# Inducing Morphemes Using Light Knowledge

MICHAEL TEPPER and FEI XIA
Department of Linguistics, University of Washington



Allomorphic variation, or form variation among morphs with the same meaning, is a stumbling block to morphological induction (MI). To address this problem, we present a hybrid approach that uses a small amount of linguistic knowledge in the form of orthographic rewrite rules to help refine an existing MI-produced segmentation. Using rules, we derive underlying analyses of morphs—generalized with respect to contextual spelling differences—from an existing surface morph segmentation, and from these we learn a morpheme-level segmentation. To learn morphemes, we have extended the Morfessor segmentation algorithm [Creutz and Lagus 2004; 2005; 2006] by using rules to infer possible underlying analyses from surface segmentations. A segmentation produced by Morfessor Categories-MAP Software v. 0.9.2 is used as input to our procedure and as a baseline that we evaluate against. To suggest analyses for our procedure, a set of language-specific orthographic rules is needed. Our procedure has yielded promising improvements for English and Turkish over the baseline approach when tested on the Morpho Challenge 2005 and 2007 style evaluations. On the Morpho Challenge 2007 test evaluation, we report gains over the current best unsupervised contestant for Turkish, where our technique shows a 2.5% absolute $F$-score improvement.

Categories and Subject Descriptors: I.2.6 [**Learning**]: Language acquisition; I.2.7 [**Natural Language Processing**]: General

General Terms: Algorithms, Languages

Additional Key Words and Phrases: Morphological induction, allomorphy, machine learning, computational linguistics

**ACM Reference Format:**
Tepper, M. and Xi, F. 2010. Inducing morphemes using light knowledge. ACM Trans. Asian Lang. Inform. Process. 9, 1, Article 3 (March 2010), 38 pages. DOI = 10.1145/1731035.1731038. http://doi.acm.org/10.1145/1731035.1731038.

## 1. INTRODUCTION

Morphological analysis is useful for a variety of computational linguistics tasks. Morphological segmentation, a type of shallow analysis that indicates

Authors' address: University of Washington Department of Linguistics, Box 354340, Seattle, WA 98195; email: mtepper@gmail.com.

Table I. Some Examples of Allomorphs in English and Turkish

| Language | Type | Morpheme | Meaning(s) | Allomorphs |
|---|---|---|---|---|
| English | Stem | */grab/* | grab | *grabb-*, *grab* |
| | Suffix | */-s/* | +PL, +3-SG | *-s*, *-es* |
| Turkish | Stem | */kanaD/* | 'wing' | *kanad*, *kanat* |
| | Suffix | */-DA/* | '+LOC | *-da*, *-de*, *-ta*, *-te* |

morphological boundaries, has resulted in improvements in speech recognition error rates [Creutz 2006; Kurimo et al. 2006] as well as improvements in information retrieval [Kurimo et al. 2007] particularly for highly inflected languages such as Finnish and Turkish. Morphological analysis may also benefit in areas where stemming has been successfully employed, such as statistical word alignment [Corston-Oliver and Gamon 2004], because like stemming, morphological analysis can be used to identify stems.

Rule-based approaches to morphological analysis have been in development since the late 1970s, and typically produce stable and accurate analyzers. Of these, the finite state transduction approach has been particularly successful [Karttunen and Beesley 2005], and is quite popular. The main drawback to rule-based approaches is that they rely on a morphological lexicon, a resource that may be prohibitively expensive to acquire, or too difficult or time intensive to produce, in many situations.

Automatic morphological analysis, trained by machine learning, typically requires fewer resources. It can be done supervised, with an annotated corpus of previously moprhologically analyzed words, or unsupervised from a variety of resources. We focus on the class of unsupervised approaches that do not use prior knowledge of the morpheme lexicon, collectively known as Morphological Induction (MI) approaches. These approaches can be speedily adapted to new languages as long as training resources (often just a word list) exist.

One of the drawbacks to induction approaches is that they produce noisier results than rule-based approaches for a number of reasons, including imperfect/incomplete models and noisy training data. Another drawback is that allomorphic variation, or variation among morphs with the same meaning, is often not modeled by machine learning systems. For applications that attempt to retrieve meanings or a map between meanings, like information retrieval and machine translation, there is a benefit to knowing certain morphs are actually representations of the same morpheme. For example, in Table I, the English morpheme */grab/* has two regular variants: one, [*grabb*], occurring in a limited set of contexts such as before vowel-initial affixes (e.g., *grabb*+*ing*), and another, [*grab*], occurring everywhere else. If one were to query 'land *grabb*ing,' presumably one would be interested in a remark such as 'Their goal was to *grab* the land.' Likewise, in Turkish there are two variants of */kanat/* (wing), [*kanat*] and [*kanad*]. If one were to query 'bülbülün *kanad*ı' (the nightingale's wing), presumably one would also be interested in phrases like 'bülbüldeki *kanat*' (the wing on a nightingale), though it has a superficially different morph. Finally, failing to account for allomorphic variation adds noise to the learning process, dividing up the frequency of a morpheme among several

allomorphs, some of which may be frequent, others quite infrequent, creating a scenario in which infrequent allomorphs of frequent morphemes may go unlearned.

To address the problem of allomorphic variation in orthography[1] head on, we present a hybrid approach that uses a small amount of linguistic knowledge in the form of orthographic rewrite rules to refine an existing *morph* segmentation and simultaneously learn a *morpheme* segmentation. In order to do this, we extend the Morfessor algorithm [Creutz and Lagus 2004, 2005] by adding segmentation analyses generated by orthographic rewrite rules along with a statistical framework to predict when analyses should be used as morphemes. This technique has resulted in improvements over the Morfessor baseline system and other state-of-the-art morphological induction systems when tested on the Morpho Challenge 2005 and 2007 data [Kurimo et al. 2006, 2007].

The article is organized as follows: In Section 2, we provide a brief review of previous work on computational morphology including Morfessor *Categories-MAP* – our baseline system. In Section 3, we describe the methodology of the proposed approach and explain how rewrite rules are used in our system. The details of our system are provided in Section 4 and the experimental results are reported in Section 5.

## 2. BACKGROUND AND PREVIOUS WORK

Before providing a brief review of the previous work on computational morphology, we will first summarize the linguistic concepts that are relevant to the discussion in the article.

### 2.1 Linguistic Preliminaries

Morphology is the study of word structure and word formation in natural language. Similar to the way sentences are formed from words, words are formed from more basic units, called morphemes. In this section, we review morphemes, morphotactics (constraints that govern how morphemes form words), allomorphy (morpheme variation), and end with a short discussion of morpheme ambiguity.

2.1.1 Morphemes. The basic units of morphology are morphemes. Trost [2003] defines morphemes as "the smallest unit in language to which a meaning may be assigned or, alternatively, as the minimal unit of grammatical analysis." This definition indicates the dual role of morphemes in language. There are morphemes that have semantic content (traditionally known as content morphemes), such as English *cat*, *boy*, and *walk*. And, there are morphemes that are associated with grammatical functions (known as function morphemes), such as the English plural morpheme *-s* and past-tense morpheme *-ed*.

To introduce a bit more terminology, a word that consists of at least one content morpheme is a base. There are multi-morphemic bases, such as *organ+ist*

[1]In this work we focus on learning allomorphs in orthography only; phonology is not considered.

or *garden+er*, and uni-morphemic bases such as *organ* and *garden*. Monomorphemic bases are called roots. In computational linguistics literature, stem is often used interchangeably with base. Just note that stem has a more specific denotation as the variant to which affixes attach, for example, the *wak-* in *wak+ing* and the *stepp-* in *stepp+ing*.

Function morphemes that attach to stems are called affixes. Depending on what information is carried, the role of an affix can be derivational or inflectional. Derivational affixes modify the part of speech category, and/or the semantic content of the stem they attach to. Inflectional affixes, on the other hand, convey grammatical information such as number and case for nouns, and tense and aspect for verbs, and typically form paradigms, or sets of affixes for a stem class. For example, the English inflectional paradigm {ø, *-ing*, *-ed*, *-s*} attaches to the regular verb stem class, such as *start* and *walk*. Our system, which learns morphology from raw text input with no part of speech markup, currently does not make the distinction between inflectional and derivational affixes.

2.1.2 Morphotactics. Word structure is governed by morphotactics. A key element of morphotactics is morpheme order, which is typically fixed. For example, in English you can have *over+extend+ed*, but not **over+ed+extend*. A related concept is morphological selectivity, or the constraints that govern which morphemes can occur together. Such constraints are tied to morpheme features such as part of speech, phonology, and prosody. For example, prosodic properties allow English comparative suffixes *-er* and *-est* to combine with adjectival/adverbial stems that are either monosyllabic or disyllabic ending in -y (e.g. *green+er*, *pretti+est*), but not with higher degree polysyllabic stems (e.g., **redundant+er*).

2.1.3 Allomorphy. The mapping between morphemes and what actually occurs on the surface, that is, in actual written words, is not one-to-one. Whereas morphemes are abstract linguistic units, typically corresponding to one meaning or grammatical function, morphemes quite typically have several versions on the surface. Surface versions of morphemes are called morphs. The group of morphs that belong to a morpheme are *allo*morphs of that morpheme. The English plural morpheme and a subset of its allomorphs is presented in Figure 1.

Regular allomorphs are conditioned on their surrounding orthographic contexts. For instance, in Figure 1, we can observe that there are some contexts that trigger the variant *-es*, such as occurring immediately following an *s* or *tch*. A complementary set of contexts triggers the variant *-s*. Irregular allomorphs, on the other hand, are lexically conditioned, occurring only with specific lexical items. For example, the irregular plural suffix *-i* will only occur only with a small set of nouns borrowed from Latin, such as *alumn+i*.

2.1.4 Ambiguity. The mapping between morphemes and their semantic content or grammatical function is not necessarily one-to-one either, causing ambiguity. Morpheme ambiguities arise when a morpheme shares the form (i.e., the same regular allomorphs and triggering contexts) with one or more

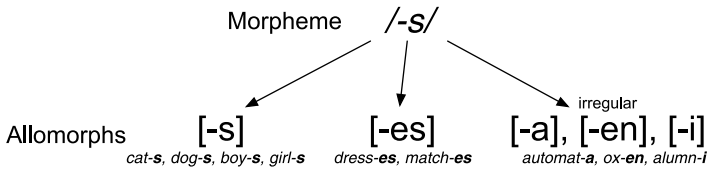



Fig. 1. The English plural morpheme *-s* and its allomorphs.

other morphemes. A prominent example of ambiguity in English is the plural and third-person singular present-tense morphemes, which share the same regular allomorphs, *-s* and *-es*. Global, or word-external, ambiguities arise because English noun and verb stems are often also ambiguous, and when inflected by *-s* the ambiguity remains. The inflected word *boats*, for example, could be a finite verb *boat_V +3sg*, as in *"She boats."* or a plural noun *boat_N +PL*, as in the phrase *"His boats"*.

A model can try to capture the ambiguity mentioned above. In such a model, an ambiguous root or stem would be marked as belonging to multiple morphological categories, e.g. *boat* would become *boat_V* and *boat_N*, and ambiguous inflectional affixes would map to multiple tags, for example *-s* would map to *+3sg* indicating third-person singular present tense, and *+PL* indicating plural. A model can also be shallow and ignore such ambiguity; under a shallow model, *boats* might receive one analysis: *boat+s*. Such a model may postulate morphemes, but refrain from analyzing their semantic and/or functional content. Our system is of the second type, postulating morphemes but not their categories or roles, because the input to our system is only a word list with frequency, and does not include part of speech tag information.

## 2.2 Computational Morphology

Computational morphology is an applied discipline meant to implement a working model of the morphology of a given language. It has been used in NLP applications such as parsing, speech recognition, text-to-speech synthesis, information retrieval, and machine translation. In what follows, we offer a short introduction to computational morphology, focusing on approaches relevant to the current work.

First we will address finite-state morphology (FSM), which involves a hand-coded morphological lexicon and rigorous rules compiled into a finite-state transducer in order to relate surface words to analyses, and vice versa. Then we will move on to morphological induction (MI), which involves automatically learning morphology from limited resources, often just a word list. We will also briefly describe what has been done regarding capturing allomorphy in morphological induction, since that is the goal of the current work.

2.2.1 Finite-State Morphology. Finite-state morphology (FSM) is a rule-based approach which models morphology using finite-state transducers (FSTs). FSTs create a bidirectional map from one set of symbolic sequences (strings in this case) to another. When an FST traverses a valid input string,

it produces a related output string. In FSM, FSTs map between the set of surface (actually occurring) words in a language and their underlying (morphemic) analyses, achieving morphological analysis, and back, achieving morphological generation.

Koskenniemi [2003] states some of the goals of finite-state morphology. In particular, FSM should:

(1) Handle allomorphy.
(2) Identify underlying morphemes.
(3) Capture the morphotactics, that is, the morpheme ordering constraints.

To handle allomorphy, Koskenniemi postulates two-level rules [Koskenniemi 1983], which can be thought of as filters between the upper and lower tapes of an FST. These rules allow single-stage derivation from underlying to surface forms, without the need to independently order rules relative to each other.

Another option for handling allomorphy is cascaded-rewrite rules. Such rules, also known as Sound Pattern of English-style (SPE-style) rules, were developed by Chomsky and Halle [1968] for the purpose of representing phonological patterns, and were formalized for use with finite state technology in Kaplan and Kay [1981; 1994]. They have the general form:

$$\underset{\text{underlying}}{\alpha} \rightarrow \underset{\text{surface}}{\beta} \; / \; \underset{\text{l. context}}{\gamma} \; \_ \; \underset{\text{r. context}}{\delta}$$

Rewrite rules propose a replace operation $\alpha \rightarrow \beta$ at the focus position '_' whenever the left context $\gamma$ and right context $\delta$ are met. Rules may interact—and may also be conditioned on changes made by other rules—so rule-ordering must be explicitly stated.

With either two-level rules or SPE-style rules, generating surface forms from underlying analyses is generally straightforward, while analyzing surface forms frequently leads to more than one possibility. This ambiguity, when spurious, results in the overanalysis problem [Karttunen and Beesley 2001]. However, some surface forms may be genuinely ambiguous, with several possible points of origin. Therefore, morphological analyzers must be able to recognize true ambiguity while discarding spurious analyses.

We show an example of the overanalysis problem in Figure 2. Upon generation, applying rules listed in the figure to the Turkish morpheme sequence *köpek+DAn*, glossed as *dog+FROM* and meaning "from the dog," has one rendering as a morph sequence: *köpek+ten*. However, given the morph sequence *köpek+ten*, there are four possible analyses: *köpek+DAn*, *köpek+tAn*, *köpek+Den*, and *köpek+ten*, where only the first is legitimate. In the FSM approach, this problem is solved by the upper tape of the FST, which spells only valid morphemes (determined by a lexicon) in valid combinations (determined by morphotactic constraints). In our example, the upper tape would lack *-tAn*, *-Den*, and *-ten* as possible suffixes, thus only *köpek+DAn* would be licensed.

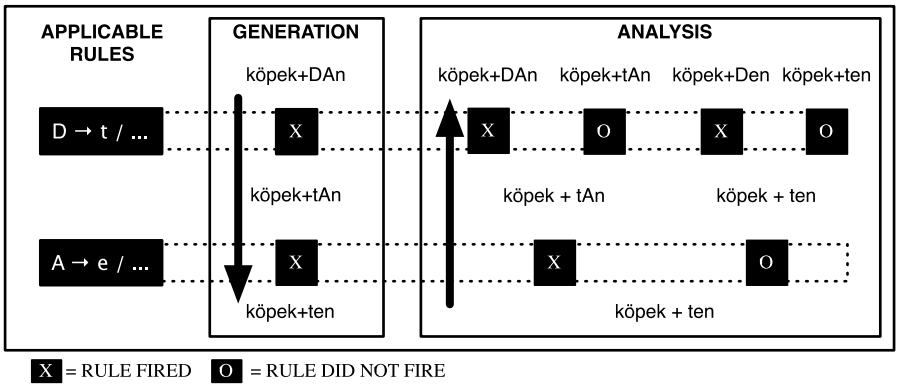



Fig. 2. Overanalysis problem in Turkish: Generation given this set of rules is straightforward while analysis is nondeterministic, as there are many possible underlying forms that can generate the surface form köpek+ten. Note "+" represents a morphological boundary.

2.2.2 Morphological Induction. Morphological induction (MI) encompasses a set of modeling approaches that fall roughly into two categories: relation-based and segmentation-based.

*Relation-Based Approaches.* The relation-based approaches attempt to learn relations between word forms; for example, that *funny*, *funnier*, and *funniest* are related by a common stem: *funn-*. A variety of techniques have been proposed to learn these sorts of relations automatically. There are alignment models, which use string and semantic similarity features to match ⟨inflected-form, root-form⟩ pairs like ⟨*funnier, funny*⟩ [Yarowsky and Wicentowski 2000]. There are also conflation set models, which attempt to learn sets of words that share the same stem using latent semantic analysis as a proxy for meaning, in combination with string-level features [Schone and Jurafsky 2001]. Then there are techniques that fall more loosely into the category, such as Goldsmith [2001]. Unlike the other approaches, Goldsmith's *Linguistica* seeks to relate words by learning which share the same inflectional paradigms, as opposed to how they cluster together according to semantic similarity and other features.

*Segmentation-Based Approaches.* The segmentation-based approaches attempt to segment words into morphs, without explicitly modeling how they are related. Some past approaches have used heuristics to find natural points of segmentation. For example, Déjean [1998] proposes a technique that uses local peaks in character perplexity to predict morphological boundaries, which has origins in Harris [1951].

Most recent approaches use mathematical models trained by unsupervised learning. Minimal description length (MDL), initially a framework for data compression, has been widely used, first by de Marcken [1996] then later by Deligne and Bimbot [1997], Goldsmith [2001], and Creutz and Lagus [2002].

The goal of MDL is to minimize the description length of a corpus, which is defined as the sum of the model bit-length and the encoding bit-length of the corpus given the model. In morphological MDL, the units in the model are the best set of morphs from the corpus, and these are used to segment (encode) the words in the corpus in the least costly way [Goldsmith 2001; Creutz and Lagus 2002].

There are also statistical approaches, such as the maximum a posteriori (MAP) approach, the goal of which is to find a model to make the observed data most likely, as well as satisfy prior distributions on what models should contain. Many MAP approaches try to maximize the probability of the model against linguistically motivated priors [Deligne and Bimbot 1997; Snover and Brent 2001; Creutz and Lagus 2005]. MDL is similar to the MAP, and in practice can be approximated as MAP estimation, using priors motivated by complexity theory.

Also in the statistical category are the pure maximum likelihood (ML) approaches, which search for the model to make the corpus most likely, but where the optimal model, that is, "the best set of morphs," is not mathematically formalized. These approaches often refine the model with heuristics, relying on categorical decisions to include one morph or another, based on linguistic properties [Creutz and Lagus 2004], or other properties, such as frequency at a particular level of segmentation granularity [Peng and Schuurmans 2001].

Some of the segmentation approaches mentioned above use a simplistic unigram model of morphology to produce the segmentation of the corpus given the model. Substrings in the model are proposed as morphs within a word based on their own likelihood, independent of phrase-, word-, and morph-contexts [de Marcken 1996; Peng and Schuurmans 2001; Creutz and Lagus 2002]. Other approaches, however, have more complex, more realistic models of morphology. These models attempt to constrain segmentations in some linguistically plausible way, in order to produce more accurate analyses. For example, work by Creutz and Lagus [2004; 2005; 2006] constrains segmentations using morphotactics, assigning morphotactic categories (prefix, suffix, and stem) to a first-pass morph segmentation, and then training an HMM using those category assignments. Other more structured models include Goldsmith's [2001] work which attempts to learn the best signatures (e.g., morph-level partial paradigms such as {ø, *-ing*, *-ed*, *-s*} for English regular verbs) and what words inflect with what signatures.

*Allomorphy in Morphological Induction.* Allomorphy, or allomorphic variation, is the process by which a morpheme varies in particular contexts, as constrained by a grammar. To our knowledge, there is only a handful of work within MI attempting to integrate allomorphy into morpheme discovery. A notable approach is the Wordframe model developed by Wicentowski [2002], which performs weighted edits on root-forms, given context, as part of a larger similarity alignment model for discovering ⟨inflected-form, root-form⟩ pairs.

A limitation of Wicentowski [2002] is that morphological complexity is fixed by a template designed for simple inflectional morphologies, constrained to

finding an optional affix on either side of a stem. Several authors have modified Wicentowski's template to represent the morphotactic properties of more complex languages. For example, Cheng and See [2006] add infixation and reduplication, boosting performance on Filipino. To the best of our knowledge, no template has been implemented for an agglutinative language such as Finnish or Turkish.

In terms of allomorphy, Wicentowski's approach succeeds at generalizing allomorphic patterns, both root-internally and at points of affixation. A major drawback is that, so far, it does not account for affix allomorphy that involves affix-internal character replacement; such allomorphy is beyond the scope of point-of-affixation insertions and deletions.

### 2.3 The Baseline System

We use the Morfessor Categories-MAP algorithm developed by Creutz and Lagus [2005; 2006] as both baseline and preprocessor for our system. In this approach, a first-pass segmentation is refined using an HMM designed to account for the morphotactic behavior of morphs.

The first-pass segmentation is produced by the recursive MDL technique described in Creutz and Lagus [2002]. This technique begins with a source text and attempts to learn a model (or codebook) to predict the optimal segmentation (or representation under the codebook) of a source text. Under the MDL approach, a segmentation is optimal when the sum of the costs of the representation and codebook is minimized. The approach is noisy and suffers from spurious segmentation. For instance, the English word *strap* may be improperly segmented as *s+trap* because "*trap*" is a frequent morph that appears on its own as a word, and "*s*", as an affix, is modeled as just another frequent morph, with no indication as to where in a word it must attach. This approach treats words as bags of morphs, so morphs are frequently posited in impossible positions.

The Categories-MAP [Creutz and Lagus 2005] approach attempts to improve the first-pass segmentation by adding a hidden layer to account for morph ordering constraints, in other words, to account for the structure that the MDL approach ignores. The Categories-MAP approach utilizes morphotactic category sequences made up of morphotactic tags like prefix, suffix, and stem, to predict how morphs such as *-s* typically behave. The sequences and segmentations are learned in an HMM model, which produces a more accurate segmentation than MDL, mainly because it predicts morphotactically plausible sequences.

The Categories-MAP approach attempts to find the best model given a corpus and thus attempts to maximize the probability of the model given the corpus:

$$\underset{model}{\mathtt{argmax}} P(model|corpus) = \underset{model}{\mathtt{argmax}} P(corpus|model) \cdot P(model) \qquad (1)$$

The model in the Categories-MAP approach is a morph lexicon. The probability of the model is calculated by the following equation, where M is the size of the lexicon, and $s_i$ is the $i^{th}$ morph in the lexicon:

$$P(model) = M! \cdot \prod_{i=1}^{M} [P(meaning(s_i)) \cdot P(form(s_i))] \tag{2}$$

Two distributions, $P(meaning(s_i))$ and $P(form(s_i))$, are combined to get the overall model probability. The factor $M!$ accounts for possible orderings of the morphs in the model. For the meaning distribution, a morph's probability is made up of priors on morph length, frequency, and usage (in the form of left- and right-perplexity). For the form distribution, a morph's probability is calculated as either the probability of its characters, or if it has any substructure, the probability of its submorphs.

The probability $P(corpus|model)$ is the product of $P(w_j|model)$, where $w_j$ is the $j^{th}$ word in the training data and W is the size of the training data, as shown in Equation (3). The probability $P(w|model)$ is calculated as a first-order HMM, where $s_1 \ldots s_n$ is the surface segmentation of a word $w$ and $C_1 \ldots C_n$ is its tag sequence, as shown in Equations (4) and (5):

$$P(corpus|model) = \prod_{j=1}^{W} P(w_j|model) \tag{3}$$

$$P(w|model) = \prod_{k=1}^{n} \left[ P(s_k|C_k) \cdot P(C_{(k+1)}|C_k) \right] \tag{4}$$

$$= \left( \prod_{k=1}^{n} P(s_k|C_k) \right) \cdot P(C_1) \prod_{k=1}^{n-1} P(C_{(k+1)}|C_k) \tag{5}$$

Categories-MAP uses a greedy search to find the best model and segmented corpus. It begins by initializing a model using the first-pass segmentation. The model is then refined in phases. One phase involves splitting morphs apart, and another involves joining shorter morphs followed by longer morphs together in a bottom-up strategy. The above-mentioned phases revise the morphs in the model and may either delete morphs, or modify them by adding substructure (submorph and tag sequences). Splitting and joining phases alternate with decoding phases and model-estimation phases. The output at the end of the procedure is a segmented corpus.

## 3. METHODOLOGY

### 3.1 The System Design

The approaches mentioned in the previous section have some limitations. Finite state morphology can produce accurate analyses that are consistent

with linguistic judgments, but in order to do so, it must have access to a lexicon containing the valid morphemes, as well as rules governing allomorphic and morphotactic properties. This is a serious practical drawback, given that creating or procuring a large morphological lexicon to cover a language's vocabulary is often quite expensive.

Induction approaches, on the other hand, can be built with limited lexicon resources, or none at all. MI learns morphological structure from resources as simple as word lists. However, MI presents a drawback in terms of overall quality, compared with FSM. MI models are never exact representations of underlying morphological phenomena, though modeling assumptions have improved over time. For example, as mentioned in Section 2.3, without a representation of morphotactic structure, MI models may produce spurious segmentations like *s+trap*. There are now models that take structure into account, as the Morfessor Categories approaches do. Another modeling difficulty is allomorphy. In allomorphically naive systems, morphological variants such as *-s* and *-es* are learned separately as distinct units, despite the fact that they are variants of the same morpheme.

Our approach combines rewrite rules from the FSM approach with an unsupervised MI system, also referred to as the baseline approach, with the goal to handle allomorphy appropriately. Hand-written rewrite rules, which are approximations meant to capture regular orthographic variation, are used to express the relationship between morphs and underlying morphemes; the MI system then chooses the best segmentation of words into a sequence of underlying morphemes.

There are several advantages to this hybrid approach. First, unlike a strictly rule-based FST approach, it does not require a morpheme lexicon. Also, the rules can overgenerate (i.e., posit some bad analyses), because the system is, in the end, choosing the most likely segmentation based on a statistical model. The main difference between our hybrid approach and the baseline approach is that, for a given word, our system will produce an underlying segmentation (i.e., a morpheme sequence) in addition to a surface segmentation (i.e., a morph sequence). To achieve that, we use rewrite rules at various stages of the system to generate the morphemes from the morphs in the surface segmentation.

### 3.2 Context-Sensitive Rewrite Rules

Our rules capture regular, context-driven spelling-changes, such as the variation of the English plural morpheme, which is *-es* after sibilants and (sometimes) vowels, and *-s* elsewhere.[2] To capture spelling-changes, we use ordered, SPE-style, context-sensitive rewrite rules. Like the rules introduced in Section 2.2.1, ours have the general form:

$$\underset{\text{underlying}}{\alpha} \quad \rightarrow \quad \underset{\text{surface}}{\beta} \; / \; \underset{\text{l. context}}{\gamma} \; \_ \; \underset{\text{r. context}}{\delta}$$

[2]To keep the rule set (and thus the amount of required knowledge) small, we shall make no attempt to cover irregular variants, such as the plural morph *-en* in *ox+en*.

Per convention, rules are written in the generation direction even though they operate in the analysis direction. The rules have the following effect: whatever is matched by the surface element $\beta$ at the focus position "_", between the context of $\gamma$ and $\delta$, is replaced by the underlying element $\alpha$. The elements (which are variables) may hold characters, character classes, character (class) sequences, as well as the empty character, "ø". The rules can reverse insertions ($\alpha = \emptyset$), deletions ($\beta = \emptyset$), or substitutions, when triggered by the context. Context elements, $\gamma$ and $\delta$, will frequently contain "+" to indicate where a segment boundary must lie. One or the other context element may be absent when the conditioning environment is on just one side of the focus.[3]

### 3.3 Underlying Mapping Function $\Upsilon$

As laid out in the previous sections, the goal of the current task is to discover underlying word structure, that is, to break words into morphemes. We can do this by specifying the spelling-change rules in a language, and then using those rules to analyze the segmentation. We define $\Upsilon(\mathbf{s}, R)$ to be a function that takes a surface segmentation $\mathbf{s}$ and a set of rewrite rules $R$ as input, and produces the set of possible underlying segmentations as output.

To avoid undesirable behavior such as infinite looping, rules are not applied to their own output. Instead, each rule applies once to each position (from left-to-right) over $\mathbf{s}$, incorporating changes as it goes. The output is then fed to the next rule which applies to each (possibly modified) position, and the process continues until all rules have been applied. This enables multi-step analyses using rules designed specifically to apply to the outputs of other rules.

We note that, as in the FSM approach, our rule system faces the problem of overanalysis. That is, given a set of rewrite rules and a surface segmentation, the number of underlying segmentations can be exponential with respect to the number of rules. This can be observed in Figure 3, which shows segmentations for English *citi+es* and Turkish *köpek+ten* (*from the dog*) and their respective underlying segmentations after applying rules. In each case there are two rules, which can either apply or not apply, leading to four possible underlying segmentations. In Section 4.2, we will show how, in our approach, we use a statistical model to select the best possible underlying segmentation from these possibilities.

### 3.4 Modifying $\Upsilon$ to Control Overanalysis

Given the unsupervised nature of our approach, there is no lexicon to confirm which underlying segmentations are correct, nor are there lexical automata built to constrain analyses to valid ones as would be the case in pure FSM systems. For practical purposes we control the exponential blowup with the simplifying assumption that either all rules that can apply will apply, or none will. Figure 4 shows how this assumption works in practice. In the figure,

[3]Some special substitution rules, such as vowel harmony in Turkish and Finnish, have a spreading effect, moving from syllable to syllable within and beyond morph-boundaries. In our formulation, these rules differ from others by not being conditioned on a morph-boundary.

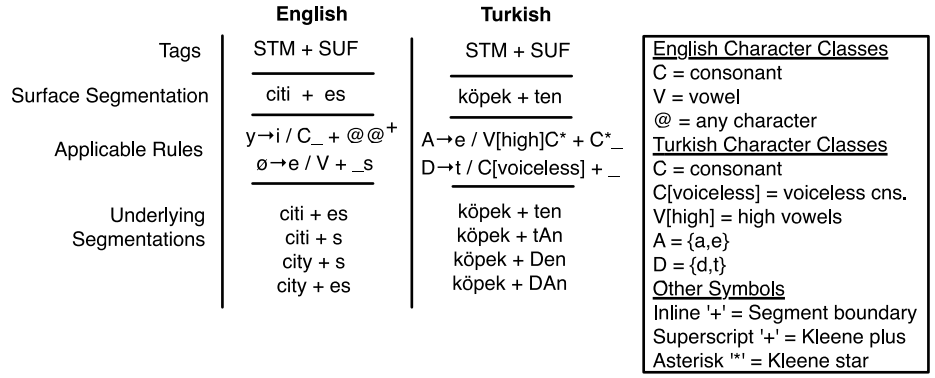



Fig. 3. Applying rules to surface segmentations generates an exponential number of underlying segmentations for English *citi + es* and Turkish *köpek + ten*.

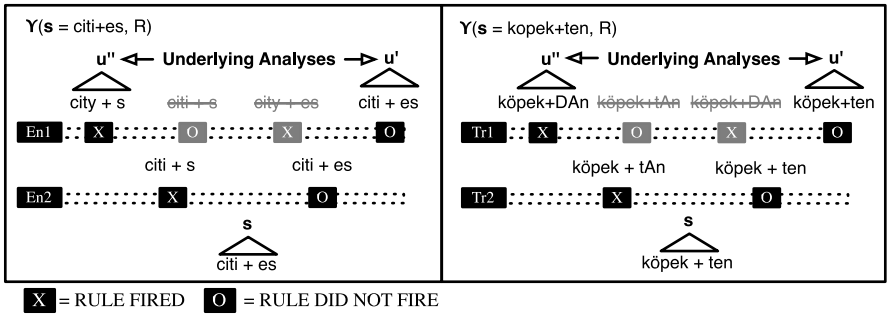



Fig. 4. The underlying mapping function $\Upsilon$ controls overanalysis by not generating intermediate possibilities.

segmentations where one rule applied but another did not, like the Turkish segmentation *köpek+Den*, are not generated by $\Upsilon$.

With this change, for a given surface segmentation **s**, the set of underlying segmentations produced by $\Upsilon$ contains only two members: $\mathbf{u}'$ and $\mathbf{u}''$, where $\mathbf{u}'$ is the underlying segmentation when no rules applied so $\mathbf{u}' = \mathbf{s}$, and $\mathbf{u}''$ is the segmentation when all rules that could have applied did apply. To distinguish $\mathbf{u}'$ and $\mathbf{u}''$ from other underlying segmentations, we refer to them as *underlying analyses*. The modified underlying mapping function $\Upsilon(\mathbf{s}, R)$ is called by Steps 1, 2, 4, and 5 of our procedure, which we will explain in the next section.

## 4. PROCEDURE

In this section, we describe the components of the system, starting with an overview, and then covering each step in some detail.

The system is modeled after the Morfessor Categories approaches, where a word $w$ is segmented into a sequence of morphological segments generated

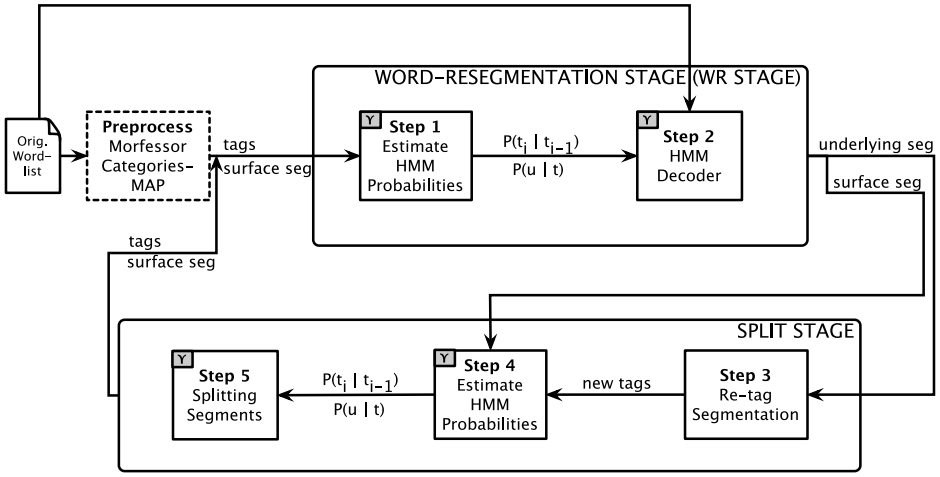



Fig. 5. Flowchart showing the entire induction procedure, going from Preprocessing, the goal of which is to produce an initial segmentation, to Word-Resegmentation and Split stages, the goals of which are to produce and refine an underlying segmentation. The output of the procedure is an underlying segmentation (words separated into morphemes) and surface segmentation (words separated into corresponding morphs).

by a morphotactic tag sequence. We use the same tags as Morfessor: {prefix, stem, suffix}. The primary difference is while Morfessor emits morphs $s_i$, our system emits morphemes $u_i$. Morphemes may be surface morphs acting as morphemes, or candidates produced by the rules, as determined by the function $\Upsilon$ defined in Section 3.4. We alter training and HMM decoding in the Morfessor system to accommodate this difference in approach.

As shown in Figure 5, the system takes a list of ⟨word, word-frequency⟩ pairs and a set of rewrite rules as input, and as output produces an underlying segmentation, tag sequence, and corresponding surface segmentation. The word list is preprocessed by Morfessor Categories-MAP. Tags and surface segmentations from preprocessing are fed to the Word-Resegmentation Stage (WR Stage), the goal of which is to find the maximum probability underlying segmentation and tag sequence. Decoding in the WR Stage also creates a surface segmentation corresponding to the underlying segmentation.

The output of the WR Stage tends to be undersegmented because it starts from conservatively segmented input. The Split Stage is introduced to address this problem. The output of WR is fed to the Split Stage which breaks some segments in the segmentation into (the most probable sequence of) smaller segments.

The Split Stage, on its own, results in massive oversegmentation, since it splits some segments without considering the greater context (i.e., the word) in which they occur. For that reason, segmentations produced by the Split Stage are fed back to the WR Stage. Here, morpheme segmentation is done at the word level once more, using a model reflecting a less redundant encoding. The procedure stops after the second pass of the WR Stage. In Steps 1, 2, 4,

and 5, $\Upsilon$ is called to generate underlying analyses, as indicated by the small shaded boxes in Figure 5.

### 4.1 Preprocessing

We use the Morfessor Categories-MAP algorithm developed by Creutz and Lagus [2005; 2006], which takes a frequency-annotated word list and produces a surface segmentation and tag sequence. For more detail on this procedure, please refer to Section 2.3.

### 4.2 Word-Resegmentation Stage

The WR Stage has two steps: Given the initial segmentation produced by the preprocessing step, Step 1 estimates the probabilities associated with an HMM model. Step 2 involves decoding of each word, that is, finding its maximum probability tag and morpheme sequence.[4]

#### Step 1: Estimate HMM Probabilities

Transition probabilities $P(t_i|t_{i-1})$ are estimated by maximum likelihood, given a tagged input surface segmentation.

For each surface segmentation $\mathbf{s}$, an underlying mapping function $\Upsilon$ is called to generate underlying analyses $\mathbf{u}'$ and $\mathbf{u}''$; all (unique) morphemes $u_i$ found in $\{\mathbf{u}', \mathbf{u}''\}$ are used to estimate $P(u_i|t_i)$. A morpheme $u_i$ can either be identical to its associated surface morph $s_i$, or different when modified by the rewrite rules.

When an underlying morpheme $u_i$ is associated to a surface morph $s$, we refer to $s$ as an *allomorph* of $u_i$. The probability of $u_i$ given tag $t_i$ is calculated by summing over all allomorphs $s$ of $u_i$ the probability that $u_i$ realizes $s$ in the context of tag $t_i$:

$$P(u_i|t_i) = \sum_{s \in \text{allom.-of}(u_i)} P(u_i, s|t_i) \tag{6}$$

$$= \sum_{s \in \text{allom.-of}(u_i)} P(u_i|s, t_i)P(s|t_i) \tag{7}$$

Both Equation (6) and Equation (7) are estimated by counting the segments in the input and output of $\Upsilon$ (for more on $\Upsilon$'s output, see Figure 4).[5]

#### Step 2: HMM Decoder

Next we resegment the wordlist into underlying morphemes.

[4]The output of Step 2 may be fed back to Step 1 in order to perform EM, if desired. However, in practice, running more than one iteration of word resegmentation has not helped performance.

[5]We show Equation (7) because it has the term $P(u_i|s, t_i)$, which may prove useful for thresholding and discounting terms of the sum where $u_i$ is rarely associated with a particular allomorph and tag.

Searching for the best breakdown of a word $w$ into morpheme sequence $\mathbf{u}$ and tag sequence $\mathbf{t}$, we maximize the probability of the following formula, where $U_w$ is the set of underlying segmentations of $w$.

$$\underset{\mathbf{u},\mathbf{t}}{\texttt{argmax}}[P(\mathbf{u},\mathbf{t}|w)] = \underset{\mathbf{u}\in U_w,\mathbf{t}}{\texttt{argmax}}[P(\mathbf{u},\mathbf{t})] \tag{8}$$

$$= \underset{\mathbf{u}\in U_w,\mathbf{t}}{\texttt{argmax}}[P(\mathbf{u}|\mathbf{t})P(\mathbf{t})] \tag{9}$$

With independence assumptions and a local time horizon, we estimate:

$$\underset{\mathbf{u},\mathbf{t}}{\texttt{argmax}} P(\mathbf{u}|\mathbf{t})P(\mathbf{t}) \approx \underset{\mathbf{u},\mathbf{t}}{\texttt{argmax}}\Big[\prod_{i=1}^{n} P(u_i|t_i)P(t_i|t_{i-1})\Big] \tag{10}$$

The search for the maximum probability tag and morpheme sequence in Equation (10) is carried out by a modified version of the Viterbi algorithm. The algorithm effectively explores all underlying segmentations $\mathbf{u} \in U_w$ derived from surface segmentations $\mathbf{s} \in S_w$, where $S_w$ is comprised of all surface span sequences in word $w$. Underlying analyses $\mathbf{u}'$ and $\mathbf{u}''$ output by $\Upsilon(\mathbf{s}, R)$, and all ordered underlying sequences constructed by mixing $u'_i \in \mathbf{u}'$ and $u''_i \in \mathbf{u}''$ are considered part of $U_w$ and are explored by the algorithm. For example, given analyses $\mathbf{u}' = citi + es$ and $\mathbf{u}'' = city + s$, $U_w$ consists of these four segmentations: *citi* + *es*, *citi* + *s*, *city* + *es*, and *city* + *s*, which are explored. Mixing allows the possibility of selecting an underlying segmentation distinct from $\mathbf{u}'$ and $\mathbf{u}''$ – one which has selected rule-modifications for some morphemes, but rejected them for others. The entire algorithm is available in Appendix A.

### 4.3 Split Stage

Many times, segments in the model will have internal substructure and yet be too frequent to be split when considered at the WR Stage. For example, a word such as *baking* may never have been segmented into *bak(e)* + *ing* because the segment *baking* was too frequent on its own. We use the Split Stage to overcome this segmentation problem, encouraging splitting of segments into sub-segments. Given $\Upsilon$, *baking* can be split into the morpheme sequence *bake* and *ing*. Therefore, we no longer need *baking* in the model lexicon. Ideally then, when probabilities are re-estimated after splitting, segments like *baking* will no longer be available, or will have significantly reduced likelihood.

This stage closely follows the Morfessor Categories-ML heuristic splitting procedure described in Creutz and Lagus [2004]. The main differences from Morfessor are (1) segments in our system are morphemes, not morphs, (2) we allow more than one split per segment (Morfessor only allows binary splitting), and (3) we introduce a typology parameter which determines whether certain types of segments are to be split. Following Creutz and Lagus [2004] we attempt to control spurious splits by first re-tagging (Step 3) to identify which morphemes are noise (fragmentary) and should not be used, and after re-tagging, re-estimating HMM probabilities (Step 4; same as Step 1) and using the new probabilities to split segments (Step 5).

### Step 3: Re-tag Segmentation

In Creutz and Lagus [2004], segments are re-tagged to identify which segments are likely to be noise by estimating a distribution $P(CAT|u_i)$ with three true categories $CAT$ = {prefix, stem, suffix} and one noise category. The probabilities of true categories are tied to characteristic features of the morphemes as well as to the value of certain cutoff parameters, the most important of which is $b$. Parameter $b$ thresholds the probability of affixes. The probability of the noise category is conversely related to the product of the true category probabilities, so when true categories become less probable, noise becomes more probable. Following Creutz and Lagus [2004], we tune the amount of noise by adjusting the parameter $b$. For more on re-tagging and a more precise definition of parameter $b$, see Appendix B.

### Step 4: Estimate HMM Probabilities

This is the same as Step 1, as discussed in Section 4.2.

### Step 5: Splitting Segments

As in Creutz and Lagus [2004], each ⟨segment, tag⟩ pair in the input surface segmentation is examined to determine whether a split is warranted. For each segment that it is possible to split, the optimal split is chosen by performing HMM decoding on the segment, a process identical to Step 2 except that the decoder only considers the segmentations that do not violate the constraints mentioned below.

There are several tag-based constraints on the splitting process: Segments with the tag "noise" are not allowed to be split; segments with the tag "stem" are split into the sequence: (prefix*+stem+suffix*); segments with affix tags (prefix or suffix) are split into segments with the same tag. We modify the approach slightly, making affix splitting an optional parameter that may be set according to typological properties of the language. If a language has rich suffixation, for example, we would hand-set this parameter to allow suffix-splitting.

## 5. EXPERIMENTS

We evaluated our system by running it on the English and Turkish data used by the Morpho Challenge contests for 2005 and 2007 [Kurimo et al. 2006, 2007]. In this section, we first give an overview of the data and evaluation metrics. Next, we describe how the system parameters were chosen. We end the section with the experimental results and some analyses.

### 5.1 Data

Morpho Challenge is part of the EU Network of Excellence PASCAL Challenge Program, and beginning in 2007, has been organized in collaboration with CLEF. Morpho Challenge 2005 uses three languages: Finnish, English, and Turkish. Morpho Challenge 2007, a follow-up to the 2005 contest, adds German to the language list. We chose English and Turkish because we were

Table II. Sizes of Morpho Challenge Datasets by Number of Unique Words

| Morpho Challenge 2005 | | | |
|---|---|---|---|
| | Training | Development | Test |
| English | 167,377 | 532 | 40,000 |
| Turkish | 582,923 | 774 | 60,000 |
| Morpho Challenge 2007 | | | |
| | Training | Development | Test |
| English | 384,903 | 410 | 117,000 |
| Turkish | 617,298 | 593 | 387,000 |

Table III. Data Examples

| **Training Data** | Gold Standard | |
|---|---|---|
| ⟨ Freq., Word ⟩ | 2005 | 2007 |
| 25 cremation | cremat+ion | cremate_V+ ion_s |
| 3 crispest | crisp+est | crisp A+SUP |
| 54 crucially | cruc+ial+ly | crux_N+al_s+ly_s |
| 741 cubic | cub+ic | cube_N+ic_s |
| 7 curtailments | curtail+ment+s | curtail_V+ment_s+PL |
| 1368 glasses | glass+es | glass N+PL |
| 811 cups | cup+s | cup_N+PL |

familiar with the languages, which allowed us to create rewrite rules without much difficulty.

For each language, the Morpho Challenge contest provides a large dataset for training, and two smaller evaluation (gold-standard) datasets for evaluation: a tiny dataset used for the development phase, and a larger one for final test. Table II shows the size of the data sets in terms of number of unique words. A more detailed description is available in the contest reports [Kurimo et al. 2006; 2007].

5.1.1 Training Data. In both contests, the training data is a list of words associated with word counts: the words were collected from a variety of sources, and the counts were the cumulative frequencies of the words in these sources. Some examples are shown in the first column of Table III. For Morpho Challenge 2007, the source text from which the words and frequencies were culled were also provided, but our experiments did not make use of it.

5.1.2 Evaluation Data. The gold standard for the 2005 and 2007 contests are substantially different, reflecting shallower and deeper levels of analysis, respectively. Some examples are given in the second and the third columns of Table III.

For 2005, the evaluation data consists of surface segmentations of words, that is, words that have been separated into substrings known as morphological segments (morphs). A word like *glasses* would be segmented *glass+es*, for instance.

For 2007, the evaluation data consists of underlying morphological analyses. The analyses consist of a lemma, derivational affixes, and inflectional tags. The lemma is tagged with its part of speech, for example, _*N* = noun. Derivational affixes are tagged with a morphotactic position, for example, _*s* = suffix, and inflectional affixes are often represented by tag alone, like

Table IV. Final Test *F*-scores

| *MC 2005-Style Evaluation* | English | | | Turkish | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| MC 2005 Top Score | 76.2 | 77.4 | **76.8**[a] | 77.5 | 65.0 | 70.7[b] |
| Morfessor MAP | 85.1 | 54.2 | 66.2 | 77.5 | 65.0 | 70.7 |
| Baseline[*] | 76.7 | 47.9 | 58.9 | 78.0 | 56.7 | 65.7 |
| Hybrid -No Rules | 66.1 | 63.6 | 64.8 | 66.0 | 67.5 | 66.7 |
| Hybrid - With Rules | 67.3 | 69.3 | 68.3 | 77.0 | 75.9 | **76.5** |

| *MC 2007-Style Evaluation* | English | | | Turkish | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| MC 2007 Top Score | 61.6 | 60.0 | **60.8**[c] | 76.4 | 24.5 | 37.1[b] |
| MC 2008 Top Score | 50.6 | 63.3 | 56.3[d] | 51.9 | 52.1 | 52.0[d] |
| Morfessor MAP | 82.2 | 33.1 | 47.2 | 76.4 | 24.5 | 37.1 |
| Baseline[*] | 81.8 | 33.0 | 47.0 | 81.1 | 20.5 | 32.8 |
| Hybrid - No Rules | 69.3 | 48.9 | 57.4 | 80.3 | 19.3 | 31.1 |
| Hybrid - With Rules | 69.2 | 52.9 | 59.8 | 61.2 | 49.2 | **54.5** |

[a] RePortS [Keshava and Pitler 2006]
[b] Morfessor MAP [Creutz and Lagus 2005]
[c] Bernhard [Bernhard 2007]
[d] ParaMor+Morfessor [Monson et al. 2008]
[*] Morfessor MAP trained by us for use in our procedure. The score is likely lower than MC Morfessor MAP because it was not tuned extensively.

*PL* = plural, which may abstract away from surface variation. For example, *glasses* has the analysis *glass_N+PL*, and *cups* had *cup_N+PL*. From these analyses we can easily identify that both words share the morpheme *PL*.

### 5.2 Evaluation Metrics

For evaluation, the recent Morpho Challenge contests use *F*-measure-based evaluation metrics, which assess the quality of system output by calculating how many key features it shares with the gold standard. *F*-measure is the harmonic mean of precision and recall, as shown below:

$$F\text{-}Measure = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{11}$$

Here, *precision* is the number of hits (components appearing in both the gold standard and the system output), divided by the number of hits plus the number of insertions (components appearing only in the system output). *Recall*, on the other hand, is the number of hits divided by the number of hits plus the number of deletions (components found only in the gold standard).

While the definition of *F*-measure remains the same, exactly what constitutes a hit, deletion, and insertion differs in the two contests, as discussed below.

5.2.1 F-Measure from Morpho Challenge 2005. This is a popular metric, and has been used extensively to measure the effectiveness of recent morphological induction methods [Creutz and Lagus 2004; 2005; Kurimo et al. 2006; Cheng and See 2006; Demberg 2007]. It tracks the extent to which

surface-segmentation boundaries match between the system output and a gold standard.

In this measure, a hit is a boundary match between the system output and the gold standard, and insertion and deletion are defined similarly. For example, suppose the word *unlikable* is segmented as *un+likable* by a system and as *un+lik+able* in the gold standard. There would be one hit, the boundary after *"un"*, one deletion, the boundary after *"lik"*, and no insertions. Therefore in this example precision would be 100% and recall 50%. We use this metric to measure the surface segmentation produced by our system.

5.2.2 F-Measure from Morpho Challenge 2007. This metric replaced the 2005 Morpho Challenge evaluation metric, and has been used to evaluate contestants starting with the 2007 contest [Kurimo et al. 2007]. It tracks the consistency of underlying morphemes shared across word pairs. For instance, there should be a morpheme shared between *glasses* and *cups*, even though the allomorphs that appear in those words, *-s* and *-es*, are different; submissions that do not have the correct morpheme correspondences between words are penalized. We use this metric to measure the underlying segmentation produced by our system. Below we explain how precision and recall are calculated.

*Calculating Precision.* First, a random sample of words is selected from the system output. For each word in the selected sample and each morpheme in the word, a "linked" word (a word that shares that morpheme) is selected at random from the system output, forming a word pair. A hit is counted when a pair of linked words selected from the system output also share a morpheme in the gold standard. An insertion is counted when words are linked in the system output, but not in the gold standard. Finally, the precision score is calculated as hits divided by the sum (hits + insertions).

For example, suppose a morphological induction method has segmented the word *cups* as *cup+s*. The pair selected for the first morpheme might be ⟨**cup**+*s*, **cup**+*holder*⟩, and for the second morpheme: ⟨*cup*+**s**, *spoon*+**s**⟩. In the gold standard, these word pairs would be analyzed as: ⟨**cup_N**+*PL*, **cup_N**+*holder_N*⟩ and ⟨*cup_N*+**PL**, *spoon_N*+**PL**⟩. Although the gold-standard morphemes appear different from the system output, what matters is the gold-standard pairs share a common morpheme in each case, just like the system output. Therefore we have two hits and no insertions, yielding a precision of 100%.

*Calculating Recall.* The recall calculation begins with a selection of the gold-standard analyses, which, though initially selected at random, is the same for all participants. For each word in the selection, and for each morpheme in the word, a linked word is selected from the gold standard, forming a pair. Analogously to the precision calculation, a hit is counted when a pair of linked words in the gold standard is also linked in the system output. A deletion is counted when words are linked in the gold standard, but not in the system output. The recall score is then calculated as hits divided by the sum (hits + deletions).

For example, in the gold standard, suppose *glasses* has the analysis *glass_N+PL*, and given that, assume the following linked pairs are selected: ⟨**glass_N**+*PL,* **glass_N**+*blow_V+VBG*⟩ and ⟨*glass_N*+**PL**, *cup_N*+**PL**⟩. Suppose, in the system output, the preceding pairs are rendered ⟨**glass**+*es,* **glass**+*blowing*⟩, and ⟨*glass+es, cup+s*⟩ respectively. The first pair is linked by a shared morpheme, yielding a hit, but the second pair is not, yielding a deletion. The recall score is then one out of two, or 50%.

### 5.3 Rewrite rules

Rewrite rules and orthographic classes used in our system were culled from linguistic literature. We currently use 6 rules for English and 10 for Turkish. These rules are displayed in Appendices D and E.

The aim was not well-tuned rule sets, but rather to show that one can get significant improvement even with a small set of imperfect rules. Therefore we include only the rules that will apply most generally; rules that apply only to exceptional cases in the lexicon are not encoded by rule. An example would be the Turkish consonant doubling rule, which affects only a small set of Arabic borrowings like *sır+Im* ∼ *sırrım*, (*secret+1Pos* ∼ *my secret*). Writing a rule set involved taking generalizations from descriptive grammars about characters that undergo changes, and encoding those generalizations into rules.

We did not explicitly quantify rule-writing effort, but we do have a pretty good sense for the approximate time it took for each language. The English rules took approximately a day to write. The Turkish rules, on the other hand, took longer. It took about a week to settle on a Turkish rule set. We found the difference in difficulty between the two languages to be due in part to the complexity of the allomorphic phenomena involved, and in part to the relative usefulness of the available linguistic literature.

Regarding the former point, English has fewer regular allomorphic phenomena than Turkish in the written lexicon, and those it does have are less complex (for instance, none involve nonlocal dependencies, such as Turkish vowel harmony). Regarding the latter point, the grammar we used for English, Cambridge Grammar of English [Huddleston and Pullum 2001], has an orthography section with a thorough treatment of English orthographic conventions and spelling changes, which we were able to follow quite closely. We were unable to locate a Turkish grammar with a similar section, likely because Turkish orthography is much closer to its phonology, and many grammars conflate the two to some degree. Because of this, rule writing had to be somewhat incremental for Turkish, as we had to make sure that each phenomenon we represented as a rule was actually an orthographic phenomenon, and not (or not just) a phonological one.

### 5.4 Parameters

Our system has several tuned parameters. The parameters for preprocessing are tuned as suggested in Creutz and Lagus 2005. The main procedure has several numerical parameters involved in re-tagging similar to those in the

preprocessing step: we set those parameters as suggested by Creutz and Lagus [2004][6], and tune parameter $b$, defined in Section 4.3 and Appendix B, on the development data. Here, and every subsequent time it is mentioned, $b$ refers to the $b$ used in the main procedure; it is distinct from the $b$ used in preprocessing which was tuned separately to 200 for English and 375 for Turkish. In the following section, we show development results for $b$ = 100, 300, and 500.

We ran the procedure represented in the flowchart in Section 4, with an iteration of the WR Stage, followed by the Split Stage to split some redundant segments, followed by an iteration of the WR Stage again. The whole procedure as run herein can be described as 1 WR + 1 Split + 1 WR. In preliminary experiments, running this entire procedure multiple times did not provide any further benefit.

Finally, as described in Section 4.3, we use a hand-set typology parameter to indicate whether segments tagged as prefixes, or suffixes, or both are allowed to be split. This parameter is introduced to distinguish morphologically rich languages (e.g., Turkish) from more impoverished languages (e.g., English).

### 5.5 Evaluation Results

Our experiments make use of contest evaluation metrics for Morpho Challenge (MC) 2005 and 2007. When evaluating output of the main procedure, surface segmentations (morphs $s_i$) are evaluated by the 2005 metric, while underlying segmentations (morphemes $u_i$) are evaluated by the 2007 metric.

For development, we present results on small-scale evaluations. For the final test, we present results of large-scale evaluations, which were conducted by Morpho Challenge contest organizers at the Helsinki University of Technology.[7]

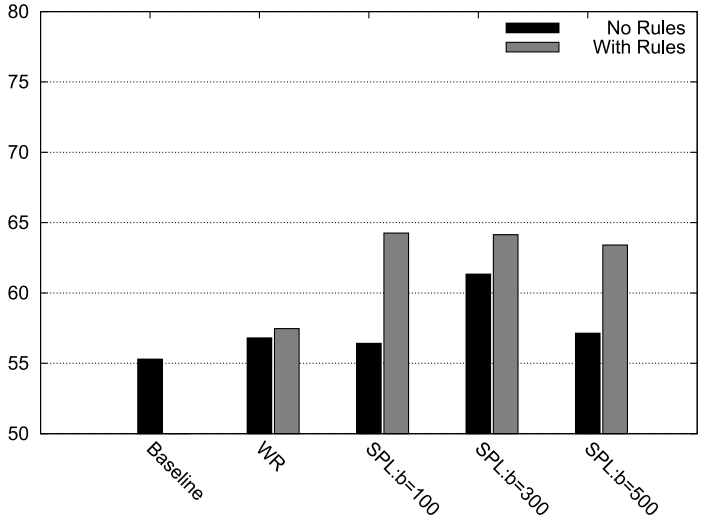
Results for the preprocessed segmentation are consistently used as a baseline. Also, in order to isolate the effect of the rewrite rules, we also ran our system with an empty rule set (the "no-rules" experiments). In effect, running without rules generates a surface-segmentation only, as the underlying and surface layers are the same.

5.5.1 Results on the Development Data. Figures 6 and 7 show the results on the development data for English and Turkish: Baseline refers to the results after the preprocessing step, *WR* to the results after the first pass of the WR Stage, and *SPL:b* = *N* to the results when the whole procedure is complete, that is, after the Split Stage (with parameter $b = N$) and another iteration of the WR Stage.

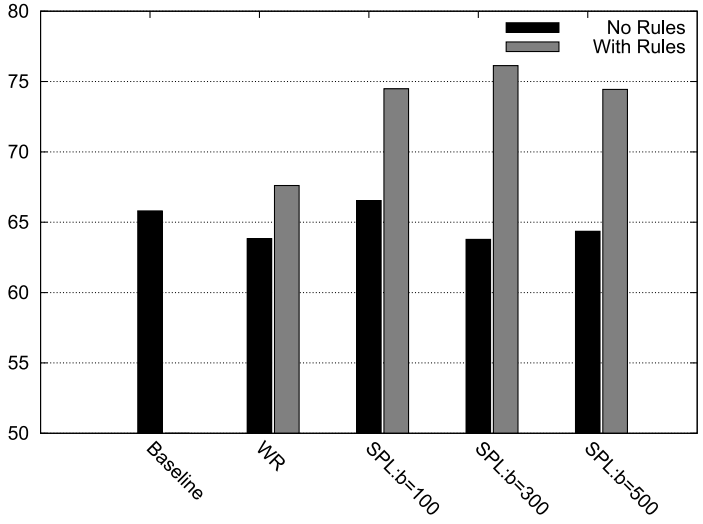
First, the overall results were very positive. For the 2005-style evaluation, with which we evaluate the surface layer of our segmentations, the largest *F*-score improvement was observed for English (Figure 6(a)), 55.30% to 64.26%,

[6]Because the procedure is a variation of the ML procedure laid out by Creutz and Lagus [2004], we use similar parameter settings.

[7]This work was submitted to Morpho Challenge 2007. Because the algorithm is not fully unsupervised, it did not formally compete, but instead was used as a reference method. This work was not submitted to Morpho Challenge 2005; evaluations on Morpho Challenge 2005 metrics were conducted nearly two years after the original contest.



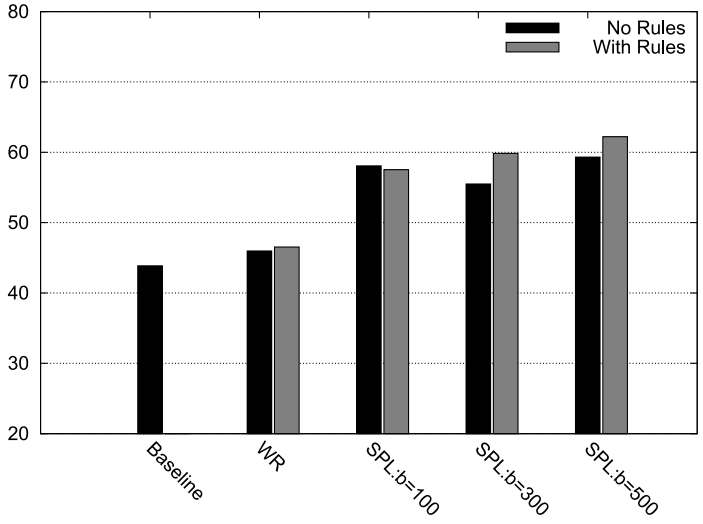

(a) English (*F*-score).



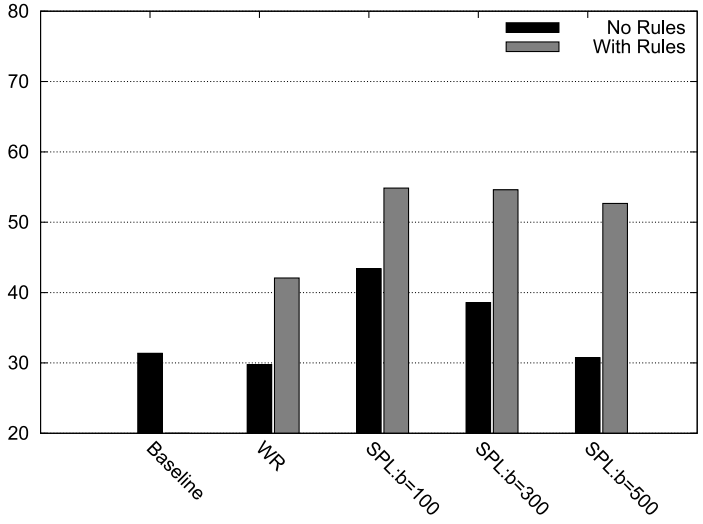

(b) Turkish (*F*-score).

Fig. 6. Development results for MC 2005-style evaluation.

an *F*-score gain of 8.96% over the baseline segmentation. The Turkish result also improves to a similar degree. For both languages, the bulk of the improvement is achieved only after the models have been refined by splitting. For the




(a) English (*F*-score).




(b) Turkish (*F*-score).

Fig. 7. Development results for MC 2007-style evaluation.

2007-style evaluation, which we take on the underlying-layer of our segmentations, the largest *F*-score improvement was observed for Turkish (Figure 6(b)), 31.37% to 54.86%, an *F*-score gain of more than 23% absolute.

Second, in all experiments with rules, the successive applications of the WR Stage and splitting result in consistent improvements in performance over the baseline. Without using allomorphic rules (no rules), the results may be negative compared to the baseline (see Figure 6(b)), or mixed (Figures 6(a) and 7(b)). A representative scenario is the 2005-style measure on the Turkish results (Figure 6(b)), which clearly shows segmentation improves reliably with rules, but not without. This indicates that segments in the model are improving as a result of the rules, since the 2005-style evaluation considers surface morphs, not morphemes. In other words, the improvement is not a result of mapping to morphemes alone, but rather, having made available consistent units that are easier to learn.

Third, large gains in *F*-score over the baseline may be observed in some no rules scenarios as well (see Figure 7(a)). One explanation for this may be that our approach is derived from the Morfessor Categories-ML, whereas the baseline segmentation is produced by Categories-MAP. Creutz and Lagus [2005] found that ML segmentations have better coverage than MAP segmentations, and sometimes the increased coverage results in a better *F*-score. When we looked at recall and precision directly, we found this explanation to be quite plausible, as we observed consistently large boosts in recall for no rules scenarios.

Fourth, there is a systematic difference between English and Turkish when it comes to the contribution made by the rewrite rules. Notice that for the 2005 evaluation, the difference by which the best with rules scenario outperforms the best no rules scenario is 2.80 for English and 9.59 for Turkish. For the 2007 evaluation, this difference is 2.90 for English and 11.46 for Turkish. Thus we see that the gap between with rules and no rules is quite a bit smaller for English than for Turkish, particularly for the 2007-style evaluation. The simplest explanation for this is that English has less allomorphy than Turkish. In English there are some morphemes that have variant forms, and those that do have at most two variants. In Turkish, nearly all morphemes have some variants, and can have up to twelve of them. In the with rules scenario, Turkish benefits much more from the amount of simplification the rules provide, while in English, the effect is more subtle.

5.5.2 Results on the Final Test Data. The results on the final test data are given in Table IV. We had three systems evaluated: the baseline used in our approach, the hybrid system in the with rules scenario, and the hybrid system in the no rules scenario.[8] The results overall are quite positive. The hybrid system with rules managed to outperform the baseline as well as the more extensively tuned MC Morfessor MAP, across all test scenarios.

For English, the top contest systems were RePorTs for 2005 [Keshava and Pitler 2006] and the Bernhard system for 2007 [Bernhard 2007]. The 2008 contest used the same evaluation and test data as 2007 and its top score was lower than 2007 because the top system in 2007 did not participate. Our hybrid

[8]For each system, the parameter $b$ was set to whatever value performed the best for that system during development.

system's performance for the 2007 dataset was comparable to the 2007 and 2008 top systems. Its performance for the 2005 dataset was lower than the 2005 top system. There are several possible explanations for why we were not the top performer on English. Our splitting constraint for stems, which allows them to split into stems and chains of affixes, is suited for rich morphologies, and does not seem particularly well suited for English morphology. Also, as we have already observed for the development data, the allomorphic rules do not have as great an impact on improving induction in our system for English. Our rewrite-rules might also be improved by increasing their coverage over additional allomorphic phenomena, and by refining them so that they produce fewer spurious analyses.

Keshava and Pitler's RePorTs and Bernhard's procedure do a terrific job predicting affixes using transition probabilities between substrings [Keshava and Pitler 2006; Bernhard 2007], which are particularly reliable for English. Neither models allomorphy, and so both are complementary with our allomorphic-learning approach, suggesting that a better system may be possible by combining the most beneficial aspects of their approaches with our approach.

For Turkish, our system outperforms the baselines and the top systems for all three years.[9] The gap between our system and the 2008 top system is smaller because that system, ParaMor+Morfessor, combines both ParaMor [Monson et al. 2008] and Morfessor MAP segmentations as options for each word. On its own ParaMor performs second best for that year, scoring 46.5% *F*-score, and our system beats this by 9.0% F-score. The ParaMor approach is also complementary to our strategy, and attempts to learn and cluster partial paradigms, so here again it may be possible to combine aspects of both approaches and acheive an even better score.

The fact that there is a gap between our allomorphic-learning approach and the unsupervised approaches emphasizes the importance of handling allomorphy for a highly inflected, highly allomorphic language such as Turkish. A Turkish suffix, for instance, may undergo multiple spelling rules, and can have as many as twelve variant forms. Knowing that these variants all come from the same morpheme makes a difference.

## 6. CONCLUSION

Morphological analysis is important for a variety of NLP tasks. What kind of analyzer to use depends on many factors, including the application and available resources. The FST approach can produce accurate analyses, but it requires a morphological lexicon to work, which is difficult to come by and often quite expensive to produce. The advantage of MI techniques, on the other hand, is the fact that they are data driven, requiring just a word list in the simplest case. MI approaches are typically quite portable and can be retrained with ease. The problem is they tend to produce noisy segmentations, especially for a morphologically rich language such as Turkish.

[9]Like English, the dataset for 2008 is the same as the one for 2007.

In this work, we have proposed and evaluated a hybrid approach that uses a small amount of linguistic knowledge to augment a MI procedure, taking the idea of using rewrite rules to produce abstractions from the FST approach, and the idea to learn which abstractions are valid from the MI approach. Our experiments show that by adding even this small amount of knowledge, one can improve unsupervised segmentations significantly, particularly for complex languages such as Turkish. In MC 2007 test results, we get an improvement for the Turkish segmentation of nearly 22% against our baseline, and 2.5% against the state-of-the-art unsupervised approach.

There has been recent work on discovering allomorphic phenomena automatically [Dasgupta and Ng 2007; Demberg 2007]. It is hoped that our work can inform these approaches, if only by showing what variation is possible, and what is relevant to particular languages. For example, variation in inflectional suffixes, driven by vowel harmony and other phenomena, should be captured for a language like Turkish.

We are currently in the process of developing rule sets for additional languages, as test data in those languages becomes available. More languages will help determine how extensible this approach is, particularly whether it might be adapted to languages whose morphology is more fusional than that of English or Turkish (e.g., Russian).

The current system requires hand-coded rules. In the future we plan to learn those rules automatically from data. This might involve instantiating variable-featured rule-templates using seed corpora containing aligned morphs and morphemes. By collecting rules automatically we will bypass the need of a human expert and therefore we can apply the same techniques to many languages. It is also possible that rule sets collected in such a manner will be more complete.

## APPENDIX

### A. MODIFIED VITERBI ALGORITHM

During the Viterbi resegmentation step of our word-resegmentation procedure, we find the morpheme sequence $\hat{\mathbf{u}}$ and tag sequence $\hat{\mathbf{t}}$ that maximizes Equation (12) for a word $w$ using a modified version of the Viterbi algorithm.

$$P(\mathbf{u}|\mathbf{t})P(\mathbf{t}) \approx \left[\prod_{i=1}^{n} P(u_i|t_i)P(t_i|t_{i-1})\right] \tag{12}$$

Given a word, underlying morphemes are generated from all possible surface spans (i.e., substrings) by context-sensitive rewrite rules. The function $\Upsilon$ given in Section 3.4 produces underlying analyses, $u'$ and $u''$, for each span. Each analysis is indexed to the span's ending position and length.

Instead of a brute force search for the best underlying morpheme and tag sequence over underlying morphemes produced from all surface spans, we are able to utilize a Viterbi procedure to perform an efficient search for each word.

### A.1 Goal of Viterbi Procedure

The procedure's goal is to find the most likely underlying morpheme and hidden tag sequence for a word, given a model. The variable $\delta_{ktl}$ stores the probability of the best path leading to each possible tag and morpheme combination in the model ending at position $k$; $L_k$ is the length of a surface span $s(L_k)$, the right boundary of which lies at position $k$, and $C_k$ is its tag. The underlying mapping function $\Upsilon(s(L_k), R)$ maps to the morphemes $u(L_k) = u'(L_k)$ or $u''(L_k)$, as defined in Section 3.4.

$$\delta_{ktl} = \max_{C_0, u(L_0), \cdots, C_{(k-L_k)}, u(L_{(k-L_k)})} P(C_0, u(L_0), \cdots, C_{(k-L_k)}, u(L_{(k-L_k)}), C_k = t, L_k = l | model) \tag{13}$$

### A.2 Initialization

$$\delta_{k=0,t,l=0} = \pi_t \qquad 1 \leq t \leq N(tags) \tag{14}$$

The induction is initialized in Equation (14) by $\pi_t$, transition probabilities from the start state to the first tag, $t$.

### A.3 Induction

The induction proceeds by calculating $\delta_{ktl}$ according to Equation (15), for each position $k$ from left to right in the word, with lengths equal to $L_k = l$ and tags $t$ for morphemes ending at $k$. The position $k$ and length $L_k$ uniquely determine where the right boundary of the previous morpheme ($k'$) should be, distance $L_k$ from $k$: $k' = k - L_k$. The length of the previous morpheme is $l'$.

Whenever a morpheme is the first in the word (i.e., $l = k$), the position of the previous morpheme $k'$, as well as its length $l'$, are both equal to 0. In Equation (15), $\delta_{k',t',l'}$ stores the probability of the best paths up to the previous morpheme with tag $t'$, ending at position $k'$ with length $L_{k'} = l'$, $a_{[t',t]}$ is the transition probability of going from state $t'$ to $t$ and $b_{t,u(L_k)}$ is the emission probability for the best-scoring underlying morpheme $u(L_k)$ given $t$, selected from $u(L_k)'$ and $u(L_k)''$:

$$\delta_{ktl} = \max_{1 \leq t' \leq N(tags), 1 \leq l' \leq k'} \delta_{[k',t',l']} a_{[t',t]} b_{[t,u(L_k=l)]}; \quad 1 \leq t \leq N(tags), \; l \leq k \tag{15}$$

### A.4 Store Backtrace

The backtrace $\psi_{ktl}$ stores the argument which maximizes Equation (15), and is given in Equation (16). It stores three essential pointers for both tags and morphemes: the best previous tag $t' = \hat{t}'$, the best previous morpheme length $l' = \hat{l}'$, and a binary valued variable $\hat{z}$, which indicates if the best previous morpheme is of type $u'$ or $u''$.

$$\psi_{ktl} = \operatorname{argmax}_{1 \leq t' \leq N(tags), 1 \leq l' \leq k'} \delta_{[k',t',l']} a_{[t',t]} b_{[t,u(L_k=l])}; \; 1 \leq t \leq N(tags), \; l \leq k \tag{16}$$

### A.5 Termination and Path Readout (Backtracking)

When $k$ is equal to the last position of the word, we calculate the best final state and store the resulting underlying morpheme $u(L_{\hat{k}} = \hat{l})$ and tag $\hat{t}$. We use this to backtrack through the most probable tag and morpheme sequence. We read out the best tag sequence $\hat{\mathbf{t}}$ and morpheme sequence $\hat{\mathbf{u}}$ by iterating backward through the backtrace $\psi_{ktl}$ until we reach the beginning of the word:

(1) $\hat{t}', \hat{l}', \hat{z} = \psi_{\hat{k}\hat{t}\hat{l}} \quad \hat{k}' = \hat{k} - \hat{l}$,

(2) store $\hat{t}', u(L_{\hat{k}'} = \hat{l}')$ in $\hat{\mathbf{t}}, \hat{\mathbf{u}}$,
where $u(L_{\hat{k}'}) = u(L_{\hat{k}'})'$ if $\hat{z} = 0$,
$u(L_{\hat{k}'})''$ otherwise.

(3) set $\hat{t} = \hat{t}', \hat{k} = \hat{k}', \hat{l} = \hat{l}'$. goto (1).

## B. SPLIT STAGE DETAIL

Here we offer more detail on the tagging phase used in the splitting procedure used in Creutz and Lagus [2004], and extended in our approach. It is important for understanding parameter $b$, the one of the numerical parameters adjusted during tuning.

### B.1 Re-tagging Morphemes

To identify noise morphemes, we use the same categories from Creutz and Lagus [2004] to estimate a distribution $P(CAT|u_i)$:

$$CAT \in \{PRE, STM, SUF\} = \mathcal{T},$$

and one noise category, $CAT = NOI$. This distribution is estimated from the underlying segmentation output by the WR Stage. Once again the variable $u_i$ refers to the underlying morpheme with index $i$ in the underlying segmentation.

Then, categories are randomly assigned to morphemes $u_i$ in the underlying segmentation. The probability of each morphological category is proportional to a characteristic function $f$. Since stems are typically longer than affixes, their characteristic function is morpheme-length:

$$f_{STM}(u_i) = length(u_i) \tag{17}$$

Affixes have the characteristic property of attaching to a large variety of morphemes. For prefixes this occurs on the right, while for suffixes, on the left. The functions used to characterize this property are right- and left-perplexity, respectively:

$$f_{PRE}(u_i) = \Big[ \prod_{v \in \text{right-of}_{(u_i)}} P(v|u_i) \Big]^{\frac{1}{N_{u_i}}} \tag{18}$$

$$f_{SUF}(u_i) = \Big[ \prod_{v \in \text{left-of}_{(u_i)}} P(v|u_i) \Big]^{\frac{1}{N_{u_i}}} \tag{19}$$

These characteristic functions are mapped by sigmoid functions $y$ to values between 0 and 1:

$$y_{PRE}(u_i) = \left[1 + e^{-a(f_{PRE}(u_i)-b)}\right]^{-1} \tag{20}$$

$$y_{SUF}(u_i) = \left[1 + e^{-a(f_{SUF}(u_i)-b)}\right]^{-1} \tag{21}$$

$$y_{STM}(u_i) = \left[1 + e^{-c(f_{STM}(u_i)-d)}\right]^{-1} \tag{22}$$

Probability for each category is thus tied to the value of sigmoid parameters $a,b,c,$ and $d$, the most important of which is the cutoff $b$, which thresholds the probability of affixes and is adjusted during tuning.

The probability of the noise category is conversely related to the product of true category sigmoids; when the sigmoids are adjusted down, noise becomes more probable:

$$P(\text{NOI}|u_i) = \prod_{CAT \in \mathcal{T}} [1 - y_{CAT}(u_i)] \tag{23}$$

Then, the remainder of the probability mass is proportionally distributed to true categories according to the value of their sigmoid functions:

$$P(CAT \in \mathcal{T}|u_i) = \left[y_{CAT}(u_i) \Big/ [\sum_{CAT \in \mathcal{T}} y_{CAT}(u_i)]\right] \cdot \left[1 - P(\text{NOI}|u_i)\right] \tag{24}$$

Finally, all the morphemes in the segmentation are randomly re-tagged according to the distribution $P(CAT|u_i)$, and the new tags are sent to the next step.

## C. SAMPLE SEGMENTATIONS

In this section we provide samples of the segmentations produced by the procedure. The samples are provided in tables with columns for the results of preprocessing (Baseline), the first round of Word-Resegmentation Stage (WR Stage), and the Split Stage plus second round of WR Stage using two values of $b$ (SPL:b = 300 and SPL:b = 500). For the segmentations produced using rules, the first line is the surface segmentation and the second is the underlying segmentation.

### C.1 English Sample Segmentations

Table V shows a sample of segmentations for English when all rules are used. Manual segmentations have been provided for comparison. For each word in the table, the surface segmentation is listed above the underlying segmentation.

Table VI are the results when no rules are used. In this case, the surface segmentation and underlying segmentation are identical.

Table V. Sample of English Segmentations – With Rules

| Baseline | WR | SPL:$b$ = 300 | SPL:$b$ = 500 | Manual |
|---|---|---|---|---|
| happen ing s | happen ing s | happ e n ing s | happen ing s | happen ing s |
| | happen ing s | happ e n ing s | happen ing s | happen ing s |
| happen s | happen s | happ e n s | happen s | happen s |
| | happen s | happ e n s | happen s | happen s |
| happier | happier | happi er | happi er | happi er |
| | happier | happy er | happy er | happy er |
| happiest | happiest | happ i est | happiest | happi est |
| | happiest | happ y st | happiest | happy est |
| happily | happily | happi ly | happi ly | happi ly |
| | happily | happy ly | happy ly | happy ly |
| happiness | happiness | happi ness | happiness | happi ness |
| | happiness | happy ness | happiness | happy ness |

Table VI. Sample of English Segmentations – No Rules

| Baseline | WR | SPL:$b$ = 300 | SPL:$b$ = 500 | Manual |
|---|---|---|---|---|
| happen ing s | happen ing s | hap p en ing s | happen ing s | happen ing s |
| happen s | happen s | hap pen s | happen s | happen s |
| happier | happier | happi er | happi er | happi er |
| happiest | happiest | happi e st | happiest | happi est |
| happily | happily | happi ly | happi ly | happi ly |
| happiness | happiness | happi ness | happiness | happi ness |

Table VII. Sample of Turkish Segmentations – With Rules

| Baseline | WR | SPL:$b$ = 300 | SPL:$b$ = 500 | Manual |
|---|---|---|---|---|
| bastığı | bastığı | bas tığı | bas tığı | bas tığ ı |
| | bastIğI | bas DIğI | bas DIğI | bas DIK I |
| bastığı m | bastığı m | bas tığı m | bas tığı m | bas tığ ım |
| | bastIğI m | bas DIğI m | bas DIğI m | bas DIK Im |
| bastığı n da | bastığı nda | bas tığı nda | bas tığı nda | bas tığ ı nda |
| | bastIğI DE | bas DIğI DE | bas DIğI DE | bas DIK I DE |
| geldiği | geldiği | gel diği | gel diği | gel diğ i |
| | geldIğI | gel DIğI | gel DIğI | gel DIK I |
| geldiği m | geldiği m | gel diği m | gel diği m | gel diğ im |
| | geldIğI m | gel DIğI m | gel DIğI m | gel DIK Im |
| geldiği nde | geldiği nde | gel diği nde | gel diği nde | gel diğ i nde |
| | geldIğI DE | gel DIğI DE | gel DIğI DE | gel DIK I DE |

### C.2 Sample Segmentation for Turkish

Table VII shows a sample of segmentations for Turkish when all rules are used. Manual segmentations have been provided for comparison. For each word in the table, the surface segmentation is listed above the underlying segmentation.

Table VIII are the results when no rules are used. In this case, the surface segmentation and underlying segmentation are identical.

## D. REWRITE RULES FOR ENGLISH

The set of rules used for English is listed in Table IX. Rules were derived from Huddleston and Pullum [2001] in combination with the author's native intuitions about English spelling.

Table VIII. Sample of Turkish Segmentations – No Rules

| Baseline | WR | SPL:$b$ = 300 | SPL:$b$ = 500 | Manual |
|---|---|---|---|---|
| bastığı | bastığı | bastığı | bastığı | bas tığ ı |
| bastığı m | bastığı m | bastığı m | bastığı m | bas tığ ım |
| bastığı n da | bastığı nda | bastığı n da | bastığı nda | bas tığ ı nda |
| geldiği | geldiği | geldiği | geldiği | gel diğ i |
| geldiği m | geldiği m | geldiği m | geldiği m | gel diğ im |
| geldiği nde | geldiği nde | geldiği n de | geldiği nde | gel diğ inde |

### D.1 Alternation Between *-s* ∼ *-es*

Huddleston and Pullum indicate that this alternation occurs in the English plural, as well as in the 3rd-person singular present-tense morphemes. According to the authors the alternation can be described by analyzing the conditions in which *-es* occurs, while letting *-s* occur as the default alternate, that is, everywhere else.

There are several conditions in which to expect the *-es* allomorph to occur. The first is after surface vowels. For instance, *-es* occurs after a *y*-final base that alternates with a surface-form ending in the vowel *i*, as in *try* ∼ *tri+es*. Also, it occurs after the vowel, *o*, as in *potato* ∼ *potato+es.* We capture this with Rule 1a, which says that any surface-vowel-final morph should be followed by an *-es* alternate.

This is not the analysis suggested by Huddleston and Pullum, and for good reason: it overgenerates badly. It leads to incorrect predictions for inflections of vowel-final stems like *taxi* ∼ **taxi+es* and *papa* ∼ **papa+es*. However, this type of overgeneration is not a problem for our system because its goal is only to learn morphemes from analyzing surface morphs, and not to generate the surface morphs accurately.

In the analysis direction the rule works well, except in cases where *V-es* occurs, but not as the result of morpheme alternation. This does happen, and will result in spurious analyses, many of which will be ignored, but some of which will persist. For example, an analysis like *sees* ∼ **se+es* should have a low probability, as *se* does not occur as a morph. However, *bees* ∼ **be+es* will persist, as the proposed stem, *be*, collides with a high probability morph. If such collisions occur frequently enough, this rule should be revised to try to avoid them.

The second major condition where the *-es* alternate occurs is after a base ending in a sibilant, or sibilant-final character. A sibilant-final character is a character that is pronounced as ending in a sibilant. The character *x* is an example; it is pronounced *ks*, with the final-sibilant *s*. Some examples of this condition are *fox* ∼ *fox+es* and *dress* ∼ *dress+es*. This is captured by Rule 1b.

### D.2 Final *e*-deletion

According to Huddleston and Pullum, a base-final *e* is usually dropped before suffixes that begin with a vowel. This rule is true with a few exceptions in the case of the so-called *mute* or *silent e* that follows a consonant, as in *wake* ∼

Table IX. English Rules

| | | |
|---|---|---|
| | **Suffix Rules** | |
| 1a | $\emptyset \rightarrow e$ / $[+V]$ + _$s$ | *-s* ∼ *-es* alternation |
| 1b | $\emptyset \rightarrow e$ / $[+SIB]$ + _$s$ | ” |
| | **Stem Rules** | |
| 2 | $e \rightarrow \emptyset$ / $[+V][+C]$_ + $[+V]$ | *e*-deletion |
| 3 | $y \rightarrow i$ / $[+C]$ $+^{?}$ _ + $[+ANY]$ | *y*-replacement |
| 4a | $\emptyset \rightarrow \alpha[+STOP]$ / $\alpha[+STOP]$_ + $[+V]$ | consonant doubling |
| 4b | $\emptyset \rightarrow \alpha[+STOP]$ / $\alpha[+STOP]$_ + $[+GLI]$ | |

*wak+ing* or *pipe* ∼ *pip+ed.* We handle this variation with Rule 2, which says that a vowel-initial morph should trigger deletion of *e* (following a consonant) in its preceding morph.

Huddleston and Pullum mention that when *e* follows certain vowels it is also deleted, but that it is less common than *e*-deletion after consonants. There are examples like *sue* ∼ *su+ing* and *free* ∼ *fre+ed,* but not *free* ∼ **fre+ing*. Because *e*-deletion following vowels is more complex and less common, we chose not to write a rule for it.

### D.3 Final *y*-replacement

In English, a base-final *y* is replaced by the character *i* before another morph is attached, as in *pony* ∼ *poni+es* and *cleanly* ∼ *cleanli+ness*. Rule 3 captures this by replacing the final *y* following a consonant with *i* before morphs that begin with any character. As Huddleston and Pullum point out, the *y*-replacement does not take place if *y* forms a diphthong with its preceding vowel as it does in *stay* and *buy*. Because the *y* undergoing variation could be a morpheme attached to another stem, as in *paper+y* ∼ *paper+i+ness*, the rule allows an optional morpheme boundary between *y* and its preceding consonant.

### D.4 Consonant Doubling

This phenomenon involves doubling a base-final consonant before attaching a vowel- or glide-initial suffix, as in *bat* ∼ *batt+ing* or *cat* ∼ *catt+y.* According to Huddleston and Pullum, the alternation does not occur with multi-character consonants (e.g., *dress* ∼ **dresss+es*), consonants preceded by multi-character vowel symbols (e.g., *seat* ∼ **seatt+ed*), consonants in a non-stressed syllable (e.g., *offer* ∼ **offerr+ing*), nor does it occur at all for the set of consonants {*h, w, y,* and *x*}.

The rules written to handle this phenomenon, 4a and 4b in Table IX are too specific, only indicating the doubling of stops. The doubling phenomenon also affects the other consonants (except the set mentioned explicitly above). The rules are incomplete because they were written at an early stage of the project; they have not been updated so as to maintain experimental consistency.

## E. REWRITE RULES FOR TURKISH

The set of rules used for Turkish is listed in Table X. The Turkish rules were primarily derived from Göksel and Kerslake [2005] with additional input from Lewis [1967].

### E.1 Vowel Harmony

According to Göksel and Kerslake [2005], vowel harmony is a process that primarily affects how suffix allomorphs are realized. Here, different vowels will obtain depending on which vowel precedes them, that is, the last vowel in the preceding base morph. There are two classes of vowel harmony in Turkish, *A*-type and *I*-type harmony. Suffixes in Turkish exhibit either one or the other, but not both. The use of a capital letter as a character implies that the "default variant" is underspecified for one or more features; it always takes on some feature-values of characters in its immediate context.

In *A*-type harmony the underlying vowel, before it undergoes change, is represented by the character *A*. It is unrounded (lips are not round) and non-high (tongue is low in the mouth), but is underspecified for frontness (how far forward the tongue is in the mouth).

Its frontness depends on the frontness of the vowel preceding it. This is captured by Rule 1a using the variable feature $\alpha F$, where $\alpha$ is instantiated to the frontness value of the preceding vowel; this determines whether it is realized as the nonfront (back) variant *a* or the front variant *e*. For example, the dative case suffix *-A* is realized as front *-e* in *ev* ∼ *ev+e*, (*house* ∼ *to the house*) or as back *-a* in *bulut* ∼ *bulut+a* (*cloud* ∼ *to the cloud*).

In *I*-type harmony the underlying vowel is represented by the character *I*. It is high, while underspecified for both roundness and frontness. Similar to what was just described, *I* takes on roundness and frontness features of the vowel preceding it. This is captured in Rule 1b using variable features where variables are instantiated by features of the preceding vowel. Because each feature is binary valued, there are four possibilities for a vowel undergoing *I*-type harmony.

As an example we show two of four harmony possibilities for first-person possessive suffix *-I*m: it can be realized as front and rounded *-üm* after a front and rounded vowel, e.g. *gül* ∼ *gül+üm* (*rose* ∼ *my rose*), or as non-front and unrounded *-ım* after a non-front and non-round vowel, for example, *at* ∼ *at+ım* (*horse* ∼ *my horse*).

### E.2 Suffix-Initial Consonant Voicing Alternation

Suffix-initial consonant voicing alternation affects suffixes that start with ⟨voiced, voiceless⟩ consonant pairs like ⟨*c*, *ç*⟩, represented by *C*, or ⟨*d*, *t*⟩, represented by *D*. The pattern driving this alternation is simple: the voiced variant obtains when the preceding character is voiced, and the voiceless variant when the preceding character is voiceless. This pattern is encoded by Rule 2 in Table X.

Table X. Turkish Rules

| | | |
|---|---|---|
| | **Suffix Rules** | |
| 1a | $\underset{[+V,-H]}{A} \rightarrow [\alpha F] \,/\, [\alpha F][+C]^* + [+C]^*\_$ | V harmony of *A* |
| 1b | $\underset{[+V,+H]}{I} \rightarrow [\alpha F, \beta R] \,/\, [\alpha F, \beta R][+C]^* + [+C]^*\_$ | V harmony of *I* |
| 2 | $[+C] \rightarrow [\alpha VOI] \,/\, [\alpha VOI] + \_[+V]$ | C alternation |
| 3a | $\emptyset \rightarrow y \,/\, [+V] + \_[+ANY]$ | suffix-initial C |
| 3b | $\emptyset \rightarrow n \,/\, [+V, +H] + \_[+V]$ | insertion |
| 3c | $\emptyset \rightarrow n \,/\, [+V, +H] + \_d$ | ” |
| 3d | $\emptyset \rightarrow s \,/\, [+V] + \_[+V, +H]$ | ” |
| | **Stem Rules** | |
| 4a | $[+C] \rightarrow [-VOI] \,/\, [+V]\_\#$ | stem-final C |
| 4b | $[+C] \rightarrow [-VOI] \,/\, [+V]\_ + [+C]$ | alternation |
| 4c | $[+C] \rightarrow [+VOI] \,/\, [+V]\_ + [+V]$ | ” |

For example, the agentive suffix *-CI* will be realized as the voiceless-initial *çi* after a voiceless character like *k* in *çömlek* ∼ *çömlek+çi* (*pot* ∼ *potter*). However, it will be voiced-initial *-ci* after a voiced character like *z*, as in *deniz* ∼ *deniz+ci* (*sea* ∼ *mariner*).

### E.3 Suffix-Initial Consonant Insertion

This phenomenon, which is also known as consonant deletion, involves consonants that appear or fail to appear as the initial character of a suffix, depending on the preceding character and grammatical properties of the preceding suffix. This phenomenon involves the consonants *y, n* and *s*.

Several examples include the accusative suffix *-(y)I*, and possessive suffix *-(s)I*, which share the homographic allomorph *-I* in the context immediately following a consonant, but are distinct after a vowel: *-yI* and *-sI* respectively.

Because of ambiguous allomorphs like *-I*, an analysis that attempts to reverse a *deletion* by inserting a consonant is problematic: given a word like *at-ı*, one cannot know whether it came from *at+yI* (*horse +ACC*), or *at+sI* (*horse +GEN*) without referring to word-external context. Therefore, both possibilities would need to be generated by an analyzer.

Since we currently generate only one rule-derived form with rule-proposed alterations,[10] our rules (see 3a-d in Table X) represent this type of variation as consonant-*insertion* instead, which is possible to reverse deterministically. The main drawback of this style of approach is that it creates ambiguous morphemes like *-I*, where there ought to be distinct morphemes: *-(y)I* and *-(s)I*.

On the Morpho Challenge 2007 evaluation metric, which was discussed in Section 4.4, positing ambiguous morphemes will likely result in lower precision

[10]In the future, we plan to alter the procedure to allow multiple rule-derived analyses (with rule-proposed changes) to compete, enabling the procedure to choose the more frequent analysis as the morpheme whenever it encounters an ambiguous allomorph like *-I*. This may be useful in cases where morphemes sharing an allomorph have vastly different overall frequencies; in such a case it may improve results to attribute the unambiguous allomorphs to distinct morphemes and the ambiguous allomorph to the more frequent (distinct) morpheme.

scores. However, since a greater number of valid (even if ambiguously labeled) morphemes are often discovered this way, the recall score is likely to improve.

### E.4 Stem-Final Consonant Voicing Alternation

This is a change induced in stems by adding suffixes. It involves a set of ⟨voiced, voiceless⟩ consonant pairs, similar to the set that undergoes voicing alternation in suffixes. Here, the voiced and voiceless variants alternate at the end of a stem. The alternation depends on phonological properties of the suffix that attaches to it. As in suffix-initial voicing alternation, the pattern behind the variation is simple. Word-finally, as well as before a consonant-initial suffix, the final consonant of a word will be voiceless. Before a vowel-initial suffix, it will be voiced. This is captured by Rules 4a-c.

For example, attaching the vowel-initial first-person possessive suffix *-Im* to *kanaD* results in *kanad+ım* (*my wing*), where the final character is the voiced *d*. As a word on its own, uninflected, it is rendered *kanat*. Upon attaching a consonant-initial suffix like locative *-DE*, it is also rendered with a final *t*, as in *kanat+ta* (*on the wing*).

There is a class of exceptional stems that do not undergo this type of alternation. These are stems that end in a voiceless final-consonant that always remains voiceless, even in the presence of a vowel-initial suffix. An example would be *sanat* ∼ *sanat+ım* (*art* ∼ *my art*). Such exceptions are not a problem for Rules 4a-c, however. Upon analyzing *sanat+ım*, the conditions would not be met for any of these rules to fire. Therefore, the only morpheme proposed for the stem would be the surface form, *sanat*, which would be correct.

## REFERENCES


BERNHARD, D. 2007. Simple morpheme labeling in unsupervised morpheme analysis. In *Working Notes for the Workshop on Cross-Language Evaluation Forum (CLEF'07)*.

CHENG, C. K. AND SEE, S. L. 2006. The revised wordframe model for the Filipino language. *J. Res. Sci. Comput. Eng.*

CHOMSKY, N. AND HALLE, M. 1968. *The sound pattern of English*. Harper & Row: New York.

CREUTZ, M. AND LAGUS, K. 2004. Induction of a simple morphology for highly inflecting languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON'04)*. 43–51.

CREUTZ, M. 2006. Induction of the morphology of natural language: Unsupervised morpheme segmentation with application to automatic speech recognition. Ph.D. thesis, Helsinki University of Technology.

CREUTZ, M. AND LAGUS, K. 2002. Unsupervised discovery of morphemes. In *Proceedings of the Workshop on Morphological and Phonological Learning of the Association for Computational Linguistics (ACL'02)*. 21–30.

CREUTZ, M. AND LAGUS, K. 2004. Induction of a simple morphology for highly inflecting languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology (SIGPHON)*. 43–51.

CREUTZ, M. AND LAGUS, K. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05)*. 106–113.

CREUTZ, M. AND LAGUS, K. 2006. Morfessor in the morpho challenge. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes (PASCAL'06)*.

DASGUPTA, S. AND NG, V. 2007. High performance, language-independent morphological segmentation. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (HLT-NAACL'07)*.

DE MARCKEN, C. G. 1996. Unsupervised language acquisition. Ph.D. thesis. Massachusetts Institute of Technology.

DEJEAN, H. 1998. Morphemes as necessary concept for structures discovery from untagged corpora. In *Proceedings of the Workshop on Paradigms and Grounding in Natural Language Learning (CoNLL'98)*. 295–299.

DELIGNE, S. AND BIMBOT, F. 1997. Inference of variable-length linguistic and acoustic units by multigrams. *Speech Comm. 23*, 223–241.

DEMBERG, V. 2007. A language-independent unsupervised model for morphological segmentation. In *Proceedings of the Association for Computational Learning (ACL'07)*.

GÖKSEL, A. AND KERSLAKE, C. 2005. Turkish: A comprehensive grammar. Routledge: London.

GOLDSMITH, J. 2001. Unsupervised learning of the morphology of a natural language. *Comput. Linguist. 27*, 2, 153–198.

HARRIS, Z. S. 1951. *Methods in structural linguistics*. University of Chicago Press.

HUDDLESTON, R. AND PULLUM, G. K. 2001. *The Cambridge Grammar of the English Language*. Cambridge University Press.

KAPLAN, R. M. AND KAY, M. 1981. Phonological rules and finite-state transducers. In *Linguistic Society of America Meeting Handbook*. New York.

KAPLAN, R. M. AND KAY, M. 1994. Regular models of phonological rule systems. *Comput. Linguist. 20*, 3.

KARTTUNEN, L. AND BEESLEY, K. R. 2001. A short history of two-level morphology. In *Proceedings of the European Summer School in Logic, Language, and Information (ESSLLI'01)*.

KARTTUNEN, L. AND BEESLEY, K. R. 2005. Twenty-five years of finite-state morphology. In *Inquiries into Words, Constraints, and Contexts: Festschrift for Kimmo Kosenniemi on his 60th Birthday*. CSLI Publications.

KESHAVA, S. AND PITLER, E. 2006. A simpler, intuitive approach to morpheme induction. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes (PASCAL'06)*.

KOSKENNIEMI, K. 1983. *Two-Level Morphology: A General Computational Model for Word-Form Recognition and Production*. University of Helsinki, Helsinki, Finland.

KOSKENNIEMI, K. 2003. Computational morphology. In *International Encyclopedia of Linguistics*, E-Reference Ed., W. J. Frawley Ed., Oxford University Press.

KURIMO, M., CREUTZ, M., AND VARJOKALLIO, M. 2007. Unsupervised morpheme analysis evaluation by a comparison to a linguistic gold standard – Morpho Challenge 2007. In *Working Notes for the CLEF Workshop*.

KURIMO, M., CREUTZ, M., VARJOKALLIO, M., ARISOY, E., AND SARAÇLAR, M. 2006. Unsupervised segmentation of words into morphemes – Morpho Challenge 2005, an introduction and evaluation report. In *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*.

LEWIS, G. 1967. *Turkish Grammar*. Oxford University Press.

MONSON, C., CARBONELL, J., LAVIE, A., AND LEVIN, L. 2008. Paramor and morpho challenge 2008. In *Working Notes for the CLEF Workshop*.

PENG, F. AND SCHUURMANS, D. 2001. A hierarchical EM approach to word segmentation. In *Proceedings of the 4th International Conference on Intel. Data Analysis (IDA)*. 238–247.

SCHONE, P. AND JURAFSKY, D. 2001. Knowledge-free induction of inflectional morphologies. In *Proceedings of the NAACL'01*.

SNOVER, M. G. AND BRENT, M. R. 2001. A bayesian model for morpheme and paradigm identification. In *Proceedings of the ACL'01*. 482–490.

TROST, H. 2003. Morphology. In *The Oxford Handbook of Computational Linguistics*. Oxford University Press, 25–47.

WICENTOWSKI, R. 2002. Modeling and learning multilingual inflectional morphology in a minimally supervised framework. Ph.D. thesis, Johns Hopkins University.

YAROWSKY, D. AND WICENTOWSKI, R. 2000. Minimally supervised morphological analysis by multimodal alignment. In *Proceedings of the Association for Computational Linguistics (ACL'00)*. 207–216.

Received March 2009; revised July 2009; accepted August 2009