From Segmentation to Analyses: A Probabilistic Model for Unsupervised Morphology Induction

From Segmentation to Analyses: A Probabilistic Model for Unsupervised Morphology Induction

Toms Bergmanis
School of Informatics
University of Edinburgh
T.Bergmanis@sms.ed.ac.uk

Sharon Goldwater
School of Informatics
University of Edinburgh
sgwater@inf.ed.ac.uk

Abstract

A major motivation for unsupervised morphological analysis is to reduce the sparse data problem in under-resourced languages. Most previous work focuses on segmenting surface forms into their constituent morphs (e.g., *taking: tak +ing*), but surface form segmentation does not solve the sparse data problem as the analyses of *take* and *taking* are not connected to each other. We extend the MorphoChains system (Narasimhan et al., 2015) to provide morphological analyses that can abstract over spelling differences in functionally similar morphs. These analyses are not required to use all the orthographic material of a word (*stopping: stop +ing*), nor are they limited to only that material (*acidified: acid +ify +ed*). On average across six typologically varied languages our system has a similar or better F-score on EMMA (a measure of underlying morpheme accuracy) than three strong baselines; moreover, the total number of distinct morphemes identified by our system is on average 12.8% lower than for Morfessor (Virpioja et al., 2013), a state-of-the-art surface segmentation system.

1 Introduction

Most previous work on unsupervised morphological analysis has focused on the problem of **segmentation**: segmenting surface forms into their constituent morphs (Goldsmith, 2001; Creutz and Lagus, 2007; Poon et al., 2009; Lee et al., 2011; Virpioja et al., 2013; Sirts and Goldwater, 2013). However, the focus on surface segmentation is largely due to ease of model definition and implementation rather than linguistic correctness. Even in languages with primarily concatenative morphology, spelling (or phonological) changes often occur at morpheme boundaries, so that a single morpheme may have multiple surface forms. For example, the past tense in English may surface as *-ed* (*walked*), *-d* (*baked*), *-ted* (*emitted*), *-ped* (*skipped*), etc.

A major motivation for unsupervised morphological analysis is to reduce the sparse data problem in under-resourced languages. While surface segmentation can help, the example above illustrates its limitations: for more effective parameter sharing, a system should recognize that *-ed*, *-d*, *-ted*, and *-ped* share the same linguistic function. The importance of identifying underlying morphemes rather than surface morphs is widely recognized, for example by the MorphoChallenge organizers, who in later years provided datasets and evaluation measures to encourage this deeper level of analysis (Kurimo et al., 2010). Nevertheless, only a few systems have attempted this task (Goldwater and Johnson, 2004; Naradowsky and Goldwater, 2009), and as far as we know, only one, the rule-based MORSEL (Lignos et al., 2009; Lignos, 2010), has come close to the level of performance achieved by segmentation systems such as Morfessor (Virpioja et al., 2013).

We present a system that adapts the unsupervised MorphoChains segmentation system (Narasimhan et al., 2015) to provide morphological analyses that aim to abstract over spelling differences in functionally similar morphemes. Like MorphoChains, our system uses an unsupervised log-linear model whose parameters are learned using contrastive estimation (Smith and Eisner, 2005). The original MorphoChains system learns to identify child-parent pairs of morphologically related words, where the child (e.g., *stopping*) is formed from the parent (*stop*) by adding an affix and possibly a spelling transformation (both represented as features in the model). However, these spelling transformations are never used to output underlying morphemes, instead the system just returns a segmentation by post-processing the inferred child-parent pairs.

We extend the MorphoChains system in sev-

eral ways: first, we use the spelling transformation features to output underlying morphemes for each word rather than a segmentation; second, we broaden the types of morphological changes that can be identified to include compounds; and third, we modify the set of features used in the log-linear model to improve the overall performance. We evaluate using EMMA (Spiegler and Monson, 2010), a measure that focuses on the identity rather than the spelling of morphemes. On average across six typologically varied languages (English, German, Turkish, Finnish, Estonian, Arabic), our system outperforms both the original MorphoChains system and the MORSEL system, and performs similarly to the surface segmentation system Morfessor. These results are (to our knowledge) the best to date from a system for identifying underlying morphemes; moreover, the total number of distinct morphemes identified by our system is on average 12.8% lower than for Morfessor, suggesting that it does a better job of abstracting over surface spellings and inducing a compact representation of the data.

2 Morphological Chains and Analyses

We base our work on the MorphoChains segmentation system (Narasimhan et al., 2015),[1] which defines a **morphological chain** as a sequence of **child-parent pairs**. Each pair consists of two morphologically related words where the child must be longer than the parent. To analyse a word we want to find the best parent for that word; we do so recursively until we conclude that the **stop condition** is met (i.e. a word doesn't have a morphological parent). The word *standardizes*, for example, produces the following chain:

$$\textit{standardizes} \rightarrow \textit{standardize} \rightarrow \textit{standard}$$

which consists of the child-parent pairs (*standardizes*, *standardize*), (*standardize*, *standard*) and (*standard*, STOP). Each child-parent pair is annotated with a **type** indicating the kind of **transformation** that relates the child-parent pair. The set of transformations defined by MorphoChains is: **suffixation** as in (*dogs, dog*), **prefixation** as in (*undone*, *done*), **deletion** as in (*baked*, *bake*)[2], **repetition** as in (*stopped*, *stop*), and **modification** as in (*worried*, *worry*). We add a sixth type, **compounding** as in (*darkroom*, *room*). The delete, repeat, and modify types all assume the change occurs to the final character of the stem, while compounding can simply concatenate the two stems, or can introduce an extra character (as in *higher-rate* or German *schilddruese +n+ krebs* ('thyroid cancer').

Word	MorphoChains	Our Model
stopping	stopp +ing	stop +ing
doubled	doubl +d	double +ed
acidified	acid +ifi +ed	acid +ify +ed

Table 1: Examples outputs of two models.

Any word (*undone*) has many possible parents, some linguistically plausible (*undo*, *done*) and others not (*und*, *ndone*). Our system, like MorphoChains, learns a log-linear model to discriminate plausible from implausible parents amongst the complete **parent candidate set** for each word. The candidate set is generated by taking all possible splits of the word and applying all possible transformation types. For example, some parent candidates for the word *dogs* include (*dog*, suffix), (*do*, suffix), (*gs*, prefix), (*doga*, delete), (*dogb*, delete), (*doe*, delete), and (*doe*, modify). The last two imply analyses of *doe +gs* and *doe +s*, respectively.

The examples above indicate how analyses can be induced recursively by tracking the transformation type and orthographic change associated with each parent-child pair. However, the original MorphoChains algorithm did not do so, instead it only used the transformation types to predict morph boundaries. Table 1 contrasts the word segmentation into morphs produced by the original MorphoChains model and the morpheme analysis produced by our model. Table 2 provides additional examples of our recursive analysis process.

2.1 Model

We predict child-parent pairs using a log-linear model, following Narasimhan et al. (2015). The model consists of a set of features represented by a feature vector $\phi : \mathcal{W} \times \mathcal{Z} \rightarrow \mathbb{R}^d$, where $\mathcal{W}$ is a set of words and $\mathcal{Z}$ is the set of $(parent, type)$ pairs for words in $\mathcal{W}$. The model defines the conditional probability of a particular $(parent, type)$ pair $z \in \mathcal{Z}$ given word $w \in \mathcal{W}$ as:

$$P(z|w) = \frac{e^{\theta \cdot \phi(w,z)}}{\sum_{z' \in C(w)} e^{\theta \cdot \phi(w,z')}}, z \in C(w) \quad (1)$$

[1] We modified the implementation available at `https://github.com/karthikncode/MorphoChain`.

[2] The system could in principle learn that *bake* is the parent of *baked* with type *suffix*, which would imply the analysis *bake +d*. However, we hope it learns instead the type *delete*, which implies the (correct) analysis *bake +ed*. Similar alternative analyses are possible for the other example types shown.

Step	Word	Child-Parent Pair	Type	Change	Analysis
1	**acidified**	(*acidified*, *acidify*)	modify	add *+ed*	*+ed*
2		(*acidify*, *acid*)	suffix	add *+ify*	*+ify +ed*
3		(*acid*, NONE)	stop	add *acid*	**acid +ify +ed**
1	**doubled**	(*doubled*, *double*)	delete	add *+ed*	*+ed*
2		(*double*, NONE)	stop	add *double*	**double +ed**
1	**higher-rate**	(*higher-rate*, *rate*)	compound	keep *higher*	
2		(*rate*, NONE)	stop	add *rate*	*rate*
3		(*higher*, *high*)	suffix	add *+er*	*+er rate*
4		(*high*, NONE)	stop	add *high*	**high +er rate**

Table 2: Examples of step-by-step derivations of morphological analysis of English words.

where $C(w) \subset \mathcal{Z}$ denotes the set of parent candidates for w and θ is a weight vector.

Our goal is to learn the feature weights in an unsupervised fashion. Following Narasimhan et al. (2015), we do so using Contrastive Estimation (CE) (Smith and Eisner, 2005). In CE every training example $w \in \mathcal{W}$ serves as both a positive example and a set of implied negative examples—strings that are similar to w but don't occur in the corpus. The negative examples are the source of the probability mass allocated to the positive examples. The word w and its negative examples constitute the **neighbourhood** $\mathcal{N}(w)$ of w.

Given the list of words $\mathcal{W}$ and their neighbourhoods, the CE likelihood is defined as:

$$L_{CE}(\theta, \mathcal{W}) = \prod_{w^* \in W} \frac{\sum_{z \in C(w^*)} e^{\theta \cdot \phi(w^*, z)}}{\sum_{w \in N(w^*)} \sum_{z \in C(w)} e^{\theta \cdot \phi(w, z)}}. \quad (2)$$

We use the same neighbourhood functions as Narasimhan et al. (2015). Specifically, for each word w in the corpus $\mathcal{W}$, we create neighbours in two ways: by swapping two adjacent characters of w (*walking→walkign*) and by swapping two pairs of adjacent characters, where one pair is at the beginning of the word, and the other at the end of the word (*walking→awlkign*).

We use LBFGS-B (Zhu et al., 1997) to optimize the regularized log-likelihood of the model:

$$LL_{CE}(\theta, \mathcal{W}) = \sum_{w^* \in W} \left[\log \sum_{z \in C(w^*)} e^{\theta \cdot \phi(w^*, z)} - \log \sum_{w \in N(w^*)} \sum_{z \in C(w)} e^{\theta \cdot \phi(w, z)} \right] - \lambda \|\theta\|^2 \quad (3)$$

2.2 Features

MorphoChains used a rich set of features from which we have kept some, discarded others and added new ones to improve overall performance. This section describes our set of features, with examples shown in Table 3.

Presence in Training Data We want features that signal which parents are valid words. Narasimhan et al. (2015) used each word's log frequency. However the majority of words in the training data (word frequency lists) occur only once, which makes their frequency information unreliable.[3] Instead, we use an out-of-vocabulary feature (OOV) for parents that don't occur in the training data.

Semantic Similarity Morphologically related words exhibit semantic similarity among their word embeddings (Schone and Jurafsky, 2000; Baroni et al., 2002). Semantic similarity was an important feature in MorphoChains: Narasimhan et al. (2015) concluded that up to 25 percent of their model's precision was due to the semantic similarity feature. We use the same feature here (COS). For a child-parent pair (w_A, w_B) with word embeddings v_{w_A} and v_{w_B} respectively we compute semantic similarity as:

$$cos(w_A, w_B) = \frac{v_{w_A} \cdot v_{w_B}}{\|v_{w_A}\| \|v_{w_B}\|} \quad (4)$$

Affixes Candidate pairs where the child contains a frequently occurring affix are more likely to be correct. To identify possible affixes to use as features, Narasimhan et al. (2015) counted the number of words that end (or start) with each substring

[3] The prevalence of singleton word types in the MorphoChallange 2010 training data for English, German, Turkish ranges from 50.73 to 58.76 %.

No	Child	Parent Candid.	Active Features
1	*dog*	(*og*, prefix)	OOV, PREF=*d*
2	*decided*	(*decide*,delete)	DELETED=*e*, SUF=*ed*, SUFLIST
3	*decided*	(*decids*, delete)	OOV, DELETED=*s*, SUF=*ed*, SUFLIST
4	*stopped*	(*stop*, repeat)	REPEATED=*p*, RENV2=*op*, RENV1=*o*, SUF=*ed*, SUFLIST
5	*worried*	(*worry*, suffix)	MODIFIED=*y-i*, SUF=*ed*, SUFLIST
6	*higher-rate*	(*rate*, compound)	HEAD=*rate*, MODIFIER=*higher*, CONNECTOR=-, COMPOUND
7	*ratepayer*	(*payer*, compound)	HEAD=*payer*, MODIFIER=*rate*, COMPOUND
8	*decided*	(*deci*, compound)	OOV, HEAD=*deci*, MODIFIER=*ded*
9	*high*	(-, stop)	STOPLEN=*4*, STOPCOS=*0.2*
10	*decided*	(-, stop)	STOPLEN=*7*, STOPCOS=*0.5*
11	*unstable*	(*able*, prefix)	PREF=*unst*
12	*unable*	(*able*, prefix)	PREF=*un*, PREFLIST

Table 3: Examples illustrating which of the binary features in the model are active for various potential child-parent pairs. Not shown here is the real-valued semantic similarity feature COS, used in all examples except 9 and 10, where it is replaced by the binary feature STOPCOS=y, for y in increments of 0.1.

Prefixes
al, ar, ba, be, bo, ca, car, co, de, dis, en, ha, ho, in, inter, la, le, li, lo, ma, mar, mc, mi, mis, mo, out, over, pa, po, pre, pro, ra, re, ro, se, ta, to, un, under, up
Suffixes
a, age, al, an, ar, ary, as, ation, b, ble, ch, e, ed, el, en, er, ers, es, est, et, ful, i, ia, ic, ie, ies, in, ing, ings, is, ism, ist, ists, land, le, led, les, less, ley, ling, ly, m, man, ment, ments, ner, ness, o, or, p, s, se, son, t, ted, ter, ters, th, ting, ton, ts, y

Table 4: The likely English affixes found by using Letter Successor Entropy.

and selected the most frequent ones. However, all words that end with *ing* also end with *ng* and *g*, which means that they also become affix candidates. Furthermore, there are more words that end with *ng* or *g* than with *ing*, therefore valid affixes might be excluded from the list because of their more frequent substrings.

We therefore modify the affix features in two ways. First, we identify a more precise set of likely affixes using Letter Successor Entropy (LSE) values (Hafer and Weiss, 1974), which are typically high at morph boundaries. LSE is computed at each point in the word as the entropy of the distribution over the next character given the word prefix so far. When selecting likely affixes, we use an LSE threshold value of 3.0 as suggested by Hafer and Weiss (1974), and we require that the affix has appeared in at least 50 types with a corpus frequency of at least 100. We then define two features (PREFLIST, SUFLIST), which are active if the proposed prefix or suffix for a parent-child pair is in the set of likely prefixes or suffixes. Table 4 shows the list of likely English affixes found by using LSE (62 suffixes and 42 prefixes). For German and Turkish, our other two development languages (see §3), the lists contain 498 suffixes/183 prefixes and 181 suffixes/35 prefixes, respectively.[4]

In addition, we use a much larger set of affix features, PREF=x and SUF=x, where x is instantiated with all possible word prefixes (suffixes) for which both w and xw (wx) are words in the training data.

Transformations To help distinguish between probable and improbable transformations, we introduce transformation-specific features. For deletion we use the deleted letter (DELETED). For repetition we use the repeated letter and its preceding 2- and 1-character contexts (REPEATED, RENV2 RENV1). For modification we use the combination of the involved letters (MODIFIED). Finally, for compounding we use the headword (i.e. the parent of the compound), the modifier and the connector, if such exists (HEAD, MODIFIER, CONNECTOR). Since these compound features can be very sparse, we also add a single COMPOUND feature, which is active when both parts of the compound are present in the training data.

Stop Condition To identify words with no parents we use two types of binary features suggested

[4] In MorphoChains, the number of affixes was set manually for each language tested: 300 for English, 500 for Turkish, and 100 for Arabic.

by Narasimhan et al. (2015). STOPCOS=y is the maximum cosine similarity between the word and any of its parent candidates (using bins of size 0.1), and STOPLEN=x is instantiated for all possible word lengths x in the training data. For illustration, if we are considering whether *decided* is a word with no parents (Table 3 Example 10), the binary features STOPLEN=7 and STOPCOS=0.5 become active.

We discard the starting and ending character unigram and bigram features used by MorphoChains, because of the large number[5] and the sparsity of these features.

2.3 Data Selection

Most unsupervised morphology learners are sensitive to the coverage and the quality of training data. In a large corpus, however, many word types occur only once because of the Zipfian distribution of word types. Low-frequency types can be either rare but valid words or they can be foreign words, typos, non-words, etc. This makes learning from low-frequency words unreliable, but discarding them dramatically reduces the size of the training data (including many valid words).

To seek balance between the quality and the coverage of the training data we try to identify which low-frequency words are likely to provide useful statistical support for our model, so we can include those in the training data and discard the other low-frequency words. First, we set a frequency-based **pruning threshold** (PT) at the frequency for which at least 50% of the words above this frequency have a word embedding (see §3); next we set a **learning threshold** (LT) at the median frequency of the words with frequencies above PT; finally we adopt the algorithm by Neuvel and Fulop (2002) to decide which words with frequencies below PT can be useful to analyse the words with frequencies above LT. We filter out any remaining words with frequencies below PT.

The outline of the adapted version of the algorithm by Neuvel and Fulop (2002) is:

1) For every word pair in the top 20k most frequent words in training data:

1.1) We find the pair's **orthographic similarities** as the longest common subsequence: ***rece****ive*⇔***rece****ption*.

1.2) We find the pair's **orthographic differences** with respect to their orthographic similarities: *rec**eive***⇔*rec**eption***.

2) For all word-pairs with the same orthographic differences we merge their similarities and differences into **Word Formation Strategies** (WFS): so ***rece****ive*⇔***rece****ption*, ***conce****ive*⇔***conce****ption*, ***dece****ive*⇔***dece****ption* give ***##ceive**⇔***##ception***, where * and # stand for the optional and mandatory character wild cards respectively.

3) We discard those WFS that that are suggested by less than 10 word pairs;

4) For each WFS and for each word with a frequency below PT:

4.1) if a word w matches either of the sides of a WFS and the other side of a WFS predicts a word w' with a frequency above the LT, we keep w in the training data, otherwise we discard it.

For more detailed description of the algorithm see Neuvel and Fulop (2002).

3 Experiments

Data We conduct experiments on six languages: Arabic, English, Estonian, Finnish, German and Turkish. For the word embeddings required by our system and the MorphoChains baseline, we used `word2vec` (Mikolov et al., 2013) to train a Continuous Bag of Words model on a sub-sample of the Common Crawl (CC) corpus[6] for each language (Table 5 lists corpus sizes). We trained 100-dimensional embeddings for all words occurring at least 25 times, using 20 iterations and default parameters otherwise.

For all languages except Estonian, we train and evaluate all systems on the data from the Morpho Challenge 2010 competition.[7] The training data consists of word lists with word frequencies. The official test sets are not public, but a small labelled training and development set is provided for each language in addition to the large unannotated word list, since the challenge included semi-supervised systems. Thus, for experiments on Arabic, English, Finnish, German and Turkish we evaluated on the annotated training and development gold standard analyses form the Morpho Challenge 2009/2010 competition data sets. The gold standard labels include part of speech tags and functional labels for inflectional morphemes, with multiple analyses given for words with part of speech ambiguity or

[5] For English there can be 676 different letter bigrams of which 99% occur at least once at the beginning of some word in the word frequency list.

[6] Common Crawl `http://commoncrawl.org`

[7] Morpho Challenge 2010: `http://research.ics.aalto.fi/events/morphochallenge2010`

Lang	Train (#types)	Test (#cases)	Embeddings (#tokens)
ARA	MC-10 (19K)	MC-09 (690)	CC (461M)
ENG	MC-10 (878K)	MC-10 (1569)	CC (1.78B)
EST	CC (470K)	S&G2013 (1500)	CC (329M)
FIN	MC-10 (2.9M)	MC-10 (1835)	CC (1.18B)
GER	MC-10 (2.3M)	MC-10 (1779)	CC (856M)
TUR	MC-10 (617K)	MC-10 (1760)	CC (1.21B)

Table 5: Data statistics. MC-09/10: Morpho Challenge 2009/2010. CC: A sub-sample of Common Crawl. S&G2013: Sirts and Goldwater (2013)

functionally different but orthographically equivalent inflectional morphemes. For example, *rockiness* is analysed as *rock_N y_s ness_s*, while *rocks* has two analyses: *rock_N +PL* and *rock_V +3SG*.

For Estonian we train on word lists extracted from Common Crawl and test on data prepared by Sirts and Goldwater (2013). The Estonian test set contains only surface segmentation into morphs (e.g. *kolmandal* is analysed *kolmanda l*). Table 5 provides information about each dataset.

Since we are developing an unsupervised system, we want to make sure that it generalizes to new languages. We therefore divide the languages into three **development languages** (English, German, Turkish) and three **test languages** (Finnish, Estonian, Arabic). We used the development languages to choose features, design the data selection procedure and select best values for hyperparameters. The system that performed best on those languages was then used unmodified on the test languages.

Hyperparameters In addition to threshold values described above, we use the same $\lambda = 1$ (Equation 3) as Narasimhan et al. (2015). To control for under segmentation we downscale weights of the stop features by a factor of 0.8. We set the maximum affix length to 8 characters and the minimum word length to 1 character.

Evaluation Metric We test our model on the task of unsupervised morpheme analysis induction. We follow the format of Morpho Challenge 2010 and use Evaluation Metric for Morphological Analysis (EMMA) (Spiegler and Monson, 2010) to evaluate predicted outputs. EMMA works by finding the optimal one-to-one mapping between the model's output and the reference analysis (i.e., the spelling of the morphemes in the analysis doesn't matter). These are used to compute precision, recall, and F-score against the reference morphemes.

Baselines We compare our model to three other systems: Morfessor 2.0 (Virpioja et al., 2013), MORSEL (Lignos et al., 2009; Lignos, 2010) and MorphoChains (Narasimhan et al., 2015). We also use a trivial baseline *Words* which outputs the input word.

Morfessor (*Morf.2.0*) is a family of probabilistic algorithms that perform unsupervised word segmentation into morphs.Since the release of the initial version of Morfessor, it has become popular as an automatic tool for processing morphologically complex languages.

MORSEL is a rule-based unsupervised morphology learner designed for affixal morphology. Like our own system, it outputs morphological analyses of words rather than segmentations. MORSEL achieved excellent performance on the Morpho Challenge 2010 data sets.

MorphoChains (*MC*) is the model upon which our own system is based, but as noted above it performs segmentation rather than analysis. In contrast to Morfessor and MORSEL, which analyse words based only on orthographic patterns, MorphoChains (like our extension) uses both orthographic and the semantic information.

All three baselines have multiple hyperparameters. Since performance tends to be most sensitive to the treatment of word frequency (including possibly discarding low-frequency words), for each system we tuned the hyperparameters related to word frequency to optimize average performance on the development languages, and kept these hyperparameters fixed for the test languages.

4 Results and Discussion

Table 6 gives the performance of all models on the three development languages. Our model outperforms all baselines on every language, and is a clear improvement over the original MorphoChains.[8]

[8]In the results reported by Narasimhan et al. (2015), MorphoChains appeared to outperform Morfessor, whereas we find the opposite. There are several possible reasons for the discrepancy. First, Narasimhan et al. (2015) used a segmentation-based metric rather than EMMA, so the scores are not comparable. Second, Narasimhan et al. (2015) appear to have tuned

Lang	Method	Prec	Recall	F-1
ENG	*Words*	0.750	0.362	0.489
	Morf.2.0	**0.788**	0.712	0.749
	MORSEL	0.784	0.725	0.752
	MC	0.685	0.729	0.706
	Our Model	0.787	**0.741**	**0.763**
GER	*Words*	0.776	0.258	0.387
	Morf.2.0	**0.690**	0.468	0.558
	MORSEL	0.670	0.449	0.538
	MC	0.649	0.397	0.492
	Our Model	0.590	**0.548**	**0.568**
TUR	*Words*	0.702	0.201	0.313
	Morf.2.0	0.598	0.338	0.432
	MORSEL	**0.626**	0.324	0.427
	MC	0.577	0.330	0.420
	Our Model	0.596	**0.351**	**0.442**
AVG	*Words*	0.743	0.274	0.396
	Morf.2.0	0.692	0.506	0.580
	MORSEL	**0.693**	0.449	0.572
	MC	0.637	0.485	0.539
	Our Model	0.658	**0.547**	**0.591**

Table 6: Results on development languages. Scores calculated using EMMA. *Words*=Trivial baseline which outputs the input word.

Lang	Method	Prec	Recall	F-1
ENG	Model-D	0.710	0.746	0.728
	Model-C	0.811	0.705	0.754
	Model-B	0.775	0.738	0.756
	Model-A	0.755	0.751	0.753
	Full Model	0.787	0.741	**0.763**
GER	Model-D	0.633	0.497	0.557
	Model-C	0.666	0.456	0.541
	Model-B	0.608	0.530	0.566
	Model-A	0.636	0.455	0.530
	Full Model	0.590	0.548	**0.568**
TUR	Model-D	0.554	0.365	0.440
	Model-C	0.601	0.344	0.438
	Model-B	0.604	0.344	0.437
	Model-A	0.588	0.346	0.436
	Full Model	0.596	0.351	**0.442**

Table 7: Ablation analysis. -D=no data selection, -C=no compound transformations, -B=no PREFLIST, SUFLIST, COMPOUND features, -A=no other LSE-based affix features.

To see where the benefit is coming from, we performed ablation tests (Table 7). Results show the importance of the LSE-based affix features (Model-A). Using these features gives gains of +1.0%, 3.8% and 0.6% F-1 absolute on English, German and Turkish respectively over using the raw affix occurrence frequencies as used by Narasimhan et al. (2015). We can see that our data selection scheme (Model-D) is important for English (+3.5%) and German (+1.1%). Although we expected that the data selection scheme would have the biggest impact on Turkish because of its small training data, it has very little effect on this language. As expected, the compounding transformation (Model-C) has a prominent impact on German (+2.7%) and a modest effect on English and Turkish. The three features PREFLIST, SUFLIST, COMPOUND (Model-B) have the least impact on the model's performance (on average 0.5% F-1 absolute), however the effect is substantial considering that this gain is achieved by merely 3 features as opposed to a new feature type with many instantiations.

Table 8 shows some example outputs for English, German and Turkish. These analyses include some correctly identified spelling changes (Example 1) compounds (Example 4), and purely concatenative morphology (Example 6). In Example 2, *+ble* is counted as incorrect because our model predicts *+ble* both for *deplorable* and *reproducible* while the reference analysis uses *able_s* and *ible_s*, respectively. Since EMMA uses one-to-one alignment, it deems one of the alignments wrong. The opposite problem occurs in Example 4: our model analyses *aus* in two ways, either as a prefix *aus+* or as a separate word *aus* (part of a compound), whereas the reference analysis always treats it as a separate word *aus*. Example 6 illustrates an oversegmentation error, caused by encountering two similar forms of the verb *giy*, *giymeyin* and *giymeyi*.

Performance of all models on the three test languages is shown in Table 9. On average, our model does better than MorphoChains and MORSEL, but slightly worse than Morfessor. However, this difference is mainly due to Morfessor's very good performance on Estonian, which is the only test set using gold standard segmentations rather than analyses. All systems perform poorly on Arabic since they do not handle templatic morphology; nevertheless our model and Morfessor perform considerably

their hyperparameters separately for each language. Finally, it is not clear how they tuned the frequency-related hyperparameters for Morfessor. We found that Morfessor performed better than MorphoChains when either low frequency words are pruned from its input, or its log-frequency option is used rather than raw frequency.

No	Lang.	Test Word	Reference Analysis	Our Model
1	ENG	acknowledging	ac_p knowledge_N +PCP1	ac+ knowledge +ing
2	ENG	reproducible	re_p produce_V **ible_s**	re+ produce ~~+ble~~
3	GER	wohnstuben	wohn_V stube_N +PL	wohn stube +n
4	GER	ausdrueckliche	**aus** drueck_V lich +ADJ-e	~~aus+~~ drueck +lich +e
5	TUR	budaklara	budak +PL +DAT	budak +lar +a
6	TUR	giymeyin	giy +NEG_ma **+P2_PL**	giy +me ~~+yi +n~~

Table 8: Examples morpheme analyses produced by our model on the development languages. Reference analyses in **bold** correspond to the predicted analyses that are ~~incorrect~~. See text for further explanation.

Lang	Method	Prec	Recall	F-1
ARA	*Words*	1.000	0.112	0.202
	Morf.2.0	0.719	**0.224**	**0.342**
	MORSEL	**0.993**	0.135	0.238
	MC	0.988	0.125	0.221
	Our Model	0.839	0.206	0.331
FIN	*Words*	0.902	0.282	0.430
	Morf.2.0	**0.704**	0.389	0.501
	MORSEL	0.698	0.504	**0.585**
	MC	0.557	0.483	0.518
	Our Model	0.588	**0.514**	0.549
EST*	*Words*	0.951	0.572	0.715
	Morf.2.0	**0.858**	**0.785**	**0.820**
	MORSEL	0.686	0.777	0.729
	MC	0.840	0.611	0.707
	Our Model	0.756	0.763	0.760
AVG	*Words*	0.951	0.322	0.449
	Morf.2.0	0.760	0.466	**0.554**
	MORSEL	**0.792**	0.472	0.517
	MC	0.785	0.413	0.492
	Our Model	0.728	**0.494**	0.547

Table 9: Results on test languages. Scores calculated using EMMA. *=reference analysis contains word segmentation.

better than the others. Overall, our model performs consistently near the top even if not the best for any of the three languages.

Lexical Inventory Size One of the motivations for unsupervised morphological analysis is to reduce data sparsity in downstream applications, which implies that for a given level of accuracy, systems that produce a more compact representation (i.e., a smaller morpheme inventory) should be preferred. To see how compactly each model represents the test set, we count the number of unique morphemes (or morphs, or labels) in the predicted output of each model and compare it with the number of labels in the reference analysis and the number of words in the test set. Table 10 summarizes this information[9]. For all languages except Estonian our model finds the most compact set of items. The number of distinct morphemes identified by our model is only about 5%, 4.5% and 8.0% larger than in the reference analysis for English, German and Finnish respectively. On average our model identified 12.8% and 14.8% fewer morphemes than Morfessor and MorphoChains respectively, while on average performing no worse or better than the two word segmentation systems. MORSEL produces the second most compact output with only a 3.2% larger set of distinct morphemes than our model, leaving the two word segmentation systems, Morfessor and MorphoChains, in the third and the forth place respectively. These results suggest that systems that attempt to output morphological analysis succeed in reusing the same morphemes more frequently than the systems that perform surface segmentation.

Method	ENG	GER	TUR	FIN	EST
Morf.2.0	1439	2005	1873	2586	1620
MC	1442	2004	1912	2718	1635
MORSEL	1373	1865	1748	2177	**1562**
Ours	**1336**	**1600**	**1725**	**2114**	1616
Ref.Anal	1257	1520	1361	1966	1548
Words	1569	1779	1760	1835	1500

Table 10: The number of distinct morphemes identified by each model. The number of distinct labels used in the reference analysis and the number of words in unanalysed test sets are given for comparison.

5 Conclusion

We presented an unsupervised log-linear model that learns to identify morphologically related words

[9] Arabic is excluded because of the low overall performance of all models (maximum recall 22.4%).

and the affixes and spelling transformations that relate them. It uses these to induce morpheme-level analyses of each word and an overall compact representation of the corpus. In tests on six languages, our system's EMMA scores are considerably better than its inspiration, the segmentation system MorphoChains, and it also outperformed the rule-based analysis system MORSEL. Our system achieved similar EMMA performance to Morfessor but with a more compact representation—the first probabilistic system we are aware of to do so well. In future work, we hope to investigate further improvements to the system and perform extrinsic evaluation on downstream tasks.

References

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 48–57. Association for Computational Linguistics, July.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions of Speech and Language Processing*, 4(1):1–34, February.

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

Sharon Goldwater and Mark Johnson. 2004. Priors in Bayesian learning of phonological rules. In *Proceedings of the Seventh Meeting of the ACL Special Interest Group in Computational Phonology*, pages 35–42, Barcelona, Spain, July. Association for Computational Linguistics.

Margaret A Hafer and Stephen F Weiss. 1974. Word segmentation by letter successor varieties. *Information storage and retrieval*, 10(11):371–385.

Mikko Kurimo, Sami Virpioja, Ville Turunen, and Krista Lagus. 2010. Morpho Challenge 2005-2010: Evaluations and results. In *Proceedings of the 11th Meeting of the ACL Special Interest Group on Computational Morphology and Phonology*, pages 87–95, Uppsala, Sweden, July. Association for Computational Linguistics.

Yoong Keok Lee, Aria Haghighi, and Regina Barzilay. 2011. Modeling syntactic context improves morphological segmentation. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 1–9, Portland, Oregon, USA, June. Association for Computational Linguistics.

Constantine Lignos, Erwin Chan, Mitchell P Marcus, and Charles Yang. 2009. A rule-based unsupervised morphology learning framework. In *CLEF (Working Notes)*.

Constantine Lignos. 2010. Learning from unseen data. In Mikko Kurimo, Sami Virpioja, and Ville T. Turunen, editors, *Proceedings of the Morpho Challenge 2010 Workshop*, pages 35–38, Helsinki, Finland, September 2–3. Aalto University School of Science and Technology.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at the 2013 International Conference on Representation Learning*.

Jason Naradowsky and Sharon Goldwater. 2009. Improving morphology induction by learning spelling rules. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, pages 1531–1536.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. 2015. An unsupervised method for uncovering morphological chains. *Transactions of the Association for Computational Linguistics*, 3:157–167.

Sylvain Neuvel and Sean A. Fulop. 2002. Unsupervised learning of morphology without morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 31–40. Association for Computational Linguistics, July.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 209–217, Boulder, Colorado, June. Association for Computational Linguistics.

Patrick Schone and Daniel Jurafsky. 2000. Knowledge-free induction of morphology using latent semantic analysis. In *Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning*, pages 67–72, Lisbon, Portugal. Association for Computational Linguistics.

Kairit Sirts and Sharon Goldwater. 2013. Minimally-supervised morphological segmentation using Adaptor Grammars. *Transactions of the Association for Computational Linguistics*, 1(May):231–242.

Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 354–362, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Sebastian Spiegler and Christian Monson. 2010. EMMA: A novel evaluation metric for morphological analysis. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1029–1037, Beijing, China, August. Coling 2010 Organizing Committee.

Sami Virpioja, Peter Smit, Stig-Arne Grönroos, Mikko Kurimo, et al. 2013. Morfessor 2.0: Python implementation and extensions for morfessor baseline.

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.