Unsupervised Discovery of Morphological Chains Using Data from Multiple Languages

Zygimantas Straznickas

6.806 Project
zygi@mit.edu

I. Introduction

The project is based on a previous paper by Narasimhan et al. (2015) where the authors propose an unsupervised method of segmenting words by first producing their morphological chains. For a word w, its morphological chain is a chain of words that follows their derivation relations, starting with w, followed by a word w was derived from, and ending with a base word that is not derived. An example morphological chain for the word "repressiveness" is

repressiveness → repressive → repress → press

While the original model only uses data for a single language to train the model, it can be noticed that different languages often share some morphological derivation rules. For example, the morphological chain for the word "repressiveness" in French ("répressivité") is

répressivité → repressive → réprimer

While it does not exactly match the English chain, the original word is also formed by adding a suffix to the parent word. Our approach is to use this information to improve morphological chain generation accuracy and, consequently, the word segmentation accuracy of the model. Our proposed model will use two languages - the main language and the side language. It will produce morphological chains and segmentations for the main language, but use both the main and the side languages for model training and prediction.

II. Background

I. MorphoChain

The original MorphoChain paper by Narasimhan et al. (2015) proposes to construct morphological chains by using a log-linear model where the conditional probability of a candidate given a word is

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

where w is a word we are interested in, z is the tuple of a candidate parent word and a change type (suffix, deletion etc.), and $C(w)$ is the set of candidates of the word w. For a word $w = apples$, possible candidates would include *(apple, Suffix)* and *(pples, Prefix)*.

The feature vector $\phi(w, z)$ contains both semantic features (the word embedding distance between the word and the candidate) and othographic features (e.g. suffix correlation features precomputed from a word list).

The model minimizes the likelihood of seeing all the words in the word frequency list by marginalizing over possible candidates of each word:

$$p(D, \theta) = \prod_{w \in D} p(w) = \prod_{w \in D} \sum_{z_C(w)} p(w, z)$$

After the model is trained, a morphological chain is constructed greedily by predicting the most likely parent for each previous word until the *STOP* parent is most likely.

II. Low-rank Tensor Models

The low-rank tensor method as described by Lei et al. (2014) works by approximating the weighted sum of combinations of different feature vectors. Ideally, if the problem can be described by features of a few, e.g. 3, different types - one could construct three feature vectors u, v, w and consider all of their combinations by forming a tensor

$$(u \otimes v \otimes w)_{i,j,k} = u_i v_j w_k$$

where $\otimes$ is the Kronecker product. The the feature score could then be expressed as the element-wise product of this tensor and a weight tensor A:

$$s(A, u, v, w) =< A, u \otimes v \otimes w >$$

however, even for small feature vectors the size of A is very big and it is almost always unfeasible to store it all. However, if it is assumed that A has rank r and that r is small, this score function can be expressed as

$$s(U, V, W, u, v, w) = \sum_{i=1}^{r} [Uu]_i [Vv]_i [Ww]_i$$

where U, V and W are weight matrices that are reasonably sized and can be learned efficiently.

III. Method

In order to capture information from both the main and the side language, three feature vectors are constructed from each word-candidate pair: ϕ_{main}, ϕ_{side} and ϕ_{sim}.

The ϕ_{main} feature vector contains the features derived from the word and the candidate in the main language. It contains semantic and orthographic features from the original MorphoChain model with some additional features, e.g. the Levenshtein distance between the words, added.

The ϕ_{side} vector contains the features derived from the word and candidate translations. If either the word or the candidate is missing from the dictionary file, all feature values other than *bias* are set to 0. Otherwise, the vector contains the same types of features as ϕ_{main}, using statistics derived from the side language data files.

ϕ_{sim} captures the similarity between words in the main language and their translations in the side language. In particular, it includes features like

- The word has a translation
- The candidate has a translation
- Both the word and the candidate have a translation
- The length ratio of the word and its translation

Because the feature vectors will be created for many words that are artificially constructed when using Contrastive Estimation or words that are real but are missing from the dictionary file, the majority of ϕ_{side} vectors will be filled with zeros when training the model. This could potentially be a problem when training because ϕ_{main} and ϕ_{side} are usually sparse. It is important that the model is able to distinguish between ϕ_{side} values being 0 because of sparsity and because the translations do not exist. The low-rank tensor model can capture this relation by considering the combination of ϕ_{side} and the values of ϕ_{sim} that specify whether the translations exist.

In order to apply the low-rank tensor approach model for this model, we use a scoring function

$$s(\omega, z, U, V, W) = \sum_{i=1}^{r} [U\phi_{main}]_i [V\phi_{side}]_i [W\phi_{sim}]_i$$

where matrices U, V and W are the model parameters. The gradient of this scoring function is

$$\frac{\partial s(\omega, z, U, V, W)}{\partial U} =$$
$$\frac{\partial}{\partial U} \sum_{i=1}^{r} [U\phi_{main}]_i [V\phi_{side}]_i [W\phi_{sim}]_i =$$
$$\frac{\partial}{\partial U} (U\phi_{main})^T d =$$
$$\frac{\partial}{\partial U} \phi_{main}^T U^T d =$$
$$d\phi_1^T$$

where d is the component-wise product of $[V\phi_{side}]$ and $[W\phi_{sim}]$. The gradients with respect to V and W can be found similarly.

The rest of the model is close to the one in the original MorphoChain. The candidate probability given a word is logarithmic (although no longer log-linear):

$$p(z|w) = \frac{e^{s(w,z,U,V,W)}}{\sum_{z' \in C(w)} e^{s(w,z',U,V,W)}}$$

Where $C(w)$ is the set of candidates of w. The model tries to maximize the likelihood of each word by marginalizing over all candidates:

$$p(D, \theta) = \prod_{w \in D} p(w) = \prod_{w \in D} \sum_{z_C(w)} p(w, z)$$

$$p(D, \theta) \sim \prod_{w \in D} \sum_{z \in C(w)} \frac{p(z|w)}{\sum_{w' \in N(w)} p(z|w')} \qquad (1)$$

Ideally we would want to calculate the denominator in Eq. 1 by summing over all the possible strings w'. However, since this is computationally infeasible, we (similarly to original MorphoChain) use Contrastive Estimation (Smith and Eisner, 2005) to generate a set $N(w)$ of negative examples - strings that most likely aren't real words - by reordering first n or last n letters in the original word.

By taking the logarithm of this likelihood and adding a normalization term we get the log-likelihood expression

$$LL = \sum_{\omega^* \in D} \Bigg[\log \sum_{z \in C(\omega^*)} e^{s(\omega^*, z, U, V, Z)} - \log \sum_{\omega \in N(\omega^*)} \sum_{z \in C(\omega)} e^{s(\omega, z, U, V, Z)} \Bigg] + \lambda(|U| + |V| + |W|)$$

where $|M|$ is the Frobenius norm of the matrix, and the gradient

$$\frac{\partial LL}{\partial U} = \sum_{\omega^* \in D} \left[\frac{\sum_{z \in C(\omega^*)} \frac{\partial s}{\partial U}(\omega^*, z, U, V, W) e^{s(\omega^*, z, U, V, W)}}{\sum_{z \in C(\omega^*)} e^{s(\omega^*, z, U, V, W)}} - \frac{\sum_{\omega \in N(\omega^*)} \sum_{z \in C(\omega)} \frac{\partial s}{\partial U}(\omega^*, z, U, V, W) e^{s(\omega^*, z, U, V, W)}}{\sum_{\omega \in N(\omega^*)} \sum_{z \in C(\omega)} e^{s(\omega^*, z, U, V, W)}} \right] - 2\lambda U$$

The model is trained using a gradient-based method (specifically for this project, L-BFGS) by finding the gradients of the parameter matrices and the updating them. All three matrices are updated in each step. The model requires that not all weights are zero so two weight initialization methods were tried: random initialization and uniform non-zero initialization.

As in original MorphoChain, the morphological chains are produced greedily, by choosing a most likely parent candidate of a word in each level until the *STOP* candidate type is found to be most likely.

IV. Datasets

Five datasets are needed to train the model:

- Main language word frequency list
- Side language word frequency list
- Main language word embeddings
- Side language word embeddings
- Main $\rightarrow$ side language translation dictionary

The evaluation was done using English and French as the main and side languages respectively. While for these languages the first four datasets are widely available, a good translation dictionary was difficult to find. We tried two different types of dictionaries:

- A scraped Wiktionary English-French dictionary (27k entries). This dictionary was created manually so the translations in it are almost completely correct. However, this dictionary only contains the "base" forms of words and omits derived forms like plurals, word declensions etc.
- A dictionary file extracted from a Moses machine translation system phrase table (60k entries). This dictionary did contain derived word forms but because it was generated automatically it also had a lot of errors, especially for rare words.

V. Evaluation

Unfortunately, the model is unable to consistently outperform the original single-language MorphoChain model. Since the segmentation accuracy depends on the values the model was initialized with, in one case when using random initialization the multi-language model beat the original version. However, on average and when using constant initialization it resulted in lower accuracy. The results, evaluated using tensor rank $r = 20$, $\lambda = 15$ and random weight initialization can be seen in Table 1. When testing different translation files, it was found that the Machine Translation file consistently produced worse results than the manually created Wiktionary dictionary, despite the fact that it was much smaller. Therefore, it looks like the accuracy of translations is more important for this model than their existence itself.

In order to understand why the model does not produce improved results, we looked at the morphological chains it produces to see if any patterns could be found. We discovered four different categories of chains:

1. Trivial chains - chains of only one word (tape | bande $\rightarrow$ STOP)
2. No translation chains - chains where the majority of English words did not have translations (homogenizes | (no transl.) $\rightarrow$ homogenize | (no transl.) $\rightarrow$ STOP)
3. Contrastive translation chains - chains where the translations exist but do not follow the derivation relation ofthe English words (suburban | banlieue $\rightarrow$ urban | urbain $\rightarrow$ STOP)
4. Good chains - chains where the all translations exist and the derivation relations can be clearly seen (devaluation | dévaluation $\rightarrow$ evaluation | évaluation $\rightarrow$ evaluate | évaluer $\rightarrow$ STOP)

The first 105 words in the golden segmentation file were checked to count the numbers of chains in each category. The results can be seen in Table 2. Interestingly, even though the Wiktionary dictionary produced fewer good translations, there were also much fewer contrastive translations. This improved performance overall, however a future topic to explore is a better source of translations that would both be accurate and contain derived word forms. Also, it would be useful to investigate methods to pre-initialize weight values before optimization to get the best-case results in a more reliable way.

Table 1: *Multilangue MultiChain results (highest, average, lowest values out of 10 runs)*

Method	Precision	Recall	F-1 Score
Model-A	0.810	0.713	0.758
Single language MorphoChain	0.807	0.722	0.762
Multilanguage MC - Highest	0.806	0.735	0.769
Multilanguage MC - Average	0.778	0.720	0.748
Multilanguage MC - Lowest	0.797	0.687	0.738

References

[Lei et al., 2014] Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola (2014). Low-Rank Tensors for Scoring Dependency Structures.

[Narasimhan et al., 2015] Karthik Narasimhan, Regina Barzilay and Tommi Jaakkola (2015). An Unsupervised Method for Uncovering Morphological Chains.

[Smith and Eisner, 2005] Noah A. Smith and Jason Eisner (2005). Contrastive Estimation: Training Log-Linear Models on Unlabeled Data.

Table 2: *Numbers of chains of each type*

Chain type	Machine Translation	Wiktionary
Trivial	37	34
No translation	34	61
Contrastive translation	23	3
Good	11	7