Resource-light acquisition of inflectional paradigms

Radoslav Klíč, Jirka Hana

[1] Matfyz or Geneea
I.Ekeland@princeton.edu,
WWW home page: http://users/~iekeland/web/welcome.html
[2] Université de Paris-Sud, Laboratoire d'Analyse Numérique, Bâtiment 425,
F-91405 Orsay Cedex, France

Abstract: This paper presents a resource-light acquisition of morphological paradigms and lexicon for fusional languages. It builds upon Paramor [10], an unsupervised system, by extending it: (1) to accept a small seed of manually provided word inflections with marked morpheme boundary; (2) to handle basic allomorphic changes acquiring the rules from the seed and/or from previously acquired paradigms. The algorithm has been tested on Czech and Slovene tagged corpora and has shown increased F-measure in comparison with the Paramor baseline.

1 Introduction

Morphological analysis is used in many computer applications ranging from web search to machine translation. As [6] shows, for languages with high inflection, a morphological analyzer is an essential part of a successful tagger.

Modern morphological analysers based on supervised machine learning and/or hand-written rules achieve very high accuracy. However, the standard way to create them for a particular language requires substantial amount of time, money and linguistic expertise. For example, the Czech analyzer by [7] uses a manually created lexicon with 300,000+ entries. As a result, most of the world languages and dialects have no realistic prospect for morphological analyzers created in this way.

Various techniques have been suggested to overcome this problem, including unsupervised methods acquiring morphological information from an unannotated corpus. While completely unsupervised systems are scientifically interesting, shedding light on areas such as child language acquisition or general learneability, for many practical applications their precision is still too low. They also completely ignore linguistic knowledge accumulated over several millennia, often failing to discover rules that can be found in basic grammar books.

Lightly-supervised systems aim to improve upon the accuracy of unsupervised system by using a limited amount of resources. One of such systems for fusional languages is described in the paper.

Using a reference grammar, it is relatively easy to provide information about inflectional endings, possibly organized into paradigms. In some languages, an analyzer built on such information would have an acceptable accuracy (e.g., in English, most words ending in *ed* are past/-passive verbs, and most words ending in *est* are superlative adjectives). However, in many languages, the number of homonymous endings is simply too high for such system to be useful. For example, the ending *a* has about 19 different meanings in Czech [4].

Thus our goal is to discover inflectional paradigms each with a list of words declining according to it, in other words we discover a list of paradigms and a lexicon. But we do not attempt to assign morphological categories to any of the forms. For example, given an English corpus the program should discover that *talk, talks, talking, talked* are the forms of the same word, and that *work, push, pull, miss,...* decline according to the same pattern. However, it will not label *talked* as a past tense and not even as a verb.

This kind of shallow morphological analysis has applications in information retrieval (IR), for example search engines. For the most of the queries, users aren't interested only in particular word forms they entered but also in their inflected forms. In highly inflectional languages, such as Czech, dealing with morphology in IR is a necessity. Moreover, it can also be used as a basis for a standard morphological analyzer after labeling endings with morphological tags and adding information about closed-class/irregular words.

As the basis of our system, we chose Paramor [10], an algorithm for unsupervised induction of inflection paradigms and morphemic segmentation. We extended it to handle basic phonological/graphemic alternations and to accept seeding paradigm-lexicon information.

The rest of this paper is organized as follows: First, we discuss related work on unsupervised and semi-supervised learning. Follows a section about baseline Paramor model. After that, we motivate and describe our extension to it. Finally, we report results of experiments on Czech and Slovene.

2 Previous work

Perhaps the best known unsupervised morphological analysers are Goldsmith's Linguistica [5] and Morfessor [1, 2, 3] family of algorithms.

Goldsmith uses minimum description length (MDL; [12]) approach to find the morphology model which allows the most compact corpus representation. His Linguistica software returns a set of *signatures* which roughly correspond to paradigms.

Unlike Linguistica, Morfessor splits words into morphemes in a hierarchical fashion. This makes it more suitable to agglutinative languages, such as Finnish or Turkish, with a large number of morphemes per word. A probabilistic model is used to tag each morph as a prefix, suffix or stem. [9] improve the results of Morfessor by providing a small set (1000+ for English, 100+ for Finish) of correctly segmented words. While the precision slightly drops, the recall is significantly improved for both languages. [15] use handwritten rewrite rules to improve Morfessor's performance by recognising allomorphic variations.

The approaches by [16] and [13] aim at combining different information sources (e.g., corpus frequencies, edit distance similarity, or context similarity) to obtain better analysis, especially for irregular inflection.

A system requiring significantly more human supervision is presented by [11]. This system takes manually entered paradigm specification as an input and generates a finite-state analyser. The user is then presented with words in a corpus which are not accepted by the analyser, but close to an accepted form. Then the user may adjust the specification and the analyser is iteratively improved.

[8, 4] build a system which relies on a manually specified list of paradigms, basic phonology and closed-class words and use a raw corpus to automatically acquire lexicon. For each form, all hypothetical lexical entries consistent with the information about the endings are created. Then competing entries are compared and only those supported by the highest number of forms are retained. Most of the remaining entries are still non-existent; however, in the majority of cases, they licence the same inflections as the correct entries, differing only in rare inflections.

3 Paramor

Our approach builds upon Paramor [10], another unsupervised approach for discovery of inflectional paradigms.

Due to data sparsity, not all inflections of a word are found in a corpus. Therefore Paramor does require full paradigms, but instead works with partial paradigms, called *schemes*. A scheme contains a set of c(andidate)-suffixes and a set of c(andidate)-stems inflecting according to this scheme. The corpus must contain the concatenation of every c-stem with every c-suffix in the same scheme. Thus, a scheme is uniquely defined by its c-suffix set. Several schemes might correspond to a single morphological paradigm, because different stems belonging to the paradigm occur in the corpus in different set of inflections.

The algorithm to acquire schemes has several steps:

1. Initialization: It first considers all possible segmentations of forms into candidate stems and endings.

2. Bottom-up Search: It creates schemes by joining endings that share a large number of associated stems.

3. Scheme clustering: Similar schemes (as measured by cosine similarity) are merged.

4. Pruning: Schemes proposing frequent morpheme boundaries not consistent with boundaries proposed by a character entropy measure are discarded.

Paramor works with types and not tokens. Thus it is not using any information about the frequency or context of forms. Below, we describe some of the steps in more detail.

3.1 Bottom-up Search

In this phase, paramor performs a bottom-up search of the scheme lattice. It starts with schemes containing exactly one c-suffix. For each of them, Paramor ascends the lattice, adding one c-suffix at a time until a stopping criterion is met. C-suffix selected for adding is the one with the biggest c-stem ratio. (Adding a c-suffix to a scheme reduces number of the stems and the suffix reducing it the least is selected. C-stem ratio is ratio between number of stems in the candidate higher-level scheme and the current scheme.) When the highest possible c-stem ratio falls under 0.25, search stops.

3.2 Scheme Clustering

Resulting schemes are then subjected to agglomerative bottom-up clustering. To determine proximity of two clusters, sets of words generated by the clusters are measured by cosine similarity.[1] Cluster generates a set of words which is the union of sets generated by the schemes it contains. In order to be merged, clusters must satisfy some conditions, e.g. for any two suffixes in the cluster, there must be a stem in the cluster which can combine with both of them.

3.3 Pruning

After the clustering phase, there are still too many clusters remaining and pruning is necessary. In the first pruning step, clusters which generate only small number of words are discarded. Then clusters modelling morpheme boundaries inconsistent with letter entropy are dropped.

4 Our Approach

4.1 Overview

We have modified the individual stems in Paramor's pipeline in order to use (1) a manually provided seed of inflected words divided into stems and suffixes; and (2) to take into account basic allomorphy of stems. Figure 1 shows phases of Paramor on the left with dashed boxes representing our alterations.

[1] $\text{proximity}(X,Y) = \frac{|X \cap Y|}{\sqrt{|X||Y|}}$

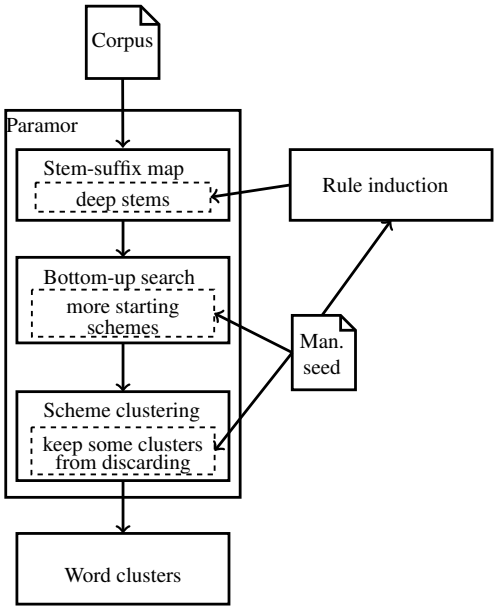

Figure 1: Altered Paramor's pipeline

In the bottom-up search phase and the scheme cluster filtering phase, we use manually provided examples of valid suffixes and their grouping to sub-paradigms to steer Paramor towards creating more adequate schemes and scheme clusters. The data may also contain allomorphic stems, which we use to induce simple stem rewrite rules. Using these rules, some of the allomorphic stems in the corpus can be discovered and used to find more complete schemes.

4.2 Scheme seeding

The manual seed contains a simple list of inflected words with marked morpheme boundary. A simple example in English would be:

talk+0, talk+s, talk+ed, talk+ing
stop+0, stop+s, stopp+ed, stopp+ing
chat+0, chat+s, chatt+ed, chatt+ing

This can be written in an abbreviated form as:

talk, stop/stopp, chat/chatt + 0, s / ed, ing

The data are used to enhance Paramor's accuracy in discovering the correct schemes and scheme clusters in the following way:

1. In the bottom-up search, Paramor starts with single-affix schemes. We added a 2-affix scheme to the starting scheme set for every suffix pair form the manual data belonging to the same inflection. Note that we cannot simply add a scheme containing all the suffixes of the whole paradigm as many of the forms will not be present in the corpus.
2. Scheme clusters containing suffixes similar to some of the manually entered suffix sets are protected from the second phase of the cluster pruning. More precisely, a cluster is protected if at least half of its schemes share at least two suffixes with a particular manual suffix set.

4.3 Allomorphy

Many morphemes have several contextually dependent realizations, so-called allomorphs due to phonological/-graphemic changes or irregularities. For example, consider the declension of the Czech word *matka* 'mother' in Table 1. It exhibits stem-final consonant change (palatalisation of *k* to č) triggered by the dative and local singular ending, and epenthesis (insertion of *-e-*) in the bare stem genitive plural.

Case	Singular	Plural
nom	mat**k**+a	mat**k**+y
gen	mat**k**+y	mat**ek**+0
dat	mat**c**+e	mat**k**+ám
acc	mat**k**+u	mat**k**+y
voc	mat**k**+o	mat**k**+y
loc	mat**c**+e	mat**k**+ách
inst	mat**k**+ou	mat**k**+ami

Table 1: Declension of the word *matka* "mother". Changing part of the stem is in bold.

Paramor ignores allomorphy completely (and so do Linguistica and Mofessor). There are at least two reasons to handle allomorphy. First, linguistically, it makes more sense to analyze *winning* as *win+ing* than as *winn+ing* or *win+ning*. For many applications, such as information retrieval, it is helpful to know that two morphs are variants of the same morpheme. Second, ignoring allomorphy makes the data appear more complicated and noisier than they actually are. Thus, the process of learning morpheme boundaries or paradigms is harder and less successful.

This latter problem might manifests itself in Paramor's bottom-up search phase: a linguistically correct suffix triggering a stem change might be discarded, because Paramor would not consider stem allomorphs to be variants of the same stem and c-stem ratio may drop significantly. Further more, incorrect c-suffixes may be selected.

For example, suppose there are 5 English verbs in the corpus: *talk, hop, stop, knit, chat*, together with their *-s* (*talks, hops, stops, knits, chats*) and *-ing* (*talking, hopping, stopping, knitting, chatting*) forms. Let's already have a scheme {*0, s*} with 5 stems. Adding *ing* would decrease number of stems to 1, leaving only *talk* in the scheme. C-stem ratio would be 0.2 and *ing* would not be accepted. Moreover, incorrect c-suffixes as *ping* and *ting* have c-stem ratio 0.4 and may be accepted.

However, for most languages the full specification of rules constraining allomorphy is not available, or at least

is not precise enough. Therefore, we automatically induce a limited number of simple rules from the seed examples and/or from the scheme clusters obtained from the previous run of algorithm. Such rules both over and undergenerate, but nevertheless they do improve the accuracy of the whole system. For languages, where formally specified allomorphic rules are available, they can be used directly along the lines of [15, 14]. For now, we consider only stem final changes, namely vowel epenthesis (e.g., *matk-a – matek-0*) and alternation of the final consonant (e.g., *matk-a – matc-e*). The extension to other other processes such as root vowel change (e.g., English *foot – feet*) is quite straightforward, but we leave with for future work.

Stem change rule induction and application Formally, the process can be described as follows. From every pair of stem allomorphs in the manual input, $s\delta_1, s\delta_2$, where s is their longest common initial substring,[2] with suffix sets f_1, f_2 we generate a rule $*\delta_1 \rightarrow *\delta_2$ / (f_1, f_2) and also a reverse rule $*\delta_2 \rightarrow *\delta_1$ / (f_2, f_1). Notation $*\delta_1 \rightarrow *\delta_2$ / (f_1, f_2) means "transform a stem $x\delta_1$ into $x\delta_2$ if following conditions hold:"

1. $x\delta_2$ is a c-stem present in the corpus.
2. C-suffix set f_1^x (from the corpus) of the c-stem $x\delta_1$ contains at least one of the suffixes from f_1 and contains no suffix from f_2.
3. C-suffix set f_2^x of the c-stem $x\delta_2$ contains at least one of the suffixes from f_2 and contains no suffix from f_1.

Induced rules are applied after the initialisation phase. So-called *deep* stems are generated from the c-stems. A deep stem is defined as a set of surface stems.

To obtain a deep stem for a c-stem t, operation of *expansion* is applied. Expansion works as a breadth-first search using a queue initialised with t and keeping track of the set D of already generated variants. While the queue is not empty, the first member is removed and its variants found by application of all the rules. (Result of applying a rule si non-empty only if the rule is applicable and its right hand side is present in the corpus.) Variants which haven't been generated so far are added to the back of the queue and to D. When the queue is emptied, D becomes the deep stem associated with t and all other members of D.

Bottom-up search and all the following phases of Paramor algorithm are then using the deep stems instead of the surface ones.

Stem change rule induction from scheme clusters In addition to deriving allomorphic rules from the manual seed, we also use a heuristic for detecting stem allomorphy in the scheme clusters obtained from the previous run of algorithm. Stem allomorphy might increases the sparsity problem and might prevent Paramor to find some paradigms. However, if the stem changes are systematic and frequent, Paramor does create the appropriate scheme clusters. However, it considers the changing part of the stem to be a part of suffix.

As an example, consider again the declension of the Czech word *matka* "mother" in Table 1. Paramor's scheme cluster with suffixes *ce, ek, ka, kami, kou, ku, ky, kách, kám* has correctly discovered 9 of 10 paradigm's suffixes,[3] but fused together with parts of the stem. Presence of such scheme cluster in the result is a hint that there may be a *c/k* alteration and epenthesis in the language.

First phase of the algorithm for deciding whether a scheme cluster with a c-suffix set f is interesting in this respect is following:

1. If f contains a c-suffix without a consonant, return *false*.
2. c_c = count of unique first consonants found in c-suffixes in f.
3. If $c_c > 2$ return *false*. If $c_c = 1$ and f doesn't contain any c-suffix starting with a vowel, return *false*.
4. Return *true*.

If a scheme cluster passes this test, each of its stems' subparadigms is examined. Subparadigm for stem s consists of s and f_s – all the c-suffixes from f with which s forms a word in the corpus. For example, let's have a stem $s = mat$ with $f_s = \{ce, ek, ka, ku, ky\}$. Now, the morpheme boundary is shifted so that it is immediately to the right from the first consonant of the original c-suffixes. In our example, we get 3 stem variants: *matk + a, u, y*, *matc + e*, *matek + 0*. To reduce falsely detected phonological changes, we check each stem variant's suffix set whether it contains at least one of the c-suffixes that Paramor has already discovered in other scheme clusters. If the condition holds, rules with same syntax as the manual data are created. For example, *matk / matc / matek + a, u, y / e / 0*. All generated rules are gathered in a file and can be used in the same way as the manual seed or just for the induction of phonological rules.

5 Experiments and results

We tested our approach on Czech and Slovene lemmatised corpora. For Czech, we used two differently sized subsets of the PDT 1 corpus. The first, marked as **cz1**, contains 11k types belonging to 6k lemmas. The second, **cz2**, has 27k types and 13k lemmas. The Slovene corpus **si** is a subset of the jos100k corpus V2.0 (`http://nl.ijs.si/jos/jos100k-en.html`) with 27k types and 15.5k lemmas.

The manual seed consisted of inflections of 18 lemmas for Czech and inflections of 9 lemmas for Slovene.

[2] should δ_1 or δ_2 be 0, one final character is removed from s and prepended to δ_1 and δ_2

[3] Except for vocative case singular, which is rarely used.

In both cases, examples of nouns, adjectives and verbs were provided. They were obtained from a basic grammar overview. For Czech we also added information about the only two inflectional prefixes (negative prefix *ne* and superlative prefix *nej*). The decision which prefixes to consider inflectional and which not is to a certain degree an arbitrary decision (e.g., it can be argued that *ne* is a clitic and not a prefix), therefore it makes sense to provide such information manually.

5.1 Evaluation method

We evaluated the experiments only on types at least 6 characters long which Paramor uses for learning. That means 8.5k types and 4500 lemmas for **cz1**, 21k types and 10k lemmas for **cz2** and 21k types and 12k lemmas for **si**.

Since corpora we used do not have morpheme boundaries marked, we could not use the same evaluation method as authors of Paramor and Morfessor – measuring the precision and recall of placing morpheme boundaries. On the other hand, corpora are lemmatised and we can evaluate whether types grouped to paradigms by the algorithm correspond to sets of types belonging to the same lemma.

We use the following terminology in this section: a *word cluster* is a set of words returned by our system, a *word paradigm* is a set of words from the corpus sharing the same lemma, an *autoseed* is a seed generated by the heuristic described in Section 4.3.

Precision and recall of the word clusters can be computed in the following way: To compute precision, start with $p = 0$. For each word cluster, find a word paradigm with the largest intersection. Add the intersection size to p. Precision = p / total number of words. For computing recall, start with $r = 0$. For each word paradigm, find a word cluster with the largest intersection. Add the intersection size to r. Recall = r / total number of words. F1 is the standard balanced F-score.

5.2 Results

Results of the experiments are presented in the tables 2 – 4. We used the following experiment settings:

1. *no seed* – the baseline, Paramor was run without any seeding
2. *man. seed* – manual seed was used
3. *autoseed* – autoseed was used for induction of the stem change rules
4. *both seeds* – Paramor run with manual seed, stem change rules were induced from manual and autoseed.
5. *seed + pref.* – manual seed was used together with additional rules for two Czech inflectional prefixes, otherwise same as 2.
6. *both seeds + pref* – manual seed was used together with additional rules for two Czech inflectional prefixes, otherwise same as 4.

Experiment	Precision	Recall	F1
no seed	97.87	84.61	90.76
man. seed	97.96	87.52	92.44
autoseed	98.19	84.58	90.88
both seeds	97.96	87.52	92.44
seed + pref.	97.84	89.40	**93.43**
both seeds + pref.	97.84	89.40	**93.43**

Table 2: Results for the **cz1** corpus.

Experiment	Precision	Recall	F1
no seed	97.36	87.02	91.90
man. seed	97.04	89.30	93.01
autoseed	97.30	87.72	92.26
both seeds	96.78	89.30	92.89
seed + pref.	96.68	92.35	**94.46**
both seeds + pref.	96.31	92.49	94.36

Table 3: Results for the **cz2** corpus.

Experiment	Precision	Recall	F1
no seed	95.70	93.00	94.33
man. seed	95.62	94.44	95.02
autoseed	95.69	93.13	94.40
both seeds	95.56	94.76	95.16

Table 4: Results for the **si** corpus.

As can be seen from the results, the extra manual information indeed does help the accuracy of clustering words belonging to the same paradigms. What is not shown by the numbers is that more of the morpheme boundaries make linguistic sense because basic stem allomorphy is accounted for.

6 Conclusion

We have shown that providing a very little of easily obtainable information can improve the result of a purely unsupervised system. In the near future, we are planning to model a wider range of allomorphic alternations, try larger (but still easy to obtain) seeds and finally test the results on more languages.

References

[1] Mathias Creutz and Krista Lagus. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop*

on Morphological and phonological learning - Volume 6, MPL '02, pages 21–30, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[2] Mathias Creutz and Krista Lagus. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05)*, pages 106–113. Finland: Espoo, 2005.

[3] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4:3:1–3:34, February 2007.

[4] Anna Feldman and Jirka Hana. *A resource-light approach to morpho-syntactic tagging*. Rodopi, Amsterdam/New York, NY, 2010.

[5] John A. Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2):153–198, 2001.

[6] Jan Hajič. Morphological Tagging: Data vs. Dictionaries. In *Proceedings of ANLP-NAACL Conference*, pages 94–101, Seattle, Washington, USA, 2000.

[7] Jan Hajič. *Disambiguation of Rich Inflection: Computational Morphology of Czech*. Karolinum, Charles University Press, Praha, 2004.

[8] Jiri Hana, Anna Feldman, and Chris Brew. A resource-light approach to Russian morphology: Tagging Russian using Czech resources. In Dekang Lin and Dekai Wu, editors, *Proceedings of EMNLP 2004*, pages 222–229, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[9] Oskar Kohonen, Sami Virpioja, and Krista Lagus. Semi-supervised learning of concatenative morphology. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, SIGMORPHON '10, pages 78–86, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[10] Christian Monson. *ParaMor: From Paradigm Structure to Natural Language Morphology Induction*. PhD thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2009.

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

[12] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific Publishing Co, Singapore, 1989.

[13] Patrick Schone and Daniel Jurafsky. Knowledge-free induction of inflectional morphologies. In *Proceedings of the North American chapter of the Association for Computational Linguistics*, pages 183–191, 2001.

[14] Michael Tepper and Fei Xia. A hybrid approach to the induction of underlying morphology. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP-2008), Hyderabad, India, Jan 7-12*, pages 17–24, 2008.

[15] Michael Tepper and Fei Xia. Inducing morphemes using light knowledge. *ACM Trans. Asian Lang. Inf. Process.*, 9:3:1–3:38, March 2010.

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