Adaptor Grammars for Unsupervised Paradigm Clustering

Kate McCurdy **Sharon Goldwater** **Adam Lopez**
School of Informatics
University of Edinburgh
kate.mccurdy@ed.ac.uk, {sgwater, alopez}@inf.ed.ac.uk

Abstract

This work describes the Edinburgh submission to the SIGMORPHON 2021 Shared Task 2 on unsupervised morphological paradigm clustering. Given raw text input, the task was to assign each token to a cluster with other tokens from the same paradigm. We use Adaptor Grammar segmentations combined with frequency-based heuristics to predict paradigm clusters. Our system achieved the highest average F1 score across 9 test languages, placing first out of 15 submissions.

1 Introduction

While the task of supervised morphological inflection has seen dramatic gains in accuracy over recent years (e.g. Cotterell et al., 2016, 2017, 2018; Vylomova et al., 2020), unsupervised morphological analysis remains an open challenge. This is evident in the results of the 2020 SIGMORPHON Shared Task 2 on Unsupervised Morphological Paradigm Completion, in which no submission consistently outperformed the baseline (Kann et al., 2020; Jin et al., 2020).

The 2021 Shared Task 2 (Wiemerslage et al., 2021) focuses on a subproblem from the 2020 task: given raw text input, cluster tokens together based on membership in the same *morphological paradigm*. For example, given the sentence "My dog met some other dogs", a successful system would assign "dog" and "dogs" to the same paradigm because they are two inflected forms of the same *lemma* "dog", while each other word would occupy its own cluster. Furthermore, a successful system needs to cluster typologically diverse, morphologically rich languages such as Finnish and Navajo, with inflectional paradigms which are much larger than English paradigms.

2 Adaptor Grammars

Our approach is based upon Adaptor Grammars, a framework which achieves state-of-the-art results on the related task of unsupervised morphological segmentation (Eskander et al., 2020).

2.1 Model

Adaptor Grammars (AGs; Johnson et al., 2007b) are a class of nonparametric Bayesian probabilistic models which learn structured representations, or parses, of natural language input strings. An AG has two components: a Probabilistic Context-Free Grammar (PCFG) and one or more adaptors. The PCFG is a 5-tuple $(N, W, R, S, \boldsymbol{\theta})$ which specifies a base distribution over parse trees. Parse trees are generated top-down by expanding non-terminals N (including the start symbol $S \in N$) to non-terminals N (excluding S) and terminals W, using the set of allowed expansion rules R with expansion probability θ_r for each rule $r \in R$. PCFGs have very strong independence assumptions; the adaptor component relaxes these assumptions by allowing certain nonterminals to *adapt* to a particular corpus, meaning they can cache and re-use subtrees with probabilities conditioned on that corpus.

An AG extends a PCFG by specifying a set of *adapted nonterminals* $A \subseteq N$ and a vector of adaptors $\boldsymbol{C}$. For each adapted nonterminal $X \in A$, the adaptor C_X stores all subtrees previously emitted with the root node X. When a new tree rooted in X is sampled, the adaptor C_X either generates a new tree from the PCFG base distribution or returns a previously emitted subtree from its cache. The adaptor distribution is generally based on a Pitman-Yor Process (PYP; Pitman and Yor, 1997), under which the probability of C_x returning a particular subtree σ is roughly proportional to the number of times X has previously expanded to σ. This leads to a "rich-get-richer" effect as more

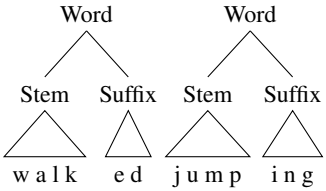
Proceedings of the Seventeenth SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology,pages 82–89

Word → Stem Suffix
Word → Stem
Stem → Chars
Suffix → Chars
Chars → Char
Chars → Char Chars

(a) Example grammar. Adapted nonterminals are underlined.

Word	Segmentation
walked	walk-ed
jumping	jump-ing
walking	walk-ing
jump	jump

(b) Toy corpus with target segmentations



(c) Example target morphological analyses, showing only the top 2 levels of structure

Figure 1: A possible morphological analysis (1c) learned by the grammar in (1a) over the corpus shown in (1b) (from Johnson et al., 2007b)

frequently sampled subtrees gain higher probability within the conditional adapted distribution. Given an AG specification, MCMC sampling can be used to infer values for the PCFG rule probabilities $\boldsymbol{\theta}$ (Johnson et al., 2007a) and PYP hyperparameters (Johnson and Goldwater, 2009).

2.2 AGs for Morphological Analysis

The probabilistic parses generated by adaptor grammars can be used to *segment* sequences. In cases where the grammar specifies word structures, the segmentations may reflect morphological analyses. For example, an AG trained with the simple grammar shown in Table 1a may learn to cache "jump" and "walk" as Stem subtrees, and "ing" and "ed" as Suffix subtrees, ideally producing the target segmentations shown in Figure 1c. In practice, researchers have successfully applied AGs to the task of unsupervised morphological segmentation (Sirts and Goldwater, 2013; Eskander et al., 2016). Eskander et al. (2020) found that a language-independent AG framework achieved state-of-the-art results on 12 typologically distinct languages.

3 System description

3.1 Overview

The task of unsupervised paradigm clustering is closely related to morphological segmentation, but we are not aware of previous applications of AGs to the current task. To use AGs for paradigm clustering, we need a method to group words together based on their AG segmentations. The example segmentations shown in Figure 1 suggest a very simple approach to paradigm clustering: assign all forms with the same *stem* to the same cluster. For example, "walked" and "walking" would correctly cluster together with the shared stem "walk". Our system builds upon this intuition.

As a preliminary step, we **select grammars** to sample from, looking only at the development languages. We build simple clusters and heuristically select grammars which show relatively high performance, as described in Section 3.2. In this case we select two grammars. Once the grammars have been selected, we discard the simple clusters in favor of a more sophisticated strategy.

We implement[1] a procedure to generate clusters for both development and test languages. First, we sample 3 separate AG parses for each corpus and each grammar, resulting in 6 segmentations for each word. We then use frequency-based metrics over the segmentations to identify the language's **adfix direction**, i.e. whether it is predominantly prefixing or suffixing, as described in Section 3.3. Finally, we iterate over the entire vocabulary and apply frequency-based scores to generate paradigm **clusters**, as described in Section 3.4 .

3.2 Grammar selection

An adaptor grammar builds upon an initial PCFG specification, and many such grammars can be applied to model word structure. As a first step, we evaluate various grammar specifications on the development languages and select the grammars for our final model.

To train the adaptor grammar representations, we use MorphAGram (Eskander et al., 2020), a framework which extends the adaptor grammar implementation of Johnson et al. (2007b). Eskander et al. (2020) evaluated nine different PCFG grammar specifications on the task of unsupervised word segmentation. Each grammar specifies the range of possible word structures which can be learned under that model. We evaluated six of their nine proposed grammars on the development languages (Maltese, Persian, Portuguese, Russian, and Swedish). Following their procedure, we extracted a vocabulary V of word types as AG inputs.[2]

[1] https://github.com/kmccurdy/paradigm-clusters

[2] Although AGs can also model token frequencies (Goldwater et al., 2006), which could conceivably improve perfor-

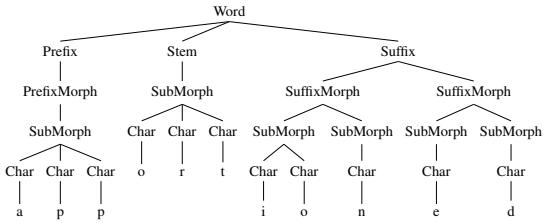
To evaluate grammar performance, we follow the intuition in Section 3.1 and group by AG-segmented stems. Grouping by stem can be more difficult for complex words. For example, an AG with a more complex grammar might segment the plural noun "actionables" into "action-able-s", with "action" as the stem (see also the example in Figure 2a); however, the target paradigm for clustering includes only "actionable" and "actionables", not "action" and "actions". To address this issue for our clustering task, we make the further simplifying (but linguistically motivated; e.g. Stump, 2005, 56) assumption that inflectional morphology is generally realized on a word's periphery, so a segmentation like "action-able-s" implies the stem "actionable" (in a suffixing language like English, where the prefix is included in the stem). As all of the development languages were predominantly suffixing (with the partial exception of Maltese, which includes root-and-pattern morphology), we simply grouped together words with the same AG-segmented Prefix + Stem.

We selected two grammars with the following desirable attributes: 1) they reliably showed good performance on the development set, relative to other grammars; and 2) they specified very similar structures, making it easier to combine their outputs in later steps. Both grammars model words as a tripartite Prefix-Stem-Suffix sequence. Both grammars also use a SubMorph level of representation, which has been shown to aid word segmentation (Sirts and Goldwater, 2013), although we only consider segments from the level directly above SubMorphs in clustering. The full grammar specifications are included in Appendix A.

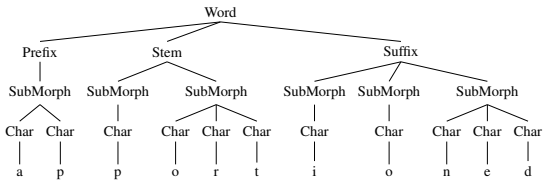
- *Simple+SM*: Each word comprises one optional prefix, one stem, and one optional suffix. Each of these levels can comprise one or more SubMorphs.

- *PrStSu+SM* Each word comprises zero or more prefixes, one stem, and zero or more suffixes. Each of these levels can comprise one or more SubMorphs. Eskander et al. (2020) found that this grammar showed the highest performance in unsupervised segmentation across the languages they evaluated.

Sampling from an adaptor grammar is a non-deterministic process, so the same set of initial parameters applied to the same data can predict different segmentation outputs. Given this variability, we run the AG sampler three times for each of our two selected grammars, yielding 6 parses of the lexicon for each language. The number of grammar runs was heuristically selected and not tuned in any way, so adding more runs for each grammar might improve performance (for example, Sirts and Goldwater, 2013, use 5 samples per grammar). We then combine the resulting segmentations using the following procedure.



(a) PrStSu+SM



(b) Simple+SM

Figure 2: Two example parses of the word "apportioned" from our two distinct grammar specifications, learned on the English test data.

3.3 Adfix direction

The first step is to determine the *adfix direction* for each language, i.e. whether the language uses predominantly prefixing or suffixing inflection. We heuristically select the adfix direction using the following automatic procedure.

First, we count the frequency of each *peripheral segment* across all 6 parses of the lexicon. A peripheral segment is a substring at the start or end of a word, which has been parsed as a segment above the SubMorph level in some AG sample. For instance, in the parse shown in Figure 2a, "app-" would be the initial peripheral segment, and "-ed" would be the final peripheral segment. By contrast, for the parse shown in Figure 2b,"ap-" would be the initial peripheral segment, and "-ioned" would be the final peripheral segment.

Next, we rank the segmented adfixes by their frequency, and select the top N for consideration, where N is some heuristically chosen quantity. In light of the generally Zipfian properties of linguistic

mance on this task, we did not explore this option.

distributions, we chose to scale N logarithmically with the vocabulary size, so $N = log(|V|)$.

Finally, we select the majority label (i.e. "prefix" or "suffix") of the N most frequent segments as the adfix direction. This simple approach has obvious limitations — to name just one, it neglects the reality of nonconcatenative morphology, such as the root-and-pattern inflection of many Maltese verbs. Nonetheless, it appears to capture some key distinctions: this method correctly identified Navajo as a prefixing language, and all other development and test languages as predominantly suffixing.

3.4 Creating paradigm clusters

Once we have inferred the adfix direction for a language, we use a greedy iterative procedure over words to identify and score potential clusters. Our scoring metric is frequency-based, motivated by the observation that inflectional morphology (such as the "-s" in "actionables") tends to be more frequent across word types relative to derivational morphology (such as the "-able" in "actionables"). Yarowsky and Wicentowski (2000) have demonstrated the value of frequency metrics in aligning inflected forms from the same lemma.

We start with no assigned clusters and iterate through the vocabulary in alphabetical order.[3] For each word w which has not yet been assigned to a cluster, we identify the most likely cluster using the following procedure.

Find possible stems Identify each possible stem s from all of the segmentations for w, where the "stem" comprises the entire substring up to a peripheral adfix. For example, based on the two parses shown in Figure 2, "apportion" and "apport" would constitute possible stems for the word "apportioned". The word w in its entirety is also considered as a possible stem.

Find possible cluster members For each stem s, identify other words in the corpus which might share that stem, forming a potential cluster c_s. A word potentially shares a stem if it shares the same substring from the non-adfixing-direction — so a stem is a shared prefix substring in a suffixing language like English, and vice-versa for a prefixing language like Navajo. For each word w_i that is identified this way, the rest of the string outside of the possible stem s is a possible adfix a_i. In the example from Figure 2, if "apportions" were also in the corpus, it would be added to the cluster for the stem "apportion", with "-s" as the adfix a_i. Similarly, it would also be considered in the cluster for the stem "apport", with adfix "-ions".

Score cluster members For each word w_i in c_s, calculate a score x_i:.

$$x_i = \sqrt{A_i^w} log(A_i) \quad (1)$$

where A_i is the normalized overall frequency of the ith adfix a_i (suffix or prefix) per 10,000 types in the corpus of 6 segmentations, and A_i^w is the proportion of segmentations of the ith word w_i which contain the adfix a_i. For example, if "apportioned" were in consideration for a hypothetical cluster based on the stem "apportion", A_i would be the normalized corpus frequency of "-ed", and A_i^w would be .5 (assuming only the two segmentations shown in Figure 2). For a cluster with the stem "apport", A_i would be the normalized frequency of "-ioned", and A_i^w would still be .5.

Intuitively, when evaluating a single word, Eq. 1 assumes that adfixes which appear frequently in the segmented corpus overall are more likely to be inflectional, so words with more frequent adfixes are more likely paradigm members (the $log(A_i)$ term). For instance, the high frequency of the "-s" suffix in English will increase the score of any word with an "-s" suffix in its segmentation (e.g. "apportion-s"). Eq. 1 also assumes that, for all segmentations of this particular word w_i, adfixes which appear in a higher proportion of segmentations are more reliable (the $\sqrt{A_i^w}$ term), so the more times some AG samples the "apportion-s" segmentation, the higher the score for "apportions" membership in the "apportion"-stem paradigm. The square root transform was selected based on development set performance, and has not been tuned extensively.

Filter and score clusters For each possible stem cluster c_s, filter out words whose score x_i is below the *score threshold* hyperparameter t, to create a new cluster c'_s. Calculate the cluster score x_s by taking the average of x_i for only those words in c'_s, i.e. only words with score $x_i \geq t$. The value for t is selected via grid search on the development set. We found that setting $t = 2$ maximized F1 across the development languages as a whole.

Select cluster Select the potential cluster c'_s with the highest score, and assign w to that cluster, along with each word w_i in c'_s.

[3] The method is relatively insensitive to order, except reversed alphabetical order, which is worse for most languages.

Language	Precision	Recall	F1
Maltese	.30	.30	.30
Persian	.54	.52	.53
Portuguese	.92	.91	.91
Russian	.83	.82	.82
Swedish	.85	.81	.83
Mean	.69	.67	.68

Table 1: Performance on development languages

Language	Precision	Recall	F1
Basque	.33	.32	.32
Bulgarian	.83	.80	.82
English	.91	.90	.90
Finnish	.61	.60	.60
German	.79	.79	.79
Kannada	.82	.59	.69
Navajo	.43	.42	.42
Spanish	.85	.82	.84
Turkish	.74	.73	.73
Mean	.70	.66	.68

Table 2: Performance on test languages

4 Results and Discussion

Performance was evaluated using the script provided by the shared task organizers. Table 1 shows the results for the development languages, and Table 2 shows the results for the test languages. While the average F1 score ends up being quite similar for both development and test languages, it's clear within both groups that there are large differences in performance across different languages.

4.1 Error analysis and ways to improve

Noncontiguous stems The clustering method described in Section 3.4 makes an unjustifiably strong assumption that stems are contiguous substrings, which effectively eliminates its ability to represent nonconcatenative morphology. This limitation contributes to the low score on Maltese, a Semitic language which includes root-and-pattern morphology for certain verbs. The model further assumes that the left or right edge of a word — the side opposite from the adfix direction — is contiguous with the stem. This leads to errors on German, as most verbs have a circumfixing past participle form "ge- + -t" or "ge- + -en". For example, the model correctly assigns "ändern", "änderten", and "ändert" to the same cluster, but incorrectly assigns "geändert" to a separate cluster. We estimate that roughly 30% of the model's incorrect German predictions stem from this issue. This limitation also contributed to our model's poor performance on Basque, which, like Maltese, uses both prefixing and suffixing inflection to express polypersonal agreement.[4]

One obvious way to improve this issue would be to use an extension of the AG framework which can represent nonconcatenative morphology. Botha and Blunsom (2013) present such an extension, replacing the PCFG with a Probabilistic Simple Range Concatenating Grammar. They report successful results for unsupervised segmentation on Hebrew and Arabic. On the other hand, it's unclear whether such a nonconcatenative-focused approach could also adequately represent concatenative morphology. Fullwood and O'Donnell (2013) explore a similar framework, using Pitman-Yor processes to sample separate lexica of roots, templates, and "residue" segments; they find that their model works well for Arabic, but much less well for English. In addition, Eskander et al. (2020) report state-of-the-art morphological segmentation for Arabic using the *PrStSu+SM* grammar which we also use here. Their findings suggest that, rather than changing the AG framework, we might attempt a more intelligent clustering method based on noncontiguous segmented subsequences rather than contiguous substrings.

Irregular morphology The strong assumption of contiguous substrings as stems also hinders accurate clustering of irregular forms of any kind, from predictable stem alternations (such as umlaut in German and Swedish, or theme vowels in Portuguese and Spanish) to more challenging suppletive forms such as English "go"-"went". The latter likely requires additional input from semantic representations, but semiregular alternations in forms could also be handled in principle by a more intelligent clustering process. On this point, we note that some small but significant fraction of AG parses of Portuguese verbs grouped verbal theme vowels and inflections together (e.g. parsing "apresentada" as "apresent-ada" rather than "apresenta-da", "apresentarem" as "apresent-arem" rather than "apresenta-rem", and so on), and these parses were crucial to our model's relatively high performance on Portuguese.

[4]We thank an anonymous reviewer for bringing this to our attention.

Derivation vs. inflection Another issue is that the parses sampled by AGs do not distinguish between inflectional and derivational morphology. This is apparent in Figure 2, where both grammars parse "apportioned" with "-ioned" as the suffix. We seek to address this issue with frequency-based metrics in our clustering method, but frequent derivational adfixes often score high enough to be assigned a wrong paradigm cluster. For example, in English our model correctly clusters "allow", "allows", and "allowed" together, but it also incorrectly assigns "allowance" to the same cluster.

A straightforward way to handle this within our existing approach would be to allow language-specific variation of the score threshold t. As we had no method for unsupervised estimation of t for unfamiliar languages, we did not pursue this; however, a researcher who had minimal familiarity with the language in question might be able to select a more sensible value for t based on inspecting the clusters. Beyond that, the distinction between inflectional and derivational morphology is an intriguing and contested issue within linguistics (e.g. Stump, 2005), and the question of how to model it computationally requires much more attention.

4.2 Things that didn't work

We attempted a number of unsupervised approaches beyond AG segmentations, with the goal of incorporating them during the clustering process; however, we could not consistently improve performance with any of them. It seems likely to us that these methods could still be used to improve AG-segmentation-based clusters, but we could not find immediately obvious ways to do this.

FastText As the AG framework only models word structure based on form, we hoped to use the distributional representations learned by FastText (Bojanowski et al., 2017) to incorporate semantic and syntactic information into our model's clusters. We tried several different approaches without success. 1) We trained a skipgram model with a context window of 5 words, a setting often used for semantic applications, in hopes that words from the same paradigm might have similar semantic representations. Agglomerative clustering on these representations alone yielded much worse clusters than the AG method, and we could not find a way to combine them successfully with the AG clusters. 2) Erdmann et al. (2020) trained a skipgram model with a context window of 1 word and a minimum subword length of 2 characters, and used it to cluster words from the same *cell* rather than the same paradigm (e.g. clustering together English verbs in the third person singular such as "walks" and "jumps"). We attempted to follow this procedure, but it proved too difficult, as paradigm cell information was not explicitly included in the development data for this shared task. 3) We used the method described by Bojanowski et al. (2017) to identify important subwords within a word, in hopes of combining them with AG segmentations. However, the identified subwords did not consistently align with stem-adfix segementations as we had hoped, and did not seem to provide any additional benefit.

Brown clustering Part of speech tags could provide latent structure as a higher-order grouping for paradigm clusters — for example, verbs would be expected to have paradigms more similar to other verbs than to nouns. Brown clusters (Brown et al., 1992) have been used for unsupervised induction of word classes approximating part of speech tags. We used a spectral clustering algorithm (Stratos et al., 2014) to learn Brown clusters, but they did not reliably correspond to part of speech categories on our development language data.

5 Conclusion

The Adaptor Grammar framework has previously been applied to unsupervised morphological segmentation. In this paper, we demonstrate that AG segmentations can be used for the related task of unsupervised paradigm clustering with successful results, as shown by our system's performance in the 2021 SIGMORPHON Shared Task.

We note that there is still considerable room for improvement in our clustering procedure. Two key directions for future development are more sophisticated treatment of nonconcatenative morphology, and incorporation of additional sources of information beyond the word form alone.

Acknowledgments

This work was supported in part by the EPSRC Centre for Doctoral Training in Data Science, funded by the UK Engineering and Physical Sciences Research Council (grant EP/L016427/1) and the University of Edinburgh, and a James S McDonnell Foundation Scholar Award (#220020374) to the second author.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Jan A Botha and Phil Blunsom. 2013. Adaptor grammars for learning non-concatenative morphology. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 345–356.

Peter F. Brown, Peter V. Desouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Géraldine Walther, Ekaterina Vylomova, Arya D. McCarthy, Katharina Kann, Sabrina J. Mielke, Garrett Nicolai, Miikka Silfverberg, David Yarowsky, Jason Eisner, and Mans Hulden. 2018. The CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection. In *Proceedings of the CoNLL–SIGMORPHON 2018 Shared Task: Universal Morphological Reinflection*, pages 1–27, Brussels. Association for Computational Linguistics.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, Gėraldine Walther, Ekaterina Vylomova, Patrick Xia, Manaal Faruqui, Sandra Kübler, David Yarowsky, Jason Eisner, and Mans Hulden. 2017. CoNLL-SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection in 52 Languages. In *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, pages 1–30, Vancouver. Association for Computational Linguistics.

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 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 10–22, Berlin, Germany. Association for Computational Linguistics.

Alexander Erdmann, Micha Elsner, Shijie Wu, Ryan Cotterell, and Nizar Habash. 2020. The Paradigm Discovery Problem. *arXiv:2005.01630 [cs]*. ArXiv: 2005.01630.

Ramy Eskander, Francesca Callejas, Elizabeth Nichols, Judith Klavans, and Smaranda Muresan. 2020. MorphAGram, Evaluation and Framework for Unsupervised Morphological Segmentation. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 7112–7122, Marseille, France. European Language Resources Association.

Ramy Eskander, Owen Rambow, and Tianchun Yang. 2016. Extending the Use of Adaptor Grammars for Unsupervised Morphological Segmentation of Unseen Languages. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 900–910, Osaka, Japan. The COLING 2016 Organizing Committee.

Michelle Fullwood and Tim O'Donnell. 2013. Learning non-concatenative morphology. In *Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*, pages 21–27, Sofia, Bulgaria. Association for Computational Linguistics.

Sharon Goldwater, Mark Johnson, and Thomas L Griffiths. 2006. Interpolating between types and tokens by estimating power-law generators. In *Advances in neural information processing systems*, pages 459–466.

Huiming Jin, Liwei Cai, Yihui Peng, Chen Xia, Arya D. McCarthy, and Katharina Kann. 2020. Unsupervised Morphological Paradigm Completion. *arXiv:2005.00970 [cs]*. ArXiv: 2005.00970.

Mark Johnson and Sharon Goldwater. 2009. Improving nonparameteric Bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado. Association for Computational Linguistics.

Mark Johnson, Thomas Griffiths, and Sharon Goldwater. 2007a. Bayesian Inference for PCFGs via Markov Chain Monte Carlo. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 139–146, Rochester, New York. Association for Computational Linguistics.

Mark Johnson, Thomas L. Griffiths, and Sharon Goldwater. 2007b. Adaptor Grammars:A Framework for Specifying Compositional Nonparametric Bayesian Models. In Bernhard Schölkopf, John Platt, and Thomas Hofmann, editors, *Advances in Neural Information Processing Systems 19*. The MIT Press.

Katharina Kann, Arya D. McCarthy, Garrett Nicolai, and Mans Hulden. 2020. The SIGMORPHON 2020 Shared Task on Unsupervised Morphological Paradigm Completion. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 51–62, Online. Association for Computational Linguistics.

Jim Pitman and Marc Yor. 1997. The Two-Parameter Poisson-Dirichlet Distribution Derived from a Stable Subordinator. *The Annals of Probability*, 25(2):855–900. Publisher: Institute of Mathematical Statistics.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-Supervised Morphological Segmentation using Adaptor Grammars. *Transactions of the Association for Computational Linguistics*, 1:255–266.

Karl Stratos, Do-kyum Kim, Michael Collins, and Daniel Hsu. 2014. A spectral algorithm for learning class-based n-gram models of natural language. *Proceedings of the Association for Uncertainty in Artificial Intelligence*.

Gregory T Stump. 2005. Word-formation and inflectional morphology. In *Handbook of word-formation*, volume 64, pages 49–71. Springer, Dordrecht, The Netherlands.

Ekaterina Vylomova, Jennifer White, Elizabeth Salesky, Sabrina J. Mielke, Shijie Wu, Edoardo Maria Ponti, Rowan Hall Maudslay, Ran Zmigrod, Josef Valvoda, Svetlana Toldova, Francis Tyers, Elena Klyachko, Ilya Yegorov, Natalia Krizhanovsky, Paula Czarnowska, Irene Nikkarinen, Andrew Krizhanovsky, Tiago Pimentel, Lucas Torroba Hennigen, Christo Kirov, Garrett Nicolai, Adina Williams, Antonios Anastasopoulos, Hilaria Cruz, Eleanor Chodroff, Ryan Cotterell, Miikka Silfverberg, and Mans Hulden. 2020. SIGMORPHON 2020 Shared Task 0: Typologically Diverse Morphological Inflection. In *Proceedings of the 17th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 1–39, Online. Association for Computational Linguistics.

Adam Wiemerslage, Arya McCarthy, Alexander Erdmann, Garrett Nicolai, Manex Agirrezabal, Miikka Silfverberg, Mans Hulden, and Katharina Kann. 2021. The SIGMORPHON 2021 Shared Task on Unsupervised Morphological Paradigm Clustering. In *Proceedings of the 18th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*. Association for Computational Linguistics.

David Yarowsky and Richard Wicentowski. 2000. Minimally Supervised Morphological Analysis by Multimodal Alignment. pages 207–216.

A PCFGs

Our system uses the following two grammar specifications, developed by Eskander et al. (2016, 2020). Nonterminals are adapted by default. *Non-adapted* nonterminals are preceded by 1, indicating an expansion probability of 1, i.e. the PCFG always expands this rule and never caches it.

A.1 Simple+SM

```
1 Word --> Prefix Stem Suffix

Prefix --> ^^^ SubMorphs
Prefix --> ^^^

Stem --> SubMorphs

Suffix --> SubMorphs $$$
Suffix --> $$$

1 SubMorphs --> SubMorph SubMorphs
1 SubMorphs --> SubMorph
SubMorph --> Chars
1 Chars --> Char
1 Chars --> Char Chars
```

A.2 PrStSu+SM

```
1 Word --> Prefix Stem Suffix

Prefix --> ^^^
Prefix --> ^^^ PrefMorphs
1 PrefMorphs --> PrefMorph PrefMorphs
1 PrefMorphs --> PrefMorph
PrefMorph --> SubMorphs

Stem --> SubMorphs

Suffix --> $$$
Suffix --> SuffMorphs $$$
1 SuffMorphs --> SuffMorph SuffMorphs
1 SuffMorphs --> SuffMorph
SuffMorph --> SubMorphs

1 SubMorphs --> SubMorph SubMorphs
1 SubMorphs --> SubMorph
SubMorph --> Chars
1 Chars --> Char
1 Chars --> Char Chars
```