Computational Morphology in Low Resource Settings

Synthesis Paper

Sarah R. Moeller

1 Introduction

This paper examines how computational methods can be applied to the analysis of morphology in low resource contexts, particularly with language documentation and description (LDD) data.[1] It attempts to identify "[w]hat [computational] methods…can [and have been used to] detect [morphological] structure in small, noisy data sets, while being directly applicable to a wide variety of languages" (**?**, page 472). Such methods have the potential speed and improve language documentation and language description and to determine where natural language processing (NLP) methods could grow beyond the current state of the art.

Morphology comprises word-building properties in human languages and their accompanying (morpho-)syntactic phenomena. Historically, computational linguists and "paper-and-pencil linguists" have taken different and sometimes seemingly incompatible approaches to morphology (**?**, page 80). In general linguistics, analyzing the morphological system is a first step in describing a language. Morphological description of a broad range of languages must support any reasonable linguistic theory.

Computational morphology attempts to parse or generate valid inflected forms according to the language's morphological structure in order to improve downstream NLP tasks such as ma-

MH

Not sure if it's a good idea to center the overview of your paper around a verbatim quote. It leads the reader to assume someone else has done this before(?).

[1] "Language documentation and description" (LDD) data refers to resources that results from linguistic fieldwork and basic linguistic analysis. "low resource languages" (LRL), according to LORELEI (a US-government funded project for building language technology in low resource languages), refers to languages for which no automated human language technology exists, typically because of a lack of available linguistic resources. Other terms are sometimes used: "under-described languages" could be described as having minimal published descriptive linguistic resources, what **?**) calls "very scarce-resource language;" "under-documented languages", or Duong's "extremely scarce-resource languages," lack sufficient raw or annotated data to write a full descriptive grammar; "endangered languages" are predicted to have no native speakers within a generation or two and most are under-documented and/or under-described. These distinctions are rarely crucial in this paper and can be understood almost interchangeably.

chine translation or voice recognition. Among some computational linguists there seems to be a belief that linguistic theory has little benefit for computational methods (**?**). Hypothesizing an all-encompassing theory of language does not induce from data a workable NLP tool. It has even been questioned whether computational linguistics needs morphological analysis at all, particularly with the recent trend of end-to-end training which have the potential to improve downstream tasks without explicitly modeling morphology. Might NLP goals benefit equally well by representing words as strings of characters? Although character-level models can learn relationships between orthographically similar words, comparing results on ten language shows that such models are too easily led up the garden path by orthographic signals (**?**). Language models trained on morphological patterns provide greater predictive accuracy.

Despite their out-of-sync approaches, both computational linguistics and "traditional" linguistics benefit from morphological analysis (**?**, page 165). What's more, the field of linguistics is not interested solely in establishing a theory of language. Early linguistics focused on increasing human understanding of language by describing a language's structure, including morphology, using recorded data. This focus has been revived since the 1990's with language documentation, a subfield that is also keenly interested in practical downstream goals such as language technology, language conservation, and language learning.[2]

MH

Why?

Morphological analysis is particularly important for morphologically complex languages, such as agglutinating (common in central and north Asia, South America, central and southern Africa, Australia), polysynthetic (North America, the Far East of Russia), or non-concatenative (north Africa and the Middle East, southeast Asia). Languages that build words from multiple morphemes or via significant morphophonological changes produce a high number of inflected and compound words which appear to the machine as brand new, unrelated words (**?**; **?**; **?**; **?**; **?**). The fact that computational linguists feel able to ask whether morphological analysis is even necessary hints at the out-sized role that so-called high resource languages have played in natural language processing. The most common languages used in computational linguistics are European languages,

[2] A casual perusal of the flagship journal *Language Documentation and Conservation* journal attests this interest.

Chinese, and Arabic, most of which have comparatively simple fusional or isolating morphology, with a very few (e.g. Finnish) that belong to the somewhat more complicated agglutinative or non-concatenative types (i.e. Arabic). None belong to the much more complicated polysynthetic type, none are under-documented, none are endangered languages. While field of linguistics has slowly but surely widened the world's knowledge about human language, thanks in part to a recent emphasis on documentary fieldwork, computational linguistics has yet to truly expand beyond a handful of economically or politically powerful languages. As an example, when **?**) compared morpheme-level versus character-level representation, they selected 10 languages that in fact represent quite a wide range of morphological phenomena, but still only from six language families. Three are Indo-European languages characterized by fusional morphology (though English tends toward the even simpler isolating type). One (Finnish) is a European, but not Indo-European, language with agglutinative morphology and many other interesting morphological and phonological alternations. Four others (Japanese, Turkish, Indonesian and Malaysian) are also agglutinative. The other two languages (Hebrew and Arabic) are Semitic languages, famous for the templatic type of non-concatenative morphology. Even this narrow dataset demonstrated that the more complex a language's morphology is, the more crucial the role that morphological analysis plays in the final results.

The rest of this paper assumes that morphology does have a role to play in both computational and general linguistics. Section 2 defines and compares the tasks of morphology learning in computational linguistics and language documentation and description. This paper also assumes that NLP algorithms and methods should be expanded into more low resource languages. Section 3 reviews the literature on various computational approaches to morphology, how they have been applied to low resource languages, and what their advantages and disadvantages are in low resource settings.

Two issues arise throughout the paper. First, how can machine learning be improved when data is limited? Second, what are the most promising computational methods for LDD? In addition to the reviews in section 3, section 4 presenting specific techniques that have overcome data limitations.

Section 5 addresses the second question but goes beyond specific methods by envisioning how computational linguistics and language documentation and description could together address the issue of limited data which plagues both fields.

2 Morphology

Historically, the study of morphology is the inference of rules that govern a language's word building strategies including morphophonological changes in morpheme shapes (pronunciation), and the discovery of systematically related word forms (derivational morphology and inflectional paradigms) (**?**). According to **?**), an ideal morphological model answers the following questions:

- What are the morphemes in each word? What are their meanings or functions?
- What morphological paradigms exist in the language? What morphological features distinguish them from one another?
- What allomorphs, or alternative pronunciations, does each morpheme have? Under what conditions do the allomorphs appear?
- What combinations of morphemes does the language permit on each lexical category (part of speech (POS))?
- What are the morphological processes that significantly change a word's meaning or part of speech? How productive are these derivational processes?

Such a complete morphological model is difficult to accomplish and to evaluate computationally. It would need to be computable, robust, interpretable, while accounting for all the language's morphophonology, allomorphy, and for any ambiguous inflected forms (**?**). In other words, a computational model that reaches the ideal would be nearly as good as human descriptive linguists, who are also able to account for the ever-present inconsistencies and unsystematic exceptions that occur in natural languages. While descriptive linguists attempt to answer all the above questions (and more), most effort in computational morphology has been expended on the first set of questions, with some significant effort given to the second. This section compares how computational linguistics and language documentation and description (LDD) approach these two sets of questions and their related tasks.

Attempts to answer the first set of questions about the number, shape, and meaning of morphemes in a language is the core part of morphological analysis. Traditionally, morphemes are first identified, for example, *-ed* would be recognized as a word segment with its own minimal meaning. Whether or not individual morphemes are identified, the meanings, or functions, that each morpheme contributes to a word must be deduced. For example, *-ed* can mean 'PAST' or 'PARTICIPLE'. After this, it becomes possible to examine questions about morphological paradigms, delving deeper into a language's morphology. Systematic rules that govern how morphemes combine on a word and which can or cannot co-occur are inferred from the analyzed data. For example, once the morpheme is identified in natural occurring texts, it should be clear that English uses *-ed* as a suffix to add past tense or participial meaning to a verb stem. Subsets of rules may govern certain classes of morphemes and these classes and their morphological patterns can be exemplified by paradigms such as Table 1. Thus, for example, the class of "regular" English uses the suffix *-ed* (versus "irregular" verbs: *swim, swam, swum*, etc.).

MH

I wonder if a caveat about orthographic and phonological representations would be motivated somewhere early on? For example, when you talk about a suffix -ed, I would like to know whether you're referring to the orthographic form or its phonetic allomorphs (as an abstraction). Of course, in NLP we're dealing with

2.1 Morphological Analysis and Annotation

Morphological analysis can be separated into two core tasks (**?**; **?**; **?**; **?**). The first task is identifying morphemes by determining their shapes and marking boundaries between them, as done for the Lezgi [lez] noun in (1b). The task is sometimes called (unlabeled) morpheme segmentation (**?**; **?**). The second task is deducing each morpheme's meaning, known as parsing, or sometimes, by itself, as morphological analysis. If morpheme segmentation is done first, the second task usually involves associating each morpheme with a label (gloss or tag) that indicates its meaning or function as, for example, the OBL (oblique stem) and GEN (genitive case) in (1c). This is sometimes known as glossing, tagging, or labeled morpheme segmentation. These two tasks are a large part of linguistic data annotation.

(1) a. paçahdin

b. paçah-di-n

c. king-OBL-GEN

d. ‘king’s’

Discovering morphemes’ meanings does not necessarily require the first task: morpheme identification and segmentation. It is possible to parse a word’s meaning without identifying the individual morphemes which compose that meaning.[3] Performing the second task without the first task is much more common in computational approaches. Computational morphology aims to take texts from any human language and build a model that accounts for every word in them and generalizes with high accuracy to unseen words in new texts (**?**). If the immediate task can be accomplished without it, a model may skip morpheme identification and segmentation, as, for example, in one CoNLL-SIGMORPHON 2018 shared task, shown in (2), where words were inflected without identifying the language’s morphemes (**?**).

(2) a. **INPUT:** The _____ are barking. *dog*

b. **OUTPUT:** dogs

?) divide the morphological analysis subtasks slightly differently, making a distinction between morphological “analysis” and morphological tagging. They describe morphological analysis as a combination of segmentation and labeling, but later state that “morphological tagging can be performed as a downstream application of morphological analysis” **?**, page 211), thereby adhering the same two distinctions described above. **?**) adds a third task under morphological analysis: identification of morphologically related words. Identifying morphologically related words is essentially a first step in identifying inflectional classes and the systematic rules that govern them (see section 2.2.

Computationally, morpheme segmentation algorithms are quite similar to word segmentation. Segmentation provides a lexicon of morphemes which is more generalizable than a vocabulary of word forms as they appear in running text (**?**). Segmentation divides strings of text without regard to their potential relation (**?**). Computational approaches can be lexicon-based, focusing on

[3] One field linguist (p.c.) claims the reverse is possible - native speakers could segment words into their minimally meaningful parts without being able to identify those meanings. But it does not seem likely that this would achieve good results since the speaker could not grasp the semantic or syntactic factors that determine a morpheme’s shape.

detecting morpheme shapes or the approach can be a model that focuses on detecting morpheme boundaries (**?**). Most lexicon-based approaches learn a generative model. They create a model of word forms and can generate them. Most boundary detection models perform discriminative learning, probabilistically estimating the segmentation boundaries in a given word. Computational morphological segmentation tends to perform better on concatenative morphology, where words are built from morphemes like beads on a string such as in example 1. Computational modeling of non-concatenative morphology, particularly when it occurs in a language that combines with concatenative morphology, remains "arguably one of the main current challenges of the field" and attempts "have been mostly applied to Arabic" (**?**).

During morpheme segmentation, the machine is not usually tasked to identify allomorphy, though some learning of allomorphy and morphophonogical changes may happen before this step (**?**). Ideally, the second task of morphological parsing or labeling accounts for allomorphy and other complicated morphophonogical issues. This is difficult to do without some previous analysis so it is no surprise that the most accurate, and historically most popular, computational morphological models are finite state transducers (FSTs) because they can take advantage of published linguistic descriptions and manually construct a complete morpheme lexicon and collection of a language's morphophonological rules.

In descriptive linguistics, the two tasks are not usually distinguished because they are typically tackled simultaneously. It is more likely that general linguistics would categorize as morphological analysis all the tasks leading to the identification of morphemes, their meanings, and the description of systematic rules and relationships between words. The mechanical parts of these tasks—segmenting and glossing morphemes—are sometimes considered together as (morphological) analysis or annotation. Those steps, together with rough translations of sentences is called interlinearization illustrated in Figure 2.1 (see section 5.

In computational morphology, the two tasks, if both are attempted, are often tackled sequentially. First, the machine finds morpheme boundaries, then attaches labels. In a narrow sense, since a machine cannot make analytical decisions that a human can (though it can find complicated patterns

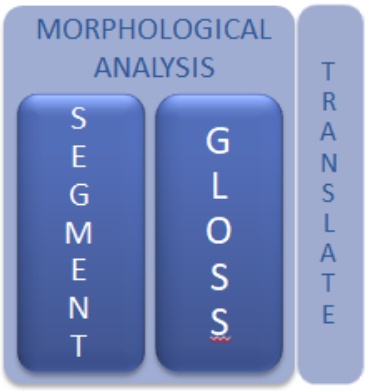

Figure 1: Interlinearization. Words are segmented and morpheme boundaries are marked. Each morpheme is glossed (parsed). Finally, each sentence is translated in a language of wider communication to clarify how the morphemes compose meaning.

and make predictions which may seem like analytical decisions), all computational morphology could be seen as annotation. However, some computational models have taken a tip from LDD workflow and united the two tasks. In some cases, this yields higher accuracy. It also allows the machine to go beyond annotation and "learn a probabilistic model of morphotactics [rules about the ordering and co-occurrence of morphemes on a word]" (**?**, page 165). Morphological segmentation and tagging should be distinguished from syntactic annotation, of which the most common type is part-of-speech (POS) tagging, but this distinction can be problematic because some POS tag sets such as the expanded set of Universal Dependency tags (**?**) include morphologically-marked syntactic features, such as number or grammatical case. In fact, "morphological" tag sets and "morphological" annotation could benefit from even more syntactic information (**?**). LDD distinguishes morphological annotation from syntactic annotation even less strictly than computational linguistics. IGTs often include lexical categories (POS) and information related to syntax, phonology, and even non-verbal communication. The fact is that the divisions (morphology, syntax, phonology, etc.) are convenient for linguistic science but are not as clear in natural language. The division between morphology and syntax is more obvious in the relatively simple morphological structures that many high resource languages have, such as the fusional morphology of most European languages or the isolating morphology of Chinese languages. Both fusional and isolating morphologies use independent function words (e.g. prepositions, particles, etc.) rather than

bound morphemes to express a significant amount of syntactic information. Even some more complex morphological structures, such as the non-concatenative, templatic Arabic, also expresses a fair amount of syntactic information with independent words. These strategies stand in contrast to agglutinative and polysynthetic morphology characterizing many low resource languages, which express most or all syntactic information by bound morphemes on the inflected word forms.

2.2 Morphological Paradigm Induction

The study of a language's morphology begins with the analysis of morphemes but extends to the inference of the system of rules that dictate how morphemes build words and interact with each other. This inference is based on three assumptions (**?**). First, within a lexical category groups of lexemes inflect according to a pattern that is dictated by subsystem of rules. Russian nouns, for example, can be generalized into three simplified patterns of inflection, shown in Table 1. Patterns in a table like this is sometimes called inflectional paradigms. Second, these patterns are triggered by context and the morphological rules governing the patterns can be inferred from the triggering context. Descriptive studies look to phonology for the triggering context or some combination of phonological structure and the semantic content of the lexemes. Computational models, due to the nature of their input texts, hinge upon orthographic context. Third, a stem morpheme is consistently inflected. Understanding a language's morphology means describing inflectional patterns and grouping lexemes that inflect according to a subsystem of rules. These groups of morphemes are inflectional classes (sometimes called "declensions" for nouns and adjectives and "conjugations" for verbs). A class's complete inflectional pattern is described by an inflectional paradigm which display either just the inflectional affix(es), as in Table 1, or a lexeme with all its inflected forms.

These tables are challenging to induce from raw data. Within one language, inflectional classes are not always unique or regular, they may have overlapping patterns and isomorphic morphemes that result in ambiguous forms. For example, an Arapaho verb ending in *-ót* might mean "You alone do something to <u>him/her/it</u>" or it might mean "You alone do something to <u>them</u>." Some languages use suppletive forms that have no similarity in shape to their "base" form (e.g. English

	Class 1		Class 2		Class 3	
	SG	PL	SG	PL	SG	PL
NOM	-a	-ɨ	Ø	-ɨ	-j	-i
ACC	-u	-ɨ/Ø	Ø/-a	-ɨ/-ov	-j	-i/-jej
GEN	-ɨ	Ø	-a	-ov	-i	-jej
DAT	-je	-am	-u	-am	-i	-jam
INST	-oj	-ami	-om	-ami	-ju	-jami
PREP	-je	-ax	-je	-ax	-i	-jax

Table 1: Example of Inflectional Paradigms for Russian Nouns. The second row and leftmost column indicate the morphosyntactic features that each slot represents.

"go" vs "went"). Field linguists can refer these difficulties to native speakers but machine inputs are typically limited to written texts sourced from published material.

?) state two guiding principles for computational paradigm induction that are applicable for any approach that is limited to textual input. First, "in any given corpus, a particular lexeme will likely not occur in all possible inflected forms". Even with a large corpus, attempts to recreate a paradigm as in Table 1 will result in empty gaps. Language documentation best practice field methods encourages elicitation of inflectional paradigms in focused sessions—despite the subfield's emphasis on natural language—precisely because complete paradigms so rarely appear in natural language (**?**). Certain inflectional forms will be much more common than others. In fact, for some lexemes, certain inflectional forms may never occur in spoken language even though they are grammatically possible (**?**). It may be possible to find all inflectional forms of the most frequent lexemes, but frequent words may follow irregular patterns such as the English "be" verb. Therefore, paradigms need to be completed by inducing the patterns from several incomplete paradigms. The second principle is that we expect inflected forms of a single lexeme to be similar. This second principle of consistency plays an important role when identifying related words forms. It is not always true—languages abound with exceptions—but it is a solid working assumption.

Descriptive linguistics infer inflectional paradigms from annotated data. An ideal analysis strives to describe all possible patterns of inflection. Theoretically, this means that every inflectional form of every word needs to be examined, but in practice, paradigmatic patterns can be inferred fairly

quickly and the linguist can then concentrate on matching lexemes to a paradigm and identifying irregularities. Most analysis is done with spreadsheets and charts in computer programs such as Microsoft Excel.

Computational models have successfully induced frequent and regular paradigms with high accuracy even in low resource settings (**?**; **?**; **?**). Most early work on computational morphological paradigm induction applied unsupervised learning to concatenative morphology (**?**; **?**; **?**). Supervised and semi-supervising models have been more recently applied on concatenative and non-concatenative languages (**?**; **?**). ParaMor (**?**) is an example of an unsupervised model. It begins by searching the data for sets of strings that resemble inflectional paradigms. It does this by identifying candidate "suffixes" in the word-final substrings. A candidate suffix is a substring that repeats at the end of different strings. The sets are refined to become partial paradigms. Inflectional paradigms induced by unsupervised learning like this has three flaws. First, suffixes may be in multiple paradigms because suffixes overlap between partial paradigms, for example "-s" on an English word may mark plurality on a noun or third person singular subject agreement and present tense on a verb. Second, most predicted partial paradigms contain "many fewer candidate suffixes than do the true paradigms" (**?**, page 903). This is another way of stating Monson's first principle: induction from a corpus of natural text leaves gaps in paradigms because some inflected forms are rare or never used and some are very frequent. Third, some suffixes are inevitably identified at incorrect morpheme boundaries. For example, the English word "ally" could be easily misidentified as a root "all" plus the adverbializing suffix "-y". **?**) solve the first two flaws by leveraging Monson's second principle of relatedness and similar inflection and clustering candidate suffixes with a similarity score. The third flaw is addressed by filtering paradigms that do not include a threshold number of forms.

MH

It sounds like a little more detail about the approach would be warranted…

Paradigm learning with supervised machine learning means at least partially complete paradigm tables are needed to train a model. A common source of inflectional tables is Wiktionary, a crowd-sourced online dictionary. Wiktionary contributors usually include an inflectional table for each entry. Sometimes, if known, the lexeme is identified with its paradigm or inflectional class. Some

supervised learning has been applied to tasks related to paradigm learning but that are not direct attempts to identify a language's inflectional classes but focus instead on generating correct inflected forms. For example, **?**) applied supervised deep learning in the form of recurrent neural networks to answer the Paradigm Cell Filling Problem (PCFP) (**?**), illustrated in Figure 2.2, which asks how new speakers infer the inflected forms of a lexeme when they have not seen all forms? Malouf et al.'s model took as input the "base" form represented as a one-hot vector and the morphosyntactic features tags of the inflected form to generate. For example, if the input is a representation of "walk" and the features tense = PRES, person = 3, number = SG, then the correct output is "walks". The model was successfully applied to Irish, Maltese, and Khaling [klr], as well as some major languages. One interesting observation was that regularization, which normally keeps the model from overfitting, was found to harm performance because the less frequent inflectional patterns look like noise. Regularization tends to guide the model away from those patterns, when it actually needs to learn them. **?**) also investigated the PCFP. With the same amount of input their results were very similar Malouf et al.'s but their model is more realistic for LDD. They do not use the lexeme, which, in morphologically complex languages, is rarely found in naturally-occurring texts. Instead, they trained on inflected forms, which could be extracted from a LDD corpus.

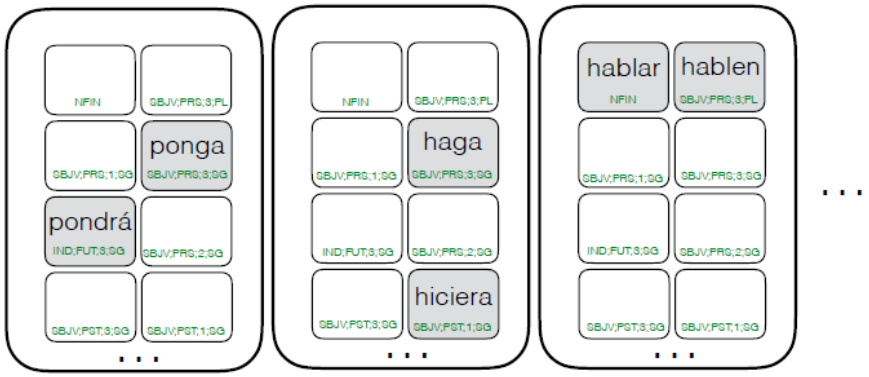

Figure 2: **?**): Illustration of the Paradigm Cell Filling Problem with Spanish verb inflectional tables. If speakers are only exposed to such partially filled paradigms, how do they fill the missing cell with the correct form?

?) use a deep learning approach to morphological re-inflection. Re-inflection attempts to predict

MH

Crucially, it was a character-based model: it would learn to inflect by translating a string to another string. Malouf's model never used strings a input, just a "number" for the lexeme, and one-hot features for the slot. Hence, it could, for example not work

a correct inflected form from another form or from morphosyntactic tags or *vice versa*. This is essentially the same task as the approaches to the PCFP, but without reference to the theoretical question, and therefore, is not compelled to limit the input: Kann et al. used completed inflection tables. Their approach does not learn inflectional patterns directly but it outlines possible first steps toward morphological paradigm induction in low resource settings. The model draws inspiration from the theoretical linguistic notion of principal parts (**?**) which refers to minimal subset of forms that allow maximum predictability. This subset can be thought of as the smallest number of inflected forms needed to identify the lexeme's inflectional class. Kann et al. added encoders that incorporate multiple inflected forms, with the assumption that these multiple forms provide complementary information about a pattern and allows the model to predict a complete paradigm. Such "multi-source" learning performs better than a single data source by working around "holes" in data. It can be applied to fully or only partially annotated data.

A similar approach can be applied to low resource settings (**?**; **?**) using a small number of inflectional tables from Wiktionary which are similar to Table 1 but include the whole word instead of the just the inflectional morphemes. With this data, groups of similarly behaving words were extracted from annotated data. Several of these groups were used to identify paradigmatic patterns. As with Kann et al. the immediate goal is to predict the correct inflected forms of unseen words, but the generalized patterns used to make these predictions are basically inflectional paradigms. The patterns are discovered by abstracting the longest common subsequence in a word and clustering those abstract patterns with same or similar patterns. This is illustrated in Figure 2.2[4]. The three parts of the longest common subsequence (LCS) *rng* or *swm* are extracted from the inflectional table on the left (step 1) and represented abstractly, in this case as x_1 and x_2. The abstract symbols replace the longest common subsequence in every word form (step 2). In theory, the characters that remain, in this case *i*, *a*, *u*, are the inflectional morphemes. Since the unique part of the each root morphemes is now represented identically, words with the same inflectional patterns will be identical and can be generalized into paradigmatic patterns (steps 3 and 4). It seems quite

[4]Acknowledgements to Mans Hulden and Ling Liu.

possible that exceptions or irregularities could be accounted for by collapsing similar patterns. The experiment was quite successful on some Indo-European languages (German, Spanish, Catalan, French, Galician, Italian, Portuguese, Russian), as well as Maltese and Finnish.

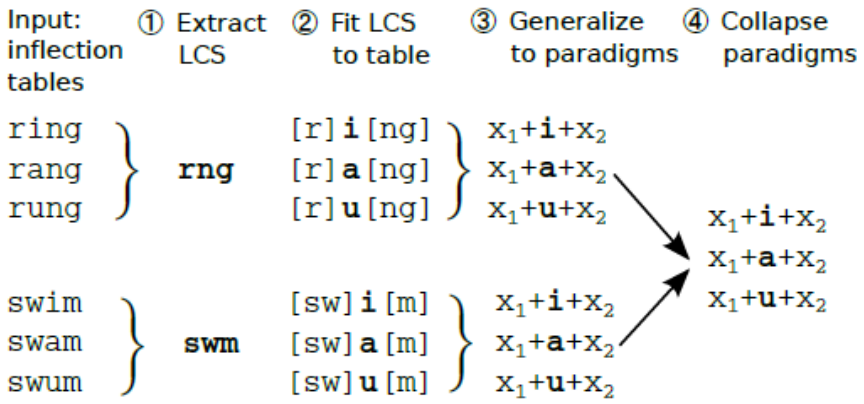

Figure 3: Ahlberg et al. (2015): Abstracting inflectional paradigmatic patterns from inflections and annotated data.

Attempts at paradigm induction has contributed to computational morphology in low resource contexts in a singular way: those attempting it have also attempted to estimate the smallest amount of data necessary to perform it with reasonable accuracy. **?**) found that paradigms from the 20 most frequent inflectional classes cover 95% of German word forms. **?**), who use hand-written rules to fill paradigm slots, started with complete inflectional tables and gradually dropped information from them. In the end, they found that one-two infected forms per paradigm are sufficient to achieve 90% accuracy in English, Swedish, French, and Finnish. **?**), using a neural model that learns from a small number of randomly chosen forms per paradigm table, found that two inflected forms gave about 85% accuracy in Finnish, Georgian, Turkish and some Indo-European languages and three forms bumped the results over 90% for all but Georgian nouns and Latvian verbs.

3 Approaches to Computational Morphology

This section explores the exciting development of computational morphology and its application to low resource languages. The four major approaches are either rule-based (finite state transducers) or machine learning approaches that induce models from texts. Machine learning strategies dif-

fer in whether the texts have been annotated completely (supervised), partially (semi-supervised), or not at all (unsupervised). Until the late-1990s and mid-2000s, the literature on computational morphology was dominated by finite-state machines (**?**; **?**; **?**). Gradually, machine learning started to become popular (**?**; **?**). Most early machine learning of morphology used unsupervised, probabilistic models (**?**). Supervised approaches came later, due in part to computer memory limitations (**?**). Semi-supervised learning has also grown in popularity, due perhaps in part to "the increased availability of machine-readable inflection tables" (**?**, page 18).

Currently, supervised end-to-end neural networks, or deep learning methods, are dominating the field. Although deep learning approaches have achieved high results in many NLP tasks, they did not become popular until recently because they train more slowly than non-neural methods for some tasks (**?**) and require greater computing power. More significantly, the models and methodology for training have greatly improved. They do require expertise to customize and this still presents a practical hurdle to efficient NLP application in LDD. General linguistics training does not include computer programming and, as yet, few, if any, easy-to-learn graphic user interfaces exist.

MH

Also, seq2seq is a recent development...

Computational morphological approaches have been tested on a growing number of languages since 2000 (**?**). The latest CoNLL-SIGMORPHON Shared Tasks (**?**; **?**) indicate how this number has grown. Nevertheless, the task's list of 100+ languages is still small in comparison to nearly 6,000 remaining languages. Nor has the number grown in a representative way. The distribution is not proportional to language families or basic morphological types. As an example, the 2018 CoNLL-SIGMORPHON task included only one language classified as polysynthetic, and that language, Navajo, is often classified as agglutinative or fusional.

It is worth noting that comparing the performance of different approaches, and the different models within one approach, is not straightforward. Morphological analysis involves many small steps; if any one step is the basis for evaluation, the assessment changes. For example, a predicted segmentation can miss the correct morpheme boundary by one letter or by many. Or, in labeled segmentation, precision and recall can be a "micro-average" measuring the accuracy of segmen-

tation or a "macro-average" measuring the accuracy of both segmentation and tagging. For the latter measure, the total count for precision, recall, and F1 measure could be based on types or on tokens. A word type count may obscure whether all types are handled identically. A word token count may make the model seem less accurate if one very frequent word is incorrectly parsed. Some evaluations may consider a parse correct when it produces one of all possible ambiguous parses; others may count a correct parse only when it is the right one in the given context (**?**). **?**) surveyed these issues through five years of MorphoChallenges and found no solution to the varying details, but they concluded that an evaluation based on segmentation is the most simple, robust, and intuitive. Beyond the difficulties caused by varying standards of comparison, computational linguistics evaluates models in two ways. The first is indirect, or extrinsic evaluation, which evaluates the effect a model has on a complete NLP system. This is complicated and time-consuming and so rarely done. Direct, or intrinsic, evaluation is can be performed either automatically against gold standard, annotated data or manually by native speakers or domain expert linguists. **?**, page 53) dismiss manual evaluation because human expert decisions are "subjective" and "the amount of work involved restricts its usage." This disregards the fact that any gold standard data was originally assembled with the same amount of human subjectivity and hard work. Such attitudes reveal the drawback of intrinsic evaluation. It sets a narrow focus on how a model performs competitively against other models on the same dataset, rather than how it performs across datasets or languages.

MH

This definition of micro/macro average sounds misleading. Usually, macro = average the individual percent accuracies over, say, many languages, micro = collect all counts correct/incorrect over all languages, then produce one accuracy score. I'm not sure I follow your interpretation.

3.1 Finite State Transducers

Finite state transducers (FSTs) are bidirectional, rule-based models that have successfully modeled linguistic patterns (**?**; **?**; **?**). They were the first computational model to successfully both parse and generate word forms (**?**) (see Figure 3.1). Since they easily represent regular relations between underlying and surface forms, moving left-to-right, they held early importance as a way to represent **?**)'s rewrite rules (**?**) and model the theory of two-level morphology.

FSTs are constructed in two steps. The first step specifies the lexicon and morphotactics in a finite-state lexicon compiler (*lexc*), illustrated in Figure 3.1. This is where concatenative morpho-

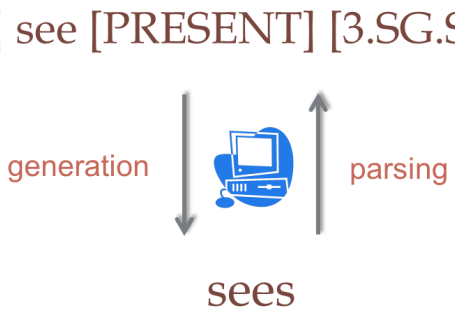

Figure 4: A bidirectional computational morphological model generates an inflected word form from a stem and morphosyntactic tags and parses inflected words forms into stem and tags.

logical rules and morphological irregularities are addressed. The second step implements morphophonogical rewrite rules. These rules apply changes in specified contexts, allowing the FST to move beyond simple concatenation of morpheme strings and generate well-formed inflected surface forms. For example, a rule would specify that the final letter in the Arapaho stem *noohow* 'see' transforms into a *b* if a vowel-initial morpheme is suffixed after it.

```
LEXICON Stem
...
@U.StemType.TA@@U.StemInitial.Yes@noohow@U.StemFinal.Yes@:
@U.StemType.TA@@U.StemInitial.Yes@^noohow^@U.StemFinal.Yes@        OrderInflection;
...
```

Figure 5: A snippet of an Arapaho *lexc* file for one stem morpheme *noohow* 'see'. The @ symbols surround flag diacritics that communicate how the rewrite rules should handle transformations unique to the stem's inflectional class or other exceptional morphophonological behavior.

FSTs remain important to computational morphology. As recently as 2017, it was claimed that computational morphological learning methods are essentially still finite state methods (**?**), They have been found to be superior for many applications over a semi-CRF supervised machine learning model (**?**). With enough linguistic expertise and time for development, FSTs are capable of correctly analyzing any well-formed word in a language. However, FST "grammars" take significant effort to develop, maintain, and update (**?**; **?**). Since the "grammar" must be manually-constructed, FSTs require a thorough knowledge of the language and finite state machines, although in at least one case a FST morphological analyzer has been constructed from labeled inflection tables; it was,

however, limied to inflectional morphology (**?**).

FSTs work well with languages that have strong constraints on how each lexical category (POS) can be inflected, such as Arapaho, with its unique polysynthetic verb forms. But while the bulk of the FST can be developed relatively quickly with basic descriptive resources (e.g. the Boasian Triad), its reliance on language-specific lexicons and rules means that a full FST grammar needs a language that has been thoroughly documented and described. FSTs can be made to generalize to unseen forms, and if both descriptive resources and a language expert trained to construct the *lexc* and rewrite rules are available, an FST can be a good solution for simple morphological modeling. However, FSTs do not help resolve ambiguity because they output all possible outputs instead of ranking them by probability given surrounding context. An FST's output is always correct but sometimes it more practical for downstream tasks to have one "best guess". FSTs cannot be used for active learning easily since they do not learn patterns from new annotated examples. These issues can be resolved with machine learning.

3.2 Unsupervised Machine Learning

Unsupervised (computational) learning of morphology (ULM) began as an attempt to prove American structuralism and Bloomfield's "inductive generalizations," as well as a possible language acquisition model (**?**). ULM models attempt to induce morphological patterns from raw, unannotated texts. They take as input natural language data and, with as little supervision (i.e. parameters, thresholds, human intervention, model selection, etc.) as possible, outputs a "description" of the language's morphological structure, according to **?**). However, their use of the word "description" is a bit misleading. What ULM produces is only a first step to morphological description in general linguistics. **?**, pp. 33-34) depicts ULM more accurately as exploiting the "latent structure [in the data] alone to find meaningful patterns" or features. It outputs the data organized "in a meaningful way", usually via a clustering algorithm that finds natural groupings within the data.

According to **?**), unsupervised morphological models have followed three main stages of development. Early unsupervised algorithms drew inspiration from Z. Harris (**?**; **?**), who extended a descriptive methodology pioneered by Bloomfield that focuses on what elements in a language can

co-occur or not. Harris observed that phonemes do not co-occur unpredictably. The probability of the second phoneme in a word is dictate by the first, the third is dictated by the first two, and so on, illustrated in Figure 3.2. **?**) was the first to use character predictability to identify morpheme boundaries. Second, models inspired by Harris used the Minimum Description Length principle (MDL). At the third stage of ULM development, models began to leverage patterns akin to inflectional paradigms in order to find inflectional relationships between words and identify their affixes.

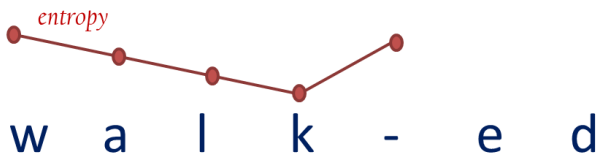

Figure 6: Harris' character predictability/entropy. In the word form "walked" the possible letters that can follow the "k are more than those that could follow the "l", so we predict a morpheme boundary.

MDL remains a leading method for unsupervised learning of morphology (**?**). It extends the Kolgomorov complexity that asks, in essence, "What's the shortest (in terms of memory) computer program that generates some text?" MDL simplifies this somewhat abstract question into a weaker but computable model. MDL-based approaches attempt to model the morphological structures by minimizing the memory cost of the input data and the model together. A proposed model encodes hypothesized morpheme strings with binary codes (essentially, a encoded lexicon or "codebook" of morpheme candidates) and replaces those strings with the codes whenever they appear in the input texts. The memory cost of the "codebook" and the encoded texts are calculated. The best morphological model is the one that where the cost of the "codebook" plus the encoded texts have the smallest possible memory cost (see Figure 3.2)[5]. Measuring the cost of the model inhibits overlearning and measuring the cost of the data, where the most frequent morphemes have the shortest codes, encourages frequent strings to be identified as repeated morphemes. Two downsides to MDL are that it provides no hint where the model should start looking for morpheme boundaries

MH

I think the costs shown in this figure is the cost of the MDL description of the unsupervised corpus used,

[5]Generated at https://asr.aalto.fi/morfessordemo/

(what hypothesis it should start with) and it cannot it provide a coherent analysis across a whole language (**?**).

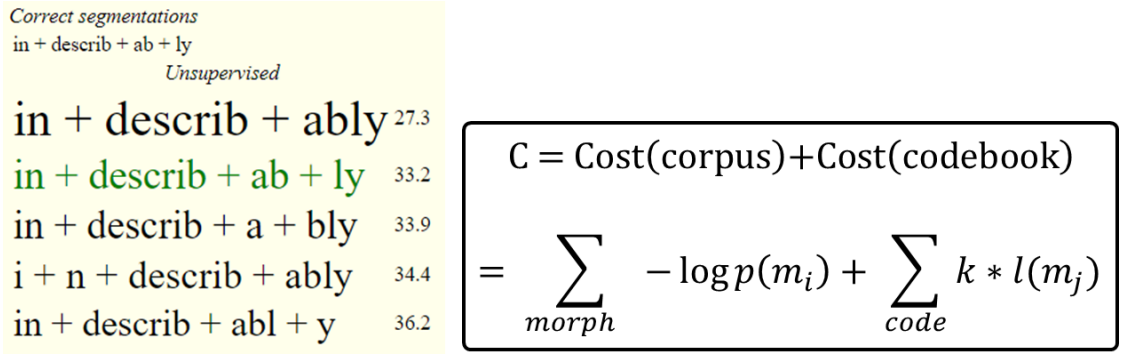

Figure 7: Minimum Description Length. The model makes random splits (marked by + on left) in a corpus, resulting in strings of candidate morphemes, or "morphs". A "codebook" of binary codes that represent each morph is constructed so that more frequent "morphemes" are represented with shorter codes, reducing the average cost per morpheme. Then each occurrence of a morph in the corpus is replaced with its code. Finally, the total cost *C* of the model (codebook) and the data (encoded corpus) is summed. In the one word "corpus" on the left, this sum *C* is shown next to the word. (Note: the lowest "score" here demonstrates typical segmentation errors in unsupervised learning of morphology.) The random splits that result in the lowest total cost is considered the correct morphological model. The cost of the encoded corpus is calculated by summing the negative log likelihood of each morph's maximum likelihood $p(m_i)$ (i.e. the count of a morph's m_i occurrences divided by the total number of morphs in the corpus). The codebook cost is calculated by multiplying the number of the bits *k* needed to code a character by the character length of each morph $l(m_j)$.

Leading MDL-based models are Linguistica (**?**; **?**) and the Morfessor family of algorithms (**?**; **?**). Linguistica is a foundational work in ULM that attempts to discover "signatures", shown in Figure 3.2, which are sets of affixes somewhat akin to inflectional paradigms. Even though it is unsupervised, Linguistica needs to set an affix length limit and this requires at least enough knowledge of the language's morphology to make a hypothesis. The latest version, Linguistica 5 (**?**), attempts to make ULM more accessible by including a graphical user interface and an open-source, modular Python library. The library is meant to extend the model beyond morphology, taking syntactic context into account (**?**). Morfessor (**?**; **?**; **?**) attempts to identify the most likely "morphs" (candidate morpheme strings) and the most likely morpheme boundaries. Like many unsupervised models, it builds a kind of a morph lexicon and this reduces their cross-linguistic

adaptability. The Morfessor model is built on two parameters that require some hypothesis about the language's morphology: "(i) our prior belief of the most common morph length, and (ii) our prior belief of the proportion of morph types that occur only once in the corpus" (**?**, page 281). Compared to Linguistica which is less eager to split words, earlier versions of Morfessor used Recursive MDL which tended make too many segments, but later versions of Morfessor stand somewhere in between early Morfessor and Linguistica in terms of segmentation accuracy (**?**).

MH

Why?

	Signature	Stem count
1	NULL/s	2327
2	's/NULL	813
3	NULL/ly	587
4	NULL/d/s	346
5	NULL/d	314
6	ed/ing	197
7	'/NULL	190
8	's/NULL/s	181
9	d/s	175
10	ies/y	173
11	NULL/ed/ing/s	151
12	NULL/ed	134

NULL/ed/ing/s (number of stems: 151)

abound	administer	affirm	aff
appeal	arrest	assault	att
awaken	award	beckon	be
bloom	bolt	broaden	bu
claw	click	climb	clu
coil	compound	concern	cor
confront	contact	contrast	cra
crown	decay	deck	dia
display	drill	drown	du
eschew	escort	exceed	ex
extend	filter	flounder	fro
haunt	hoot	hover	ho

Figure 8: Screenshot of Linguistica 5. The column on the left displays hypothesized "affixes" that were found in complementary distribution on "stems", one group of which is displayed on the right.

ParaMor is another unsupervised algorithm (**?**; **?**; **?**; **?**). Like Linguistica, it exploits patterns in the data similar to inflectional paradigms, but unlike Linguistica, it allows multiple segmentations per word. Paramor treats paradigm induction and segmentation separately but it does not address morphophonology as well as Linguistica. It requires more data but not as much memory. **?**, page 49) claim that Paramor's unsupervised induction of morphology could facilitate quick development of morphological learners for LRL, but they refrain from explaining how the system, which requires a significant amount of data, could do this.

Most ULM approaches are biased towards a certain affixing pattern or morphological type. Linguistica and ParaMor are tuned to suffixing morphology. Morfessor is better at prefixing+suffixing languages than the other two, but Linguistica does better with suffixing-only languages. All three are most suitable to agglutinative languages but also assume a small number of morphemes per

word, often only one prefix/suffix. This does not bode well for morphologically complex languages.

In theory, unsupervised models are the most exciting and attractive for LDD because they do not require costly data annotation. In fact, an early vision for ULM was to support language technology for minority language communities. This idealistic aim has not materialized because successful ULM demands "a sufficiently large set of unannotated words in electronic form" (**?**, page 92). Languages that lack a basic morphological analysis generally lack the amount of raw data that accurate ULM requires. "Sufficient" data for unsupervised learning is in the order of a hundreds of thousands or a million words (**?**). A small corpus is defined as 1,000-100,000 words (**?**). Since LRL includes well-documented languages, sometimes obtaining sufficient data may be simply a matter of digitizing existing literature, but for most, a limited written history means that texts must be first recorded orally and then transcribed before this a small corpus is usable. Although language documentation aims to generate a great deal of minimally annotated data (**?**; **?**), LDD projects rarely produce even a 100,000 words and the transcribed portion tends to be much smaller. Despite this, the claim is still being made that ULM methods, "provide an inexpensive means of acquiring a type of morphological analysis for low resource languages" (**?**, page 92). Another drawback is that unsupervised methods are less accurate than other machine learning approaches. **?**) were able to overcome this drawback, at least for some Indo-European languages by using unsupervised word embeddings (word meanings represented in vector space). However, even if word embedding increase accuracy, it is an open question whether they address the large data requirements since successful word embeddings are also built with large amounts of texts.

?) raise other problematic issues. For example, unsupervised approaches have to start with some assumptions about a language's morphology. Should we start with random parameters or should we use knowledge about the target language? The latter is the most efficient, but using prior linguistic knowledge, as Goldsmith, Lee, and Xanthos and **?**) insinuate, means the approach is not truly unsupervised. It also makes ULM even less attractive for under-described languages. With such languages, we can presumably hypothesize from related or geographically adjacent languages, but

this opens another question that is not clearly addressed in the literature: how well do current ULM methods transfer across languages, and can we even measure this reliably? This cannot be addressed until ULM models have been tested against a wider range of languages. Currently, English is the *de facto* language in ULM (**?**). The tension between expanding computational models to a wider range of languages and the prohibitive cost of data to test and evaluate the models will continue to influence the development of computational linguistics unless better data production methods are devised.

Although unsupervised machine learning is most suitable to resource-rich languages, ULM has been applied in low resource contexts. **?**) incorporated unsupervised morphological segmentation as a first step towards semi-automated IGT production (interlinearization) in Uspanteko [usp], a Mayan language of Guatemala with some 4,000 speakers (**?**). Words were assumed to be morphologically related if they were orthographically similar. Affix candidates were generated by assuming stems are longer than affixes and then filtering for statistically significant co-occurrences. **?**) also applied ULM to Uspanteko with a method that assumed suffixation only and, most notably, incorporated awareness of document boundaries when generating and clustering candidate morphemes. This technique assumes spelling is likely to be consistent within a document but vary across documents. Considering that minority languages may not have a standardized orthography and transcribers may have little formal education in the language, this is a simple yet very practical twist on the "one sense per discourse" effect used in computational semantics which observes that if a polysemous word such as "saw" appears more than once in a document all occurrences are highly likely to express the same sense (**?**). Document boundary awareness assumes one inflectional paradigm per stem per document; that is, a morpheme that may belong to two lexical categories is highly likely to appear as only one within a single document. This technique reduces noise and improves results over Morfessor and Linguistica (for English). Remarkably, the performance degrades when the corpus size is increased, perhaps because of increased noise from spurious candidate morphs. **?**) applied ULM to a corpus of Kilivila [kij], an Austronesian language of Papua New Guinea with 20,000 speakers (**?**). Inspired by **?**) and **?**), they identified morpholog-

ically related words by orthographic similarity and a context co-occurence vector. They claim that the method prefers languages with infixes but it is unclear how they determined this with only one language.

3.3 Supervised Machine Learning

Supervised learning of morphology means that models are trained on gold standard annotated data rather than unannotated raw texts. It requires significantly less data than unsupervised learning. However, the data must be annotated. Annotated data makes supervised machine learning qualitatively different than unsupervised learning. Unsupervised learning clusters underlying patterns or features which may facilitate human analysis and description of the data. Supervised learning is trained on data that has already been analyzed and labeled. It learns the labeling and generalizes it, predicting, of with high accuracy, what labels unseen instances should have.

The best performing non-neural supervised model for sequence prediction tasks such as morphological analysis is conditional random fields (CRF) (**?**; **?**; **?**). A CRF is a sequence classifier that considers the whole sequence when making a prediction for each symbol in the sequence. In morphology, CRFs can work as boundary detection (segmentation) and labeler. According to **?**), discriminative learning methods such as CRF are better than generative models because they optimize the accuracy of segmentation boundaries and generalize better to unseen forms, assuming they have sufficient training data. A discriminative model's strength lies in its ability to define features beyond the previous label and allow arbitrary weights. CRFs apply logistic regression which allows it to make generalized predictions from arbitrary and possibly dependent features (**?**).

A CRF can look at a number of features as it makes a prediction. A feature function input for a (linear-chain) CRF in morphological labeling might be 1) a whole word segmented into morphemes, 2) the position of the current morpheme in the word, and 2) the label of the previous morpheme. Each feature is weighted during training and, with these weights, labeling predictions are scored over the whole sequence, then transformed into a probability. For example, in a segmentation task where the previous word is "am/is/are/was/were", if the target word ends in "ing" and a weighted feature is the previous word the CRF might give a high weight to "-ing" as a separate

morpheme. In labeling, the CRF might decide a word-final "s" marks a plural noun and not the 3rd person singular simple present tense by highly weighting the POS tag of the stem. However, the quality of the results relies heavily on the choice of features. This could be a drawback for under-described languages, because if little linguistic description is available then how does one know which linguistic features are optimal? Fortunately, CRFs have been shown to perform reasonably well using primarily language-independent features (i.e. surrounding substrings) (**?**; **?**), including a model trained on approximately 3,000 words of Lezgi [lez], an agglutinative Nakh-Daghestanian language with about 600,000 speakers (**?**). However, it is not clear if the performances were more dependent on language-specific or task-specific features.

Supervised (and semi-supervised, see section 3.5) learning requires less data and almost always achieves better results than unsupervised learning (**?**) and, therefore, seems more promising for LDD. However, it may be necessary to augment textual data with other descriptive resources such as grammars and dictionaries since the size of annotated LDD corpora are still "inadequate for supervised learning" (**?**, page 18). This is not an impractical expectation for LDD since descriptive linguists traditionally result in the Boasian triad, described in Figure .

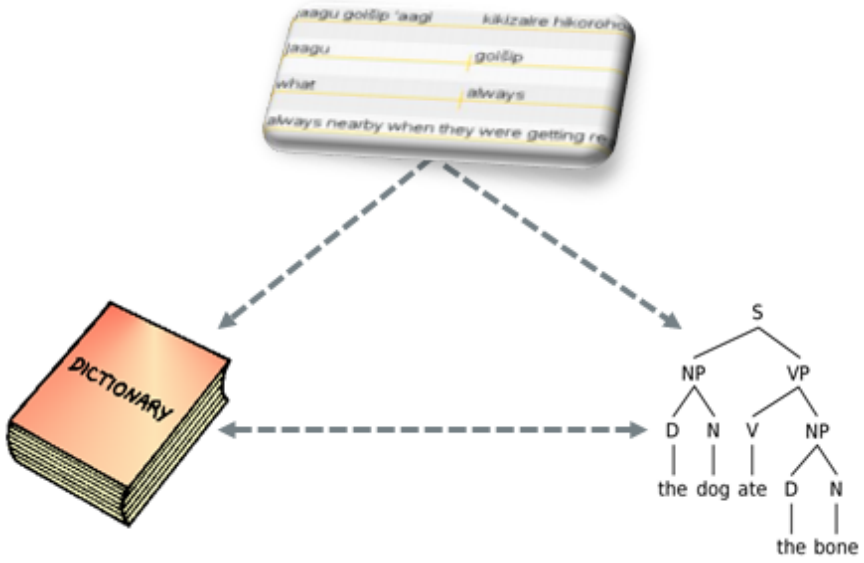

Figure 9: The Boasian Triad. Traditionally, linguists working in new languages concentrate on producing a corpus of interlinearized texts, a grammar, and a dictionary. The interlinearization is a pre-step and feeds directly into the other two steps.

3.4 Supervised Neural Networks or Deep Learning

Neural networks (NN), or deep learning, is a family of machine learning techniques that builds layers of perceptrons that can learn complex patterns. Vector representations of the data are received by an input layer, fed into and transform in "hidden" layers then arrive at the output, or activation, layer. Each unit in each layer is connected to each unit in the adjacent layers. The connections are represented by learnable weights; the higher the weight the more influence a unit has on the result. Since deep learning is supervised[6] the weights are adjusted via stochastic gradient descent with backpropagation using "feedback" from annotated data. Many flavors of deep learning exist and different models work best for different tasks. Recurrent neural networks (RNN) (**?**) with long short-term memory gates (LSTM) (**?**) are currently considered state-of-the-art for most sequence-based NLP tasks. RNNs may have one layer and recurrently add its output to next input and feed that into that layer. Many may have multiple layers and the recurrence occurs only for each time step in the network. This works as a feedback loop. RNNs can input and output sequences of values. A probabilistic prediction of the next item in a sequence is conditioned on the entire history of transformations and previous predictions. Unfortunately, the typical deep learning problem of vanishing gradients, or decay of information through time, is much worse for RNN. That is why the LSTM memory gate is so important. The LSTM decides how much of the previous output should feed into the next recurrence and how much should be "forgotten". This makes training RNNs much faster because "forgotten" information does not needs to be calculated during backpropagation.

MH

It's more relevant that the gradients now propagate much more efficiently over the (long) sequence of time-steps.

In morphology, deep learning methods can be used to classify, generate sequences, or to performs sequence-to-sequence mapping. Classifiers classify data points into categories or classes. A data point might be a morpheme and the classes morphosyntactic tags. The hidden layers feed into a final logistic function layer (i.e. softmax) that outputs a prediction of each possible class as a probability between 0 and 1. Morphological generation produces fully inflected forms from morphosyntactic or contextual information (**?**) or completes morphological paradigms (**?**).

[6] Deep learning morphological segmentation has been performed on unsupervised texts with some success (**?**)

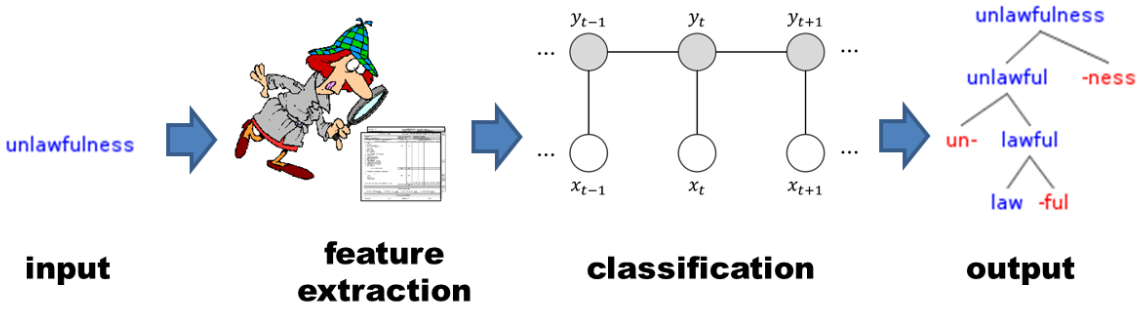

Figure 10: Classical machine learning. In feature-based models a human expert must analyze and define optimal features from the data. A CRF usually uses gradient descent to assign weights to those features during training. A discriminative model such as Conditional Random Fields, can assign arbitrary weights.

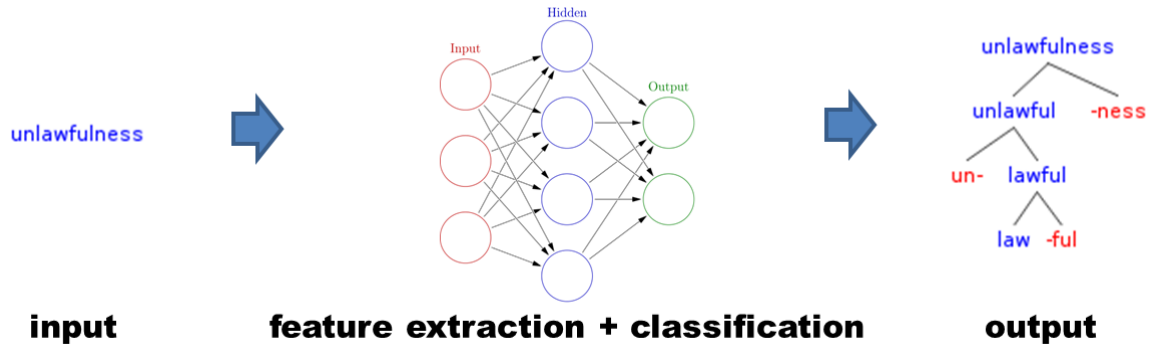

Figure 11: In deep learning, hidden layers create intermediate representation of the data. These essentially serve the function of feature engineering in non-neural machine learning.

Currently, the strongest generator models use an encoder-decoder, also known as sequence-to-sequence (seq2seq), architecture of a RNN (**?**; **?**). Encoder-decoders map one sequence to another of a different length by encoding an input sequence of symbols (e.g. the letters of a word) and decoding it as another sequence of symbols (e.g. morphosyntactic tags). Encoder-decoders are also state-of-the-art for morphological tagging (**?**), morphological segmentation (including low resource settings) (**?**), and morphological paradigm completion (**?**).

Until recently, neural networks were considered to have the same drawback as unsupervised learning: large amounts of training data for good results (**?**). While it is true, for example, that top performing neural networks generate inflected forms with less than 60% average accuracy when

given only 100 training examples, results can increase significantly (over 85%) when trained on 1,000 and 10,000 examples (up to 96%) (**?**; **?**; **?**). Improved results cannot always be accomplished by the tuning tricks common with normal data sizes. For example, adding hidden layers can actually reduce accuracy with smaller amounts of data (**?**). Low data settings require unique techniques that include adjusting the model, training an intermediate step, and artificially augmenting the data. As an example of adjusting the model, **?**) found that using a gated recurrent unit (GRU) to control access to previous states in the RNN, rather than the NLP state-of-the-art LSTM, gave better results with up to 1,000 training examples. Successful intermediate steps in morphology include training the model to make edits to a string instead of decoding a new sequence, so, for example, the model learns an INSERT operation will generate "runs" from "run" (these models use some sort of alignment to reduce the number of edit operations between related forms) (**?**; **?**). Other successful intermediate steps include the whole sentence where an inflected form is or should used or train a model to first produce abstract underlying forms (e.g. "impossible" $\rightarrow$ "in-possible" $\rightarrow$ "NEG-possible") the final inflected or parsed form (**?**; **?**). In recent CoNLL-SIGMORPHON submissions, participants found that these, and other new techniques considerably outperformed the state-of-the-art encoder-decoder baseline (**?**). These successes suggests that developing unique techniques for morphological analysis of low resource languages is a promising area for exploration. A few of these techniques are discussed further in section 4.3.

3.5 Semi-Supervised Machine Learning

Unsupervised learning may require less costly annotated data and supervised training may beat unsupervised training in accuracy but combining the two may be the best of both worlds. Semi-supervised, or minimally supervised, learning combines annotated data with a larger set of unannotated data when available annotated data is not adequate to effectively train a supervised model (**?**; **?**). Like unsupervised learning, semi-supervised learning exploits the underlying structure of raw texts, but with the goal of improving the machine's predictions, rather than discovering new ones (**?**).

Semi-supervised learning has been widely researched for computational morphology and a wide

variety of techniques have been employed (**?**). Some ULM algorithms, such as Morfessor Baseline, have proven adaptable to semi-supervised learning (**?**). **?**) used a "mostly unsupervised" method that employs a generative probabilistic model and 10 million unannotated words. It does POS-tagging, segments and tags morphemes, and identifies candidate inflectional paradigms. **?**) predicted general paradigmatic patterns using longest common substrings and some unannotated data (LCS) (see section 2.2 for more details). The latter two models presuppose a few completed inflectional paradigms which are provided by native speakers either directly, or as computational linguists tend to prefer, through the indirect means of an expert who pre-processes the data or through crowdsourced resources such as Wiktionary.

While semi-supervised learning was early praised for its greater effectiveness and practicability, real applications were rare well into the late 2000's (**?**). One reason may be that, like unsupervised learning which requires some initial hypothesis, many semi-supervised models make strong assumptions about the data. These assumptions may hold true in pre-processed datasets but "tend to be violated in real-world data" (**?**, page 1). In addition, unannotated and annotated data cannot be simply combined together. Annotated data has to be weighted so that the larger, unannotated part does not overwhelm it. Overall, results in semi-supervised learning "strongly suggest that it is crucial to use the few available annotated training instances as efficiently as possible before…incorporating large amounts of unannotated data" (**?**, page 35).

Semi-supervised learning is promising for LDD because even though small amounts of annotated data are "easy to get by manual annotation" (**?**, page 49), typical documentation corpora are only partially annotated. However, the results suggest that all available descriptive data for LRL should be exploited as much as possible before relying on unsupervised data. This heavy dependence on annotated data still poses a challenge (**?**).

4 Exploiting Resources

This section explores ways to improve results for some of the approaches discussed in section 2 when applied in low resource contexts. The high cost of annotating texts or manually constructing

FSTs may be prohibitive, and even though supervised and semi-supervised methods requires less data than unsupervised or neural network models, they still need significantly more data than is typically available for low resource languages. "Understanding the available resources is the first step in building a natural language processing framework for a target low resource language" (**?**, page 13). The spotty nature of LDD data necessitates using whatever resources are available (**?**). This section present some techniques to exploit available resources other than textual data, such as artificially generated "textual" data, descriptive linguistic knowledge, and resources in other languages.

4.1 Non-Textual Resources

Even though annotated textual data is scarce for most languages, many have been described to some extent and some linguistic resources exist for them. In practice, language documentation projects are rarely undertaken (and more rarely funded) except to support descriptive analysis and publication (**?**; **?**; **?**; **?**). Describing a language's morphological structure is relatively easy (**?**) and linguists tend to tackle it in the first stages. Minimally, a few inflectional paradigms can be easily elicited in a language documentation project. The bulk of a language's morphology is described in structured data. Structured data might be any organized or pre-processed data that is not a naturally occurring text, such as inflectional tables or word lists.

Crowdsourced websites such as Wiktionary sometimes have inflectional tables for LRL. **?**) successfully exploit spell checkers and Wiktionary. **?**) use publically available inflection tables when combining semi-supervised learning with a feature-based Support Vector Machine (SVM). **?**) also exploit Wiktionary for inflectional tables as the supervised part of a semi-supervised model. They claim that this allows their model to "extend to other languages [with inflectional tables in Wiktionary] without change" but it is to be expected some of the 150 or so languages on Wiktionary have very few tables.

If data is thoroughly annotated, computational morphology can improve results from labeling that are not strictly morphological in nature. For example, **?**) reduced CRF training time by performing a coarse POS-tagging then fine-grained morpheme tagging. The POS-tagging leaves

MH

The semi-supervised part was in the 2014 paper (no SVM), the SVM was in the 2015 paper for classifying unseen lexemes into inflectional classes.

the algorithm with fewer possible tags to process for each instance.

Exploiting non-textual resources does not always improve results. Semi-supervised learning with a discriminative sequence learner like a CRF performs worse when non-textual resources are integrated because the external resources receive undue weight. **?**) claim that models, such as a Maximum Entropy Hidden Markov Model, that generate a lexicon are less likely to degrade in performance because the lexicon essentially acts as an external resource and does not compete with the annotated corpus.

4.2 Transfer and Joint Learning

Transfer learning is a common technique for low resource languages that can be applied in semi-supervised and unsupervised learning (**?**). Transfer learning assumes that what succeeds on one language will work well on another and attempts to transfer that success. It may apply one model to another language or attempt to transfer annotated tags by aligning words across languages (**?**). Transfer learning performs better when trained on more than one source language. **?**) found that single segmentation model trained simultaneously on four related polysynthetic languages performed as well as and sometimes better than single language models.

?) applied transfer learning principles by using an English word list to identify affixes in unlabeled data of Tagalog [tgl] and Zulu [zul], low resource languages (but clearly not endangered – both have over 20 million speakers) spoken in the Philippines and South Africa respectively. The strategy was motivated by the realization that many low resource languages are spoken in multilingual contexts and have borrowed vocabulary from more economically powerful languages. A word list in that language identifies some root morphemes. Splitting substrings from these roots provides a list of prefixes and suffixes. With some knowledge of the language's morphological patterns, it was even possible to identify infixes in Tagalog. However, the strategy does not do well when a word has multiple affixes.

Transfer learning is promising for low resource languages but it has its limitations. When RNNs performed cross-lingual morphological tagging on 18 languages from four language families, the conclusion was informative but perhaps unsurprising to those familiar with linguistic typology—

MH

It might be good to know that the vocabulary is evolving. Transfer learning in the last couple of years has come to mean learning from one task to another, while I've seen the term "cross-lingual learning" more often for generalizing across languages.

the closer the languages' relationship, the more accurate were the results (**?**). The precise correlation between linguistic genetics and results in transfer learning is largely unexplored (**?**; **?**), although interest is growing and new experiments are forthcoming (**?**). It is still not clear, for example, whether morphological structure or phonological (orthographic) similarity is a stronger factor. It *is* clear that the amount of morphologically marked information common across languages influences results, so if a word "in the source language does not overtly mark a grammatical category [that is marked] in the target language, it is nigh impossible to expect a successful transfer" (**?**, page749). Transfer learning shows sharply the difficulties in paradigm induction. Paradigms do not generalize across languages. Each language has its own unique "layout" for classes within each lexical category and these can differ considerably even in closely-related languages.

Since transfer learning can potentially overcome deep learning's data hunger, discovering how to improve deep learning results in when transferring from high resource to low resource settings would be "a boon for low resource computational linguistics" (**?**, page 752). **?**) points out that automatic projection of annotation (specifically, POS-tagging, noun phrase chunking, and dependency parsing) between high and low resource languages admits error at multiple points. The results have to be corrected or adjusted. Any projection-based method requires some seed data to perform well (**?**), so one question ripe for exploration is how much data does the low resource target language need to balance a high resource source language's greater data? **?**) claim only a "small amount of annotation" is required; but they also claim the necessary annotation should not take too long. Conspicuously, they refrain from defining "small amount" and "not too long".

Joint learning trains simultaneously on different types of linguistic information. For example, since segmentation and tagging are generally separate task in computational morphology, training a semi-CRF to do both tasks would be considered joint learning (**?**). It is based on the intuition that one type of linguistic knowledge (e.g. syntax) can improve results in another domain (e.g. morphology), and vice versa (**?**). One of the earliest joint learning attempts was **?**)'s incorporation of semantic, orthographic, and syntactic information into unsupervised learning of morphology. The semantic information came from Latent Semantic Analysis which represents meanings as patterns

of words that appear together in text, for example “morphology”, “morpheme”, “inflection”’. Semantic information helped avoid the typical unsupervised segmentation error (e.g *all-y* instead of *ally*). It was supplemented by the probability of two affixes alternating on one “stem”, which was calculated using orthographic information. Syntactic information was derived from the frequency of words that occur around pairs of possibly related inflected forms. The output was “conflation sets” which are paradigm-like groups (e.g. “abuse”, “abused”, “abuses”, “abusing”).

Since morphology carries a great deal, and in some languages most, syntactic information, syntax is important information for morphological analysis. Joint learning of syntax and morphology can be as simple as incorporating POS tags as features or extending POS tagging methods to morphological tagging (**?**; **?**). Syntax can also be incorporated into supervised learning by clustering words with similar final substrings which serve as a proxy for grammatical agreement and are assumed to indicate lexical categories (**?**). Successful joint learning of semantics and morphology has used the semantic information in word embeddings (**?**).

Joint learning in low resource settings has been successfully applied using information commonly produced by LDD projects (**?**; **?**), particularly POS tags.

4.3 Data Augmentation

Another “resource” to exploit is artificially generated data to augment naturally occurring textual data. **?**, page 38) found that the “main benefit of the various data augmentation methods is providing a strong bias towards … regularizing the model, with a slight additional benefit obtained by learning the typical character sequences in the language." **?**) found that with data augmentation a neural model can outperform Morfessor and match CRF accuracy. The success of these methods suggests that data augmentation may be a fruitful area of experimentation for applying neural models in low resource settings.

The most successful data augmentation strategies in low resource settings bias an encoder-decoder model to copy strings (**?**; **?**; **?**; **?**). For example, **?**) and **?**) auto-encode (train a model to output a string identical to the input) words from the training corpus. Their performance varied between languages, perhaps due to how well a language’s morphological features were represented

in the small training datasets, but **?**) found this method performed better than transfer learning and **?**), who refer to the approach as "multi-task learning", found it superior to **?**)'s strategy of replacing common substrings in the test data with random strings from the training data. Simplistically, we might assume that copying "base forms" (e.g. "run") would work well for morphologically simple languages that add few affixes to the base form, but the foundation of morphological analysis is the systematic patterns of inflection which means that all languages tend to repeat identical or very similar substrings across morphologically related forms.

5 Computational Morphology for Language Documentation and Description

As we have seen, computational linguistics has a solid history in morphological analysis and paradigm learning and its success in low resource settings is expanding. Unfortunately, general linguistics has witnessed very little evidence of this success. Basic morphological analysis and annotation stands as a bottleneck in production of new and accessible language data (**?**). This is doubly unfortunate because limited data hinders both general linguistics and computational linguistics. If computational morphology were incorporated into the documentation and early description of endangered languages, new data could be annotated more quickly. This section explains the bottleneck and suggests how the models and technique described in this paper might overcome it.

Once new language data is recorded and transcribed,[7] the next step is an annotation process called interlinearization. Narrowly defined, interlinearization marks boundaries between morphemes (segmentation) with hyphens, labels morphemes with morphosyntactic or lexical meaning (glossing), and gives a rough equivalent of each sentence in a language of wider communication (free translation). Interlinearization stands on the fuzzy line between documentary linguistics and descriptive linguistics. It goes beyond mere documentation of a language but it is still a "preprocessing step" that lays the foundation for deeper analysis and fuller descriptions in dictionaries and published grammars (**?**).

Linguistic annotation is performed primarily by hand. While manual work may provide reliable analysis—a human is much better than a computer at interpreting contextual cues—once the

[7] Transcription is the first bottleneck in data production. Fortunately, machine learning is already being applied to this step.

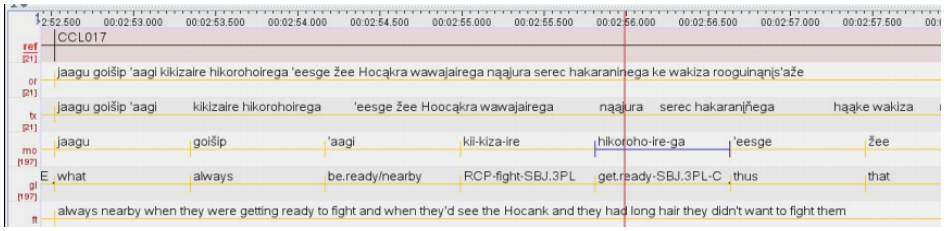

Figure 12: Interlinearization in ELAN.

initial analysis is complete, manual annotation is extremely inefficient (**?**; **?**; **?**). It is repetitive, monotonous, and, therefore, prone to typos and inconsistencies. It is costly, requiring money to hire and train annotators until the last word has been processed. And it is time-consuming. This is true even in computational linguistics, where it took three years to annotate just one layer of the Penn Treebank (**?**) (**?**; **?**). Whereas a documentary field project may record several hours of valuable and endangered data, annotation is rarely completed within funding and time limits.

Popular software for linguistic annotation and analysis do not incorporate computational morphological models, so unfortunately, effective machine learning is not available to linguists who do not have computer programming skills. Two popular software for linguistic analysis and annotation ELAN, pictured in Figure 5[8], and FLEx, pictured in Figure 5[9], do suggest morpheme boundaries and glosses (tags), but only when the word form is identical to one that was previously annotated manually or fits hand-constructed rules.

Incorporating machine learning into the morphological analysis and annotation has been demonstrated to be effective (**?**; **?**; **?**). With some manual pre-annotation, supervised learning can speed morpheme segmentation and glossing. The results could serve as a springboard for machine learning applied to related tasks in the LDD workflow. Morphologically annotated data could serve as input to machine translation models that would complete the third line in interlinear glossed texts. It could also provide data for models that support further description of a language's structure and dictionary development. Paradigm tables could be induced and root morphemes could be clustered into inflectional classes. Paradigm tables are important features of published grammar descriptions

[8]Source: **?**)
[9]Source: http://software.sil.org/fieldworks/resources/tutorial/interlinearize-texts/

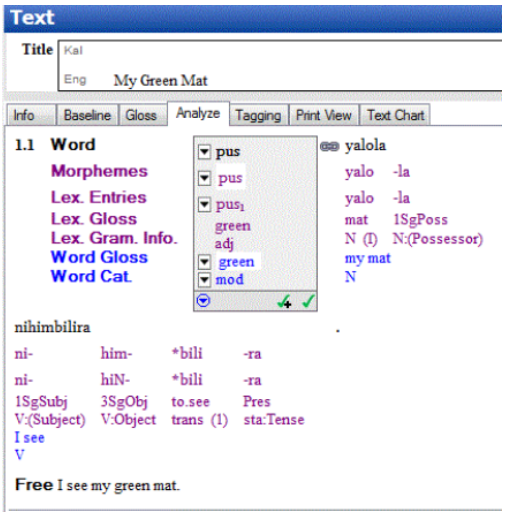

Figure 13: Interlinearization in Fieldworks Language Explorer (FLEx).

and inflectional class identification is useful information in a dictionary entry. The candidate tables and clusters could serve the linguist as working hypotheses.

As this paper has described, several computational models and techniques achieve good accuracy in low resource settings. Many can be expanded and improved and new ones will be discovered, but there is a practical problem in the LDD workflow that can be addressed now by applying a successful model to interlinearization. For example, one might begin with the CRF's success on one agglutinative language using only 3,000 training words and simple, cross-linguistic features (**?**). Replicating the experiment on documentary data in other languges using identical, or similar features, would test whether some morphological types need more supervised input or whether specific features should added or customized for different language families. If the success on other languages is not satisfactory other techniques could be explored. For example, augmenting data with other resources such as including surrounding words in a texts as input data in a RNN morphological model or perhaps just the POS tags of surrounding words, if available.

By speeding and improving the production of segmented and glossed data, the third line of interlinearized texts could be finished with machine learning. For example, an encoder-decoder could learn to translate the glossed lines into "bad English" and from that into "good" English[10].

[10]Or Spanish or Russian or any major language used as the language of analysis.

This progression might look like example 3 which begins with glosses from a Russian sentence.

(3) evening-INST 1SG run-PAST.SG.FEM in store-ACC ⟶ 'Evening with I ran in store to' ⟶ 'In the evening I ran to the store.'

Bringing computational morphology into the LDD workflow carries implications beyond the production of interlinearized texts. The models can assist more advanced analysis. The logical next step is to provide automated aid in the discovery of inflectional classes and their paradigm tables. For example, the approach used by **?**) and illustrated in Figure 2.2 already finds candidate inflectional patterns. It could be adjusted to also output the words that were used to identify each pattern. Software programs for linguistic analysis already use interlinearized texts to automatically build lexicons so it would be natural to add inflectional class identification.

Although computational morphology reaches reasonably high accuracy in low resource settings, the simple fact is that more data yields better results. It enough to introduce one or two computational models into LDD workflow is not enough to address the data production bottleneck. **?**) propose two stratagems to improve and speed the annotation: reduce the cost of annotating new data or get more miles out of previously annotated data. Crowdsourcing is an example of the first strategem, but it has proved less effective in low resource contexts (**?**; **?**). For the second strategem, Baldridge et al. recommend active learning. Active learning is a machine learning technique that directs human annotators towards the most informative unannotated instances (i.e. instances most dissimilar to what the machine has been trained on). Annotators vet and correct the machine predictions for those instances. The ongoing input is used to iteratively retrain the model and improve accuracy with less manual effort. Experiments in computer-aided machine translation (**?**) and interlinearizing LDD data (**?**; **?**; **?**) indicate that active learning is a promising route for automated assistance in language documentation and description. After training a machine learning model on available Uspanteko data (**?**) the responses of annotators during active learning gave the machine learning model's accuracy a significant boost. **?**) even found that if the model's patterns are identified, the vetting and correction process can be automated as well. Ideally, interlinearization and other linguistic annotation will someday assume minimal manual annotation followed by a

MH

???

few cycles of training a supervised machine learning model, vetting and correcting its results, and retraining the model until it achieves near perfect accuracy.

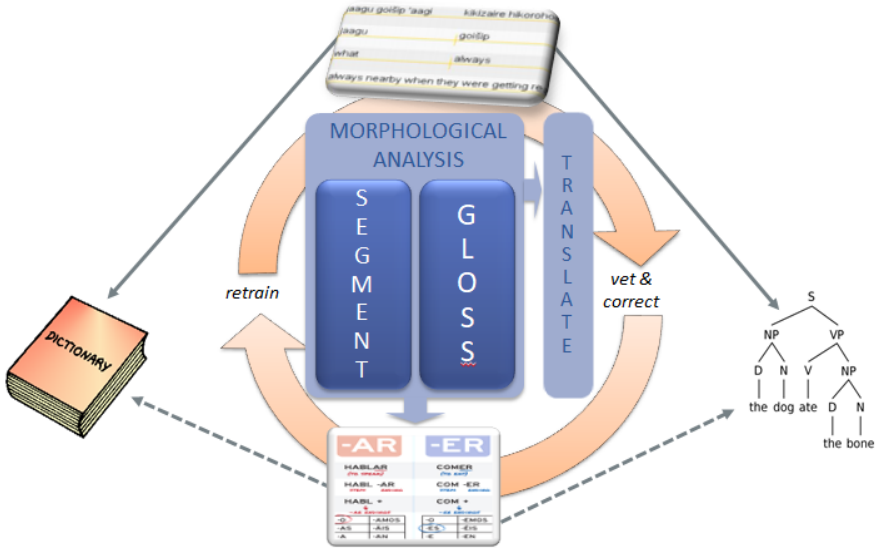

Figure 14: A practical application of computational morphology to language documentation and description would focus on addressing the time costs of interlinearizing transcribed texts and provide working hypotheses of inflectional patterns and classes. This would support the development of dictionary and full grammatical descriptions. Computational morphological models will continually improve if a cycle of active learning is incorporated into the workflow, where language experts vet and correct machine predictions and use that information to retrain the machine learning models.

Although this paper does not have space to consider the issue properly, it must be remembered that the ideas suggested here to bridge the technological gap for for ordinary working linguists and the average native speaker are only practical if they are accompanied by graphical user interfaces. A well-designed user interface is particularly important to incorporate the iterative aspects of active learning. It might be new features or an add-on in popular software programs such as ELAN (**?**) and FLEx (**?**). Another option is a new program that is compatible with popular tools, as, for example, the web-based ELPIS (**?**) transcription app, which intakes and outputs data in ELAN-compatible files, or similar to the interface **?**) developed, where users saw the most informative tag predictions and a ranked list of other tags to choose from. Without requiring a class in machine learning, a good interface would walk users through the process of training and re-training the model and draw their attention to the machine predictions that most need to be vetted or corrected

Bridging the gap between computational morphology and language documentation and description will bring to light new questions for future research. Experimenting with real world data from actual LDD projects may guide NLP to unforeseen problems in low resource settings and to new solutions. For example, how accurate does a model need to be to be practical? It would be wonderful to begin machine-aided morphological analysis as soon as transcribed data becomes available, but it turns out that automated support is more time-consuming than manual annotation if performs poorly (**?**; **?**). The optimal threshold of accuracy will likely vary among language families or morphological types. Another question is, how will machine learning affect linguistic field methods? Will the results all be positive? Hopefully, exhibiting the practical benefits will motivate field linguists to spend their limited time and money on producing data that fuels the most efficient computational models. However, solutions from the computational side must weigh against other concerns. For example, a joint learning model that requires syntactic parsing might prove highly efficient, but training someone to perform syntactic annotation might cost more than another, less optimal task. The favored computational models, recommended best practices, and priority outputs should be a combination that optimizes benefits for the fields of computational linguistics, general linguistics, and for the communities who speak low resource and often endangered languages.

6 Conclusion

Until very recently, success in computational morphology depended on high resource contexts, which was only a trivial percentage of the world's approximately 7,000 languages could provide. This retards the development of computational linguistics and its potential contribution to broader linguistic science. Recently, computational approaches are being increasingly applied with reasonable success to low resource languages. Promising machine learning techniques for limited data settings are being discovered.

Our understanding of morphological structure in the world's languages rests on empirical studies of language data. Computational morphology, in particular, will not achieve helpful results without sufficient published data. The most efficient computational models are served best by

publically available morphologically annotated corpora, because, although unsupervised models achieve some success with unannotated data, accuracy is much higher in supervised and semi-supervised machine learning. Fortunately, morphological annotation (interlinearization) of texts serves as a foundation to traditional linguistic analysis. Documentary and descriptive linguistics also regularly produce other structured data resources such as grammars and dictionaries. Semi-supervised learning, in particular, are well suited to contexts where a combination of textual and non-textual linguistic data exists. Exploiting a variety of resources and a range of machine learning models is a creative way to improve accuracy with limited data. Nevertheless, for all models—finite state transducers, unsupervised, supervised feature-based or neural networks, and semi-supervised—more data means better results.

Despite the importance of annotated data, the computational linguistics literature repeatedly returns to the unfortunate reality that "annotating sufficient data...is expensive" (**?**, page 1954) and retards the development of computational models. To this the literature often adds another perceived inconvenience: that annotation "relies on linguistic expertise" (*ibid*). A language expert—native speaker or linguist—must be involved in recording, analyzing, and annotating data to produce the gold standards annotations necessary for training and evaluating machine learning models or for building finite state machines. Strange as it may seem, it is fortunate these same costs and inconveniences have long stood as a bottleneck to documentary and descriptive linguistics data production (**?**; **?**; **?**). The this same "inconvenient" reliance on people who actually know the language has been an integral part of linguistics for as long as it has existed as a science. Language documentation and descriptive linguistic literature is brimming with proven field methods for making the most of human linguistic expertise. What remains, then, is to reduce the cost of annotation and speed its production. This relies the success of computational morphology in low resource settings.