PROMODES: A probabilistic generative model for word decomposition

Sebastian Spiegler, Bruno Golénia, Peter Flach
Machine Learning Group, Computer Science Department, University of Bristol, UK
{spiegler,goleniab,flach}@cs.bris.ac.uk

Abstract

For the Morpho Challenge 2009 we present an algorithm for unsupervised morphological analysis called PROMODES[1] which is based on a probabilistic generative model. The model considers segment boundaries as hidden variables and includes probabilities for letter transitions within segments. PROMODES purely concentrates on segmenting words whereas its labeling method is simplistic. Morpheme labels are the segments themselves. The algorithm can be employed in different degrees of supervision. For the challenge, however, we demonstrate three unsupervised versions. The first one uses a simple segmenting algorithm on a small subset of the data which is based on letter succession probabilities in substrings and then estimates the model parameters using a maximum likelihood approach. The second version estimates its parameters through expectation maximization. Independently of the parameter estimation, we utilized each model to decompose words from the original language data. A third method is a committee of unsupervised learners where each learner corresponds to the second version, however, with different initializations of the expectation maximization. The solution is then found by majority vote which decides whether to segment in a word position or not. In this paper, we describe the details of the probabilistic model, how parameters are estimated and how the most likely decomposition of an input word is found. We have tested PROMODES on Arabic (vowelized and non-vowelized), English, Finnish, German and Turkish. All three methods achieved competitive results in the Morpho Challenge 2009.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: I.2.6 Learning; Parameter learning; I.2.7 Natural Language Processing; Language models

Keywords

word morphology, word decomposition, probabilistic generative model, expectation maximization, committee of unsupervised learners

1 Introduction

The goal of the Morpho Challenge is to advance algorithms for unsupervised morphological analysis. In general, *morphological analysis* is a subdiscipline of linguistics and can be defined as the study of the internal structure of words [3]. There are four tasks assigned to it: 1) decomposing words into morphemes, 2) building a morpheme dictionary, 3) defining morphosyntactical rules

[1] PROMODES stands for *PRObabilistic generative MOdel for different DEgrees of Supervision.*

which state how morphemes can be combined to valid words and 4) defining morphophonological rules which specify phonological changes when morphemes are combined [9]. Unsupervised means absence of supervision in *machine learning*. Most generally, machine learning refers to a computational program which improves its results with increasing experience in terms of data and intervention of an external teacher [13]. If there is no intervention of a teacher and the data does not contain any information towards the task to be solved, the learning process is referred to as *unsupervised*. For the Morpho Challenge, the task is to perform unsupervised morpheme analysis for words contained in a word list using a generic algorithm without any further information. Test languages are Arabic, English, German, Finnish and Turkish.

1.1 Related work

Over the recent years, a number of algorithms for unsupervised morphological analysis have been produced. Goldsmith [8] presents a morphological analyzer called *Linguistica* which learns signatures from a word list. *Signatures* are sets of suffixes which are used with certain sets of stems. They are built by initially splitting words into stem and affix candidates and then by heuristically changing stem-affix boundaries until the description length of the signatures is minimized. A similar approach has been chosen by Monson [14] who developed *Paramor*, an algorithm which learns paradigmatic structures of morphology as sets of mutually substitutable morphological operations. Another frequently cited morphological analyzer is *Morfessor* developed by Creutz et al. [4, 5] who implemented a model family for unsupervised morphology induction. The two main methods are *Morfessor baseline* and *Morfessor Categories-MAP*.[2] The first one is a recursive algorithm for morphological decomposition based on minimum description length (MDL) and the latter method is based on a probabilistic maximum a posteriori (MAP) framework and furthermore distinguishes between different categories of morphemes (prefixes, stems, suffixes). Linguistica, Paramor and Morfessor carry out morphological analysis in terms of word decomposition, learning a morpheme dictionary and finding morphosyntactical rules. Other approaches [2, 6] focus on word decomposition by analyzing words based on *transition probabilities* or *letter successor variety* which originates in Harris' approach [11]. Snover [15, 16] describes a *generative model* for unsupervised learning of morphology, however, it differs from ours in the fact that Snover searches, similar to Monson, for paradigms and we are interested in word decomposition based on the probability of having a boundary in a certain position and the resulting letter transition of morphemes.

The remainder of this paper is structured as follows. In Section 2 we will present our algorithm Promodes, its mathematical model, different ways of estimating its parameters and different methods for word decompositions. In Section 3 and 4 experiments are explained, results analysed and conclusions drawn.

2 Algorithm

In this section we will explain our algorithm Promodes. Its name stands for probabilistic generative model for different degrees of supervision. It can be applied in an unsupervised manner where model parameters are estimated using expectation maximization (EM) [7] or (semi-) supervised by computing maximum likelihood estimates (MLE) from a pre-segmented training set. For the Morpho Challenge, we used a simple unsupervised segmenting algorithm to segment words in the training set and then estimated the model parameters by MLE. Independently of the parameter estimation, we applied the generative model to the original data and decomposed all words. Apart from the two different ways of estimating the model parameters, we also carried out experiments with a *committee of unsupervised learners* where we combined η different results by majority vote.

In Figure 1, an overview over the algorithm is given. The probabilistic generative model which will be introduced in Section 2.1 supplies us with an *objective function* which assigns a probability to a single word analysis. In Section 2.2 two ways of estimating parameters will be explained. In Section 2.3 we will show how we used a committee of unsupervised learners to decompose words.

[2] Both morphological analyzers are the reference algorithms for the Morpho Challenge.

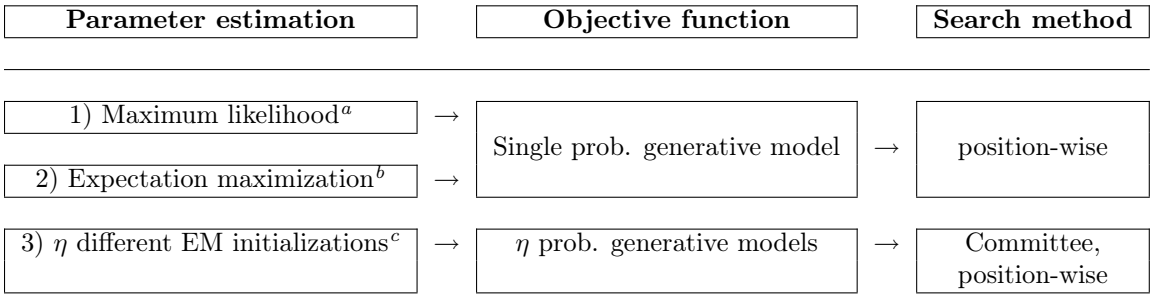

Figure 1: Overview over PROMODES algorithm

[a] PROMODES 1
[b] PROMODES 2
[c] PROMODES COMMITTEE

2.1 Probabilistic generative model

In a general way, one can say that a *probabilistic generative model (PGM)* is used to describe the process of data generation based on observed variables X and hidden or target variables Y with the goal of forming a conditional probability distribution $P(Y|X)$. In morphological analysis, we try to decompose words into morphemes. Therefore, the observables correspond to the original words and the hidden variables to their segmentation or boundaries. Knowing the parameters of the model we can find the best segmentation of a given word. However, we can also learn the parameters from a word list by using either maximum likelihood estimates or expectation maximization as described in Section 2.2.1 and 2.2.2. Subsequently, we will describe how the PGM has been constructed.

Words and segmentations The data we are dealing with is a list W of words w_j with $1 \leq j \leq |W|$. A word consists of n letters and has $m = n - 1$ positions where a boundary can be inserted. We denote a segmentation of a word with b which describes a binary vector $\langle b_1, \ldots, b_m \rangle$ with $1 \leq i \leq m$. In position i a boundary value b_i can be $\{0, 1\}$ for putting a boundary or not.

Letter transition probability distribution In the Markovian spirit we describe a word by transitions from a letter x to a letter y within a morpheme where y is drawn from an alphabet A and x from $A_{\mathcal{B}} = A \cup \mathcal{B}$ where $\mathcal{B}$ is a silent start symbol pointing to the first letter of the morpheme. By introducing such a silent start symbol it is guaranteed that different segmentations of a word have the same number of transitions.

$$p_{x,y} = Pr(X_{i+1} = y | X_i = x) \text{ with } \sum_{y \in A} p_{x,y} = 1 \; \forall x \in A_{\mathcal{B}} \text{ and } 1 \leq i \leq m. \tag{1}$$

Probability distribution over boundary and non-boundary positions A simple way of describing a segmentation is in a *position-independent and identically distributed* manner. However, assuming that boundaries are equally likely over the segmentation is a strong simplification. For this reason we chose a *position-dependent and non-identical distribution*. Each position i is therefore assigned to a Bernoulli random variable Z_i and the existence of a boundary corresponds to a single trial with positive outcome.

$$Pr(Z_i = 1|m) = p_{z_i,m} \text{ with } Pr(Z_i = 0|m) + Pr(Z_i = 1|m) = 1 \text{ and } 1 \leq i \leq m \tag{2}$$

with $Z_i \in Z = [Z_1, \ldots, Z_m]$ for the entire segmentation. Both above distributions describe the parameters of our generative model which can be summarized as

$$\theta = \{X, Z\} \tag{3}$$

with X being the probability distribution over transitions and Z the probability distribution over boundary and non-boundary positions in a segmentation.

Position-based perspective and probability of segmenting or not in position i Instead of looking for the best segmentation b_{jk} given a word w_j in an exponential search space with $1 \leq k \leq 2^m$, we make a decision in each position i and therefore turn the search into a linear problem. The probability of putting a boundary in position i or not is then defined as

$$Pr(b_i|m,\theta) = p_{b_i,m} \tag{4}$$

where $p_{b_i,m}$ is the probability of having a boundary value $b_i = \{0,1\}$ in position i given the length of the segmentation m. For later derivations we rewrite this equation as

$$Pr(b_i|m,\theta) = \prod_{r=0}^{1} {p_{r,m}}^{\mu_{r,b_i}} \tag{5}$$

$$\mu_{r,b_i} = \begin{cases} 1, & \text{if } b_i = r, \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where we iterate over possible boundary values $r = \{0,1\}$ and have a function $\mu_{r,i}$ which eliminates all r's in the product which do not correspond to b_i.

Probability of a letter transition in position i If the segmentation of the word is known we can assign a probability to its letter structure in terms of letter transitions (which are length-independent).

$$Pr(w_{ji}|b_i,\theta) = p_{x,y} \tag{7}$$

where in the jth word the letter x is in position i and y in $i+1$ given the information whether there is a boundary or not in position i expressed by b_i. For later derivations, we rewrite this equation such that we iterate over the alphabet using x' and y', and eliminate all probabilities which do not correspond to the original x and y by using the function $\mu_{xy,x'y'}$.

$$Pr(w_{ji}|b_i,\theta) = \prod_{y' \in A} {p_{x,y}}^{\mu_{xy,x'y'}} \;\; x' \in A_{\mathcal{B}} \tag{8}$$

$$\mu_{xy,x'y'} = \begin{cases} 1, & \text{if } x' = x \text{ and } y' = y \text{ in } w_j \text{ at } i\text{th position}, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Finding the best segmentation of a word is done as follows.

$$b_{max,i} = \begin{cases} 1, & \text{if } Pr(Z_i = 1|m) \cdot Pr(X_{i+1} = y|X_i = \mathcal{B}) > Pr(Z_i = 0|m) \cdot Pr(X_{i+1} = y|X_i = x) \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

$$b_{max} = \langle b_{max,1}, \ldots, b_{max,m} \rangle \tag{11}$$

If the product of the transition probability from the abstract start symbol $\mathcal{B}$ to letter $i+1$ and the probability for segmenting in i is higher than product of the transition from letter i to $i+1$ and not segmenting in i, segment, otherwise do not segment.

2.2 Parameter estimation

Before applying the probabilistic model described in the previous section, its parameters have to be estimated. For the Morpho Challenge we present two ways of doing this, at first by maximum

likelihood estimates and then by expectation maximization. A training set was prepared which constituted a subset of the original data. Approximately 100,000 word forms[3] were selected for each language which did not contain special characters like apostrophes or hyphens.

2.2.1 Maximum likelihood estimates

We segmented each training set using a heuristic similar to the *successor variety* [11] in a *separate* pre-processing step. All possible substrings of every word were collected in a forward trie[4] along with statistical information as their frequency. A particular word was then analysed from the first letter onwards where the conditional probability $P(l_{t+1}|l_1, \ldots, l_t)$ with l being any letter from the alphabet was evaluated. If the conditional probability dropped in $t+1$, the word was segmented in position t and the evaluation process was reinitiated in position $t+1$. Since this is a rather crude method, it tends to over-segment and solutions for similar words might vary a lot. From the segmentations we estimated the parameters for PROMODES 1 using maximum likelihood estimates.

2.2.2 Expectation maximization

Parameter estimation by expectation maximization (EM) [7] was used in PROMODES 2. The EM algorithm iteratively alternates between two distinctive steps, the expectation or E-step and the maximization or M-step, until a convergence criterion is met. In the E-step the log-likelihood of the current estimates for the model parameters are computed. In the M-step the parameters are updated such that the log-likelihood is maximized. First, we will describe how the likelihood is calculated and then we will show how the re-estimation is done using partial derivatives.

The Q function as the expected value of the log likelihood function is defined as follows:

$$Q(\theta, \theta_t) = \sum_{j=1}^{|W|} \sum_{i=1}^{m_j} \sum_{r=0}^{1} P(b_i = r|w_{ji}, \theta) \log P(w_{ji}, b_i = r|\theta_t) \tag{12}$$

$$\theta_t^* = \arg\max_{\theta_t} Q(\theta, \theta_t) \tag{13}$$

Our objective function $\mathcal{L}$ which we want to maximize during the *M-step* is built of the Q function from Equation 12 and includes constraints c_1 and c_2 and Lagrange multiplier λ_1 and λ_2:

$$c_1 = \sum_{y' \in A} p_{x',y'} = 1 \text{ with } x' \in A_{\mathcal{B}}, \text{ and } c_2 = \sum_{r'=0}^{1} p_{z=r',m} = 1 \tag{14}$$

$$\begin{aligned}
\mathcal{L} &= Q(\theta, \theta_t) - \lambda_1(c_1 - 1) - \lambda_2(c_2 - 1) \\
&= \sum_{j=1}^{|W|} \sum_{i=1}^{m_j} \sum_{r=0}^{1} P(b_i = r|w_{ji}, \theta) \cdot \left[\log \prod_{r=0}^{1} {p_{r,m}}^{\mu_{r,b_i}} + \log \prod_{y' \in A} {p_{x,y}}^{\mu_{xy,x'y'}} \right] \\
&\quad - \lambda_1 \left(\left[\sum_{y' \in A} p_{x,y} \right] - 1 \right) - \lambda_2 \left(\left[\sum_{r'=0}^{1} p_{z=r',m} \right] - 1 \right)
\end{aligned} \tag{15}$$

In order to re-estimate transition probabilities we use the partial derivative $\frac{\partial \mathcal{L}}{\partial p_{x,y}}$ and get the following result:

$$p_{x,y} = \frac{\sum_{j=1}^{|W|} \sum_{i=1}^{m_j} \sum_{r=0}^{1} \left(P(b_i = r|w_{ji}, \theta) \sum_{y' \in A} \mu_{xy,x'y'} \right)}{\sum_{y' \in A} \sum_{j=1}^{|W|} \sum_{i=1}^{m_j} \sum_{r=0}^{1} \left(P(b_i = r|w_{ji}, \theta) \sum_{y'' \in A} \mu_{x'y',x''y''} \right)} \tag{16}$$

[3] For Arabic we used the whole data set since there were less than 20,000 word forms given in the vowelized and non-vowelized data set.

[4] A trie is a tree structure where each node corresponds to a single letter. If a set of strings contain a common prefix, they hang off the same node and the path from the root to the last common node corresponds to the prefix.

For re-estimating the probabilities for segmenting in position i the partial derivative $\frac{\partial \mathcal{L}}{\partial p_{z_i,m}}$ has to be calculated:

$$p_{z_i,m} = \frac{\sum_{j=1}^{|W|} \sum_{i=1}^{m_j} \sum_{r=0}^{1} \left(P(b_i = r|w_{ji}, \theta) \sum_{r''=0}^{1} \mu_{r'',z_i} \right)}{\sum_{r'=0}^{1} \sum_{j=1}^{|W|} \sum_{i=1}^{m_j} \sum_{r=0}^{1} \left(P(b_i = r|w_{ji}, \theta) \sum_{r''=0}^{1} \mu_{r'',r'} \right)} \tag{17}$$

Although both derivatives are bulky, they have an intuitive interpretation. In Equation 16 we first count the occurrence of a certain letter transition from x to y multiplied by the probability $P(b_{ir}|w_{ji}, \theta_t)$ and divide it by the weighted sum of all transitions from x. In Equation 17 we calculate the weighted sum for putting a boundary in position i of words with length m_j and divide it by the weighted sum of all boundaries and non-boundaries in position i.

2.3 Committee of unsupervised learners

Since different initializations of the above described EM algorithm may converge in different local optima, the actual models might give slightly different analyses for a single word. Therefore it seems natural to average results from a set of different initializations and combine them to a single solution. Our third method for word decomposition PROMODES COMMITTEE does exactly this by using a *committee of unsupervised learners*. This approach is similar to Atwell's [1] in the Morpho Challenge 2006. In general, a *committee* is a group of individuals which have been appointed to perform a certain task or to arrive at a certain decision. In machine learning, a committee can combine results from different algorithms or in our case different initializations. Each member of the committee can vote for a certain partial or complete solution. The weight of each vote can be either uniform or non-uniform, e.g. based on the algorithm's performance or the confidence in the algorithm. The approach we are presenting is completely unsupervised and purely based on *majority vote* for putting a boundary in a certain position. Given η analyses for a single word w_j in position i we introduce $score_{j,i}$ as

$$score_{j,i} = \sum_{h=1}^{\eta} \pi_{h,j,i} \tag{18}$$

$$\pi_{h,j,i} = \begin{cases} 1, & \text{if analysis } h \text{ contains a boundary} \\ & \text{in position } i \text{ for word } w_j, \\ -1, & \text{otherwise.} \end{cases}$$

and put a boundary at the ith position of word w_j if $score_{j,i} > 0$.

3 Experimental results

For the Morpho Challenge 2009 we applied three methods, PROMODES 1, PROMODES 2 and PROMODES COMMITTEE, to five languages. Two of these languages, Turkish and Finnish, are *agglutinating languages* where words are composed mostly by joining morphemes, however, in certain situations morphemes also undergo phonological changes during the word building process. One of these phenomena is called *vowel harmony* which is defined as long-distance phonological interactions constraining what kind of vowels can co-occur in a word. The other three languages, namely German, English and Arabic, are rather *fusional languages* where morphemes are tightly fused together and are more difficult to separate. The PROMODES methods are intended for agglutinating languages. They decompose words into their morphemes. Morphosyntactic rules are implicitly stored as statistics in terms of probabilities for segmenting in certain word positions and resulting letter transitions within morphemes. There is no further grammatical analysis like building *signatures* or *paradigms*. Morpheme labels are either the morphemes themselves or simple labels consisting of *morpheme+index number* which we generated during a post-processing step. The results across all language are listed in Figure 2 and in Table 1 where the best results for precision, recall and F-measure are written in bold.

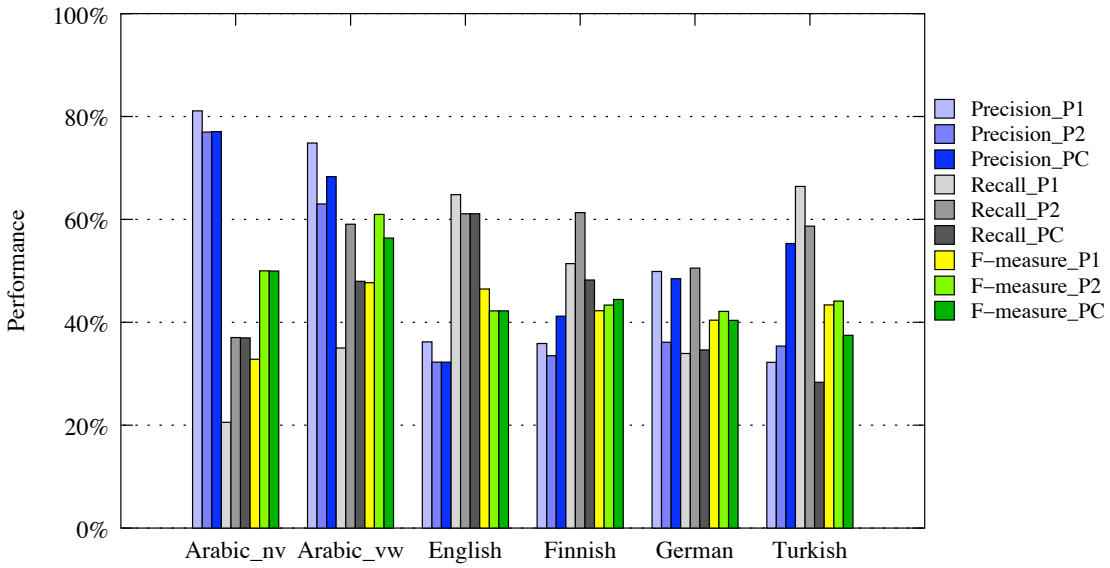

Figure 2: Results of PROMODES 1, PROMODES 2, PROMODES COMMITTEE in Competition 1

Language	Precision			Recall			F-measure		
	P1	P2	PC	P1	P2	PC	P1	P2	PC
Arabic (nv)	**.8110**	.7696	.7706	.2057	**.3702**	.3696	.3282	**.5000**	.4996
Arabic (vw)	**.7485**	.6300	.6832	.3500	**.5907**	.4797	.4770	**.6097**	.5636
English	**.3620**	.3224	.3224	**.6481**	.6110	.6110	**.4646**	.4221	.4221
Finnish	.3586	.3351	**.4120**	.5141	**.6132**	.4822	.4225	.4334	**.4444**
German	**.4988**	.3611	.4848	.3395	**.5052**	.3461	.4040	**.4212**	.4039
Turkish	.3222	.3536	**.5530**	**.6642**	.5870	.2835	.4339	**.4414**	.3748

Table 1: Results of PROMODES 1, PROMODES 2, PROMODES COMMITTEE in Competition 1

Language	Precision		Recall		F-measure	
	SA	CM	SA	CM	SA	CM
Arabic (nv)	.7113	.7800	.5387	.3576	.6131	.4904
Arabic (vw)	.4889	.6700	.8324	.3589	.6160	.4675
English	.0424	.0900	.9988	.9495	.0814	.1644
Finnish	.0489	.2700	.9997	.8767	.0932	.4128
German	.0280	.1100	.9989	.9111	.0545	.1963
Turkish	.1133	.3100	.9939	.8559	.2034	.4551

Table 2: Default *segment-all* (SA) and *one-common-morpheme-only* (CM) approach

Language	average # morpheme	average word length
Arabic (nv)	8.80	5.77
Arabic (vw)	8.75	9.90
English	2.25	8.70
Finnish	3.58	13.50
German	3.26	11.12
Turkish	3.63	10.80

Table 3: Language statistics

3.1 General setup of the experiments

For method PROMODES 1 (P1) we estimated the algorithm's parameters from a pre-segmented subset which was generated with a simple segmenting algorithm based on a forward trie that contained all possible n-grams of words. Boundaries were put in word positions where the conditional probability of seeing a certain letter after a substring suddenly dropped. This kind of pre-processing did not give precise morpheme boundaries for various reasons, mostly because it is a crude heuristic and it does not distinguish between common substrings from different locations in a word. By using maximum likelihood estimates we averaged the statistics across the subset. Subsequently, the model was applied to the entire data set to decompose all words.

PROMODES 2 (P2) was the version which used expectation maximization (EM) to estimate its parameters. Initially, words from a subset were randomly segmented and then the EM algorithm improved the parameter estimates until a convergence criterion was met. As convergence criterion we used the *Kullback-Leibler divergence* [12] which measures the difference between the model's probability distributions before and after each iteration of the EM. The resulting probabilistic model was then applied to the entire data set. Since the EM converges to local optima we restarted it with different initializations varying the rate of the random segmentation.

PROMODES COMMITTEE (PC) made use of the different initializations and the resulting analyses of PROMODES 2. Instead of having to choose a single result it combined different solutions into one by using *majority vote* for either segmenting or not in a certain position of a word. The idea was that when different initializations led to a word boundary in the same position, it should be placed and if they disagreed, the boundary can be omitted.

3.2 Analysis of results

Comparing our three methods to each other, we can say that P1 had the highest *precision* aside from Finnish and Turkish where PC came first. For the *recall*, P2 had the highest results except for English and Turkish where P1 had higher values. Looking at the F-measure as the harmonic mean of precision and recall, P2 had the highest performance for Arabic (non-vowelized, vowelized), German and Turkish. The highest performance for English was achieved by P1 and for Finnish by PC. Comparing our methods to other participant's methods in this year's challenge, we can state that P2 ranked first for vowelized and non-vowelized Arabic, and PC came third for Finnish. For the other languages, P1, P2 and PC were in top or mid ranks.

Since the analysis of precision and recall also depends on the distribution of the underlying data, we compared our methods to two default approaches shown in Table 2, segmenting words after each letter called the *segment-all* (SA) approach and only assigning a common morpheme to all words without further analysis called the *one-common-morpheme-only* (CM) approach. SA shows whether words which share a morpheme label in the gold standard also have letters in common. CM shows the different underlying distributions in terms of average number of morphemes in each language. For CM, we would expect a recall close to 100% if word pairs of the gold standard have on average a morpheme in common. For SA, we also expect a recall near 100% if words labeled with the same morpheme tag in the gold standard have a similar spelling. For the languages English, Finnish, German and Turkish the precision of all our methods were significantly above the SA and CM method. Arabic was a challenging language because it contained the most morpheme labels per word as shown in Table 3, independently from being vowelized (vw) or non-vowelized (nv). The winning method P1 still managed to have a higher precision, however, P2 and PC were slightly below the default solution. For the recall, SA and CM represented an upper bound. Since the recall has been calculated by looking at word pairs in the gold standard which share a common morpheme and should share a morpheme in the result file as well, CM would always return a correct answer because all words have a common morpheme. SA would return a correct answer if two words share at least a letter which is the case if they share a morpheme with the same or similar spelling. Expectedly, all our methods have recall values below the default solutions by SA and CM.

Looking at the overall performance expressed by the F-measure, results were significantly better than default for English, Finnish and German. For Turkish, the F-measure of our methods were slightly lower than the one of the CM approach. For vowelized and non-vowelized Arabic, our methods were also slightly below the SA approach. SA's higher F-measure has been caused by the number of morphemes per word which exceeds the number of letters. There is also a difference between the vowelized and the non-vowelized version where non-vowelized Arabic yields a lower F-measure. This can be explained by the fact that non-vowelized words are shorter and contain therefore less linguistical information.

4 Conclusions

We have presented different settings of an algorithm called Promodes which is based on a probabilistic generative model. Promodes has been developed from a machine learning perspective with little linguistical considerations and a simple labeling scheme where each morpheme has a unique morpheme label, either the morpheme itself or an index number. Two ways of estimating parameters from a subset of each language have been described, maximum likelihood estimates (Promodes 1) and expectation maximization (Promodes 2). Subsequently, we applied the two models to decompose an entire data set. As a third method we presented a committee of unsupervised learners (Promodes Committee) which combined different analyses of Promodes 2. All three methods achieved competitive results in the Morpho Challenge 2009 whereas Promodes 2 ranked first for vowelized and non-vowelized Arabic, and Promodes Committee came third for Finnish. For the other languages, our methods were in top or mid ranks.

In general, we believe that Promodes has the following strengths. The algorithm focuses on decomposing words and it predicts well how many morphemes two words have in common. The algorithm does not make assumptions about the structure of the language. It does not matter whether a language uses only suffixes or also prefixes. Furthermore, it can be applied to different degrees of supervision. In a (semi-) supervised or unsupervised setting it can be utilized to average over a small segmented set by learning its parameters from it. Another advantage is that instead of building a morpheme dictionary and morphosyntactic rules which are likely to be incomplete, it applies the statistics of a small training set to a larger test data set. Furthermore, the deployment of expectation maximization showed another way of estimating the model parameters. Doing so, we achieved similar or slightly better results than with the first method and we can choose a model, which meets our expectations the best, from different initializations. Another interesting method was the committee of unsupervised learners which combined results from different analyses.

Our future work includes investigating the behaviour of the committee in terms of number of members and how members should be best initialized. Moreover, we want to examine in greater detail what the impact of the training set size is and to continue in our ongoing optimization of the probabilistic model.

Acknowledgements

We would like to thank Aram Harrow for various fruitful discussions on the mathematical background of this paper, and our team colleagues Roger Tucker and Ksenia Shalonova for consulting us on general issues in morphological analysis. The work was sponsored by EPSRC grant EP/E010857/1 *Learning the morphology of complex synthetic languages.*

References

[1] Eric Atwell and Andrew Roberts. Combinatory hybrid elementary analysis of text (cheat). *Proceedings of the PASCAL Challenges Workshop on Unsupervised Segmentation of Words into Morphemes, Venice, Italy*, 2006.

[2] Delphine Bernhard. Simple morpheme labelling in unsupervised morpheme analysis. *Working notes for the CLEF 2007 Workshop, Budapest, Hungary*, 1:1–9, 2007.

[3] Geert Booij. *The Grammar of Words: An Introduction to Linguistic Morphology.* Oxford University Press, 2004.

[4] Mathias Creutz and Krista Lagus. Inducing the morphological lexicon of a natural language from unannotated text. *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR'05)*, 1:106–113, 2005.

[5] Mathias Creutz and Krista Lagus. Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3, 2007.

[6] Minh Thang Dang and Saad Choudri. Simple unsupervised morphology analysis algorithm (sumaa). *Proceedings of the PASCAL Challenges Workshop on Unsupervised Segmentation of Words into Morphemes, Venice, Italy*, 1, 2006.

[7] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the em algorithms. *Journal of the Royal Statistical Society*, 39:1–38, 1979.

[8] John Goldsmith. Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27:153–198, 2001.

[9] John Goldsmith. *The Handbook of Computational Linguistics, chapter Segmentation and morphology.* Blackwell, 2009.

[10] M.A. Hafer and S.F Weiss. Word segmentation by letter successor varieties. *Information Storage and Retrieval*, 10:371âĂŞ385, 1974.

[11] Zellig S. Harris. From phoneme to morpheme. *Language*, 31(2):190–222, 1955.

[12] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[13] Tom M. Mitchell. *Machine Learning.* McGraw-Hill Science/Engineering/Math, 1997.

[14] Christian Monson. *ParaMor: From Paradigm Structure To Natural Language Morphology Induction.* PhD thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 2008.

[15] Matthew G. Snover and Michael R. Brent. A probabilistic model for learning concatenative morphology. *Proceedings of Neural Information Processing Systems*, pages 1513–1520, 2002.

[16] Matthew G. Snover, Gaja E. Jarosz, and Michael R. Brent. Unsupervised learning of morphology using a novel directed search algorithm: Taking the first step. *Proceedings of the ACL-02 workshop on Morphological and phonological learning*, 6:11–20, 2002.