Automatic morphological alignment and clustering

Jackson L. Lee
University of Chicago
jsllee@uchicago.edu

May 2, 2014

Abstract

This paper describes an unsupervised algorithm, with no language-specific knowledge, which takes a list of morphological paradigms and explores cross-paradigmatic structure in terms of two computational tasks: alignment and clustering. Based on complexity computation in a minimum description length approach, the proposed algorithm learns the relationship across the paradigms based purely on surface strings and formalizes the intuitive idea that, for instance, *jumping* and *loving* belong to the same morphological category – this is alignment. Moreover, the algorithm simultaneously learns morphological groupings of the paradigms akin to conjugation and declension classes – this is clustering. The clustering analysis also reveals more fine-grained hierarchical structure among the inflectional classes. The algorithm is applied to verbal paradigms from English and Spanish. The results are useful for further work on the unsupervised learning and prediction-oriented research of paradigmatic structure. We also show the value of computational techniques in linguistics for both explicitly evaluating competing analyses and rigorously implementing analyses.

Contents

1 Introduction

This paper is concerned with the learning of natural language morphology, research on which has grown quickly in the past two decades or so; see Goldsmith (2010); Hammarström and Borin (2011) for a recent survey; also Pertsova (To appear) for a general discussion on learning inflectional morphology.[1] Morphological paradigms are considered the central object of study (Matthews 1972; Bybee 1985; Carstairs 1987), in line with research under the rubric of Word and Paradigm Morphology (Hockett 1954; Blevins 2013; Spencer 2013). Concretely, we are interested in learning morphological structure from paradigmatic data as in (1) below.

(1)

jump	jumping	jumps	jumper	jumped
walked	walks	walker	walking	walk
moving	mover	moved	moves	move
loves	love	loving	mover	loved
⋮	⋮	⋮	⋮	⋮

If we allow a more charitable reading of "paradigms", they are potentially not restricted to inflectional paradigms and may include what is often considered derivational paradigms. For instance, (1) shows English verbal inflectional paradigms where the agentive *-er* form for each paradigm could be included for the purposes of this paper.

For the particular learning tasks of interest, to be explained in detail below, we begin by the paradigms as in (1) with several characteristics. Each row represents one paradigm from a lexeme (JUMP, WALK, MOVE, etc.; small caps denote lexemes in the linguistic convention). All rows have the same number of forms for distinct morphological realizations; there are five forms in each row in (1). Moreover, all rows have forms for the exact same morphosyntactic categories. In (1), it is always the same five morphosyntactic categories of English verbal paradigms in every row: the bare form, the third singular present sense with *-s*, the simple past typically with *-ed*, and the *-ing* form. Lastly, within a row, the different forms of the paradigm can be horizontally ordered in an arbitrary way.

Given a paradigm data set like (1), we ask if we are able to learn morphological structure algorithmically. Specifically, in this paper we are interested in two

[1]Primary thanks go to John Goldsmith, without whom this paper and related ongoing work would not have existed. I am grateful to Greg Kobele and Jason Riggle for their guidance. I have also benefited from the questions and comments from the members of the linguistic community at the University of Chicago as well as the audience at the 2nd American International Morphology Meeting at the University of California, San Diego, in November 2013. All errors and shortcomings are mine.

tasks: alignment and clustering. We propose an algorithm which learns the cross-paradigmatic structure based purely on surface strings and formalizes the intuitive idea that, for instance, *jumping* and *loving* belong to the same morphological category – this is alignment. Moreover, the algorithm simultaneously learns (i) morphological groupings of the paradigms akin to conjugation and declension classes, and (ii) the hierarchical patterns among these morphological groupings – this is clustering.

This paper is written with both the general linguistic and the more computationally inclined audiences in mind. On the one hand, with a large proportion devoted to the description and explanation of an algorithm, we describe a procedure to accomplish alignment and clustering in a detailed way that permits replication of results. On the other hand, linguistic considerations figure prominently in the nuts and bolts of the algorithm. Indeed, one may ask whether linguistics is necessarily computational.[2]

The structure of this paper is as follows. Section 2 describes the learning tasks of alignment and clustering. We justify and motivate the way we do what we do, as well as position this paper in relation to other related work in section 3. Section 4 explains in detail the alignment-clustering algorithm. Section 5 discusses results using English verbal paradigmatic data. Lastly, section 6 sketches several directions for further work, and section 7 concludes the paper.

2 Learning morphology algorithmically

This section describes the two unsupervised learning tasks of alignment and clustering, which we aim at accomplishing concurrently. They are unsupervised in the sense that the system does not possess knowledge of any gold standard or any language-specific knowledge. The purpose of this section is to present what alignment and clustering are as computational tasks, while their connection with linguistics, or why linguists should care, is in subsequent sections.

2.1 Task 1: Alignment

The first task which we hope to accomplish is alignment, in the sense that, for instance, *jumps* and *loves* should be aligned together because they belong to the same morphosyntactic category, whereas *jumps* and *loved* should not.

[2]The source code in Python and datasets are available online:
`https://github.com/JacksonLLee/morph-align-cluster`

(2) a. Correct alignment:

jump	jumps	jumping	jumped	jumper
|	|	|	|	|
love	loves	loving	loved	lover

b. Incorrect alignemnt:

jump	**jumps**	jumping	**jumped**	jumper
|	|	|	|	|
love	**loved**	loving	**loves**	lover

As alluded to above, the horizontal ordering of forms in a paradigm in (1) can be random, which may look like (2b) with a wrong alignment. The task of alignment, e.g., to obtain (2a) from (2b), is whether we are able to do so based purely on surface strings from the given data without reference to morphosyntax. The intuition is that forms of the same morphosyntactic category from different lexemes are similar on the surface in some way, e.g., *-ing* shared by *jumping* and *loving*.

2.2 Task 2: Clustering

The second task of automatic morphological learning is clustering. In many languages with inflectional morphology, paradigms tend to exhibit patterns which form groups, or *clusters*. For verbal paradigms, we call them conjugation classes. For paradigms of other parts of speech, we call them declension classes. In this paper, the goal of clustering is to algorithmically find out these conjugation/declension classes given some paradigmatic data.

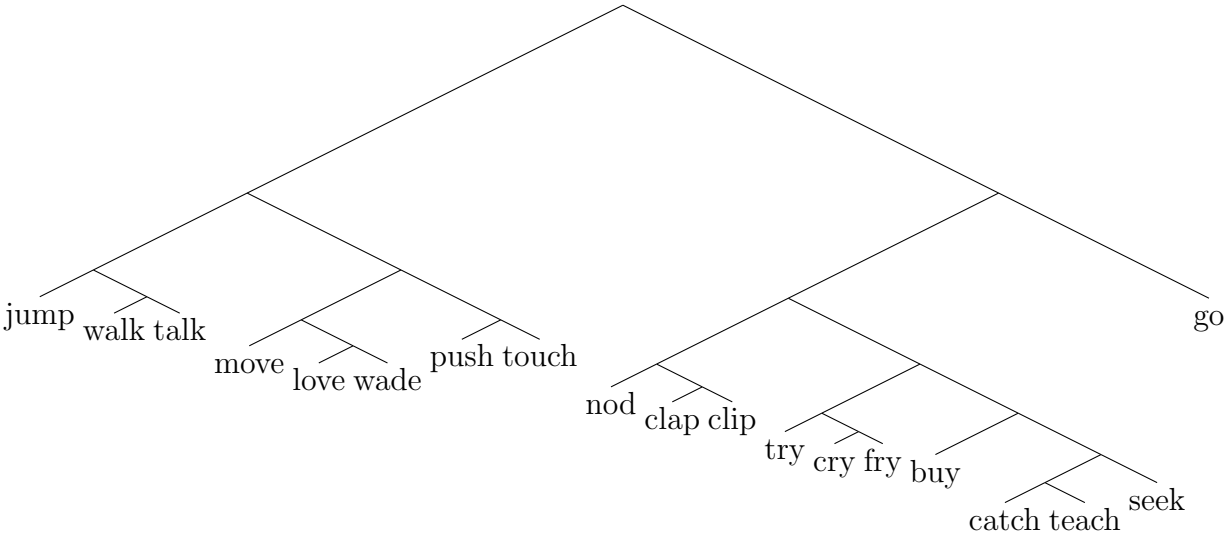
Arguably, English verbal paradigms in (1) also display conjugation classes (Bloch, 1947). For instance, there is a group of verbs which share the ablaut pattern in *sing-sang-sung, drink-drank-drunk*. If we allow a broader reading of 'conjugation' for English, then we may admit orthographical alternations as well, at least in the sense that how we write present-day English indeed reflects a non-contemporary version of English, or that the orthography shows what we observe in other languages with well-established inflectional classes. For (1) plus a few more paradigms, some 'conjugation classes' for English are as follows:

(3) Some English 'conjugation classes'

Property	**Paradigms**				
Regular	jump	jumping	jumps	jumped	jumper
	walk	walking	walks	walked	walker
With 'e'	move	moving	moves	moved	mover
	love	loving	loves	loved	lover
Consonant doubling	nod	nodding	nods	nodded	nodder
	clap	clapping	claps	clapped	clapper
'y/ie'	try	trying	tries	tried	tryer
	cry	crying	cries	cried	cryer
{*o*,*a*}*ught*	buy	buying	buys	bought	buyer
	catch	catching	catches	caught	catcher

As with the task of alignment described in the previous section, we propose to perform clustering based purely on surface strings. As a quick illustration whose details are to be discussed in subsequent sections, our algorithm produces the following hierarchical representation (very much simplified here) for an English data set similar to (1).

(4) An inheritance hierarchy for some English 'conjugation classes'



Not only do we attempt to look for the inflectional classes, we also infer higher-order structure, namely the hierarchical relationship among the inflectional classes. This is linguistically significant, because many inflectional morphological systems do not have entirely distinct string-based patterns across inflectional classes, although

the number of classes could logically be as many as there are lexemes; this observation is termed *paradigm economy* in Carstairs (1983, 1987). The different combinations of some small number of inflectional patterns are what result in the partial overlapping and similarity among inflectional classes.

3 Motivation

In this section, we explain why we are doing what we are doing, from the perspectives of grammar evaluation and algorithms, as well as position this present work in relation to other work. In particular, for clustering, we describe Haspelmath's (2002) work on inheritance hierarchies of Greek nominals in some detail as an intuitive, non-quantitative predecessor.

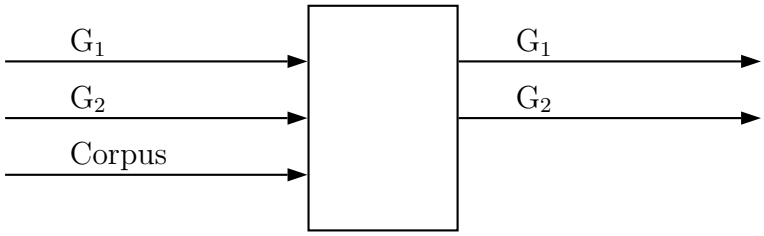
3.1 Grammar evaluation and algorithms

On the goal of linguistic theory, Chomsky (1957: ch.6) discusses several possibilities and argues for this particular one:

> "[G]iven a corpus and given two proposed grammars G_1 and G_2, the theory must tell us which is the better grammar of the language from which the corpus is drawn. In this case we might say that the theory provides an *evaluation procedure* for grammars." (Chomsky, 1957: 51; original emphasis)

This is graphically represented on the same page as follows:

(5) Chomsky (1957: 51) on the evaluation procedure of grammars



Early responses to this position appear to be somewhat cautious, but by no means negative. Garvin (1964: 36-37) wrote:

> "I do not wish to become embroiled in the ultimate issue raised, namely, whether an evaluation procedure is indeed the only reasonable goal for linguistics. In my opinion it is not.

> I should instead like to discuss a more immediate question: how to provide a practical evaluation procedure, where by practical I mean something that can actually be done in practice.
>
> It appears to me first of all that an evaluation procedure for grammars – that is, entire grammars – is rather a tall order, if the procedure is to be interpreted operationally. It seems to imply that two grammars (that is, two complete descriptions from phoneme – or morphoneme [sic] – to sentence) are to be compared to each other and to a corpus, in order to ascertain which is to be preferred. As to the criterion by which this is to be judged, let me quote Chomsky again: "Suppose that we use the word 'simplicity' to refer to the set of formal properties of grammars that we shall consider in choosing among them." For the choice to be feasible in practice, there should first of all exist two such grammars, each meeting what Chomsky calls "the external criteria of adequacy for grammars". Assuming such a condition, the two adequate grammars would then have to be compared in their entirety which, if taken seriously, might mean a comparison page by page, or statement by statement, or chapter by chapter. Each partial comparison may then result in a judgement of simplicity. If it is possible to weight each part judgement appropriately, one may assume that an overall judgement can be computed by some reasonable statistical operation. It is also thinkable that instead of this series of partial comparisons (which presupposes a matching of aprts that are not necessarily susceptible to clear-cut matches), one might take each grammar separately and by some procedure to be defined when available take an independent measure of simplicity. The two measures could then be compared and a final evaluation made.
>
> I wonder whether, at the present state of the art in linguistics, any of this is very practical.
>
> If an evaluation procedure for entire grammars is deemed impractical, under what conditions does an evaluation procedure in linguistics become practical, and is it then relevant to the objectives of linguistics? Let me give a categorical and somewhat unfashionable answer to both questions." (Garvin, 1964: 36-37)

Like Garvin, we attempt to explore and implement evaluation in linguistic analysis. If this is the way to do linguistics, the question we ask now is this: How exactly do we evaluate which grammar or analysis is the best among competing ones? The answer we adopt here is to incorporate the Minimum Description Length philosophy

(MDL; Rissanen 1989) in an algorithmic approach. There are two parts here, which have been argued for in linguistics, see Goldsmith (2011a) on MDL, and Goldsmith (2004) on algorithms.

For any grammar or analysis, we would like a way of formalizing how complex it is. One way is to compute its description length in terms of bits, that is, how many bits are required to encode an analysis. Furthermore, we also ask how good the analysis is for fitting the given data, and for this we have some way to measure it. The notion of the best analysis, then, is formalized as searching for the lowest total complexity. This is in line with the philosophy of MDL approaches in machine learning. For grammar selection, an MDL analysis asserts that the best analyis is one which minimizes the sum of the grammar complexity and the data cost given the grammar (Goldsmith 2011a and elsewhere). There are two major appealing aspects stemming from an MDL approach. First, MDL embodies Occam's razor. Minimizing grammar complexity is the computational analog to advancing the simplest analysis in theoretical linguistics. Second, MDL eschews over-fitting. In traditional theoretical linguistics, an often unnoticed assumption is the emphasis placed on accounting for *all* the given data points at all costs, typical in linguistic training, and consequently the reduced concern over increasing grammar complexity.[3] In a nutshell, the insight from MDL is this: you don't want to fit the data too well at the cost of a highly complex grammar, and at the same time you also don't want have a grammar too simple, one that fits the given data too poorly. MDL says that the best analysis is a trade-off between how complex the analysis is, on the one hand, and the goodness-of-fit by that analysis for the given data, on the other.

On algorithms, we show the value of using them as they provide a computationally explicit and falsifiable means to derive and test analyses. More often than not, in theoretical linguistics, the focus is the analysis but not *how* that analysis comes into being in the first place. This claim is supported by the way how linguists are typically trained: given a linguistic dataset, we are asked to come up with an analysis for a given question, but we are *never* asked to explicitly pin down the meticulous steps through which the analysis comes into being. The procedure which leads to an analysis is as important as, if not more, the analysis itself (Goldsmith, 2004). Such a procedure is an algorithm. An algorithmic approach is especially relevant in the computer age. With the high computational power right at our fingertips, an algorithm can easily run through a huge amount of data, perhaps from different languages. We shift our focus to the procedure resulting in an analysis and to the interpretation of the analysis.

[3]Some authors, e.g., Chan (2008: 55), contend that MDL still overfits the data, in the sense that every data point is accounted for, and that MDL finds the best fit for the *observed* data.

Going back to Garvin's (1964) lesson on the evaluation procedure in linguistic analysis, we take note of his concluding remarks:

(6) The three concluding points of Garvin's (1964: 42-43), in brief quotes
   a. "Evaluation is a necessary ingredient in the process of decision-making, which in turn is an essential part of the work of the analyst."
   b. "[W]hile an evaluation procedure for entire grammars requires the consideration of an unmanageable multiplicity of variables, one applied to partial solutions during individual steps of the analysis allows the control of all relevant conditions, since only a limited number of them has to be considered."
   c. "In order for an evaluation procedure to lead to a decision, it should be based on a criterion related to the final goal of the overall process of analysis within which a particular step is to be decided."

Our present work echoes Garvin's insights, as articulated in more general and modern terms in Goldsmith (2007). First, evaluation is at the heart of our proposed algorithm for morphological alignment and clustering (6a). Second, evaluation in our algorithm is performed quantitatively, tractably, and iteratively in a well-defined search space (6b). Third, evaluation proceeds as step-wise complexity minimization, with the final result being the best, simplest analysis in terms of complexity (6c).

3.2 On inflectional classes

This section explains why linguists should care about alignment and clustering as described in section 2. The linguistic relevance of alignment and clustering is *inflectional classes*, a well-known yet under-studied linguistic phenomenon. If inflectional morphology is what is relevant to syntax (Anderson, 1982), then the very existence of inflectional classes presents challenges to our understanding of language. Inflectional classes are the groupings, usually but not always arbitrary, of the lexemes of a given lexical category in terms of inflectional patterns; their existence is often attributed to diachronic reasons, see Dammel (2009) for a case study. A familiar example of inflectional classes is Spanish verbal morphology, which has three major groups— -AR, -ER, and -IR verbs— with no phonological, syntactic, or semantic basis. In the call for papers for the special session on "Inflectional Classes in the Languages of the Americas" at the 87$^{\text{th}}$ Annual Meeting of the Linguistic Society of America in 2013, the crux of the problem and the reason why we are interested in inflectional classes are aptly summarized:

> "Inflection classes are seemingly useless in functional terms, and yet they are widely found across languages and remarkably resilient over time. [...] Inflectional classes, as they resist a syntactic or phonological explanation, are in themselves an interesting object of study for a theory of language because they introduce into the linguistic system a layer of complexity which is purely morphological."[4]

A quick survey of morhpology textbooks and linguistics glossaries reveals that the treatment of and attention to inflectional classes are somewhat uneven (Jensen, 1990; Spencer, 1991; Trask, 1999; Haspelmath, 2002; Bauer, 2003, 2004; Matthews, 2007; Crystal, 2008; Haspelmath and Sims, 2010; Lieber, 2010; Aronoff and Fudeman, 2011). In more technical works, the situation is similar. In formal studies of inflectional morphology, so long as inflectional classes are not the focus of discussion, a popular treatment of inflectional classes is simply assign diacritic or class features of some sort to lexemes. For example, a binary feature such as [±strong] is used to distinguish strong and weak verbs in English in Distributed Morphology (Halle and Marantz, 1993).

3.2.1 The connection between inflectional classes, alignment, and clustering

This section underscores the connection between inflectional classes, alignment, and clustering. The discussion will be illustrated with the well-known Spanish conjugation classes:

(7) The Spanish present indicative suffixes

	1st conjugation	2nd conjugation	3rd conjugation
1.SG	-o	-o	-o
2.SG	-as	-es	-es
3.SG	-a	-e	-e
1.PL	-amos	-emos	-imos
2.PL	-áis	-éis	-ís
3.PL	-an	-en	-en

For alignment, we might take it for granted that *jumped* and *loved* are in the same category, for instance. This is so as long as we have access to all our knowledge, implicit or otherwise, of morphosyntactic distribution and meaning. This paper asks if

[4]From `http://linguistlist.org/callconf/browse-conf-action.cfm?ConfID=147727`

alignment can be done based purely on surface strings. Any answer proposed can be checked easily, because the gold standard is naturally available. Moreover, computationally, alignment is a follow-up step for learning cross-paradigmatic relationship in light of recent lines of research on the unsupervised learning of morphology, especially for work (e.g., Goldsmith 2000, 2001, 2006) that finds individual paradigms (e.g., *jump-jumps-jumping-jumped, love-loves-loving-loved*) from a corpus. Despite the presence of distinct inflectional classes, morphosyntactic properties in different classes often share phonological material (e.g., "-{a,e,i}*mos*" in Spanish first plural verbs). The task of alignment makes use of this to put together forms of the same morphosyntactic properties. From a linguist's perspective, alignment as performed here is to show, in a formally explicit way rather than taking it for granted, why surface word forms across morphological paradigms are aligned in the way they are as cued by phonology.

Clustering is tantamount to learning inflectional classes and their string-based hierarchical relationship. If descriptive grammars and current morphological theory recognize inflectional classes, or at least take inflectional morphology seriously (cf. Aronoff 1994; Blevins 2006; Matthews 1972; Spencer 1991; Stump 2001b, and others), one must ask whether (and how) these classes can be learned. Once the inflectional classes are established, we are also curious if there exists any structure *across* them. Cross-linguistically, it is observed that inflectional classes exhibit partial similarity – part of Carstairs' (1983; 1987) notion of paradigm economy. The Spanish conjugation system as presented in (7) provides a convenient illustration: intuitively, -ER and -IR verbs are more similar to each other than either to -AR verbs.

As the table in (7) above shows, the three Spanish conjugation classes are distinct for the first- and second-person plurals, e.g., *-amos, -emos*, and *-imos* for the first-person plurals. It is differences of this type which give rise to inflectional classes. At the same time, these conjugation classes share a great deal in common. Across all three classes, the first-person singular suffixes are *-o*, the second-person singular suffixes end with *-s*, and so forth. It is these similarities which make alignment possible. An algorithmic approach to alignment and clustering will reveal structured similarities and differences across morphological paradigms.

3.2.2 String-based inheritance hierarchies

Inflectional classes do not differ from one another in an arbitrary way. There is a good amount of partial similarity among inflectional classes (Matthews, 1991), and there appears to be an upper bound of the number of inflectional classes given the number of inflectional affixes (Müller, 2007). As such similarity is usually uneven

across inflectional classes in various combinations, it is reasonable to think of inflectional classes as having a nested structure (Corbett and Fraser, 1993; Stump, 2001a) based on the complex overlapping patterns; these patterns have been studied in depth in terms of principal parts (Stump and Finkel, 2013), specific frameworks such as Network Morphology (Brown and Hippisley, 2012), and a combination of these (Baerman, 2012). In general, to study structure of this type, clustering is a suitable and explicit tool to explore and display the hierarchical structure of inflectional classes. In the following, we discuss a highly intuitive and illustrative example on Greek nominals by Haspelmath (2002). The reader will see that some of Haspelmath's goals and results are very similar to ours in the present paper.

Haspelmath considers seven declension classes of Greek nominals, in (8), and discusses what he calls the inheritance hierarchies among them. The very fact that his data have the morphosyntactic features such as number, gender, and case means that we are not dealing with alignment. Working out the inheritance hierarchies among inflectional classes for Haspelmath is exactly our clustering problem.

(8) Seven classes of Greek nominals, data from Haspelmath (2002: 125)

	SG			PL			
Class	NOM	ACC	GEN	NOM	ACC	GEN	
os	nomos	nomo	nomu	nomi	nomus	nomon	'law (masc.)'
as	pateras	patera	patera	pateres	pateres	pateron	'father (masc.)'
us	papus	papu	papu	papuðes	papuðes	papuðon	'grandfather (masc.)
a	imera	imera	imeras	imeres	imeres	imeron	'day (fem.)'
i1	texni	texni	texnis	texnes	texnes	texnon	'art, skill (fem.)'
i2	poli	poli	poleos	poles	poles	poleon	'town (fem.)'
u	maimu	maimu	maimus	maimuðes	maimuðes	maimuðon	'monkey (fem.)'

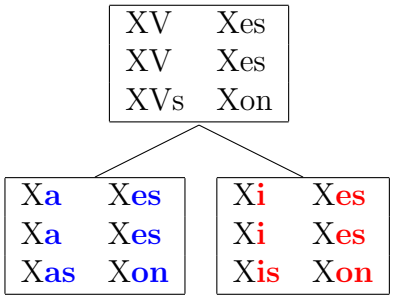
Haspelmath observes that, in terms of surface forms, these declension classes are not completely distinct from one another. The partial similarity is illustrated by comparing the *a*- and *i1*-declension classes.

(9) Greek *a*- and *i1*-declension classes, from Haspelmath (2002: 125)

		a-declension	*i1*-declension
SG	NOM	imer**a**	texn**i**
	ACC	imer**a**	texn**i**
	GEN	imer**as**	texn**is**
PL	NOM	imer**es**	texn**es**
	ACC	imer**es**	texn**es**
	GEN	imer**on**	texn**on**
		'day (fem.)'	'art, skill (fem.)

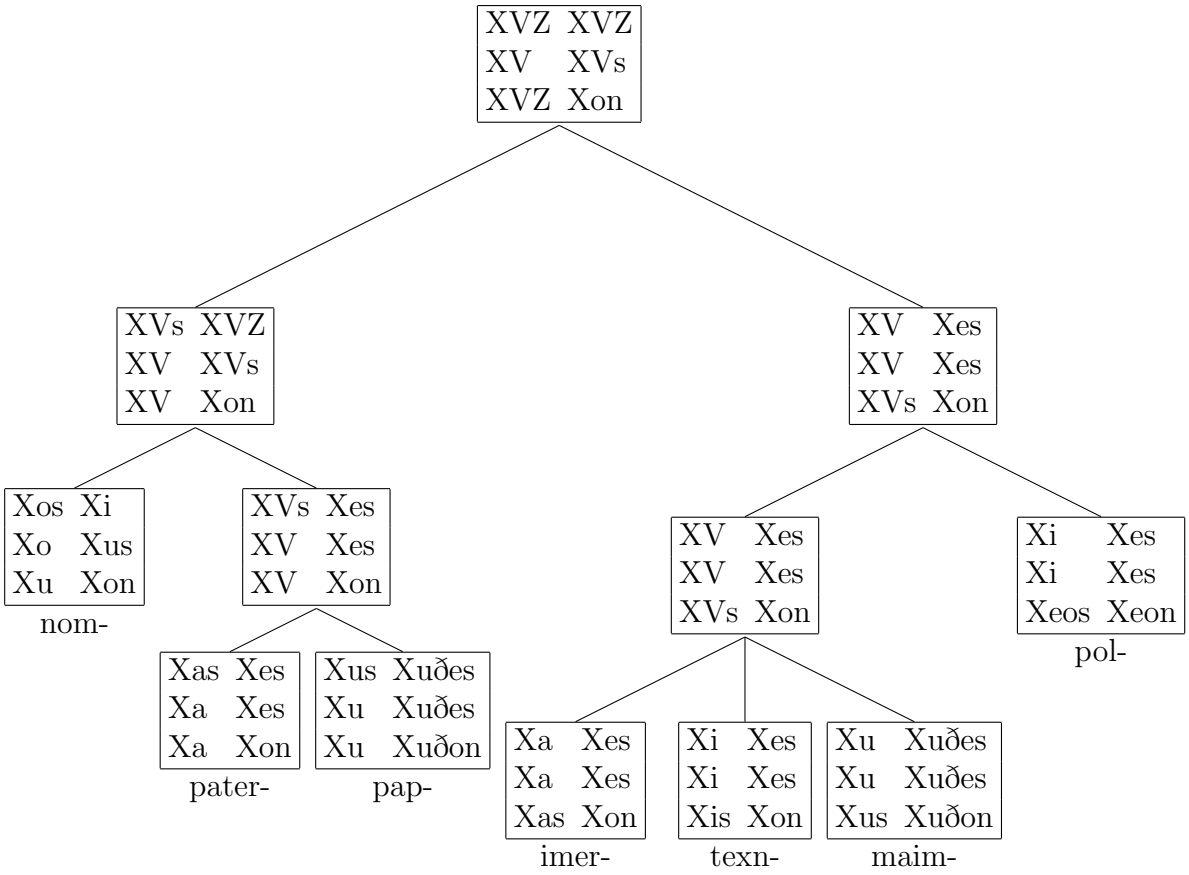
If we replace the common portion among all the forms within a class by "X", it become clear that these two classes differ by only a vowel in the singular forms. Haspelmath illustrates this by means of "V" as a variable for the vowel, and postulates a common template for both classes in a hierarhical representation:

(10) The hierarchy of *a*- and *i1*-declension classes, from Haspelmath (2002: 127)



Then, Haspelmath makes a giant leap forward, by presenting an inheritance hierarchical analysis of all the seven Greek nominal classes we saw earlier.

(11) An inheritance hierarchy of seven Greek declension classes, from Haspelmath (2002: 128)



The tree in (11) can be a sophisticated answer to the question of how many classes there are among the seven Greek paradigms concerned. It may range from two (*macro-classes* in Haspelmath's terminology), referring to masculine classes (the three leaves together on the left in (11)) and feminine ones (the other four leaves on the right), to four or seven (*micro-classes*). Furthermore, the tree also indicates the defaults and principal parts for the given data. The plural genitive can be considered a default, since all the given seven paradigms end with *-on*. The top node with "Xon" for plural genitive in (11) indicates that all paradigms in question here share, or *inherit*, such a morphological characteristic. On the other hand, the singular genitive is a possible principal part, because the seven paradigms have almost distinct realizations (with six unique ones) for this cell: *-u, -a, -u, -as, -is, -us, -eos.*

Both Haspelmath's work and ours are about learning inflectional classes and their hierarchical patterning, but we do not assume knowledge of morphosyntactic

alignment at the outset. Interestingly, Haspelmath does not explain how exactly the rest of the tree with hierarhical clusters in (11) is obtained. We work on the same clustering problem, with fewer assumptions, and more explicitly.

3.3 Other related work

What bridges alignment and clustering is the notion of inflectional class. Clustering learns inflectional classes, whereas alignment indirectly learns how the inflectional classes differ by learning morphological categorization. In the literature, there is a growing body of computational work geared in one way or another towards inflectional classes. Some of it tackles them head-on, while others inquire about their higher-order structure. Also, equally important is some other work on inflectional morphology and paradigms where the idea of inflectional classes looms large in the background. In general, our present work differs from others in that we do not assume alignment (i.e., the morphosyntactic features) is given in the input data.

Computational work that explicitly attempts to learn inflectional classes includes Goldsmith and O'Brien (2006) who use a connectionist model to learn inflectional classes (for Spanish conjugation and German nominal/article declension) modeled as one-layer hidden nodes. In the developmental literature, MacWhinney et al. (1989) also use connectionist algorithms to model inflectional patterns. In contrast, it is interesting to take note of the body of work termed "paradigm induction" by Hammarström and Borin (2011: section 3.2.3) (see Chan (2006); Zeman (2008, 2009); Hammarström (2009); Monson (2009)) which seems to go in the opposite direction by *collapsing* the inflectional classes in some explicit way. For instance, Chan (2006) neutralizes allomorphic distinctions in input data representation, while Zeman (2008, 2009) reduces the number of paradigms learned based on a subset principle to collapse sets of affixes.

On the hierarchical patterns that inflectional classes display, the line of research under the rubric of Network Morphology (Corbett and Fraser (1993); Finkel and Stump (2007); Brown and Evans (2010, 2012); Brown and Hippisley (2012), among others) captures the ideas of defaults and principal parts in a hierarchical structure. Using the conjugation of Spanish present indicative in (7) for illustration, the first-person singular is the default, because all the three conjugation groups share *-o* (a given first-person singular form does not help in signaling the conjugation group). The first-person plural is a principal part, because all the conjugation groups have distinct realizations with *-amos, -emos, -imos*. In a hierarchical representation, higher nodes (away from the leaves at the bottom) encode the defaults for characteristics shared by nodes or leaves beneath, while the leaves with specific information re-

flect the principal parts. These ideas—defaults and principal parts in inheritance hierarchies—are illustrated succinctly in section 3.2.2 with the work by Haspelmath (2002) on Greek nominals.

Lastly, there is a body of work focusing on the relations of a subset of forms in a paradigm to the others. Many researchers have observed that some forms in a paradigm are useful in predicting other forms. Bonami and Boyé (2002) formalize this observation as dependency relations couched within the HPSG framework. Ackerman et al. (2009), Ackerman and Malouf (2013), and Finkel and Stump (2009) quantify how predictive some forms of paradigms are for other forms. Relatedly, the General Minimization Learner learns the most predictive form (called the base) of some given paradigm (Albright 2002a, b; Albright and Hayes 2002, 2003; Albright 2008, see also Bonami and Boyé 2007 on French). The connection of inflectional class learning (one goal of this present paper) with respect to these works is this: the knowledge of which inflectional class a certain word form belongs to should facilitate the prediction of other forms in the paradigm. Computationally, clustering for inflectional classes restricts the relevant search space, in that a paradigm is inflectionally similar to other paradigms in a more local cluster than to those in distant ones. A key difference between the proposed algorithm in the present paper and the works just cited is that our algorithm treats all word forms of a paradigm equally and does not attempt to learn which forms within a paradigm are superior to others in any sense.

4 Algorithm

This section describes in detail the algorithm to perform alignment and clustering. In brief, we treat the tasks at hand as an iterative, greedy optimization problem. At each iteration, the *complexity* of the system, to be defined and explained below, is minimized. Here is the algorithm in brief pseudocode:

(12) The alignment-clustering algorithm

Data: n paradigms, each with k forms

Initialize **stems and affixes**;
    n paradigms $\rightarrow$ n **stemplexes**;
Initialize **overall complexity** (grammar cost + data cost);
while $n > 1$ **do**
    for $i \leftarrow 1$ **to** $\binom{n}{2}$ **do**
        (for the i-th **merging** possibility of the 2 stemplexes);
        for $j \leftarrow 1$ **to** $k!$ **do**
            Compute d_{ij}, the decrease in complexity for the j-th **alignment** choice of the i-th merging possibility;
    For the largest d_{ij}, actually perform the i-th merging for those 2 stemplexes at the j-th alignment choice;
    $n \longleftarrow n - 1$;

4.1 Initializing stemplexes

The first initialization step is to create stemplexes from the paradigms. We have felt the need to coin the word *stemplex* which refers to a new entity: a composite object with a list of stems, a list of (union) affixes, and a list of morphological paradigms which look very much like the input data. In its simplest form, a stemplex has only one morphological paradigm, together with its stem and affixes.

The starting point is a data set with n rows (paradigms), each with k forms:

(13) An $n \times k$ data set

	k **columns**				
n **rows**	jump	jumping	jumps	jumped	jumper
	clap	clapping	claps	clapped	clapper
	⋮	⋮	⋮	⋮	⋮

4.1.1 Stems and affixes

For each paradigm, the algorithm defines its stem and affixes. A stem is the multiset of the letters/sounds shared by all the forms in the paradigm, and what remains in

each form is an affix; the detailed discussion on strings as mutltisets is in section 4.1.2. Deriving stems and affixes is not the focus of this paper, but we need some hypothesis to get off the ground (as in Haspelmath's work on Greek nominals discussed above) for alignment and clustering. As it stands, the algorithm does not update the stems and affixes once they are initialized. Some paradigms with their stem and affixes are illustrated below.

(14) Deriving stems and affixes by the algorithm

Paradigm	Stem + affixes
jump, jumping, jumps, jumped	jump + {Ø, ing, s, ed}
clap, clapping, claps, clapped	clap + {Ø, ping, s, ped}
go, going, goes, went	Ø + {go, going, goes, went}

If one examines the stems and affixes in (14, it appears at first blush that some of them do not quite make sense to a linguist. In particular, the GO paradigm has the null string Ø as its stem, and all the forms of this paradigm have been shoveled to the affix slots. There is no bug here: this is necessarily the case because the verb forms *go* and *went* do not share any letters at all. Indeed, the stem and affixes thus derived for GO may look odd, but as alluded to above, getting the stems and affixes right (however one defines "right" here) is tangential to alignment and clustering, our main goals in this paper.[5] To use the airplane-versus-bird analogy as in Jurafsky and Martin (2006: 14), both airplanes and birds have wings, both fly, but airplanes do not flap their wings. In our case, we think that we do need some hypothesis of stems and affixes, as many linguists do when they work on morphology.[6] However, we need stems and affixes for other purposes, and therefore having our stems and affixes match what one would think they should look like is a non-issue. As will be clear in section 5 on results, a paradigm such as GO with a peculiar stem-affix hypothesis does not have adverse effects in alignment and clustering. Naturally, there have been explicit attempts to infer the stem and affixes of a morphological paradigm with varying assumptions (see the survey on morpheme segmentation in Goldsmith 2010; Hammarström and Borin 2011), but this is not our objective here.

[5]Goldsmith (2011b) proposes a string algebra to learn morphophonology in paradigms, which has the potential to learn stem allmorphy between Ø and *go* for GO, for instance.

[6]This is naturally true for a linguist operating within a morpheme-based framework of morphology and morphosyntax. But even in word-based approaches, it is difficult, if not impossible, to get away from the idea that a morphologically complex word typically has some phonological material (i.e., the affixal exponence) shared by other words (Anderson, 1992; Booij, 2010).

4.1.2 Strings as multisets of symbols

An important remark is in order regarding the representation of strings. Our algorithm actually treats all word forms, stems, and affixes as *bags of letters* with both adjacency and linear ordering of letters/sounds removed.[7] For example, the word *jump* is computationally represented as the alphabetized "jmpu", if it has to be represented as a string at all. This strategy has multiple advantages. First, it makes it computationally easy to derive stems and affixes. Second, it does not assume whether the language at hand is a prefixing, suffixing, or even infixing language.[8] This point is illustrated by the potential of the algorithm to deal with languages with templatic morphology with consonantal roots such as Semitic languages.[9] To give a concrete example, we use the well-known Arabic forms with the triconsonantal root *k-t-b* (loosely meaning 'to write'):

(15) Arabic *k-t-b* forms and their alphabetized representations in the algorithm [10]

	Arabic form	Alphabetized representation
'he wrote'	**k**a**t**a**b**a	aaa**bkt**
'we wrote'	**k**a**t**a**b**nā	aaā**bk**n**t**
'he writes, will write'	ya**kt**u**b**u	a**bkt**uuy
'we write, will write'	na**kt**u**b**u	a**bk**n**t**uu
'writer'	**k**ā**t**i**b**	ā**bikt**
'he dictated'	a**kt**a**b**a	aaa**bkt**
'he dictates, will dictate'	yu**kt**i**b**u	**bikt**uuy
'he asked s.o. to write s.th.'	ista**kt**a**b**a	aaa**bikst**t
(imperfect of above)	yasta**kt**i**b**u	aa**bikst**tuy
'office'	ma**kt**a**b**	aa**bk**m**t**

Using these Arabic forms for illustration, the way the algorithm derives the stem and affixes for a given paradigm is as follows. First, the shortest word form is located, i.e., *ābikt* (alphabetized for *kātib*). Then, the algorithm scans it from left to right,

[7] It is also possible to treats strings with only linear ordering while adjacency is ignored. For stem extraction, this way of treating word forms leads to the idea that the stem is the longest common *subsequence* of all word forms in a paradigm (Lee and Goldsmith, In prep).

[8] Austranesian languages, for example, are well-known to employ infixation in their inflectional morphology; see Yu (2007) for a general survey of infixation.

[9] For languages with templatic morphology, there are sophisticated unsupervised approaches such as Goldsmith and Xanthos (2009), who use a graph theoretical approach to separate vowels and consonants in order to learn consonantal roots in Arabic.

[10] Arabic forms are from `http://en.wikipedia.org/wiki/Semitic_root` accessed May 16, 2013. The way in which the orthographic *ā* (for /a:/) is alphabetized does not affect our points of interest.

asking whether each letter is present in all word forms. If it is, the letter is part of the stem, but if not, the letter is part of an affix. In this case, for *ābikt*, only *bkt* is shared by all word forms in (15), and therefore is the stem. This stem is exactly the alphabetized version of the triconsonantal root *k-t-b*. Right from Harris (1955) on the unsupervised learning of morphological structure, it is customary that the linear ordering of letters or sounds is assumed and used. Indeed, this necessarily has to be the case for tasks such as morpheme segmentation. But for objectives like ours, if discovering morphemes is at best secondary, then removing the linear ordering of letters gives rise to interesting and useful consequences.

Using our terminology, we have n stemplexes initialized from n paradigms. Next, the algorithm initializes an important measurement, the complexity of the stemplexes.

4.2 The complexity computation

What is the complexity of an analysis? The notion of complexity is frequently appealed to in theoretical linguistics. A typical scenario consists of multiple competing analyses from different frameworks for some given linguistic data, and the argumentation goes in favor of one analysis, argued to be the least complex, and therefore the particular theoretical framework within which the analysis is couched is superior. Complexity is often characterized qualitatively without rigorous quantification. This makes comparison of analyses rest on intuitive and subjective terms. This paper represents a step forward towards an objective and testable measure of complexity. For our purposes, we speak of the complexity computation of some morphological analysis. Two distinct but related questions are asked: (i) How is complexity computed? (ii) How is complexity used?

This section focuses on providing an answer to the first question. We propose a way to represent the complexity of a stemplex, using a number whose computation is detailed below. The next section is to answer the second question, where we detail the use of the complexity measurements in the algorithmic steps of alignment and clustering.

The complexity computation is explained by means of an example. Consider the JUMP stemplex with five word forms:[11]

[11] As explained above, the algorithm actually treats everything as bags of letters. For reasons of readability, however, we use the human-friendly representations throughout this paper.

(16) The JUMP stemplex

		jump	jumps	jumping	jumped	jumper	⇐ TARGET FORMS
STEM ⇒	jump	Ø	s	ing	ed	er	⇐ AFFIXES

Complexity is a trade-off between two related components:

(17) Complexity = Grammar cost + Data cost given the grammar

Some remarks are in order as to why we need both components. First, the grammar cost measures how complex a grammar is. Second, we also need a measure for the forms actually observed, and that is the data cost. We need both costs because a grammar provides only generalizations in abstract terms for some given data but not the actual linguistic forms we use. For instance, we may well utter the sentence *The dog chased the cat* but not *Article Noun Verb Article Noun* given by a part-of-speech analysis. The actual forms are generated by the grammar, and they incur a cost.

The grammar cost is dependent on the set of p stems $\{t_1, \ldots, t_p\}$ and affixes $\{x_1, \ldots, x_k\}$:

(18) Grammar cost $= \lambda \cdot (\sum_{i=1}^{p} |t_i| + \sum_{j=1}^{k} |x_j| + k)$

The grammar cost of a stemplex composed of p paradigms hinges on three terms: the length of all the p stems ($|s|$ denotes the length of a string s), the length of all the k affixes, and k (the number of word forms in each paradigm). λ is set to be 5, because $2^5 = 32$ is roughly the number of letters in the alphabet for languages of interest, and five bits are needed to encode one letter. To illustrate the computation, we use the JUMP stemplex as in (16), i.e., $p = 1$, with only one paradigm in this stemplex, and $k = 5$ for five word forms.

(19) Grammar cost of the JUMP stemplex[12]

$$\begin{aligned} & 5 \cdot (|\text{jump}| + |\text{Ø}| + |\text{s}| + |\text{ing}| + |\text{ed}| + |\text{er}| + 5) \\ =& 5 \cdot (4 + 0 + 1 + 3 + 2 + 2 + 5) \\ =& 85 \end{aligned}$$

[12] As it stands currently, the algorithm treats the null string Ø as a letter of zero letters long.

The data cost computation is arguably more complicated. Several parameter values represent the different weights for various components of generating the data. They are assigned some (arbitrarily) chosen values – more comments on these parameters in section 6.1. γ denotes the vector representing these parameters.

(20) Data cost parameters

StemUsed	4
StemNotUsed	?
AffixUsed	1
AffixNotUsed	2
Extra	?

$$\gamma = \begin{bmatrix} 4 \\ 0 \\ 1 \\ 2 \\ 0 \end{bmatrix}$$

Conceptually, the set-up has five parameters as shown in (20), which means that the linguistic system incurs complexity in terms of five different components in order to generate the observed forms. We have to pay for (i) each stem letter used, (ii) each stem letter not used, (iii) each affix letter used, (iv) each affix letter not used, and (v) each extra letter from neither the stem or the affix. Nevertheless, given the simple way how the stems and affixes are derived, only three of these data cost components currently play an actual role in the algorithm; their weights are shown in (20). In further work, if we allow changes of the stem-affix boundaries, then there may be situations where a stem letter from the grammar is not used to generate the observed word (analogous to deleting a vowel or consonant from the stem's underlying form), or where an extra letter is needed (i.e., epenthesis). Until then, the parameters StemNotUsed and Extra are inactive. We still choose to include them here for completeness.

For every target word, i.e., each of {*jump, jumps, jumping, jumped, jumper*} in (16), the goal is to use the stem and the relevant affix to generate it, and the algorithm keeps track of the cost associated. Formally, we define $u_i \subseteq t_i$ as the set of stem letters used from the stem t_i. It follows that $\bar{u}_i = t_i \setminus u_i$ is the set of stem letters not used from the stem t_i. Similarly, $v_j \subseteq x_j$ is the set of affix letters used from the affix x_j, $\bar{v}_j = x_j \setminus v_j$ is the set of affix letters not used from the affix x_j. e is the set of letters used from neither the stem nor the affix.

(21) Data cost for a target word $w_{ij} = \gamma^\top \begin{bmatrix} |u_i| \\ |\bar{u}_i| \\ |x_j| \\ |\bar{x}_j| \\ |e| \end{bmatrix}$

Take the target word *jumps* as an example. Its stem is *jump*, its affix is *s*, and so the data cost is as follows:

(22)

Data cost to generate *jumps* from stem *jump* and affix *s*

$$= \gamma^{\top} \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 0 & 1 & 2 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$= 17 \text{ (sum of all entries in } \begin{bmatrix} 16 & 0 & 1 & 0 & 0 \end{bmatrix}^{\top}, \text{the second column in (23))}$$

Performing the same procedure for all target word forms in a given stemplex results in a data cost matrix, here the one for the JUMP stemplex:

(23) Data cost of the JUMP stemplex in (16)

	jump	jumps	jumping	jumped	jumper	
jump	Ø	s	ing	ed	er	
	16	16	16	16	16	(StemUsed)
	0	0	0	0	0	(StemNotUsed)
	0	1	3	2	2	(AffixUsed)
	0	0	0	0	0	(AffixNotUsed)
	0	0	0	0	0	(Extra)

Summing all entries in the data cost matrix gives 88 as the total data cost for the JUMP stemplex. The overall complexity of this stemplex, grammar plus data, is 85 + 88 = 173.

The algorithm computes the grammar cost and data cost in the way just described for all stemplexes. The total complexity of the stemplexes is the sum of all grammar costs and data costs.

4.3 Greedy optimization and Minimum Description Length

The previous section presented in detail how complexity is computed. This section explains how the computed complexity is used for alignment and clustering.

With the stemplexes and total complexity initialized based on the input data, the algorithm undergoes an iterative, greedy process of minimizing the total complexity. This is achieved by iteratively merging two stemplexes. "Merge" here has nothing to do with Merge in the current Minimalist syntactic theory, but instead means: choose two stemplexes, compute the optimal alignment between them, and create a new stemplex based on this alignment, such that the total complexity decreases the most at the given particular iteration.

4.3.1 Union affixes and optimal alignment

The goal of alignment is to learn, for instance, that *jumps* is best aligned with *loves*, *jumped* with *loved*, and so forth. The key is the creation and comparison of *union affixes*. As an example, we consider the merging of the JUMP and LOVE stemplexes. The LOVE stemplex looks like the following, with a total complexity of 90 + 70 = 160; note that this verb is different from JUMP and belongs to the group with *e*-final stems:

(24) The LOVE stemplex

		love	loves	loving	loved	lover	⇐ TARGET WORD FORMS
STEM ⇒	lov	e	es	ing	ed	er	⇐ AFFIXES

(25) Grammar cost of the LOVE stemplex

$$\begin{aligned} & 5 \cdot (|\text{lov}| + |\text{e}| + |\text{es}| + |\text{ing}| + |\text{ed}| + |\text{er}| + 5) \\ =& 5 \cdot (3 + 1 + 2 + 3 + 2 + 2 + 5) \\ =& 90 \end{aligned}$$

(26) Data cost of the LOVE stemplex

	love	loves	loving	loved	lover	
lov	e	es	ing	ed	er	
	12	12	12	12	12	(StemUsed)
	0	0	0	0	0	(StemNotUsed)
	1	2	3	2	2	(AffixUsed)
	0	0	0	0	0	(AffixNotUsed)
	0	0	0	0	0	(Extra)

Total data cost = 70

The discussion is first in qualitative and intuitive terms, introducing the idea of union affixes. According to the algorithm (and our knowledge about English), the best alignment for the JUMP and LOVE stemplexes is in (27a); an affix is boxed if it differs from the union affix:

(27) Aligning JUMP and LOVE

a. Optimal alignment

Union affixes	**-e**	**-ed**	**-ing**	**-es**	**-er**
	jump-$\boxed{\emptyset}$	jump-*ed*	jump-*ing*	jump-$\boxed{s}$	jump-*er*
	lov-*e*	lov-*ed*	lov-*ing*	lov-*es*	lov-*er*

b. Suboptimal alignment

Union affixes	**-es**	**-ed**	**-ing**	**-es**	**-er**
	jump-$\boxed{s}$	jump-*ed*	jump-*ing*	jump-$\boxed{\emptyset}$	jump-*er*
	lov-$\boxed{e}$	lov-*ed*	lov-*ing*	lov-*es*	lov-*er*

To merge the stemplexes JUMP and LOVE into a new stemplex, the algorithm considers all 5! = 120 alignment possibilities. Given a particular alignment, the union affixes are computed by taking the union of the affixes from the same column. For example, in the fourth column in (27a), the union affix *-es* is the union of the individual affixes *-e* and *-es*.[13] For this new JUMP-LOVE stemplex, the union affixes thus derived under this particular alignment are the new affixes, and are what counts towards the grammar cost of this new stemplex.

[13] As always, we are dealing with bags of letters, not sets in the mathematical sense. So in a case with the same double letters in two individual affixes (hypothetically, *abb* and *bbc*), the union affix preserves the double letters (*abbc* as the union affix for this example).

Because the grammar has been updated, the data cost for generating the observed forms for both JUMP and LOVE has to be recalculated based on the new affixes. For instance, in (27a), to generate *jumps*, the stem is *jump-*, and the union affix is *-es*. All four stem letters from *jump-* are used, but only one letter, *-s*, from the union affix *-es* is used. The unused union affix letter *-e* incurs a cost in the algorithm (the affix *-s* is boxed for differing from the corresponding union affix). Any other alignment deviating from (27a), such as (27b) with more boxed affixes, incurs a higher data cost than that of (27a). The alignment in (27a), with the lowest overall cost, is the best alignment for merging JUMP and LOVE.

Here are the quantitative details of the alignment just explained. Before merging, the total complexity of the JUMP is 173, and that of the LOVE is 160. The combined complexity of the two stemplexes is 173 + 160 = 333. Finding the optimal alignment is to merge the stemplexes such that the resultant new stemplex has the lowest complexity. Below are the cost details of the two alignments illustrated in (27).

(28) Data costs for aligning JUMP and LOVE

a. Optimal alignment

	-e	**-ed**	**-ing**	**-es**	**-er**
jump-	jump-*Ø*	jump-*ed*	jump-*ing*	jump-*s*	jump-*er*
	16	16	16	16	16
	0	0	0	0	0
	0	1	3	2	2
	2	2	0	0	0
	0	0	0	0	0
lov-	lov-*e*	lov-*ed*	lov-*ing*	lov-*es*	lov-*er*
	12	12	12	12	12
	0	0	0	0	0
	1	2	3	2	2
	0	0	0	0	0
	0	0	0	0	0

Data cost = 92 + 70 = 162

b. Suboptimal alignment

	-es	**-ed**	**-ing**	**-es**	**-er**
jump-	jump-*s*	jump-*ed*	jump-*ing*	jump-*Ø*	jump-*er*
	16	16	16	16	16
	0	0	0	0	0
	0	1	3	2	2
	4	2	0	0	0
	0	0	0	0	0
lov-	lov-*e*	lov-*ed*	lov-*ing*	lov-*es*	lov-*er*
	12	12	12	12	12
	0	0	0	0	0
	2	1	3	2	2
	0	2	0	0	0
	0	0	0	0	0

Data cost = 94 + 72 = 166

(29) Grammar costs for aligning JUMP and LOVE

a. Optimal alignment

$$\begin{aligned} & 5(|\text{jump}| + |\text{lov}| + |\text{e}| + |\text{ed}| + |\text{ing}| + |\text{es}| + |\text{er}| + 5) \\ = \; & 5(4+ 3 + 1 + 2 + 3 + 2 + 2 + 5) \\ = \; & 110 \end{aligned}$$

b. Suboptimal alignment

$$\begin{aligned} & 5(|\text{jump}| + |\text{lov}| + |\text{es}| + |\text{ed}| + |\text{ing}| + |\text{es}| + |\text{er}| + 5) \\ = \; & 5(4+ 3 + 2 + 2 + 3 + 2 + 2 + 5) \\ = \; & 115 \end{aligned}$$

Examining the cost details in (28) and (29) reveals how the best alignment beats all other competing alignments. In terms of grammar cost, it is the differences in union affixes in different alignments. As for data cost, the locus is the mismatches between union affixes and individual affixes.

The optimal alignment has a total complexity of 162 + 110 = 272. For the suboptimal alignment discussed, the complexity is 166 + 115 = 281. Both are lower than the pre-merging complexity of 333, but the complexity of the optimal alignment is the lowest among all 5! alignment permutations by saving $333 - 272 = 61$. It is only $333 - 281 = 52$ that this particular suboptimal alignment saves. The following table shows the cost saved of the best ten alignments.

(30) Costs saved of the best ten alignments for JUMP and LOVE

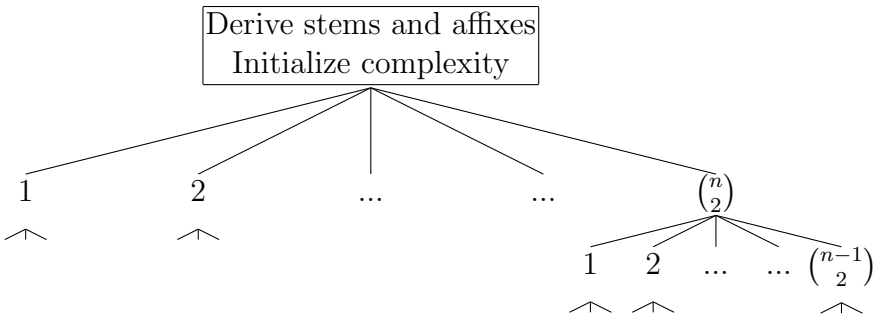
Rank	Cost saved
1	61
2	52
3	52
4	52
5	43
6	43
7	43
8	43
9	43
10	43

4.3.2 MDL-based iterative merging

With n stemplexes initially, each with k target word forms, there are $\binom{n}{2}$ distinct ways of picking a pair of stemplexes for merging. For each of these $\binom{n}{2}$ merging possibilities, there are $k!$ alignment permutations. The algorithm checks all $\binom{n}{2}k!$ options for the particular alignment in a specific merging option which lowers the total complexity the most.

Merging is performed iteratively as greedy MDL-based optimization. At each iteration, two stemplexes are merged with the best alignment between them, such that the grand total complexity for the overall data set is minimized. This is schematized as follows:

(31)



After the first iteration, the system has $n-1$ stemplexes left, and therefore the second iteration has $\binom{n-1}{2}$ merging possibilities. The iterative merging process ends when there is only one stemplex left in the system.

A remark on the greediness of the algorithm. Once two stemplexes are merged with their best alignment, the new stemplex stays as is. No un-merging or re-alignment is allowed. The algorithm does the best it can to lower complexity at each iteration. In brief, the algorithm does not look back, nor look ahead. Even if there may be *globally* less costly merging options down the road in future iterations that would require a *locally* suboptimal merging choice, the algorithm does not consider them. In fact, there is a practical, computational reason for the strict greediness: when merging involves a complex stemplex, if the algorithm took away the established alignments within that complex stemplex, then the number of (re-)alignment possibilities would grow exponentially over the factorials and the computation would take way too long.

4.3.3 The clustering effect

Finally, what is left to be accounted for is the clusters that mimic inflectional classes for conjugation and declension. The key is the MDL-based and greedy nature of the algorithm. The more similar the paradigms, the earlier they merge in the iteration. From the perspective of machine learning, since all clustering algorithms employ some notion of distance among the objects in question, we can say that this paper proposes a measure for morphological similarity among paradigms in order to perform bottom-up, agglomerative clustering.

Let us for the moment consider the merging of the JUMP and WALK stemplexes.

(32) The grammar (stems and affixes) for merging JUMP and WALK

a. Before merging, with individual affixes for each stemplex:

jump-	-Ø	-ed	-ing	-s	-er
walk-	-Ø	-ed	-ing	-s	-er

b. After merging, with union affixes for the new stemplex:

jump- walk-	-Ø	-ed	-ing	-s	-er

The union affixes thus created for JUMP and WALK are identical to the individual affixes of both paradigms. This is doubly good in terms of complexity minimization. For the new grammar, one of the two (identical) sets of the individual affixes are effectively wiped out from the system. For the data cost, there is no increase. If the two sets of individual affixes were not identical, as is the case between JUMP (regular) and LOVE (with silent *e*) discussed in detail above, there would not be such advantages. The more similar paradigms attract one another.

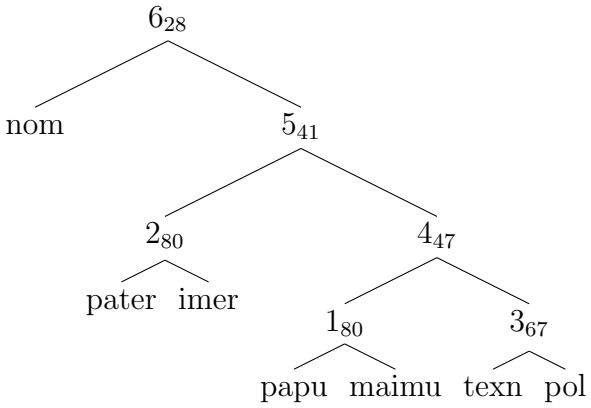
4.4 A variant: clustering with alignment given

If we now have an algorithm which learns hierarchical patterns of inflectional classes, then it should be interesting to try it with the Greek nominal data that Haspelmath (2002) considers. With everything unchanged in the algorithm, including all the grammar and data cost parameters, the seven Greek paradigms in (8) are analyzed as follows for alignment and clustering.

(33) Alignment results of the seven Greek nominal paradigms from (8)

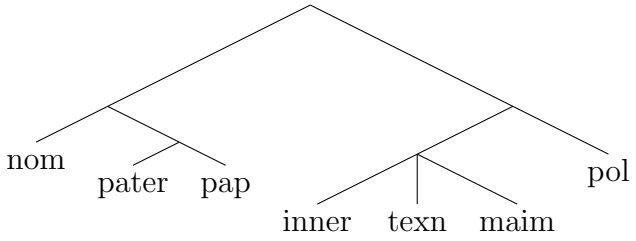
mno	nomos	nomo	nomu	nomi	nomus	nomon
aeprt	pateras	patera	patera	pateres	pateres	pateron
eimr	imeras	imera	imera	imeres	imeres	imeron
appu	papus	papu	papu	papuðes	papuðes	papuðon
aimmu	maimus	maimu	maimu	maimuðes	maimuðes	maimuðon
entx	texnis	texni	texni	texnes	texnes	texnon
lop	poleos	poli	poli	poles	poles	poleon

(34) Clustering results of the seven Greek nominal paradigms from (8)



The alignment in (33) does not match the original Greek data with morphosyntactic features indicating the correct alignment in (8). As for clustering, the resultant tree looks quite distinct from what Haspelmath has come up with. For ease of comparison, Haspelmath's tree is simplified as follows:

(35) Haspelmath's inheritance hierarchy for the Greek nominals concerned



While the Greek results may be less satisfactory than the English ones, this is not entirely bad new for us. Understanding why we have obtained our results is instructive for further research. For alignment, one factor contributing to the undesirable results is that the algorithm is not (and should not be) able to see a strong generalization within the given data separating the masculine classes from the feminine ones. The singular nominative in the given *masculine* classes all ends with *-s*, but without it in the singular genitive. This pattern is the exact opposite in the *feminine* classes.

The crucial difference between Haspelmath's work and ours here is that Haspelmath knows *a priori* the alignment by the knowledge of morphosyntactic knowledge. His goal is to deal only with clustering. Indeed, as illustrated in (35), Haspelmath's inheritance hierarchical analysis has all masculine classes cluster together, the three leaves on the left. The other four leaves on the right are the feminine classes.

Juxtaposing Haspelmath's work and ours confronts us with the question of whether we assume prior assumption that, for instance, *jumping* and *loving* belong to the same morphological category in English. It is "yes", as in Haspelmath's discussion, if we assume knowledge of morphosyntactic features, their distribution, and all that (which non-computational linguists often do), but it is the "no" side that we would like to explore in this paper and related work: we would like to explore how much we can learn if we remove assumptions which we are so used to and which we take for granted.

5 Results

This section discusses the alignment and clustering results using English verbal paradigms. As exploratory work, preliminary clustering results for Spanish verbal paradigms are briefly presented for locating possible aspects of further research.

5.1 Alignment

The alignment output is in a tabulated form where target word forms in the same column are inferred to belong to the same morphological category. With the given data cost weights, the alignment results are what we expect. In the following table, we also show the stems in their alphabetized forms (the leftmost column).

(36) Alignment results

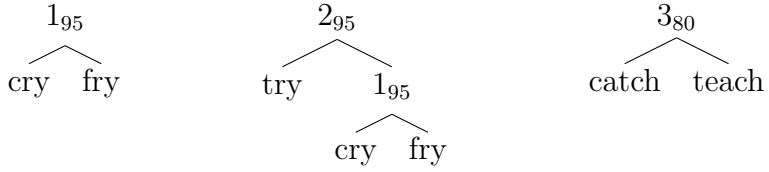
jmpu	jumps	jumping	jump	jumper	jumped
aklw	walks	walking	walk	walker	walked
aklt	talks	talking	talk	talker	talked
mov	moves	moving	move	mover	moved
lov	loves	loving	love	lover	loved
adw	wades	wading	wade	wader	waded
hpsu	pushes	pushing	push	pusher	pushed
chotu	touches	touching	touch	toucher	touched
dno	nods	nodding	nod	nodder	nodded
aclp	claps	clapping	clap	clapper	clapped
cilp	clips	clipping	clip	clipper	clipped
rt	tries	trying	try	tryer	tried
cr	cries	crying	cry	cryer	cried
fr	fries	frying	fry	fryer	fried
bu	buys	buying	buy	buyer	bought
acht	catches	catching	catch	catcher	caught
aht	teaches	teaching	teach	teacher	taught
s	seeks	seeking	seek	seeker	sought
Ø	goes	going	go	goer	went

An interesting aspect is concerned with suppletion. In principle, the strictly string-based algorithm should have a harder time dealing with suppletive forms such as the past tense forms of BUY and GO. As it turns out, though, the alignment results match what is expected: the suppletive past tense forms are aligned with other more regular past tense forms. The reason has to do with the degree of suppletiveness of a paradigm in the data set. In the paradigm GO, for instance, there is only one suppletive form, the past tense form *went*. Because the algorithm is cost-based, the best alignment of GO with respect to other paradigms minimizes complexity by having the word forms with *-es, -ing, -er* correctly aligned to other paradigms with similar affixes. The corollary is that *went* must go to the left-over slot, so to speak, which happens to be the past tense column. If there were highly suppletive paradigms such as BE, the algorithm would not work well.

5.2 Clustering

The clustering results are visualized arboreally. With the 19 English verbal paradigms as input, the following are the first three merges.

(37) The first three merges



The first merge is between the CRY and FRY stemplexes, represented by the first tree in (37). The mother node says "1", which means this is the first merge. There is also the subscript "95", which tells us that this merge saves 95 units in complexity. The second merge is between a complex stemplex, created from merge 1, and the TRY stemplex. This merge also saves 95 units, which indicates that the order of merging among CRY, FRY, and TRY does not matter; this makes good sense, as they are morphologically identical. At the third iteration, the algorithm decides that it is best to merge CATCH and TEACH, which makes the total complexity drop by 80 units.

With 19 paradigms at the outset, there are 18 merges altogether ($n - 1$ merges for n input paradigms). The following table shows the cost saved at each merge.

(38) Costs saved by merging

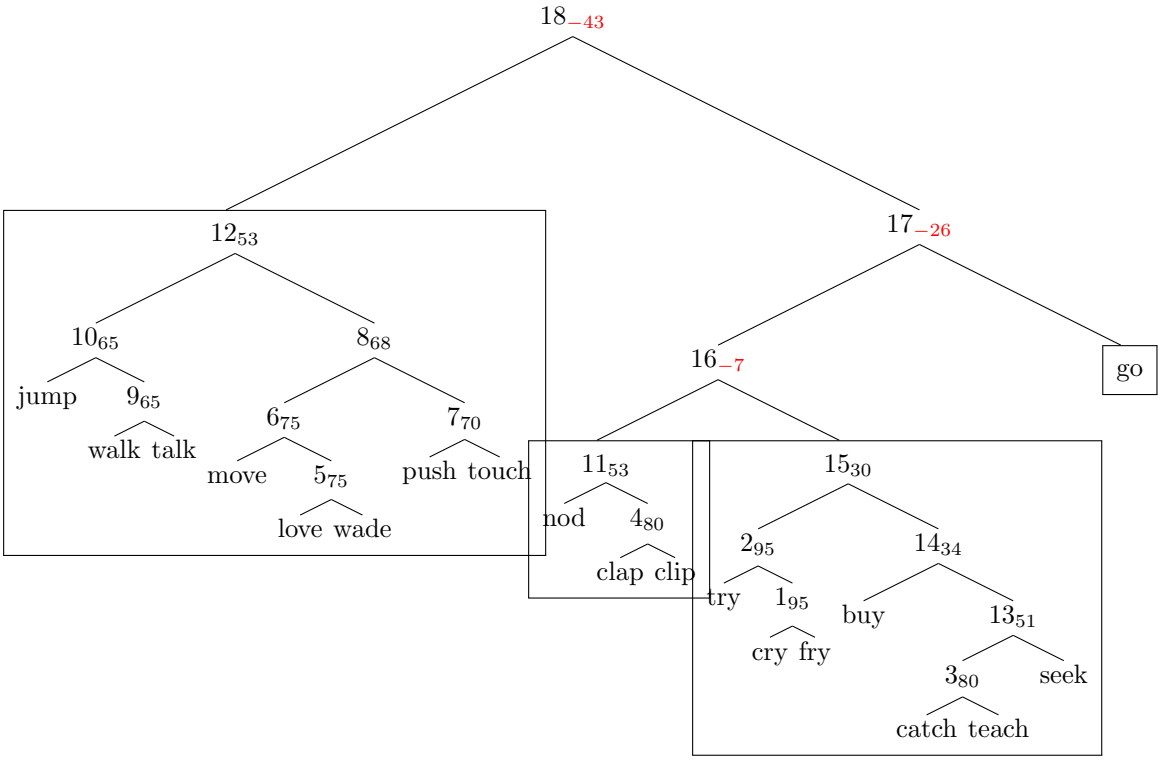
Merge	Cost saved	Merge	Cost saved
1	95	10	65
2	95	11	53
3	80	12	53
4	80	13	51
5	75	14	34
6	75	15	30
7	70	16	-7
8	68	17	-26
9	65	18	-43

Importantly, the costs saved by successive merges are decreasing, or at least non-increasing for ties between two merges. This is the case because the algorithm aims at decreasing the total complexity as quickly as possible. From the 16th merge onwards,

the costs saved are negative. The algorithm can no longer actually decrease the total complexity. Nonetheless, algorithm does not stop until there is only one stemplex left, and the best it can do is to increase complexity the least. In other words, from merge 16 through 18, the total complexity increases as little as possible.

For more interpretation of the clustering results, let us examine the complete tree:

(39) Clustering results



From (39), one can visually identify English 'conjugation groups' such as the following; they are reminiscent of Bloch's (1947) grouping of English verbal inflectional classes:

(40) Some English 'conjugation classes'

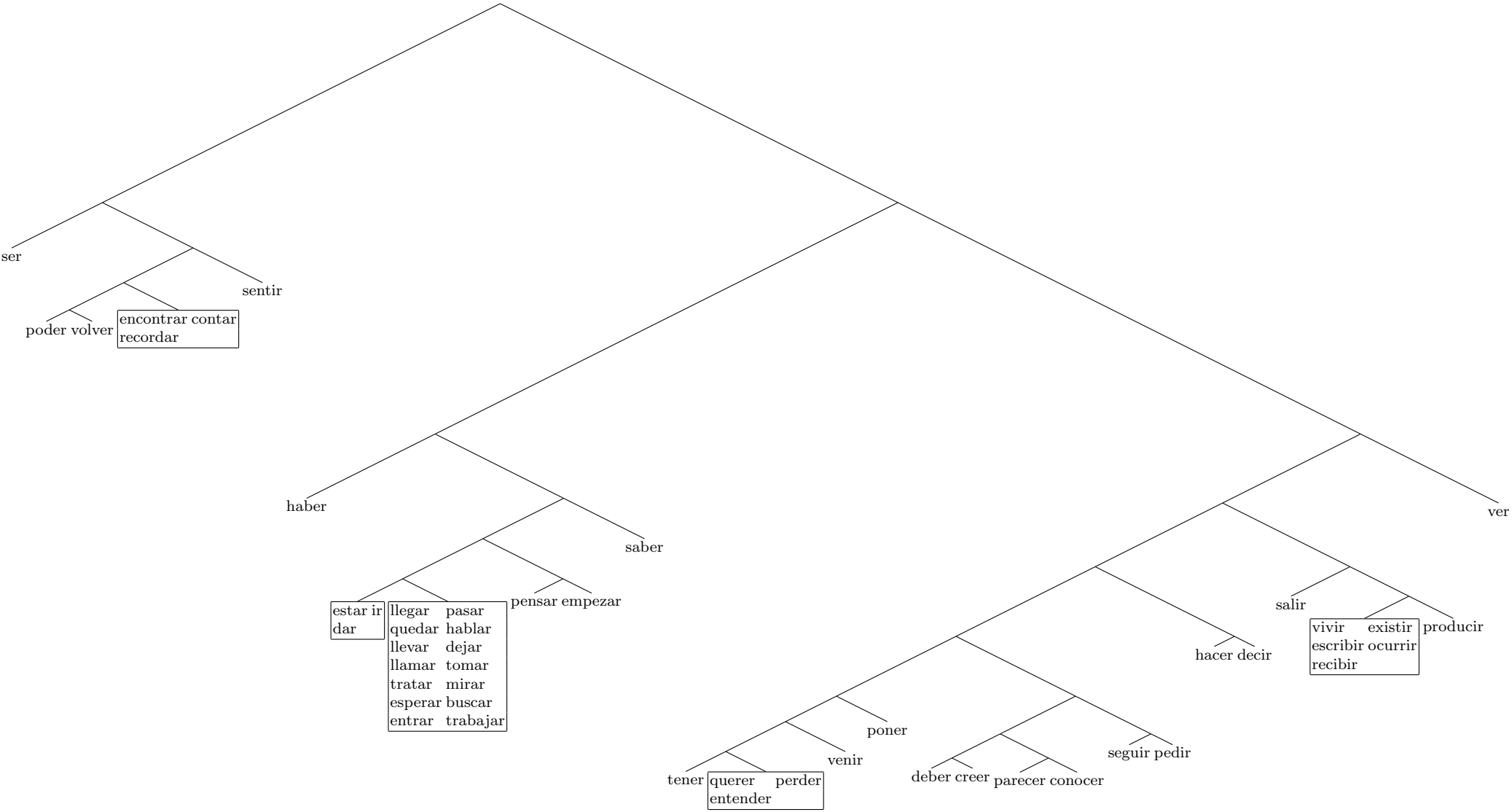
Property	**Paradigms**
Regular	JUMP, WALK, ...
Silent 'e'	LOVE, MOVE, ...
Consonant doubling	NOD, CLAP, ...
'y/ie'	TRY, CRY, ...
'es' for 3sg	PUSH, TOUCH, ...
'{a,o}ught' for past	BUY, SEEK, ...
Suppletive	GO, ...

At another level, the negative saved costs in (38) can be interpreted as indicators of more general, major clusters (Haspelmath's macro-classes). This is *a* way of interpreting them, and in this particular one, they tell us where the algorithm should not have merged, so to speak. In (39), the boxes indicate the main clusters under this interpretation. The groupings appear to reflect suppletiveness or the amount of 'morphophonological alternations' involved in the paradigms.

Given the generality of the algorithm and to point to directions of further work, we briefly examine in the following the preliminary clustering results of Spanish verbal paradigms alluded to from time to from in the discussion above.

We take as input data the 50 most common verbs in Spanish, conjugated in the present indicative; each verb lexeme has six inflected forms, such as *hablo-hablas-habla-hablamos-hablaís-hablan* for *hablar* 'to speak'. The algorithm takes the given alignment in the input data as is (section 4.4) and assumed the same cost parameter γ (20). The resultant clustering tree, with merge numbers and costs removed for simplicity of exposition, is as follows:[14]

[14]The dataset and output files are also found here:
`https://github.com/JacksonLLee/morph-align-cluster`


ser
sentir
poder volver
encontrar contar
recordar
haber
saber
estar ir
dar
llegar pasar
quedar hablar
llevar dejar
llamar tomar
tratar mirar
esperar buscar
entrar trabajar
pensar empezar
ver
salir
vivir existir
escribir ocurrir
recibir
producir
hacer decir
poner
venir
tener
querer perder
entender
deber creer
parecer conocer
seguir pedir

The boxes at some of the leaves in the tree above indicate that the paradigms in the same box are determined to be morphological identical, i.e., having the same inflectional pattern; the largest box is the one with non-stem-changing -AR verbs such as *llegar, quedar*.

On the whole, the clustering results here are less than satisfactory. We have noted above that Spanish verbs are grouped into three major classes of -AR, -ER, and -IR verbs. The results are that verbs of different classes are scattered around rather than showing the intuitively desirable group patterns. One likely reason why this is the case is that strings are treated as multisets which disregard linear ordering. The presence of stem-changing versus non-stem-changing verbs, where the stem-changing patterns involve "a", "e", and "i", introduces confusion with the theme vowels of the same symbols. To tease apart stem-internal vowels and theme vowels for inflectional classes, linear ordering in the treatment of strings is needed; this is elaborated in further work (Lee and Goldsmith, In prep).

6 Ongoing work

This section sketches some ongoing work in relation to the algorithm developed. It all has to do with pushing the assumptions and limitations of the algorithm.

6.1 Cost parameter estimation

A parameter in our algorithm which we have not said much about is the cost parameters, λ for the grammar cost and γ for the data cost. While λ is currently fixed to be 5 for a good reason, γ raises the basic question of what it should be. Due to the cost-based nature of the algorithm, changing the cost parameters would alter the results. We said above that the specific coordinates for the parameter vector γ are arbitrary, which is actually not quite true. In particular, for costs related to affixes, AffixNotUsed is 2, while AffixUsed is 1. This translates into the consequence that dropping an affix letter is more costly than using one. This conforms to the linguists' intuition of preferring the exponence of underlying information rather than dropping it (think, for example, faithfulness in Optimality Theory, Prince and Smolensky 1993/2004). Ideally, the cost parameters should be learned in a rigorous way, e.g., by providing the gold standard (if existent) of alignment and clustering for the given data to estimate the parameters that would result in the gold-standard analysis. It may also turn out that the cost parameters for different languages are different.

6.2 Using corpus-derived data

The very notion of paradigms has been assumed throughout this work. The English results discussed above depend on the dataset used. In particular, the one that we have used is a full, gapless paradigm array crafted for the purposes of trying out the proposed algorithm, i.e., a highly selective data set, as opposed to a non-selective one (in the sense of Hammarström and Borin (2011: 310)). Can we possibly use naturally occurring data as input?

There has been a good amount of work on learning paradigms from a corpus text (Dreyer 2011; Goldsmith 2000, 2001, 2006, and others). Once we examine corpus-derived paradigms, the assumptions made by the data set emerge. The following table displays a few English paradigms learned from a sizeable corpus by an algorithm similar to *Linguistica* described in Goldsmith (2000, 2001, 2006). Unsurprisingly, we shall see that paradigms derived in an unsupervised fashion are far from clean and tidy as in a full paradigm table.

(41) Some corpus-derived English paradigms[15]

Signature	Example			
Ø-s	anglican	anglicans		
Ø-ly	seasonal	seasonally		
⋮	⋮			
Ø-d-s	incorporate	incorporated	incorporates	
Ø-ed-s	charter	chartered	charters	
⋮	⋮			
Ø-ed-ing-s	represent	represented	representing	represents
ing-ion-ions-ive	reacting	reaction	reactions	reactive
⋮	⋮			

In (41), a row consists of a signature (in the sense of Goldsmith 2000, 2001, 2006), i.e., a unique set of affixes, plus an example from the corpus for that signature. The huge contrast between these corpus-derived paradigms and the handcrafted English data set used in this paper squeezes out the assumptions we have made, implicitly or otherwise. Our data set is streamlined with paradigms of the same number of word forms, but this is clearly not the case in (41). The issues are those of data sparsity and strongly skewed distributions among lexemes (Baayen, 2001).

[15]Data courtesy of John Goldsmith.

Despite the large sizes of corpora, a lexeme is typically observed to be inflected only in some but not all of its possible, legal forms. A comparison among INCORPORATE, CHARTER, REPRESENT, and REACT in (41) demonstrates this point. While they are essentially verbal lexemes, it happens that forms such as *incorporating*, *chartering*, *react*, and so forth do not appear in the corpus in question.

Even if we look at paradigms of the same size in (41), such as *anglican-anglicans* and *seasonal-seasonally*, we know (by knowledge of English morphosyntax sitting in our mind all the time) that the different word forms across paradigms do not belong to the same morphological categories: *anglican-anglicans* is about singular and plural nouns, whereas *seasonal-seasonally* is adjective versus adverb. Alignment and clustering among them would not make sense.

Running our algorithm using corpus-derived paradigms like (41) should be among the to-do list, but to overcome the challenges just mentioned, our system has to be able to deal with (i) gaps in corpus-derived paradigms due to data sparsity as well as (ii) syntactic distributions of signatures, among other potential issues. For (i), it is equivalent to collapsing the signatures, such as the verbal ones of CHARTER and REPRESENT in (41), into one signature, which Goldsmith (2009) identifies as a challenge. Intuitively, a simple subset principle might seem to be able to handle this: if signature A has all affixes in signature B, and if signature A has more affixes than signature B, then collapse signatures A and B by keeping A and removing B. Two potential issues arise out of this strategy. First, perhaps not all stems associated with signature B can actually occur with affixes of signature A (Zeman, 2008, 2009). Second, collapsing the nominal Ø-s and the verbal Ø-ed-ing-s, though obeying the subset principle, would be erroneous. This is why we need some syntactic distributional information, presumably derivable from the input raw corpus text, to avoid this; Chan (2006), for example, assumes part-of-speech tagging in paradigm induction.

6.3 From structure to prediction

One major motivation of studying structure across morphological paradigms is to better understand speakers' remarkable ability to extend morphological patterns to novel forms. To this end, research has focused on devising models capable of predicting novel forms based on training data (Albright and Hayes, 2002, 2003; Dreyer and Eisner, 2011; Durrett and DeNero, 2013). Morphological alignment and clustering as described in this paper explore cross-paradigmatic structure without explict predictive power. Further research shall incorporate them into a system capacble of dealing with, for example, gaps in corpus-derived paradigms discussed in the previous

section and predicting morphologically related forms in wug tests.

7 Conclusions

We have proposed a language-independent and computationally rigorous algorithm which explores structure across morphological paradigms in terms of alignment and clustering for inflectional classes and their hierarchical patterns. The results are useful for further work on the unsupervised learning and prediction-oriented research of morphological paradigmatic structure.

While there are desirable results under some conditions, we have also sketched what the loose ends are and where things will begin to break down. This is actually good news: If we proceed by learning as much as possible while assuming as little as possible, we are faced with potentially unnoticed assumptions and limitations of our proposal head-on, and gain a deeper understanding of it. Then, we improve the proposal based on the identified issues gradually, rather than throwing everything in *en masse*. The incremental, quantitative, and algorithmic approach is a promising method of studying natural language morphology for its intricate properties and intertwining relationship with other components of grammar.

References

Ackerman, Farrell, James P. Blevins and Robert Malouf. 2009. Parts and wholes: Implicative patterns in inflectional paradigms. In James P. Blevins and Juliette Blevins (eds.), *Analogy in Grammar: Form and Acquisition*, 55–83. Oxford: Oxford University Press.

Ackerman, Farrell and Robert Malouf. 2013. Morphological organization: the low conditional entropy conjecture. *Language* 89: 429–464.

Albright, Adam. 2002a. *Identifying the inflectional base*. Ph.D. thesis, University of California, Los Angeles.

Albright, Adam. 2002b. Islands of reliability for regular morphology: evidence from Italian. *Language* 78: 684–709.

Albright, Adam. 2008. Inflectional paradigms have bases too: Arguments from Yiddish. In Asaf Bachrach and Andrew Nevins (eds.), *Inflectional Identity*, 271–312. Oxford University Press.

Albright, Adam and Bruce Hayes. 2002. Modeling English past tense intuitions with minimal generalization. In M. Maxwell (ed.), *Proceedings of the 6th meeting of the ACL Special Interest Group in Computational Phonology*. Philadelphia: ACL.

Albright, Adam and Bruce Hayes. 2003. Rules versus analogy in English past tenses: a computational/experimental study. *Cognition* 90: 119–161.

Anderson, Stephen R. 1982. Where's morphology? *Linguistic Inquiry* 13(4): 571–612.

Anderson, Stephen R. 1992. *A-Morphous Morphology*. Cambridge: Cambridge University Press.

Aronoff, Mark. 1994. *Morphology by itself: stems and inflectional classes*. Cambridge, MA: MIT Press.

Aronoff, Mark and Kirsten Fudeman. 2011. *What is Morphology?* Wiley, 2nd ed.

Baayen, Harald R. 2001. *Word Frequency Description*, vol. 18, Text, Speech, and Language Technology. Dordrecht: Kluwer.

Baerman, Matthew. 2012. Paradigmatic chaos in Nuer. *Language* 88(3): 467–494.

Bauer, Laurie. 2003. *Introducing Linguistic Morphology*. Washington, D.C.: Georgetown University Press, 2nd ed.

Bauer, Laurie. 2004. *A Glossary of Morphology*. Edinburgh: Edinburgh Univerisity Press.

Blevins, James P. 2006. Word-based morphology. *Journal of Linguistics* 42(3): 531–573.

Blevins, James P. 2013. *Word and Paradigm Morphology*. Oxford: Oxford University Press.

Bloch, Bernard. 1947. English verb inflection. *Language* 23(4): 399–418.

Bonami, Olivier and Gilles Boyé. 2002. Suppletion and dependency in inflectional morphology. In *The proceedings of the 8th international conference on head-driven phrase structure grammar*.

Bonami, Olivier and Gilles Boyé. 2007. Remarques sur les bases de la conjugaison. In Elisabeth Delais-Roussarie and Laurence Labrune (eds.), *Des sons et des sens: données et modèles en phonologie et en morphologie*, 77–90. Lavoisier.

Booij, Geert. 2010. *Construction Morphology*. Oxford: Oxford University Press.

Brown, Dunstan and Roger Evans. 2010. Inflectional defaults and principal parts: an empirical investigation. In Stefan Müller (ed.), *Proceedings of the 17th International Conference on Head-Driven Phrase Structure Grammar*, 234–254. CSLI Publications.

Brown, Dunstan and Roger Evans. 2012. Morphological complexity and unsupervised learning: Validating Russian inflectional classes using high frequency data. In Ferenc Kiefer, Mária Ladányi, and Péter Siptár (eds.), *Current Issues in Morphological Theory: (Ir)regularity, analogy and frequency*, 135–162. Amsterdam, Philadelphia.

Brown, Dunstan and Andrew Hippisley. 2012. *Network Morphology: A Defaults-based Theory of Word Structure*. Cambridge: Cambridge University Press.

Bybee, Joan L. 1985. *Morphology: A Study of the Relation between Meaning and Form*. Amsterdam/Philadelphia: John Benjamins.

Carstairs, Andrew. 1983. Paradigm economy. *Journal of Linguistics* 19: 115–125.

Carstairs, Andrew. 1987. *Allomorphy in Inflexion*. London: Croom Helm.

Chan, Erwin. 2006. Learning probabilistic paradigms for morphology in a latent class model. In *Proceedings of the Eighth Meeting of the ACL Special Interest Group on Computational Phonology at HLT-NAACL 2006*, 69–78. New York City.

Chan, Erwin. 2008. *Structures and Distributions in Morphology Learning*. Ph.D. thesis, University of Pennsylvania.

Chomsky, Noam. 1957. *Syntactic Structures*. Paris: Mouton Publishers, The Hague.

Corbett, Greville G. and Norman M. Fraser. 1993. Network Morphology: a DATR account of Russian nominal inflection. *Journal of Linguistics* 29: 113–142.

Crystal, David. 2008. *A Dictionary of Linguistics and Phonetics*. Wiley-Blackwell, 6th ed.

Dammel, Antje. 2009. How – and why – do inflectional classes arise? A case study on Swedish and Norwegian conjugation. In Fabio Montermini, Gilles Boyé, and Jesse Tseng (eds.), *Selected Proceedings of the 6th Décembrettes*, 12–21. Somerville, MA: Cascadilla Proceedings Project.

Dreyer, Markus. 2011. *A non-parametric model for the discovery of inflectional paradigms from plain text using graphical models over strings*. Ph.D. thesis, Johns Hopkins University.

Dreyer, Markus and Jason Eisner. 2011. Discovering morphological paradigms from palin text using a dirichlet process mixture model. In *Proceedings of empirical methods in natural language processing*, 616–627.

Durrett, Greg and John DeNero. 2013. Supervised learning of complete morphological paradigms. NAACL.

Finkel, Raphael and Gregory T. Stump. 2007. A default inheritance hierarchy for computing Hebrew verb morphology. *Literary and Linguistic Computing* 22(2): 117–136.

Finkel, Raphael and Gregory T. Stump. 2009. Principal parts and degrees of paradigmatic transparency. In James P. Blevins and Juliette Blevins (eds.), *Analogy in Grammar: Form and Acquisition*, 14–54. Oxford: Oxford University Press.

Garvin, Paul L. 1964. *On Linguistic Method*. The Hague: Mouton & Co.

Goldsmith, John A. 2000. Linguistica: An automatic morphological analyzer. In John Boyle, Jung-Hyuck Lee, and Arika Okrent (eds.), *Papers from the 36th Annual Meeting of the Chicago Linguistic Society, Main Session*. Chicago: Chicago Linguistic Society.

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

Goldsmith, John A. 2004. From algorithms to generative grammar and back again. In *Proceedings of the 40th Annual Meeting of the Chicago Linguistic Society*. Chicago: Chicago Linguistic Society.

Goldsmith, John A. 2006. An algorithm for the unsupervised learning of morphology. *Natural Language Engineering* 12(4): 353–371.

Goldsmith, John A. 2007. Towards a new empiricism. *Recherches Linguistiquesa Vincennes* 36: 9–36.

Goldsmith, John A. 2009. Morphological analogy: Only a beginning. In James P. Blevins and Juliette Blevins (eds.), *Analogy in Grammar: Form and Acquisition*, 138–164. Oxford: Oxford University Press.

Goldsmith, John A. 2010. Segmentation and morphology. In Alexander Clark, Chris Fox, and Shalom Lappin (eds.), *Handbook of Computational Linguistics and Natural Language Processing*, 364–393. Oxford: Wiley-Blackwell.

Goldsmith, John A. 2011a. The evaluation metric in generative grammar. Paper presented at the 50th anniversary celebration for the MIT Department of Linguistics.

Goldsmith, John A. 2011b. A group structure for strings: Towards a learning algorithm for morphophonology. Technical Report TR-2011-06, Department of Computer Science, University of Chicago.

Goldsmith, John A. and Jeremy O'Brien. 2006. Learning inflectional classes. *Language Learning and Development* 2(4): 219–250.

Goldsmith, John A. and Aris Xanthos. 2009. Learning phonological categories. *Language* 85(1): 4–38.

Halle, Morris and Alec Marantz. 1993. Distributed Morphology and the pieces of inflection. In Kenneth Hale and Samuel Jay Keyser (eds.), *The View from Building 20: Essays in Linguistics in Honor of Sylvain Bromberger*, 111–176. Cambridge, MA: MIT Press.

Hammarström, Harald. 2009. *Unsupervised Learning of Morphology and the Languages of the World*. Ph.D. thesis, Chalmers University of Technology and University of Gothenburg.

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

Harris, Zellig S. 1955. From phoneme to morpheme. *Language* 31(2): 190–222.

Haspelmath, Martin. 2002. *Understanding Morphology*. London: Arnold, 1st ed.

Haspelmath, Martin and Andrea D. Sims. 2010. *Understanding Morphology*. London: Hodder Education, 2nd ed.

Hockett, Charles F. 1954. Two models of grammatical description. *Word* 10: 210–34.

Jensen, John T. 1990. *Morphology: Word Structure in Generative Grammar*. Amsterdam/Philadelphia: John Benjamins.

Jurafsky, Daniel and James H. Martin. 2006. *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*.

Lee, Jackson L. and John A. Goldsmith. In prep. Algorithmic approaches to inflectional stem identification.

Lieber, Rochelle. 2010. *Introducing Morphology*. Cambridge: Cambridge University Press.

MacWhinney, Brian, Jared Leinbach, Roman Taraban and Janet McDonald. 1989. Language learning: Cues or rules? *Journal of Memory and Language* 28(3): 255–277.

Matthews, Peter H. 1972. *Inflectional Morphology*. Cambridge: Cambridge University Press.

Matthews, Peter H. 1991. *Morphology*. Cambridge: Cambridge University Press, 2nd ed.

Matthews, Peter H. 2007. *Concise Dictionary of Linguistics*. Oxford University Press, 2nd ed.

Monson, Christian. 2009. *ParaMor: From Paradigm Structure to Natural Language Morphology Induction*. Ph.D. thesis, Carnegie Mellon University.

Müller, Gereon. 2007. Notes on paradigm economy. *Morphology* 17: 1–38.

Pertsova, Katya. To appear. Machine learning of inflection. In Matthew Baerman (ed.), *Oxford handbook of inflection*. Oxford University Press.

Prince, Alan and Paul Smolensky. 1993/2004. *Optimality Theory: Constraint Interaction in Generative Grammar*. Wiley-Blackwell.

Rissanen, Jorma. 1989. *Stochastic Complexity in Statistical Inquiry*, vol. 15, Series in computer science. Singapore; Teaneck, N.J.: World Scientific.

Spencer, Andrew. 1991. *Morphological Theory*. Oxford, England: Basil Blackwell.

Spencer, Andrew. 2013. *Lexical Relatedness: A Paradigm-based Model*. Oxford: Oxford University Press.

Stump, Gregory T. 2001a. Default inheritance hierarchies and the evolution of inflectional classes. In Laurel Brinton (ed.), *Historical Linguistics 1999*. Amsterdam: Benjamins.

Stump, Gregory T. 2001b. *Inflectional Morphology: A Theory of Paradigm Structure*. Cambridge: Cambridge University Press.

Stump, Gregory T. and Raphael A. Finkel. 2013. *Morphological Typology: From Word to Paradigm*. Cambridge: Cambridge University Press.

Trask, R. L. 1999. *Key Concepts in Language and Linguistics*. London: Routledge.

Yu, Alan C. L. 2007. *A Natural History of Infixation*. Oxford: Oxford University Press.

Zeman, Daniel. 2008. Unsupervised acquiring of morphological paradigms from tokenized text. In *Advances in Multilingual and Multimodal Information Retrieval: 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007*, 892–899. Budapest.

Zeman, Daniel. 2009. Using unsupervised paradigm acquisition for prefixes. In *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarthus, Denmark, September 17-19, 2008, Revised Selected Papers*, 983–990. Springer-Verlag, Berlin.