Extending Morphological Chains with Supervised Learning and Cross-Language Features

Calvin Huang (calvinh@mit.edu), 6.867
Brian Shimanuki (bshimanuki@mit.edu), 6.867
Charlotte Chen (czchen@mit.edu), 6.806

Abstract

Building on an existing algorithm to perform morphological analysis, based on using semantic features and contrastive estimation to detect morphological chains, we investigate the effects of various types of extensions to this algorithm. We extend the unsupervised model with the log-likelihood of known word segmentations, producing a semi-supervised model. Finally, we also attempt to use Turkish morphological parents and an English-Turkish translation dictionary to help detect and disambiguate English word segmentations and transformations.

1 Introduction

The division of words into morphemes–basic units of meaning, such as root words, affixes, et cetera—encodes information about how words are constructed and derive their meaning, as well as how languages evolve.

When performing morphological analysis, there are a few different cases to consider: words that share similar meaning but appear in different syntactic contexts ("German"—"Germanic"), new words generated from the combination of two separate words ("bookcase"—"book"—"case"), words with dissimilar meaning but derived from one another "insufficient"—"sufficient"). In the first and last case, words are often related by the addition of prefixes, suffixes, and other affixes. This allows us to form morphological chains - which link a base word to its derivative through a series of affixations.

Most work in the field looks at either orthographic information or semantic information. Narasimhan et al. [2015] integrates these approaches by representing a word with a series of derivations from the base form of the word, thereby capturing both types of information in each derivation. At each step, the similarity across the derivation can be analyzed using both character similarity measures and semantic similarity measures.

2 Unsupervised Model

Our model builds on work by Narasimhan et al. in constructing *morphological chains* to model the morphological segmentation of words in a language in an unsupervised manner. The idea behind morphological chains is that complex words are constructed by attaching morphemes simpler words. Thus, complex words have words from which they are directly derived, which we call *parent* words. Likewise, the derived word is a *child* word. A morphological chain is a sequence of words such that consecutive pairs form a parent-child relationship. For example, the word *unsustainable* can be constructed from *sustain*, as demonstrated in the chain *sustain* $\rightarrow$ *sustainable* $\rightarrow$ *unsustainable*. Words without parents are called *base* words. In our example, *sustain* is a base word.

Since a parent word can have multiple children (e.g., *play* $\rightarrow$ *plays* and *play* $\rightarrow$ *played*), words can be part of multiple chains. Thus mul-

tiple chains can share segments. Narasimhan et al. makes use of this shared information by constructing a model which analyzes parent-child pairs.

Because we have chains where different chains can share segments, it is natural to consider the graph formulation where each word is represented by a node and each parent-child transition is represented by a directed edge. Then if we consider only the most likely parent for a word,[1] the resulting graph forms a forest, and our objective is to find the path to from a given node to the root in an unsupervised manner without any explicit information about any of the edges.

Note that by constructing morphological chains rather than just finding segmentations, more information is gained about the words tested. Generating the chain, in addition to segmenting a word, also finds the base word and predicts the order in which the morphemes attach to the base word.

2.1 Model

In the unsupervised version of our model, we observe words in a wordlist with their word count of occurrences. Additionally, we have access to a large corpus of text from which we can obtain semantic information. Our objective is to generate morphological segmentations, which we do by generating morphological chains, as given by Narasimhan et al.

A log-linear model is used to evaluate different pairs. For this, a feature mapping $\phi : \mathcal{W} \times \mathcal{Z} \rightarrow \mathbb{R}^d$ to compute a corresponding weight vector $\theta \in \mathbb{R}^d$. $\mathcal{W}$ is the set of words being trained on, and $\mathcal{Z}$ is the set of *candidates* for the parents the words in $\mathcal{W}$. For a word $w \in \mathcal{W}$, $\mathcal{Z}$ is constructed by splitting w at many points. To capture orthographic changes in the parent word as it undergoes the derivation from $z \in Z$ to w (eg., *believe* $\rightarrow$ *believing*), the type of transition is kept as part of the candidate. Thus the candidates take the form (*parent*, *type*), where the type is the type of transition.

Note that parent words usually undergo changes only when attaching suffixes. Thus there is one *Prefix* class, but there are a variety of suffix classes. When acquiring a suffix, the parent word can:

(1) Undergo no change (*bike* $\rightarrow$ *bikes*)

(2) Repeat a character (*star* $\rightarrow$ *starring*)

(3) Delete a character (*believe* $\rightarrow$ *believing*)

(4) Modify a character (*parry* $\rightarrow$ *parried*)

These correspond to the candidates (*bike*, *Suffix*), (*star*, *Repeat*), (*believe*, *Delete*), and (*parry*, *Modify*). Finally, there is a *Stop* type, which is used to signify that the word is a base word.

2.2 Word Vectors

It has been shown that words can be mapped to word vectors in a reasonably-sized dimensional space such that semantically similar words align in similar directions [Mikolov et al., 2013], thus we can compute a *cosine similarity* as a distance metric between words which captures semantic similarity. Given vector representations of two words, the cosine similarity is measured as the ratio of the dot product between the word vectors and the product of the norms of the word vectors.

In our work, we generate 200-dimensional vectors for words using Word2Vec, which looks at word alignment and co-occurrence patterns across many documents. In generating our vectors, we use text from Wikipedia, to generate the English word vectors, along with the BOUN Web Corpus (consisting of news articles) to generate Turkish word vectors.

2.3 Features

The model uses a variety of features covering orthographic and semantic aspects of the word-candidate pairs. These features are computed for pairs $(w, z) \in \mathcal{W} \times \mathcal{Z}$.

[1] Words are not restricted to having a single valid parent (or segmentation). In cases where there are multiple parents (segmentations), we care only that the algorithm selects one of them.

Affixes Prefixes and suffixes appear in many words. For a given word with its potential parent, the affix removed can be compared against a precompiled list of potential affixes. This allows the model to learn which affixes are real and apply them to other cases with the same potential affix.

Affix Correlation Similar to comparing affixes across words, we can use the joint distribution of words and their potential affixes. There is a correlation between affixes that can usually attach to the same stem, often corresponding to the part of speech. For example, the participles formed from the suffixes -ing and -ed usually occur together. For candidates with an affix that is correlated with another affix, we can check if the parent with the other affix also occurs in the observed wordlist.

Semantic Similarity Morphologically related words should have similar meanings. We measure the cosine similarity between the word and its candidate parent as a feature to represent semantic similarity.

Transformations As discussed above, some derivations involve a change to the parent word. To allow for non-concatenative morphology, we use binary features which capture the type of transformation. We use the same types of transformations for features as we do for generating candidates: repetitions, deletions, and modifications. The set of binary features are the cartesian product of the type of transformation with the characters transformed.

Wordlist Most of the time, we want the parents to be valid words. To this effect, we compare the candidate parent against the wordlist, and add a feature for the log of the word count. Additionally, we set a binary feature corresponding to whether the word was found at all in the wordlist.

Stop Features We have a number of miscellaneous features targeted at finding the end of chains. These include orthographic information like the length of the parent, unigrams and bigrams at the beginning and end of the parent, and the highest cosine similarity between the word and any of its candidate parents.

2.4 Unsupervised Learning

Recall that in our model, we want to find weights θ for the feature vectors $\phi(w, z)$, where $w \in \mathcal{W}$ and $z \in \mathcal{Z}$. Define the probabilities of a word-candidate pair (w, z) as $P(w, z) \propto e^{\theta \cdot \phi(w,z)}$. Then the probability of a candidate z ocuring given w is

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

Let our set of observed words be D. We approach this by trying to maximize the likelihood of observing the words in D from the space of all constructible strings from the alphabet, Σ^*. We maximize the log-likelihood over θ as given by

$$\begin{aligned} L(\theta; D) &= \log \prod_{w^* \in D} P(w^*) \\ &= \log \prod_{w^* \in D} \sum_{z \in C(w^*)} P(w^*, z) \\ &= \log \prod_{w^* \in D} \frac{\sum_{z \in C(w^*)} e^{\theta \cdot \phi(w^*, z)}}{\sum_{w \in \Sigma^*} \sum_{z \in C(w)} e^{\theta \cdot \phi(w,z)}} \end{aligned} \tag{2}$$

We cannot compute $L(\theta; D)$ directly since we cannot compute over all strings in Σ^*. Instead, we approximate this distribution by considering only strings which are similar to those encountered in D.

We use the method of Contrastive Estimation [Smith and Eisner, 2005] and substitute the space of all strings Σ^* with neighbors of each word, $N(w)$. Toward this end, we transpose pairs of consecutive letters of w near both ends of the word. We also do both simultaneously. Together these form our set of neighbors for w. The neighbors form a proxy for Σ^*, and represent the set of

strings we want to reduce the probability of seeing in our model because they are not observed. This has the benefit of providing contrast in the structure of words while not requiring the model to look at the entire Σ^*.

With this substitution, we can formulate the contrastive log-likelihood as

$$L_C(\theta; D) = \log \prod_{w^* \in D} \frac{\sum_{z \in C(w^*)} e^{\theta \cdot \phi(w^*, z)}}{\sum_{w \in N(w^*)} \sum_{z \in C(w)} e^{\theta \cdot \phi(w, z)}} \tag{3}$$

With a regularization term, this becomes

$$L_C(\theta; D) = \log \prod_{w^* \in D} \frac{\sum_{z \in C(w^*)} e^{\theta \cdot \phi(w^*, z)}}{\sum_{w \in N(w^*)} \sum_{z \in C(w)} e^{\theta \cdot \phi(w, z)}} - \lambda \|\theta\|^2 \tag{4}$$

Simplifying, this becomes:

$$L_C(\theta; D) = \sum_{w^* \in D} \left(\log \sum_{z \in C(w^*)} e^{\theta \cdot \phi(w^*, z)} - \log \sum_{w \in N(w^*)} \sum_{z \in C(w)} e^{\theta \cdot \phi(w, z)} \right) - \lambda \|\theta\|^2 \tag{5}$$

This has the gradient given by

$$\frac{\partial L_C(\theta; D)}{\partial \theta_j} = \sum_{w^* \in D} \left(\frac{\sum_{z \in C(w^*)} \phi_j(w^*, z) \cdot e^{\theta \cdot \phi(w^*, z)}}{\sum_{z \in C(w^*)} e^{\theta \cdot \phi(w^*, z)}} - \frac{\sum_{w \in N(w^*)} \sum_{z \in C(w)} \phi_j(w, z) \cdot e^{\theta \cdot \phi(w, z)}}{\sum_{w \in N(w^*)} \sum_{z \in C(w)} e^{\theta \cdot \phi(w, z)}} \right) - 2\lambda\theta_j \tag{6}$$

We optimize $L_C(\theta; D)$ with its gradient using the LBFGS-B algorithm.

2.5 Prediction

The algorithm yields an optimal θ^* of weights of the features generated from D. Then given a test word w, we can predict the probability of a given parent candidate z by computing $P(z|w) = e^{\theta^* \cdot \phi(w,z)}$ as in Equation 1. To predict the parent of w, we pick the MLE candidate from parent candidates generated for w, which we can generate in the same way as we did while training.

To generate a morphological chain for w, we recursively predict the MLE parent for w, then the MLE parent of that prediction, and so forth until the *Stop* candidate is predicted. Together, this sequence forms our chain. A morphological segmentation can be constructed from the chain by splitting w at every point corresponding to where each of the edges in the chain splits its child word.

3 Semi-supervised Learning with Known Word Segmentations

In a semi-supervised learning model, we still have access to a large wordlist of observed words as well as the corpus for semantic information. In addition, we have a small amount of pre-segmented words (about 2% of the size of the wordlist). We use these pre-segmented words to generate probable morphological chains, which we can then incorporate into the likelihood model. Specifically, we want to maximize the joint distribution of observing the words in the wordlist along with observing the segmentations of the words from the list of pre-segmented words, with some weighting on each of them.

3.1 Generating Chains from Segmentations

Based on the correct segmentation of a word, we construct a probable chain. In addition to the split points of the segmentation, we have access to tags for each segment which give the original form of the segment. This information is used to construct the type (*Repeat*, *Modify*, etc) of the transformation.

Assuming no infixes, each parent must be formed by removing segments at either the beginning or the end of the word. In generating morphological chains from segmentations, we assume most segmentations are derived from single affix transitions. Our method cannot generate the correct chain for compound words where both parts are derived from parent words, for example.

Given a segmentation for a word which is not a base word, we have two candidates for the segmentation of the parent word:

(1) The parent is the child without a prefix.

(2) The parent is the child without a suffix.

We use a heuristic to decide between them. Our heuristic captures properties like the frequency of the parent word in the wordlist, the cosine similarity between the parent and child, and the length of the parent relative to the child. We select the parent candidate with the higher score.

By recursing on the parent, we can generate more word-parent pairs that we want our model to generate.

3.2 Semi-supervised Model

Using the word-parent pairs generated from the pre-segmented list, in addition to maximizing the log-likelihood of the known wordlist, we can also maximize the log-likelihood of the correct segmentation of the training data.

If a given word can be expressed as a morphological chain, the likelihood of its correct segmentation is equal to the product of the likelihoods of each word-to-parent transition within the morphological chain. Therefore, the likelihood of the segmentation of the training data is the product of the likelihoods of each correct word-to-parent transition for every morphological chain in the training data.

Let $MC(w)$ be the set of word-parent pairs constructed from the morphological chains of the pre-segmented list. We want to maximize the likelihood of the document D in the following:

$$\prod_{w,c \in D} P(c|w) = \prod_{w,c \in D} \left(\prod_{w',z' \in MC(w,c)} P(z'|w') \right) \tag{7}$$

where w is a word in D and c is its segmentation.

We can simplify this product by letting $TR(D)$ be all pairs of words and parents implied in each of the morphological chains of all words in D—then, the log-likelihood of our training data can be expressed as

$$\begin{aligned} L_S(\theta; D) &= \log \prod_{w,z \in TR(D)} P(z|w) \\ &= \sum_{w,z \in TR(D)} \Bigg(\theta \cdot \phi(w,z) \\ &\qquad - \log \left(\sum_{z' \in C(w)} e^{\theta \cdot \phi(w,z')} \right) \Bigg) \end{aligned} \tag{8}$$

We can also express its gradient as such:

$$\begin{aligned} \nabla_\theta L_S = \sum_{w,z \in D} \Bigg(& \phi(w,z) \\ & - \frac{\sum_{z' \in C(w)} \phi(w,z') e^{\theta \cdot \phi(w,z')}}{\sum_{z' \in C(w)} e^{\theta \cdot \phi(w,z')}} \Bigg) \end{aligned} \tag{9}$$

Adding this to our original log-likelihood from contrastive estimation along with an L2 regularization term, we get our objective:

$$\begin{aligned} L(\theta; D) &= L_{CE}(\theta; D) + \alpha L_S(\theta; D) - \lambda \|\theta\|^2 \\ \nabla_\theta L &= \nabla_\theta L_{CE} + \alpha \nabla_\theta L_S - 2\lambda\theta \end{aligned} \tag{10}$$

where α is a free parameter that specifies by how much to weight the labeled training data likelihood against the contrastive estimation likelihood. As before, with its gradient given above, we minimize $L(\theta, D)$ with LBFGS-B.

4 Using Cross-Language Features and Candidates

Due to the simple nature of our parent candidate generation routine, there are many cases in which the potential parent is not actually generated; for example, the morphological parent of "feet" is the singular "foot"; however, there is no prefix or suffix that turns one into the other, and therefore the parent for "feet" is not generated by our system.

However, other languages may not have such an irregular transformation for the same words, for example in Turkish the corresponding translations for "feet" and "foot" are "ayaklar" and "ayak". A properly trained Turkish morphological parent model will correctly detect "ayak" as a parent of "ayaklar", and can be used to suggest morphological parents in English after translation.

Since there are multiple possible morphological parents for a given word, and multiple possible translations in either direction, this leads to many possible candidates for a given word, we generate a set of possible parent candidates (from translation):

$$C_2(w) = \bigcup_{w_t \in T_{ET}(w)} \left(\bigcup_{w_{tp} \in MP_T(w_t)} T_{TE}(w_{tp}) \right) \tag{11}$$

where T_{ET} returns the set of possible Turkish translations for an English word; MP_T returns the set of possible parents for some Turkish word, T_{TE} returns the set of possible English translations for an English word, and this potentially large set of candidates is pruned with heuristics judging how similar the candidate is to the original word. We then extend our original parent candidate with this parent candidate set.

Additionally, we extend our feature set with features regarding the translation link between the word and potential parent, to include information such as whether or not each candidate was generated by a Turkish translation (a strong indicator for parent-ness) and how confident the Turkish model was in selecting a parent.

Therefore, our model becomes:

$$\begin{aligned} P(w) &= \sum_{z \in C'(w)} P(w, z) \\ &= \frac{\sum_{z \in C'(w)} e^{\theta \cdot \phi(w, z, T(w), T(z))}}{\sum_{w' \in N(w)} \sum_{z \in C'(w')} e^{\theta \cdot \phi(w', z', T(w'), T(z'))}} \end{aligned} \tag{12}$$

where $C'(w)$ is the potential parents of w updated with potential irregular translations, and $T(w)$ is the set of Turkish translations of w.

To build the two-language model, we first train the Turkish model with semi-supervised data. Using this model and an English-Turkish/Turkish-English dictionary, we derive an extended set of possible English parent candidates for English words, and additional features. These are then used in conjunction with training the English model on the unlabeled word lists, and also with the labeled training data.

5 Experimental Validation

5.1 Data

We used the datasets provided by 2010 Morpho Challenges. For English, this includes a wordlist with word frequencies of approximately 878,000 words scraped from the Wortschatz collection from the University of Leipzig, CLEF, and the Europarl corpus as well as gold standard segmentation training and development sets of approximately 1000 randomly selected words. The MorphoChallenge 2010 gold standard segmentation is based on the CELEX database. For Turkish, the word list contains approximately 670,000 words, with gold standards from a morphological parser developed at Boğaziçi University. The Turkish wordlist was obtained from a significantly smaller corpus—at 1 million words, compared to the English corpus of 18 million. For a word-word translation dictionary, we used a English-Turkish dictionary (retrieved from `http://www.fen.bilkent.edu.tr/~aykutlu/sozluk.txt`).

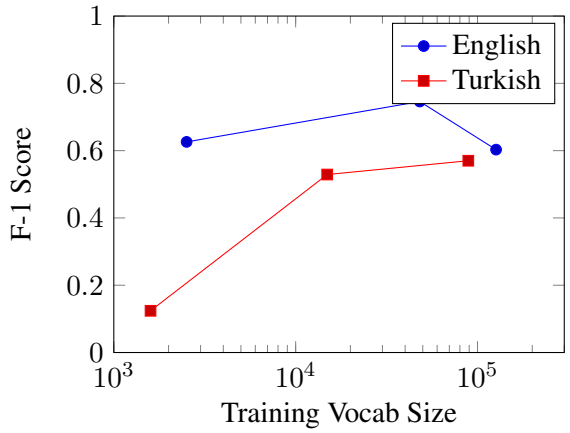

Figure 1. Performance of unsupervised model after frequency thresholding training data

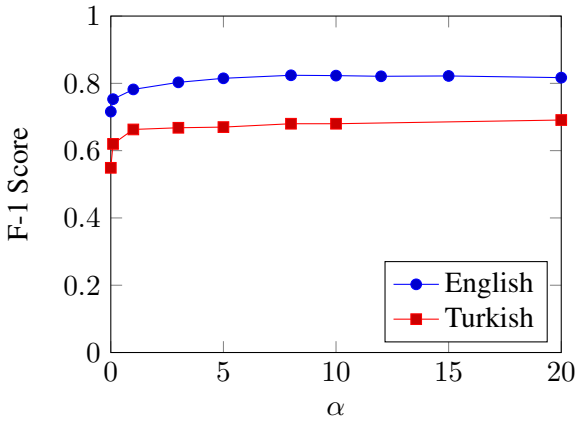

Figure 2. Performance of the semi-supervised model based on α.

The Turkish language contains characters like ç and ğ which are not in the English alphabet. In computing with them and in our results below, we have preprocessed the Turkish texts to replace characters not found in English with a capitalized letter. (We convert all words to lowercase beforehand, so there are no ambiguous symbols.)

5.2 Evaluation

Performance was measured by looking at the segmentation points within each word (i.e. the points within the word where it is split into separate morphemes) and evaluating the F1 measure, i.e. the harmonic mean of precision and recall; precision is the fraction of the segmentation points that are present in the gold segmentation; recall is the fraction of the gold segmentation points that are correctly identified by our estimator.

6 Results

We found that utilizing information from the gold segmentations, we were able to substantially increase our performance, from an accuracy of 75% to 81.7% with English, and from 54.3% to 69.7% with Turkish. This is a substantial but expected increase; observing the unsupervised model reveals that it performs many segmentations with nonexistent affixes, which are rectified by providing input about known affixes.

It should be noted that as α increases, the likelihood of the labeled training data matters more and more, and the score is positively influenced up to a point—as α increases further, the labeled data dominates the loss function, and the information from the unstructured model ceases to be used, causing a slow decline in the score.

We try using differently sized training sets by varying frequency thresholds for the words. As training sets often contain some amount of error and noise (the English wordlist contains an html tag as a word), the frequency threshhold is a good approximation of error; words with low frequency are may be mispelling or words from other languages. We find that for datasets ranging from 2500 words to 48,000 words give similar quality results, with a sharp drop off in scores afterwards.

We found that adding the additional Turkish candidates and features regarding translation did not help accuracy of training very much; this is likely due to the incompleteness of the translation dictionary only being able to provide potential translations for very few words. Furthermore,

Language	Method	Precision	Recall	F1
English	Narasimhan et al.	0.807	0.722	0.762
	Unsupervised (Baseline)	0.744	0.756	0.750
	Semi-supervised ($\alpha = 15$)	0.892	0.777	0.831
	Two language unsupervised	0.502	0.682	0.578
	Two language semi-supervised	0.519	0.789	0.623
Turkish	Narasimhan et al.	0.743	0.520	0.612
	Unsupervised (Baseline)	0.627	0.478	0.543
	Semi-supervised ($\alpha = 50$)	0.763	0.642	0.697

Table 1. Accuracies of Various Models. For our model, we used the top 48000 English words in the wordlist, and the MorphoChallenge 2010 Train/Development sets for training and testing, respectively.

Language	**Correct Segmentations**		**Incorrect Segmentations**		
	Word	Segmentation	Word	Predicted	Correct
English	suburbanite	suburb/an/ite	provokingly	provok/ing/ly	provoking/ly
	buffeted	buffet/ed	invalidated	in/validat/ed	in/valid/at/ed
	ewers'	ewer/s/'	dutifully	dutiful/ly	duti/ful/ly
	yowling	yowl/ing	ignominiously	ignominious/ly	ignomni/ous/ly
Turkish	kompozisyonudur	kompozisyon/u/dur	aklInIz	aklI/nIz	akl/In/Iz
	ayet	ayet	genCliGinizi	genC/liGi/ni/zi	genC/liG/iniz/i
	rUyalarInIz	rUya/lar/InIz	irdeleyin	irdeley/in	irdele/yin
	borular	boru/lar	indirgenemez	indirgenemez	indirge/n/emez

Table 2. Examples of Correct and Incorrect Segmentations

due to the ambiguity of translation, it's very hard to tell whether or not something would make a good parent of a word after translating the word into Turkish, taking the parent in Turkish, and translating it back. The additional candidates, for the most part, reduced accuracy by increasing the number of incorrect parent choices, as opposed to their intended result of providing parents that our candidate generation algorithms missed.

References

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013. URL `http://arxiv.org/abs/1301.3781`.

Karthik Narasimhan, Regina Barzilay, and Tommi S. Jaakkola. An unsupervised method for uncovering morphological chains. *CoRR*, abs/1503.02335, 2015. URL `http://arxiv.org/abs/1503.02335`.

Noah A. Smith and Jason Eisner. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 354–362, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219884. URL `http://dx.doi.org/10.3115/1219840.1219884`.