

# Unsupervised Segmentation of Bilingual Text

Slav Petrov  
University of California at Berkeley  
slav@petrovi.de  
Fall 2004

## Abstract

*This paper presents a system for segmenting bilingual text. A Naive Bayes classifier using  $n$ -gram counts is trained on an unlabeled corpus using the Expectation Maximization (EM) algorithm. We show that the performance can be improved when a small set of labeled documents is available: the classifier is first trained using the available labeled documents and then the EM technique of iterating (until convergence) between probabilistically labeling the unlabeled documents and re-estimating the model's parameters is applied. While the accuracy of this semi-supervised classifier is higher than the one of the unsupervised classifier, the accuracy degrades with each iteration of EM. In the second part Deterministic Annealing (DA) is applied instead of EM and is shown to converge to a better local maximum, thus achieving better segmentation results. But again the use of unlabeled data only decreases the quality of the classifier.*

## 1. Introduction

A system that is able to segment text containing two languages could help linguists to better understand how bilingual speakers use the different languages. Especially children growing up in a bilingual environment tend to mix languages heavily and even create new composite words out of words from different languages. Analyzing the way that words from different languages are combined could help us understand the way that language is acquired and represented by children. When dictionaries are available the task simplifies to finding the word stem, querying both dictionaries and resolving ambiguities when a word can be part of both languages. Here we will focus on the situation where no dictionaries are available. We will train a classifier on an unlabeled training set of bilingual text and then evaluate the performance of predicting the language of words from this training set.

Unsupervised classification, or clustering, assumes that the training data is not labeled with class information. The goal is to create structure for the data by objectively partitioning the data into homogeneous groups where the within group object similarity and the between group object dis-

similarity are optimized. A lot of work has been done on language segmentation and classification. Here are a few examples of text segmentation tasks that at first occur to be similar to our problem, but then turn out to have very different solutions. When dealing with multi-lingual corpora it is often useful to determine the language used in a document. A few sentences are usually sufficient for this task ([1]), so that it is even possible to determine the language used in short paragraphs reliably. For document retrieval it is important to cluster text-segments containing similar concepts, rather than different languages and good solutions have been proposed ([4], [3]). On a lower level people are interested in splitting words into morphemes (morphemes are the smallest meaning-bearing elements of language). For agglutinative languages like Finnish and Turkish the unsupervised construction of lexica is infeasible due to the large number of different word forms and therefore morphemes have to be used [5].

The techniques used in these text segmentations tasks cannot be applied in our environment. We need to be able to determine the language not for a whole paragraph but for single words. In the language classification task evidence from a whole paragraph can be collected and it is known where a transition between languages can occur. In our case a transition can occur between each pair of words and getting the boundaries between languages exactly will be crucial. Therefore our task turns out to be closer related to work in completely different domains. When trying to find the secondary structure of a protein [2] or trying to determine the borders between coding and non coding DNA [10] a transition can occur after any element.

Note: The initial idea was to compare a clustering method working on the character level with a more sophisticated model working on the word level. The word level model was not implemented because the data set turned out to be smaller than expected, so that sparsity would have made parameter estimation very hard.

This paper is structured as follows: after describing the data set in the next section we will introduce a Naive Bayes classifier and the corresponding graphical model in section three. This classifier will be trained with Expectation Maximization in section four. Deterministic Annealing as an

alternative training method will be presented in section five. We will conclude and give an outlook to future work in section six.

## 2. The Data Set

The data set is a collection of emails from three different authors. All of the authors are bilingual (Bulgarian and German) and mix both languages when writing. While the main language is Bulgarian, single words, phrases or even whole sentences can be in German. Very often the German stem of a word is used with a Bulgarian ending. The level in which the languages are mixed is highly dependent on the author. It should be noted that everything is written with Latin characters (a-z). Therefore the German special characters (ß and the different "Umlaut", e.g. ü) were replaced by character pairs resembling their sound (ss, ue, etc.). Since Bulgarian uses the Cyrillic alphabet, Latin transliteration is used. Because there are more Cyrillic letters than Latin letters, many of the sounds that are specific to Slavic languages are transcribed with two or three Latin letters. Unfortunately some inconsistencies exist in the way the different authors transcribe these sounds, e.g. 'ш' is written as 'sh' or 'sch'.

The task is further complicated by the fact that there exist words with different meanings, but the with the same spelling in both languages. The correct classification of these ambiguous words can only be achieved when the context is taken into consideration. Our model relies solely on character statistics and therefore cannot be expected to classify such words correctly. Hence ambiguous words will be treated separately.

The bodies of the emails are transformed into a sequence of words by removing all punctuation marks and transforming all letters to lower case. The data set is then split into an unlabeled training set (95%), and labeled validation and test sets (each 2.5%).

## 3. Naive Bayes Classifier

Assume you were given a text containing words from two languages which you have never heard and that you were asked to separate the words into two languages. How would you solve the problem? A natural way to cluster words is to examine the characters that occur within the words and to group words that inhibit similar character frequencies. Since German, an Indogermanic language, and Bulgarian, a Slavic language, are fairly different languages, it is reasonable to expect such a method to do something useful. One might be able to increase the accuracy by using tuples of characters (bigrams, trigrams or even fourgrams) instead of just single characters (unigrams). Bigrams and trigrams represent sounds that contain more information about the language than just single characters. This is especially true for Bulgarian since many typical sounds are described by a

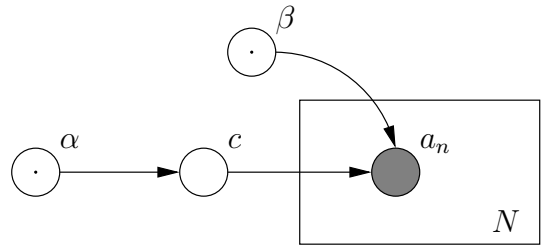


Figure 1: Graphical model representation of the character based classifier. Based on the observed characters  $a_n$  the parameters  $\alpha$  and  $\beta$  are to be estimated in order to determine the language class  $c$  for each word.

specific Cyrillic letter which has to be described by several Latin letters. However this approach has its limitations and close to perfect clustering certainly cannot be expected.

### 3.1 Graphical Model

Throughout this paper  $w = a_1 a_2 a_3 \dots a_N$  shall denote the word  $w$ , which has length  $N$  and consists of the characters  $a_n, n = 1 \dots N$ . The generative probabilistic model of a word is shown in figure 1. The basic idea is that a word is generated by picking a language  $c$ , a word length  $N$  and then generating characters (e.g.  $N$  can be chosen to be Poisson distributed, but since it is independent of the other variables its randomness can be ignored). We assume that there are only two languages classes ( $c$  can take on the values 0 and 1) and therefore  $c$  can be characterized by a Bernoulli distribution which is parameterized by  $\alpha$ . The character probabilities are parameterized by a  $2 \times A$  matrix  $\beta$ , where  $A$  is the size of the alphabet and  $\beta_{ij} = P(a_j = 1 | c_i = 1)$ . Our goal is to estimate  $\alpha$  and  $\beta$  so that we can compute the probability that the word  $w$  is from the language class  $c$ :

$$\begin{aligned} P(c|w, \alpha, \beta) &\propto P(c|\alpha) \cdot P(w|c, \beta) \\ &= P(c|\alpha) \cdot \prod_n P(a_n|c, \beta). \end{aligned}$$

Among other things, this assumes that each character is generated independently of the previous characters. This is certainly not true in general but is a useful simplification. It is relatively straight forward to extend this model to use n-grams instead of single characters (unigrams). We will be using n-grams up to  $n = 4$  in our experiments.

### 3.2. Smoothing

For n-gram models higher than unigrams sparsity becomes a problem. A new word can contain character combinations that were not seen in the training set. The maximum likelihood estimate for unseen n-grams and thus for the word will

be zero. There are two different approaches for handling unknown n-grams. One solution is to smooth the parameters, (e.g. Good-Turing Smoothing [7], which essentially is to take off some mass from the observed features and to reserve this mass for unseen features.

When clustering unlabeled data a different approach is often used: If the test data is available at the time of training, it is included into the training set (without the labels). If the test data is not available at the time of training, the classifier will be re-trained on the training set and the test data before classification. Since the test data is just a random subset of the training data and we are trying to find patterns in this data, it is reasonable to include the test data in the training step.

## 4. Expectation Maximization

Since there is no labeled training set  $P(a_n|c, \beta)$  cannot be computed directly. Instead Expectation Maximization (EM) [6] is applied in order to estimate  $P(a_n|c, \beta)$ . After initializing  $P(a_n|c, \beta)$  and  $P(c|\alpha)$  the following two steps are computed repeatedly:

**E-Step:** Use the current classifier,  $\hat{\beta}^{(k)}$  and  $\hat{\alpha}^{(k)}$ , to estimate the class membership  $y_l$  of each word, i.e., the probability that each class generated each word:

$$P(y_l^{(k+1)} = c_j | w_l, \hat{\alpha}^{(k)}, \hat{\beta}^{(k)}) = \frac{P(c_j | \hat{\alpha}^{(k)}) \cdot P(w_l | c_j, \hat{\beta}^{(k)})}{P(w_l | \hat{\beta}^{(k)})} = \frac{P(c_j | \hat{\alpha}^{(k)}) \cdot \prod_n^{|w_l|} P(a_n | c_j, \hat{\beta}^{(k)})}{\sum_{m=0}^1 P(c_m | \hat{\alpha}^{(k)}) \cdot \prod_n^{|w_l|} P(a_n | c_m, \hat{\beta}^{(k)})}.$$

**M-Step:** Re-estimate the classifier given the estimated class membership of each word. We first maximize our likelihood by assigning each word  $w_l$  the most likely label:

$$\hat{y}_l^{(k+1)} = \operatorname{argmax}_j P(y_l^{(k+1)} = c_j | w_l, \hat{\alpha}^{(k)}, \hat{\beta}^{(k)}).$$

Then  $\hat{\beta}_{ij}^{(k+1)}$  is re-estimated by counting up how often n-gram  $a_j$  occurs in words with label  $c_i$  and normalizing. The update of the class priors  $\hat{\alpha}^{(k+1)}$  follows analogously.

Iteration between E- and M-Steps will converge to a local maximum of the likelihood function. While these are the general steps of the EM-algorithm, there are slightly different ways of performing each of them. The following subsections will discuss these variations.

### 4.1. Initialization

When absolutely no labeled data is available for training there are two equivalent ways of initialization.

One way is to initialize  $P(c|\alpha)$  and  $P(a_n|c, \beta)$  uniformly (with some added noise). An alternative is to assign each word randomly to one of the classes and then estimate  $P(c|\alpha)$  and  $P(a_n|c, \beta)$  according to this assignment. The estimates for  $\beta$  will be very different but both will have in common that  $P(a_n|c = 0, \beta) \approx P(a_n|c = 1, \beta)$ . We decided to use the first initialization scheme.

Since we have a small validation set we can use it for a semi-supervised approach. Instead of initializing randomly we can use the M-Step of the outlined algorithm and estimate  $\alpha$  and  $\beta$  from the correctly labeled validation set.

### 4.2. Hard vs. Soft EM

The literature distinguishes between hard and soft EM. While hard EM assigns each data point to one cluster (as described above), soft EM uses fractional counts in the M-Step. Usually soft EM achieves slightly better results, but on the given training data there seemed to be no significant performance difference.

### 4.3. Results

The performance of the different classifiers will be evaluated on a hand labeled test set. The words in the test are labeled as either "Bulgarian", "German" or "Both". The words in the class "Both" are either ambiguous words that exist in both languages, composite words with a Bulgarian and a German part or proper names. Words from this class are not taken into account when the accuracy of the classifier is computed.

Note: Due to the limitations of the models a perfect classification cannot always be achieved, even in the most simplifying environment. If the Naive Bayes Classifier is trained (supervisedly) on the test set and evaluated on the same set the accuracy for unigrams is 86.3%, for bigrams 75.7% and above 98% for higher n-gram models. This is because short words in both languages inhibit similar n-gram counts and also because we are discarding the position of the n-grams within the word.

Since we do not know of any other work trying to segment bilingual text, we will compare our classifiers to a baseline where each word is labeled randomly. Since there are only two classes the performance of such a baseline will be 50%. The test set was generated in such a way that it contains equally many Bulgarian and German words (At first Bulgarian was with roughly 70% the more common language). This was necessary in order to ensure that the accuracy of labeling all words as one of the languages would be 50%.

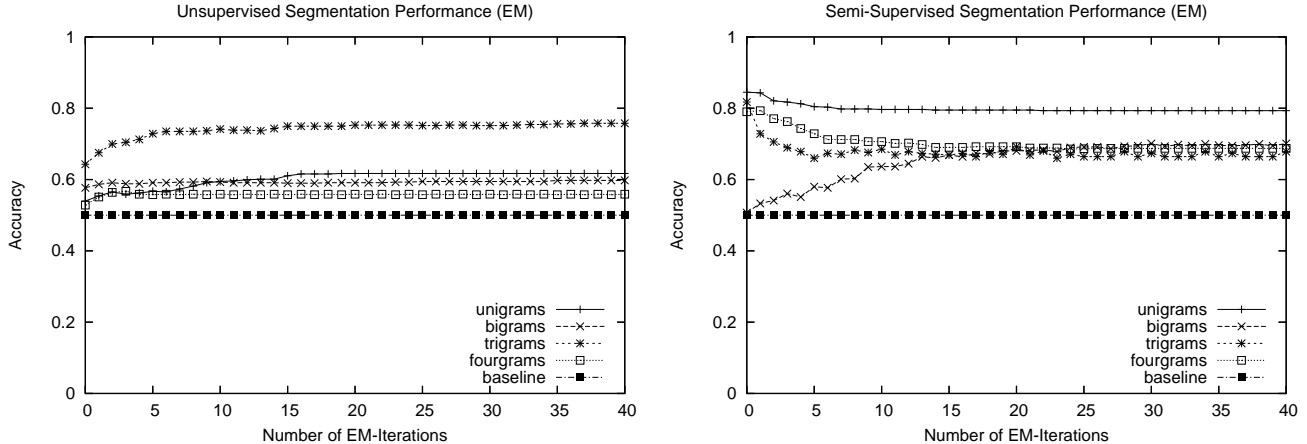


Figure 2: Classification results for the different n-gram based classifiers trained with Expectation Maximization.

Figure 2 shows how the classification performance evolves with each iteration of the EM algorithm. The performance of the unsupervised classifier turned out to be highly dependent on the random initialization. On average the trigram-model seems to perform consistently with an accuracy around 70% while the other models achieved rates of only 60%. The performance of the fourgram model is the lowest. This is a sign that we are starting to overfit the training data and therefore the accuracy on the test set drops. The amount of training data is too small for the reliable estimation of fourgram frequencies. It should be further noted that one of the clusters usually contains almost only German words (the ratio between German and Bulgarian words is ca. 9:1), while the other cluster contains the rest of the Bulgarian words and the short German words (ratio 6:4). The classifier is very good at picking out long German words, but for short words the n-gram counts for German and Bulgarian words are too similar.

The semi-supervised models find a better local maximum since their initialization is closer to the desired one. Unfortunately the accuracies drop with each EM iteration before converging (For some reason the supervised initialization of the bigram has a very low accuracy). These results are in accordance to B. Merialdo [8], who experimented with a probabilistic model to tag English text as part of speech. He was training the model with different amounts of labeled and unlabeled data and found that the accuracy "will generally degrade [...], except when only a limited amount of hand-tagged text is available".

## 5. Deterministic Annealing

Deterministic Annealing (DA) [9] is an algorithm that uses EM in an inner loop that iteratively increases a value  $\gamma$  (from  $\approx 0$  to 1), which in turn modifies the posterior com-

puted in the E-Step. While the E-Step of the EM algorithm computes  $P(c|w, \alpha, \beta) \propto P(c|\alpha) \cdot P(w|c, \beta)$ , the E-Step in the inner loop of the DA algorithm will compute:

$$P(c|w, \alpha, \beta) \propto P(c|\alpha)^\gamma \cdot P(w|c, \beta)^\gamma.$$

When we start with  $\gamma \approx 0$  the computed P will be close to a uniform distribution over the hidden variables; DA effectively ignores the current parameter estimates  $\alpha$  and  $\beta$ . For  $\gamma = 1$ , there is no difference between DA and EM. DA starts out by using EM to find the maximum of an easy concave function ( $\gamma \approx 0$ ). This solution is then used as an initialization for a slightly harder optimization problem. While DA guarantees an improvement on the objective function, it does not guarantee to find a global maximum or even a better maximum than EM will converge to. In our experiments we started out  $\gamma = 0.1$  ran EM until convergence and increased  $\gamma$  by a factor  $\delta = 1.5$  as long as  $\gamma \leq 1.0$ . Experiments with different  $\delta$  factors had similar results.

### 5.1. Skewed DA

One of the advantages of DA over EM is its insensitivity to the initialization of the parameters. In situations where a small amount of labeled data is available, this data can be used to initialize the parameters. Skewed DA makes use of this additional information.

$$P(c|w, \alpha, \beta) \propto P(c|\alpha)^\gamma \cdot P(w|c, \beta)^\gamma \cdot \hat{P}(c|w)^{1-\gamma},$$

where  $\hat{P}(c|w)$  is a posterior estimated from the labeled data. Essentially DA is interpolating in the log domain between two posterior distributions: the one given by  $\alpha, \beta$  and another one given by  $\hat{P}$ . When the interpolation coefficient  $\gamma$  is close to zero the E-Step will choose to be very close

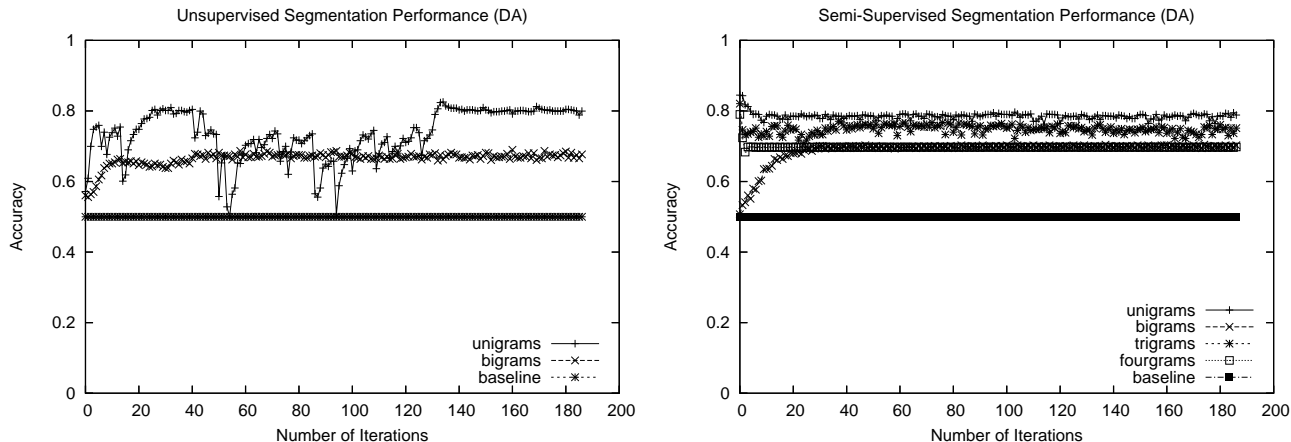


Figure 3: Classification results for the different n-gram based classifiers trained with Deterministic Annealing.

to  $\hat{P}$ . When  $\gamma$  comes closer to 1 the influence of  $\hat{P}$  will diminish and the algorithm becomes identical to EM.

## 5.2. Results

Figure 3 shows the accuracy of the classifier for different n-gram models on our test set. In the unsupervised case the DA algorithm is able to find a better local maximum than the EM algorithm and the accuracies increase to 70% - 80%. The fourgram model still has the lowest accuracy, confirming our interpretation that this model starts to overfit the data.

In the semi-supervised case the same effect as for EM can be observed: the accuracy of the supervised classifiers is decreased with each iteration. This is unsurprising since (skewed) DA is simply another search algorithm with the same criterion as EM. It improves the likelihood of the data but this does not always affect the segmentation performance positively.

DA appears to be more suitable for training the classifiers. It should also be kept in mind that this slightly higher performance comes at the cost of a significantly longer training time. Further even this increased accuracy is still not sufficient for practical applications.

## 6. Summary and Conclusions

We presented a generative n-gram based word model and two different algorithms for training a Naive Bayes classifier for the language segmentation task. While it was not hard to achieve first promising results, improving on them and building a system with almost perfect accuracy is very hard given the constraints. Although German and Bulgarian have a fairly disjunct vocabulary, segmenting words based solely on frequency counts of n-grams seems not feasible.

The best results were obtained when only the small set of labeled words were used for training. Neither EM nor DA were able to benefit from the much bigger set of unlabeled words. This indicates that while the used features (n-grams of different sizes) are useful for distinguishing languages, they are not sufficient for the unsupervised training of a classifier with EM or DA. EM and DA are search algorithms and maximize the likelihood of the data for a given model. Merialdos results [8] demonstrated that ML and maximizing accuracy are generally not the same; the EM algorithm consistently degraded the accuracy of his supervised models while increasing the likelihood of the data. These results lead to the conclusion that, when the goal is a system with high accuracy for two specified languages, it is more useful to focus the resources on generating labeled data that can be used for training, instead of developing better algorithms.

However, when the languages are not known in advance and from a scientific point of view, it would be very beneficial to have algorithms that can make use of unlabeled data. While the generation of labeled corpora is very expensive, large amounts of unlabeled data can be obtained easily. Concretely we would propose to first collect a much larger data set, in the order of several hundred thousand words (the current data set has 25,000 words). We would then build a model working on the word level, probably a (higher order) Hidden Markov Model. Such a model is capable of learning patterns in the way that languages are combined: it can learn that after certain words or word types a transition between languages is more likely to occur. However, the challenge would be to ensure that the model learns these transitions from unlabeled data. It is not completely clear how it can be enforced that, for a frequently used expression consisting of a German word and a Bulgarian word, the two words are put in different clusters, although they appear most of the time in a sequence.

## Acknowledgments

I would like to thank my friends Octavian, Kostas, Arnaud, Ben, Alexandra, Ryan, Pascal and many others for making my first semester in Berkeley so enjoyable.

## References

- [1] Kenneth R. Beesley. Language identifier: A computer program for automatic natural-language identification of on-line text. In *ATA'88*, pages 47–54, 1988.
- [2] Gill Bejerano, Yevgeny Seldin, Hanah Margalit, and Naftali Tishby. Markovian domain fingerprinting: statistical segmentation of protein sequences. *Bioinformatics*, 17(10):927–934, 2001.
- [3] D. Blei, A. Ng, and M. Jordan. Latent dirichlet allocation. In *NIPS'02*, 2002.
- [4] Marc Caillet et al. Unsupervised learning with term clustering for thematic segmentation of texts. In *RIAO'04*, pages 1–11, Avignon, France, 2004.
- [5] Mathias Creutz. Unsupervised segmentation of words using prior distributions of morph length and frequency. In *Proc. ACL'03*, pages 280–287, Sapporo, Japan, 2003.
- [6] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.
- [7] I. Good. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40:237–264, 1953.
- [8] Bernard Merialdo. Tagging english text with a probabilistic model. *Computational Linguistics*, 20(2):155–171, 1994.
- [9] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proceedings of the IEEE*, 86(11):2210–2239, 1998.
- [10] Yevgeny Seldin, Gill Bejerano, and Naftali Tishby. Unsupervised sequence segmentation by a mixture of switching variable memory Markov sources. In *ICML'01*, pages 513–520, 2001.