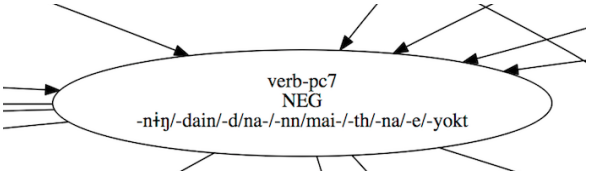

Figure 8: Chintang cluster containing circumfix *mai- -yokt.*

5.4 Matsigenka

While the k-means grammar lacks some productive edges that the overlap grammar has, it gains at least some others, which makes it possible for the k-means to parse *i-tsarog-a-i-t-an-ak-e*, since the k-means grammar does not lose the edge from *-i* class to *-t* class. With only one such example, it is difficult to conclude anything. No qualitative analysis of smaller Matsigenka graphs was done at this point. In future work, it will be possible to use a larger Matsigenka dataset, and hopefully the results will be more interesting.

5.5 ODIN Data

The ODIN datasets do not contain much variety, since the IGT come from linguistic papers' examples, and those tend to not be very diverse. At the same time, the ODIN data is rather noisy and often times it is not easy to align the gloss line to the language line. This way, many affixes never make it into the grammars and many items are not parsed. Interestingly, k-means comes closest to the baseline in this setting. The items that are parsed by both grammars are the ones that are seen in the data a lot and are therefore fairly simple for both systems to get. It seems that k-means could be

used on small and noisy field datasets, often as successfully as the baseline system, and the hope of discovering non-canonical phenomena will be higher.

6 Conclusion

The experiments described in this paper show that unsupervised methods such as clustering can be used somewhat successfully on smaller scale data such as field languages IGT collections. In case with Chintang at least, the clusters of affixes yielded by k-means sometimes roughly correspond to the position classes described in the literature. Both the baseline and the k-means systems are able to morphologically analyze (parse) more verbs than a hand-built grammar, which confirms that automatic processing is useful for field research.

Strict ordering of affixes that is easily accounted for by heuristic methods such as Wax (2014) is generally a very strong predictor for whether two affixes belong to the same position class or not. Systems that rely solely on inferring such ordering perform better than k-means in all the cases presented in this paper, but k-means achieves comparable results in noisy settings. Furthermore, approaches such as input overlap are by definition hopeless for discovering non-canonical morphotactics, while k-means seems to discover some correlations between positions that are conditioned on each other (e.g. Chintang *-yokt* and the negative prefixes). An improvement to the current approach would be weighted k-means, where immediate context (input) can be given more weight.

A system like the one described in this paper can be a useful component of an interactive linguistic analysis tool for field linguists. Kim (2015) showed that clustering results can be made more interpretable for humans in the education domain with the aid of Bayesian Case Modeling. It is possible that the same is applicable to the domain of field linguistics and morphological analysis. I showed that clusters suggest correlations between morphological features; designing a BCM-based interactive system where the linguist could guide the algorithm and look at automatically generated hypotheses in the process is a tempting direction for future work. As it is at present, k-means is a simple and extensible alternative to heuristic algorithms of inferring position classes from IGT and can serve as a stepping stone for developing expert linguistic analyses, as it can form preliminary buckets of affixes that can be considered candidates for either true position classes or for positions that are related to each other in some non-obvious way.

References

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The Grammar Matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In John Carroll, Nelleke Oostdijk, and Richard Sutcliffe, editors, *Proceedings of the Workshop on Grammar Engineering and Evaluation at the 19th International Conference on Computational Linguistics*, pages 8–14, Taipei, Taiwan.

Emily M Bender, Dan Flickinger, and Stephan Oepen. 2008. Grammar engineering for linguistic hypothesis testing. In *Proceedings of the Texas Linguistics Society X conference: Computational linguistics for less-studied languages*, pages 16–36. Citeseer.

Emily M Bender, Scott Drellishak, Antske Fokkens, Laurie Poulson, and Safiyyah Saleem. 2010. Grammar customization. *Research on Language & Computation*, 8(1):23–72. 10.1007/s11168-010-9070-1.

Emily M. Bender, Robert Schikowski, and Balthasar Bickel. 2012. Deriving a lexicon for a precision grammar from language documentation resources: A case study of Chintang. In *COLING*, pages 247–262.

Emily M. Bender, Joshua Crowgey, Michael Wayne Goodman, and Fei Xia. 2014. Learning grammar specifications from igt: A case study of chintang. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 43–53, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Emily M. Bender. 2011. On achieving and evaluating language-independence in NLP. *Linguistic Issues in Language Technology*, 6(3):1–26.

Balthasar Bickel, Goma Banjade, Martin Gaenszle, Elena Lieven, Netra Prasad Paudyal, Ichchha Purna Rai, Manoj Rai, Novel Kishore Rai, and Sabine Stoll. 2007. Free prefix ordering in Chintang. *Language*, pages 43–73.

Balthasar Bickel, Martin Gaenszle, Novel Kishore Rai, Vishnu Singh Rai, Elena Lieven, Sabine Stoll, G. Banjade, T. N. Bhatta, N. Paudyal, J. Pettigrew, M. Rai, and I. P. Rai. 2013. Tale of a poor guy. Accessed online on 15-January-2013.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational linguistics*, 21(4):543–565.

Ann Copestake. 2000. Appendix: Definitions of typed feature structures. *Natural Language Engineering*, 6(01):109–112.

Ann Copestake. 2002. The LKB system.

Greville G Corbett. 2009. Canonical inflectional classes. *Selected proceedings of the 6th Décembrettes: Morphology in Bordeaux*, pages 1–11.

Mathias Creutz and Krista Lagus. 2006. Morfessor in the morpho challenge. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*. Citeseer.

Berthold Crysmann and Olivier Bonami. 2015. Variable morphotactics in information-based morphology. *Journal of Linguistics*, pages 1–64.

Berthold Crysmann and Woodley Packard. 2012. Towards efficient hpsg generation for german, a non-configurational language. In *COLING*, pages 695–710.

Béatrice Daille, Cécile Fabre, and Pascale Sébillot. 2002. Applications of computational morphology. *Many morphologies*, pages 210–234.

Mark Johnson. 2008. Unsupervised word segmentation for sesotho using adaptor grammars. In *Proceedings of the Tenth Meeting of ACL Special Interest Group on Computational Morphology and Phonology*, pages 20–27. Association for Computational Linguistics.

Fred Karlsson. 2008. *Finnish: An essential grammar*. Routledge.

Been Kim. 2015. *Interactive and interpretable machine learning models for human machine collaboration*. Ph.D. thesis, Massachusetts Institute of Technology.

Michael Krauss. 1992. The world's languages in crisis. *Language*, 68(1):4–10.

William D. Lewis and Fei Xia. 2010. Developing ODIN: A multilingual repository of annotated language data for hundreds of the world's languages. *Literary and Linguistic Computing*, 25(3):303–319.

Lev Michael, Christine Beier, Zachary O'Hagan, Harold Vargas Pereira, and Jose Vargas Pereira. 2013. Matsigenka text written by Matsigenka authors. Accessed online on 15-September-2015: `http://www.cabeceras.org/ldm_publications/mcb_text_collection_30jun2013_v1.pdf`.

Kemal Oflazer and Gokhan Tur. 1996. Combining hand-crafted rules and unsupervised learning in constraint-based morphological disambiguation. *arXiv preprint cmp-lg/9604001*.

Kemal Oflazer, Sergei Nirenburg, and Marjorie McShane. 2001. Bootstrapping morphological analyzers by combining human elicitation and machine learning. *Computational linguistics*, 27(1):59–85.

Arfath Pasha, Mohamed Al-Badrashiny, Mona T Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *LREC*, pages 1094–1101.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Carl Pollard and Ivan A. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. The University of Chicago Press and CSLI Publications, Chicago, IL and Stanford, CA.

Robert Schikowski. 2012. Chintang morphology. *Unpublished ms, University of Zürich*.

Patrick Schone and Daniel Jurafsky. 2001. Knowledge-free induction of inflectional morphologies. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–9. Association for Computational Linguistics.

Gregory T Stump. 1993. Position classes and morphological theory. In *Yearbook of Morphology 1992*, pages 129–180. Springer.

David Wax. 2014. Automated grammar engineering for verbal morphology. Master's thesis, University of Washington.

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.