Multiple Sequence Alignment for Morphology Induction

Tzvetan Tchoukalov
Stanford University
P.O. Box 17469
Stanford, CA 94309
eurus@stanford.edu

Christian Monson and Brian Roark
Center for Spoken Language Understanding (CSLU)
OGI School of Science & Engineering
Oregon Health & Science University (OHSU)
20000 NW Walker Road
Beaverton, Oregon 97006
monsonc@ohsu.com, roark@cslu.ogi.edu

Abstract

MetaMorph is a novel application of multiple sequence alignment (MSA) to natural language morphology induction. Given a text corpus in any language, we sequentially align a subset of the words of the corpus to form an MSA using a probabilistic scoring scheme. We then segment the MSA to produce output analyses. We used this algorithm to compete in the 2009 Morpho Challenge. The F-measure of the analyses produced by MetaMorph are low for the full development corpus, but high for the corpus subsets used to generate the MSA, even surpassing the F-measure of another system used to aid MSA segmentation. This suggests that MSA is an effective algorithm for unsupervised morphology induction and may yet outperform the state-of-the-art morphology induction algorithms. Future research directions are discussed.

Categories and Subject Descriptors

I.2 [Artificial Intelligence]: I.2.7 NaturalLanguage Processing

General Terms

Algorithms, Experimentation, Languages

Key Words

Natural Language Morphology, Unsupervised Learning, Morphology Induction

1. Introduction

In this paper we describe the newly developed MetaMorph algorithm and report on its performance. The MetaMorph algorithm is an unsupervised natural language morphology induction algorithm utilizing multiple sequence alignments (MSA). MSAs are frequently implemented in biological sequence processing in order to capture information about long-distance dependencies and three-dimensional structures of protein or nucleotide sequences without resorting to polynomial complexity context-free models (Durbin et al., 1998). However, MSA techniques are rarely used in natural language processing. Furthermore, MSA has never been applied to natural language morphology induction to date. MetaMorph is thus a first attempt at utilizing multiple sequence analysis in this domain.

The task in the Morpho Challenge is to perform morphological analysis in an unsupervised fashion on a corpus of words from an arbitrary character based language. The morphological analysis of a word consists of segmentation into morphemes. Morphemes are word segments that in themselves are individual grammatical units, like roots, prefixes, suffixes, and infixes. Segmentation is done by labeling morpheme boundaries in a word and segmenting the word at those boundaries. We attempt to utilize the grouping of similar sequence elements, in our case characters, performed by MSA in order to group suffixes, prefixes, and other common morphemes so that morpheme boundaries may more easily be identified. Probabilistic models built from the MSA can then align any new sequence. If the MSA constructed from the language corpus is segmented, it can be used to perform morphological analysis.

We are unaware of any prior attempt to model morphological structure using MSA techniques. Broadening our view somewhat, MSA is at base a method for measuring distances between strings—and string edit distances *have* played a part in a variety of unsupervised morphology induction systems. Baroni et al. (2002), for example, seed a semantically based morphology induction system with pairs of words that are orthographically similar. Likewise, Wicentowski (2002) trains a statistical model of morphological structure from several weak correlates of morphological relatedness—including the Levinshtein distance between pairs of words.

While we do create MSAs, this application is fundamentally different from applications in biology. In biology, few sequences (on the order of 10) of very long length (on the order of millions) are typically aligned. In natural language processing, however, the sequences are words and are thus relatively short in length. However, there are many distinct word types and thus sequences to be aligned. A language corpus can range from 20,000 to 2,000,000 word types. Due to algorithm run time and memory constraints, not all of these word types are necessarily aligned into one MSA, but the alignments will contain significantly more sequences than those in biological applications.

2. Introduction to Multiple Sequence Alignment

MSAs are frequently used in biological sequence processing, where sets of evolutionarily related sequences are collectively aligned. The resulting alignment is in table format comprised of sequences containing gaps. Depending on the corpus of sequences being aligned, sequences are constructed from an alphabet. In the case of aligning RNA sequences, the alphabet is {A, C, T, G}, for example.

Simple statistical alignment models, typically called profile models, can be constructed for a MSA. Position specific score matrices (PSSM) can assign a cost to each of the symbols in the alphabet as well as to the gap symbol. Given a MSA, the PSSM values can be estimated as the negative logarithm of the probability of a symbol given a column. Typically Laplace's rule (add one observation per symbol per column) is used to smooth such distributions, along with relative frequency estimation. However, this model does not provide a mechanism to insert symbols between columns, which may be necessary. Such a mechanism is provided by a profile hidden Markov model and can be estimated from a given MSA, using relative frequency estimation and simple smoothing.

Dynamic programming can be implemented to efficiently achieve optimal alignment of a sequence with a PSSM or profile HMM. The dynamic programming table would have dimensions defined by the number of columns on one side and the length of the sequence on the other. The resulting alignment of a sequence of length N with a MSA which has M columns takes O(NM) in space and time.

In our application of MSA, we align large numbers of sequences obtained from a text corpus. The alphabet used is the set of characters encountered in the language corpus, which is typically the alphabet of the language and a few additional characters such as numbers or punctuation marks depending on the purity of the corpus. We implement a profile HMM in aligning sequences to generate a MSA. Figure 1 depicts a MSA. The tabular format of the MSA can be seen in the left portion of the figure. It is composed of 8 columns and 10 sequences. The table in the figure depicts the probability distributions of each of the columns of the MSA. A column distribution is composed of the MSA alphabet and a count for each of the characters of that alphabet and also for the gap

```
12345678
d-anc-es
d-anc-ed
d-anc-e-
d-ancing
r-unning
j-umping
j-ump-ed
j-ump-s-
j-ump---
laughing
```

Chars	1	2	3	4	5	6	7	8
a	1	2	5	1	1	1	5	1
c	1	1	1	1	5	1	1	1
d	5	1	1	1	1	1	1	3
e	1	1	1	1	1	1	1	1
g	1	1	1	2	1	1	1	5
h	1	1	1	1	2	1	1	1
i	1	1	1	1	1	5	1	1
j	5	1	1	1	1	1	1	1
l	2	1	1	1	1	1	1	1
m	1	1	1	5	1	1	1	1
n	1	1	1	6	2	1	5	1
p	1	1	1	1	5	1	1	1
r	2	1	1	1	1	1	1	1
s	1	1	1	1	1	1	2	2
u	1	1	7	1	1	1	1	1
gap	1	10	1	1	1	7	2	4

Figure 1: Example MSA with Column Distributions

character. These counts are the occurrences of the character in the column plus the Leplace smoothing constant of one. The probability of a character given a column is the ratio of the character count in that column's distribution to the distribution total. For example, the probability of the character 'd' given column 1 is the ratio 5/26; the cost for aligning the character 'd' to column 1 would therefore be the negative logarithm of this probability, or approximately 1.65.

3. The MetaMorph Algorithm

The MetaMorph algorithm implements progressive MSA for unsupervised natural language morphology induction. An ordered list of similar words with respect to Levinstein distance is obtained from the language corpus. Words from the ordered list are sequentially aligned to the probability distribution of the MSA using a linear gap scoring method. When alignment is complete for all words in the ordered list, multiple cycles of realignment are performed, where each word in the MSA is sequentially extracted from the alignment and aligned to the remaining alignment anew. These cycles are halted when the MSA remains unchanged or when, after an initial number of iterations, the MSA's end-of-cycle sum of column probability distribution entropy scores increases from the previous cycle's sum. At this phase, certain columns of the MSA are chosen to segment the alignment into chunks of columns. Each chunk represents a distinct morpheme of the word in each row of the MSA. The morphological analysis of the corpus is generated by first recording the words contained in the MSA and their respective analyses based on the alignment segmentation into word morpheme boundaries. Finally, the remaining words of the corpus are individually aligned to the MSA to produce their analyses based on the morpheme boundaries generated by the alignment's existing segmentation. Each of the phases of the MetaMorph algorithm are discussed in more detail in sections 3.1 through 3.4.

3.1. Step 1: Reordering

When generating a MSA, words are sequentially aligned to the MSA, which is initialized with a single word. To increase the morphological information stored within the MSA, words that are more similar with respect to Levinstein distance are selected earlier in the generation phase. We also wish to use only a small subset of the text corpus in generating the MSA. We therefore use a greedy heuristic to generate an ordered corpus subset consisting of similar words to later be aligned.

The basic steps in this heuristic are to, first, choose an initial pair of words to include in the subset, and second, iteratively add in one additional word until the subset of ordered words is of the desired size. We implement dynamic programming to generate the Levinstein distance between all pairs of words. The corpus of words that we use is sorted by word occurrence frequency in the source literature. In order to obtain an initial, locally closest pair of words to add to the subset we calculate pair-wise Levinstein distances for a small number of the corpus and choose the closest pair. The first 1000 words of the frequency sorted words of the corpus, constrained to be between 5 and 16 characters in length and containing no hyphens or numbers, constitute this set, in an attempt to target words with meaningful morphological structure. Because our corpus is sorted by word frequency with most frequent words appearing earlier in the corpus, the initial pair will in most cases contain properly spelled words. However, the algorithm will likely include misspellings of words in the corpus subset since the Levinstein distance is very small between a properly spelled word and its misspelling.

In order to complete the second step of the process we maintain a list of potential words from which to choose the next best word to add to the corpus subset. We impose a bound on its size to be the difference between the desired size of the corpus subset and the corpus subset's current size. Once a word is added to the corpus subset, we calculate the Levinstein distances between that word and each candidate word, or rather, each remaining word in the corpus that does not yet appear in the subset. If the list of potential words has not reached its size bound, a candidate word is added to the potential words list with its corresponding Levinstein distance value. If the list of potential words is full, a candidate word is only added if its Levinstein distance is less than that of the word in the potential words list with the greatest Levinstein distance, in which case that word is removed from the list. When all candidate words have been examined, the word in the list of potential words with the lowest Levinstein distance is removed and added to the corpus subset.

We note some strategies we use in our implementation of this heuristic. For the sake of algorithmic efficiency we implement two priority queues utilizing a heap-based structure to represent the list of potential words. Both priority queues contain the list of potential words but one is ordered in increasing Levinstein distance while the other in decreasing. Thus the best and worst words, or the words with the least and greatest Levinstein distance values, are easily accessible because the priority queues impose a sorting on the words. Adding words to the queues and removing the best and worst words takes logarithmic time. Since searching in the queues and removing any other words besides the best and worst takes linear time, we simply do not check to see if a candidate word appears in the list when considering it as a new candidate word, thus a candidate word may appear multiple times in this list. Similarly, we do not remove all occurrences of a word in this list when we add it to the corpus subset. Instead, if a word is returned from the priority queues as the next best word to add to the corpus subset and the word is already included in the subset, we simply ignore it and retrieve the next best. We also do not remove the best word from the priority queue we use to return the worst word, because this would take linear time. Similarly, we do not remove the worst word from the priority queue that we use to return the best word. In summary, if a word is returned by the queues that is already in the subset, we ignore that word and retrieve the next word. This results in substantial speedups of the algorithm.

Finally, the other technique we use to substantially reduce algorithm run time is if we are calculating Levinstein distance between two words to see if it is less than a certain value, at each step in the dynamic programming we determine if there is a possibility of the result being less than the value. If there is no such possibility, we terminate the dynamic programming early. Also, before we begin dynamic programming, we check to see if the difference in the size of the words will produce a value greater than our tolerance value and do not perform dynamic programming if it thus has no chance of returning a value less than the tolerance. This short-circuiting of the dynamic programming approach in determining Levinstein distance substantially decreases algorithm run time.

3.2. Step 2: Alignment

We next implement progressive alignment (Feng and Doolittle, 1987) to produce a MSA from the corpus subset we generated in the reordering phase above. Each word is considered as a sequence of characters. We implement a probabilistic scoring method with linear gap scoring, where gaps in a sequence are treated as characters in the word. The MSA is initialized to the first word in the sorted set where each character in the word is a new column of length one. Column distributions are created for each new column by initializing the counts for each alphabetical character to a predefined Leplace smoothing value and then incrementing the count of each character that appears in the column by one, and normalizing.

Then we proceed to sequentially align the remaining words in the corpus subset to the MSA. The cost of a character-to-column match or of inserting a gap in the new sequence is the negative logarithm of the probability of the specific character or gap in the column's distribution. The cost of inserting a column of gaps in the MSA is equivalent to the cost of matching a character to a column in which the character does not yet occur. Since the column probability distribution totals increase as more sequences are aligned to and inserted in the MSA, this cost increases dramatically with the size of the MSA. The score of an alignment of a sequence to the MSA is the additive combination of these character or gap to column matches or gap column insertions. This allows the use of dynamic programming with back-tracing to efficiently determine an optimal alignment, minimizing the alignment score. The modified sequence is then incorporated into the MSA and the MSA's column distributions are updated so the sequence's character counts are incremented in the respective MSA column distributions. When all words in the corpus subset have been inserted into the MSA in this fashion, the initial alignment cycle is complete.

Using a corpus of 500,000 Hungarian words paired with a gold standard morphological analysis, we ran the MetaMorph algorithm on various corpus subset sizes, column distribution smoothing values, and even implemented a modified scoring scheme that included entropy scores in determining alignment scores. We generated corpus subsets of 5,000, 10,000, and 20,000 words and used the MetaMorph algorithm to generate MSAs of these subsets using smoothing values of 1, 0.3, 0.1, 0.01, and 0.001 and also using an alternate scoring scheme besides the standard scoring scheme described above. We used this development set to establish working parameters.

The alternate scoring scheme attempts to take column probability distribution entropy into consideration when assigning a score to a character or gap to column match. In these two cases, the score of a match is calculated in the same manner as in the scoring scheme mentioned before, except that the column probability distribution entropy is subtracted from the cost. Thus a high entropy, corresponding to a uniform probability distribution, would result in a low cost for matching. We hoped this would influence the MSA in exhibiting a root region, or a region of columns with uniform probability distributions, resulting in high entropy values, which would contain the roots of the corpus subset words and thus allow the prefixes and suffixes to align more directly in the periphery of the MSA, resulting in low entropy values. Our main purpose in doing this was to better enhance the success of our implicit segmentation strategies, which we will discuss later.

This alternate scoring scheme resulted in identical MSAs as the original scoring scheme. Furthermore, varying the Leplace smoothing value used in the column probability distributions resulted in insignificant changes in the MSA. MSAs with various smoothing values would result in approximately the same MSA columns but with gap columns, or columns predominantly composed of the gap character, located in slightly different areas of the alignment. In all cases, gap columns were found uniformly scattered in the MSAs, thus the differences in the MSAs resulting from various smoothing values were insignificant. We thus used the standard scoring scheme with a smoothing value of one to produce our final results. The equivalence of the two scoring methods also suggests that the involvement of entropy in the scoring scheme was overshadowed by the magnitude of the character or gap costs in column probability distributions.

3.3. Step 3: Re-alignment

Once we have aligned all words in the subset to a MSA we perform iterative refinement (Gotoh, 1996). This is done by repeatedly performing leave-one-out realignment cycles until certain termination conditions are met. Each leave-one-out realignment cycle consists of removing each sequence of the MSA in turn, decrementing the sequence's character count in the respective column distributions, finding the best alignment of the word represented by the sequence, and reinserting this modified, currently optimal sequence back into the MSA.

At the end of each alignment cycle (both initial alignment and realignment cycles), the MSA is given a score comprising of the sum of the entropy of each column's distribution. Typically this score decreases with each cycle, but may increase in some initial realignment cycles. Therefore, the condition for terminating the execution of a subsequent realignment cycle is set to be an increase in total entropy score of the MSA after a predefined number of unhindered realignment cycles. The other condition for termination is if the MSA remains unchanged after a realignment cycle. Originally, this last condition was our only condition for terminating realignment cycles, but we observed that after many realignment cycles, the MSA occasionally converged to a cyclic re-occurrence of a number of alignment configurations. We therefore add the entropy condition.

3.4. Step 4: Segmentation

Having obtained a final MSA, we must now produce a morphological analysis of the language corpus. We had many ideas about how to do this, but settled on a simple baseline technique due to time constraints. We implement a greedy algorithm to iteratively generate segmentation schemes using a third-party (Monson et al., 2009) analysis provided by a different unsupervised morphology induction algorithm as a guide. This approach is expected to utilize the third-party analysis in identifying columns whose column boundaries likely mark morpheme boundaries. The approach also relies on the MSA alignment of sequences to properly segment a large portion of the corpus. Each segmentation scheme consists of a set of column boundaries at which we segment the MSA. The greedy algorithm is comprised of the following steps. We begin with an empty segmentation scheme, which imposes no morpheme boundaries, as the current optimal segmentation scheme. First, for each column in the MSA in turn, we produce a potential segmentation scheme that consists of this column together with those in the current optimal segmentation scheme. We then score the analysis induced on either the whole corpus or on the corpus subset against the third-party analysis. Second, we select the segmentation column which most improves the F-measure and add this column to the current optimal segmentation scheme. If no such column is found, then the algorithm terminates and the current optimal segmentation scheme is used to produce the final analysis. Third, if the process was not terminated in step two then we repeat this process from step one. The two potential scoring methods mentioned in step one produce two distinct segmentation schemes. If we score the analyses induced on the whole corpus by the potential segmentations, the segmentation scheme we receive from the algorithm is optimized for an analysis on the whole corpus. Similarly, the segmentation scheme resulting from scoring the analyses induced on the corpus subset by the potential segmentations is optimized for analyzing the corpus subset.

Once a segmentation strategy is given, words in the corpus subset are morphologically analyzed by imposing morpheme boundaries where the segmentation column boundaries in the segmentation scheme divide the sequences. Similarly, we align each word of the corpus not contained in the subset to the MSA and, as before, impose morpheme boundaries where the segmentation column boundaries in the segmentation scheme divide the word's sequence.

We developed segmentation strategies that only relied on the MSAs to define a segmentation scheme. These initial segmentation strategies involve segmenting at the left column boundaries of columns with the maximal gap probability in their distributions and columns with minimal probability distribution entropies. We defined four such segmentation strategies. Strategy one segmented at columns with locally maximum gap probability. Strategy two segmented at columns with globally maximum gap probability. Strategy three segmented at columns with locally minimum probability distribution entropy. Finally, strategy four segmented at columns with globally minimum probability distribution entropy. Figure 2 depicts the first six sequences of one of the MSAs of the Hungarian corpus, where sequence gaps are represented as dashes. The two lines of asterisks and ampersands represent the four segmentation strategies for this MSA. The asterisks and ampersand in the first of these two rows indicate a column with locally maximum gap probability and the ampersand indicates the column with globally maximal gap probability. The asterisks and ampersand in the second row indicate columns with locally minimum probability distribution entropy and the ampersand indicates the column with the globally minimum entropy. Sequences two through five are variations of the first sequence, which is a postposition, and are reasonably aligned. The final 'i' and 't' in the third sequence are distinct morphemes while the endings on the other variations of the first sequence are single morphemes. The last sequence is a verb and is linguistically

```
*   * *  *   * * * *   * * * * *  &  * * * *  *  *
*   * *   *  * * *  &  * * * * *  *  * * * *  *  *
-----k----ö---z-----ö-------t-----------t---------
-----k----ö---z-----ö-------t-----------t----i----
-----k----ö---z-----ö-------t-----------t----i-t--
-----k----ö---z-----ö-------t-----------t----e----
-----k----ö---z-----ö-------t-----------t----e-m--
-----k----ö---t-----ö-------t-----------t----e-m--
```

Figure 2: First Six Sequences of a Hungarian Corpus MSA with Indicated Implicit Segmentation Schemes

unrelated to the other sequences. As can be observed, it was found that strategies one and three always tended to be almost identical. Furthermore, these segmentation strategies exhibited no correlation with actual morpheme boundaries in the words. Because gap columns were uniformly spread out in the MSAs, segmentation strategies one and three over-segmented, producing analyses that would define morpheme boundaries between every character or every few characters of each word. While segmentation strategies two and four under-segmented, and were often different from each other, they would frequently define a morpheme boundary in the roots of words, as is the case above. It can therefore be seen that these segmentation strategies were not successful in producing meaningful morphological analyses. Due to the failure of these implicit segmentation strategies, we developed the two explicit segmentation strategies that depend on a third-party morphological analysis of the corpus data. We utilized the morphological analysis produced by this year's new unsupervised Paramor-Morfessor union morphology induction algorithm, cited in section 1.

4. Results

We ran the MetaMorph algorithm on a Hungarian corpus of 500,000 words with corpus subsets of 5,000, 10,000, and 20,000 words and a Leplace smoothing value of one. Our results suggest that there was a difference in F-measure between the two segmentation strategies. Both segmentation strategies produced potential segmentations yielding analyses scored against the third-party analysis to determine an optimal segmentation strategy. The difference between these two strategies is that the first scores full potential analyses while the second only scores the portion of the potential analyses corresponding to the corpus subset. Thus the first strategy finds the segmentation scheme that optimizes the analysis score over the whole corpus, while the second only over the corpus subset. We also observed that when using the first segmentation strategy, the corpus subset size influences F-measure. Besides this case, corpus subset size and F-measure seemed unrelated. Finally, there is a substantial difference between F-measures of the entire corpus analyses and the F-measures of only the corpus subset analyses against the gold standard.

We will refer to the first segmentation strategy mentioned in the previous paragraph as the full corpus analysis segmentation strategy, or just “full” in the figures, and the second segmentation strategy as the corpus subset analysis segmentation strategy, or just “subset” in the figures. The corpus subset analysis segmentation strategy is intended to optimize the morphological analysis of the words contained in the corpus subset while the full corpus analysis segmentation strategy is intended to optimize the morphological analysis of all words in the corpus. These predictions are observable in the actual data. As can be seen in figure 3, the morphological analysis produced using the full corpus analysis segmentation strategy resulted in a better F-measure than the analysis produced using the corpus subset analysis segmentation strategy, especially in the case where the corpus

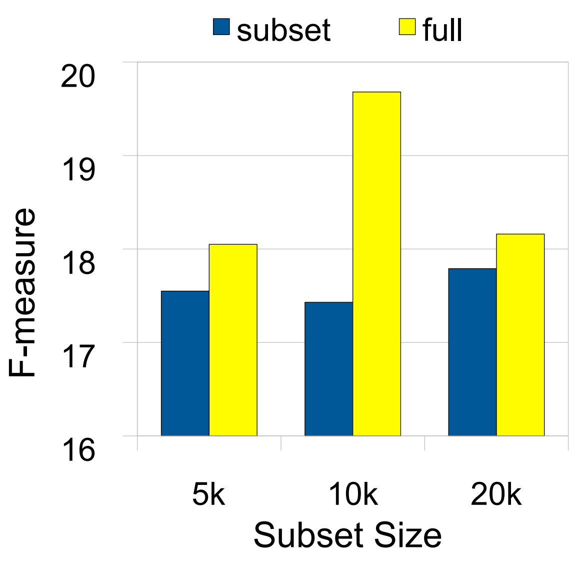

Figure 3: Full Corpus Analysis Using 3 Corpus Subset Sizes and 2 Segmentation Strategies

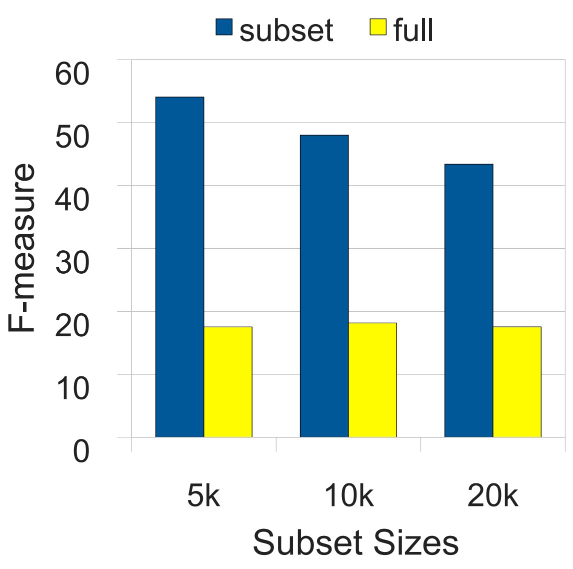

Figure 4: Corpus Subset Analysis Using 3 Corpus Subset Sizes and 2 Segmentation Strategies

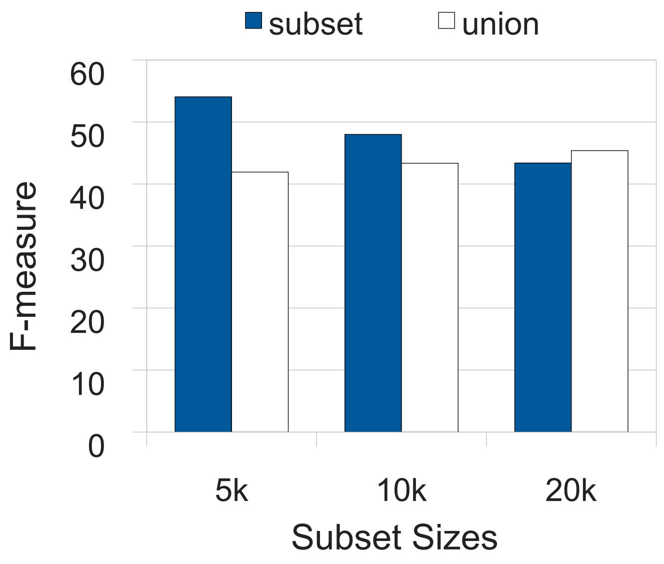

Figure 5: Corpus Subset Analysis Using 3 Corpus Subset Sizes and Corpus Subset Segmentation Analysis Vs. Paramor-Morfessor Union Analysis

subset size is 10,000 words. Analogously, as can be seen in figure 4, the morphological analyses produced on the corpus subsets alone using the two segmentation strategies also reflect the implemented strategy. The analysis produced using the corpus subset analysis segmentation strategy is substantially more accurate than that produced by the full corpus analysis segmentation strategy.

The final and more significant result, however, is that even though the corpus subset analysis segmentation scheme was produced by scoring morphological analyses of the corpus subset against the third-party analysis, the resulting morphological analysis of the corpus subset using MetaMorph greatly outperformed the third-party analysis on the corpus subset for the 5,000 and 10,000 corpus subset sizes. The degree to which the MetaMorph analysis on the corpus subsets outperforms the third-party analysis can be observed in figure 5. This final result suggests that the MetaMorph algorithm's analysis of words in the MSA is highly successful for smaller sets of words, while the analysis of words not included in the MSA is not as successful. These data suggest that MetaMorph is an effective method for aligning words in a MSA that reflects morpheme boundaries with significant accuracy and thus is an effective algorithm for morphology induction in principle, but fails to effectively utilize this MSA to accurately predict the morpheme boundaries of words in the language corpus not included in the corpus subset.

5. Conclusions and Future Directions

We conclude by discussing directions into which this research can be further extended. The two main directions we suggest include, first, modifying the existing MetaMorph algorithm to take advantage of its accurate analysis of smaller sized corpus subsets and, second, modifying the scoring scheme to define prefix, root, and suffix regions in the MSA.

In the first method, we would impose a partition of the corpus into equivalence classes or words that are mutually within a predefined Levinstein distance tolerance apart. We would then construct a MSA using each equivalence class as described in the MetaMorph algorithm. Finally, each MSA is segmented by using the corpus subset analysis segmentation strategy described in the MetaMorph algorithm. Alternatively, an implicit segmentation strategy may be discovered if the Levinstein distance tolerance used in creating the partition is adequately small. In that case, words may tend to be very similar, and perhaps column probability distribution entropy or column gap probability may successfully indicate segmentation columns correlating to morpheme boundaries.

The corpus subset analysis F-measure using the corpus subset analysis segmentation strategy was found to be high. Similarly, we anticipate that implicit segmentation strategies will also provide similar efficacy with an appropriately set Levinstein distance tolerance. These suggest that the analyses of the individual equivalence classes will also yield a large F-measure. Extrapolation of the results of figure 5 suggest that smaller corpus subset sizes yield larger F-measures. Therefore, by setting a low Levinstein distance tolerance we would expect the equivalence classes to be smaller in word number and the resulting analyses to thus yield a larger F-measure than that produced by the corpus subset of size 5,000.

The corpus may contain words that have a large Levinstein distance from almost all other words in the corpus. This may result in a number of equivalence classes in the partition with low or single word counts. Therefore, if the corpus subset analysis segmentation strategy is used, the analysis of these words will likely be identical or worse than the analysis of these words in the third-party analysis. However, the F-measure of the analysis of the words in the larger equivalence classes is expected to be much higher than the third-party analysis. This suggests that the overall F-measure will still be higher than that of the third-party analysis and thus this modified MetaMorph algorithm will outperform the third-party algorithm over the full corpus.

The second foreseeable alteration to the MetaMorph algorithm in hopes of improving its performance may be the modification of the scoring scheme. We would not focus on whether we use one or more MSAs as done above. Instead we would focus on redefining the construction algorithm of the MSA in hopes of better isolating prefixes and suffixes of words. By initializing the MSA to a certain size, we may allocate a certain number of the initial and final columns to be prefix and suffix areas and allocate the middle portion to be a root area. We may define two different scoring schemes to be used when aligning a sequence (word) to the existing MSA. In both schemes we would attempt to emphasize the role of column probability distribution entropy, which may be done by various approaches. We may do this by normalizing the negative logarithm of the probability of a character or gap in a column probability distribution to the potential range of entropy in order for this value to not overshadow the entropy's involvement in the cost function. We may altogether not assign a cost for inserting a gap into the sequence and matching this gap to a column. Or we may simply not consider a gap as a character at all and allow the column probability distribution sizes to vary from column to column depending on the number of gaps they contain.

In any case, our overall objective with the scoring schemes would be to maximize probability distribution entropies of columns in the root region and minimize those in the prefix and suffix regions. Thus the overall scoring approach is not the only one to consider. One of the two schemes is to be used if we are matching a character or gap to a column contained in the root region of the MSA. In this scoring scheme, it is intended that a high column entropy value desirably affect the cost of matching. Or rather, we may subtract the current column entropy from the column entropy with the character or gap inserted into the column. A positive result would indicate an increase in column entropy while a negative result would indicate a decrease in column entropy. Since we desire the root region of the alignment to contain characters more uniformly and thus be something of a mixing pot, a positive result would be a desirable cost, while a negative result would be an undesirable cost. The other scoring scheme is the reverse of the one discussed. We wish the probability distribution entropies of columns in the prefix and suffix regions to decrease, and thus specialize in one or a couple of characters, thus locking re-occurring prefixes and suffixes of words in these regions. Thus, by taking the difference in entropies as above, a positive result is undesirable while a negative result is desirable. The two scoring schemes just mentioned may be used alone or incorporated into the overall scoring schemes mentioned earlier. Otherwise, one of the overall scoring schemes may be used and modified to respectively take column probability distribution entropy into account when aligning sequences.

In summary, the performance of the current version of the MetaMorph algorithm is mediocre on complete language corpora. However, its results on the morphology induction of the Hungarian corpus subset suggest that the technique is very effective in principle. We suggest a couple of directions which can be followed to improve the MetaMorph algorithm.

Acknowledgments

This research was supported in part by NSF Grant #IIS-0811745 and DOD/NGIA grant #HM1582-08-1-0038. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF or DOD.

References

Baroni, Marco, Johannes Matiasek, and Harald Trost. 2002. Unsupervised Discovery of Morphologically Related Words Based on Orthographic and Semantic Similarity. *ACL Special Interest Group in Computational Phonology in Cooperation with the ACL Special Interest Group in Natural Language Learning (SIGPHON/SIGNLL)*, Philadelphia, Pennsylvania, 48-57.

Durbin, Richard, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. 1998. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, United Kingdom.

Feng, Da-Fei and Russell F. Doolittle. 1987. Progressive Sequence Alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351-360.

Gotoh, Osamu. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *Journal of Molecular Biology*, 264(4):823-38, 1996.

Monson, Christian, Kristy Hollingshead, and Brian Roark. 2009. Probabilistic ParaMor. *Working Notes for the CLEF 2009 Workshop*.

Wicentowski, Richard. 2002. *Modelling and Learning Multilingual Inflectional Mor-phology in a Minimally Supervised Framework*. Ph.D. Thesis, Baltimore, Maryland, The Johns Hopkins University.