UNSUPERVISED MORPHOLOGICAL ANALYSIS USING TRIES

Koray AK

B.S., Computer Engineering, Işık University, 2008

Submitted to the Graduate School of Science and Engineering
in partial fulfillment of the requirements for the degree of
Master of Science
in
Computer Engineering

Graduate Program in Computer Engineering
Işık University
2011

IŞIK UNIVERSITY

GRADUATE SCHOOL OF SCIENCE AND ENGINEERING

UNSUPERVISED MORPHOLOGICAL ANALYSIS USING TRIES

Koray AK

APPROVED BY:

Assist. Prof. Olcay Taner YILDIZ	Işık University	_____
(Thesis Supervisor)		
Assist. Prof. Taner ESKİL	Işık University	_____
Assist. Prof. Ümit GÜZ	Işık University	_____

DATE OF APPROVAL: 29/04/2011

Acknowledgements

I am honored to present my special thanks and deepest gratitude to my supervisor Assist. Prof. Olcay Taner YILDIZ for his guidance in this thesis. Without his endless patience and support I would not finish this work. I am feeling lucky to share his vision and knowledge throughout this thesis.

I am also grateful to my family for their support. They have always been by my side whenever I needed.

I have been partially funded by the Turkish Scientific and Technical Council (TÜBİTAK) BİDEB 2210 National Graduate Scholarship Programme.

UNSUPERVISED MORPHOLOGICAL ANALYSIS USING TRIES

Abstract

Morphological analysis or decomposition studies the structure, formation, function of words, identifies the morphemes (smallest meaning-bearing elements) of the language and attempts to formulate rules that model the language. It is widely used in different areas such as speech recognition, machine translation, information retrieval, text understanding, and statistical language modeling. Considering that the natural language processing applications are dealing with large amounts of data, it is not feasible to use linguists to analyze text corpus by hand, the complexity and real time processing requirements leads to automated morphological analysis. As an alternative to the hand-made systems, there exist algorithms that work unsupervised manner and autonomously do morphological analysis for the words in an unannotated text corpus.

In this thesis, an unsupervised learning algorithm is proposed to extract information about the text corpus and the model of the language. The proposed algorithm constructs a trie that consists of characters and the occurrences of the words as nodes. The algorithm then detects roots of the given words by examining the occurrences in the path of the word. When the root is revealed, the algorithm creates a new trie from the affix parts, left after the root for each word. The algorithm continues recursively until there is no affix left to process. Experimental results on three languages (Finnish, English and Turkish) show that our novel algorithm performs better than most of the previous algorithms in the field.

AĞAÇ YAPISI KULLANARAK GÖZETİMSİZ BİÇİMBİRİM ANALİZİ

Özet

Biçimbirim analizi ya da ayrıştırması, kelimelerin yapısını, dizilimini ve fonksiyonlarını inceler, kelimeler içindeki en küçük anlam taşıyan morfemleri belirler ve dilin modelini çıkarmaya çalışır. Konuşma işleme, bilgisayarlı çeviri, bilgi bulgetir, metin anlama ve istatiksel dil modelleme gibi alanlarda kullanılır. Biçimbirim analizi, metin içinde bir çok sözcük formu olduğundan çoğu dil için hem zor hem de gereklidir. Çekimli dillerde aynı köke ait binlerce değişik sözcük formu olabilir, bu da çekimlenmiş sözcük dizilerini oluşturmayı zor kılar. Doğal dil işleme uygulamalarının büyük verilerle çalıştığı düşünülürse bu işin dilbilimciler tarafından el ile yapılması karmaşıklık ve gerçek zamanlı işleme açısından mümkün değildir. Bu nedenle bu işlemin otomatikleşmiş biçimbirim algoritmaları tarafından yapılması gerekmektedir. Bu bağlamda öğreticisiz biçimbirim çözümleyicilerin kullanıldığı sistemlerle işlenmemiş metin bütünceleri işlenebilir.

Bu çalışmada metin bütünceleri ve dilin modeli hakkında bilgi çıkarımı yapacak bir gözetimsiz öğrenme algoritması önerilmiştir. Tasarlanan algoritma, metin bütüncesinde geçen kelimelerden oluşturduğu ağaçlar ile verilen kelimelerin kök ve eklerini kelimelerin geçme sıklığına göre bulmaya çalışmaktadır. Kelimelerin kökleri çıkarıldıktan sonra algoritma geri kalan sözcük kısımları ile ek ağaçları oluşturup özyineli bir şekilde tüm ekleri bulur. Algoritma Fince, İngilizce ve Türkçe dillerinde denenip önceki çalışmaların çoğundan iyi sonuçlar vermiştir.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

Morphological analysis is widely used in different areas such as speech recognition, machine translation, information retrieval, text understanding, and statistical language modeling. In many languages this task is both difficult and necessary, due to the large number of different word forms found in the text corpus. Highly inflecting languages may have thousands of different word forms of the same root, which makes the construction of a fixed lexicon hardly feasible. Also in compounding languages, word forms can be expressed in one single word. Considering that the natural language processing applications are dealing with large amounts of data, it is not feasible for linguists to analyze text corpus by hand, the complexity and real time processing requirements leads to automated morphological analysis.

Some morphological analyzers are designed for languages separately. But to use and maintain this kind of applications requires expert knowledge and new words should be supplied continuously whenever any change occurred in the language. As an alternative to the hand-made systems, there exist algorithms that work unsupervised manner and autonomously do morphological analysis for the words in unannotated text corpus. These algorithms are mostly using machine learning approaches.

Machine learning is a widely used methodology that allows computers to solve problems by evolving strategies from experience through example datasets [1] . Machine learning simply gathers information from the empirical training data about the

phenomena and makes intelligent decisions based its training. It also gives advantages when the problem depends on the particular environment, instead of writing multiple applications for different cases we can teach algorithm by using different training data and solve the problem in generalized manner. For example, in morphological analysis, languages that differ in morphological transformation rules can be analyzed with the same algorithm that is trained with the training data of each language.

In this thesis we will use unsupervised learning to gather information about the text corpus and the model of the language. In supervised learning input and output couple is given by a supervisor, the algorithm constructs a function that map each input to the desired output in the training data. It is obvious that training data can not cover all of the input-output possibilities. The algorithm should generalize the training data and map even unseen input to a possible correct output via set of assumptions. This set of assumptions is called inductive bias. On the other hand, in unsupervised learning there is no such supervisor and the algorithm only has input data. The algorithm models the input by examining patterns in the data and learns what to map in an input according to the relationship and properties in the data.

In this thesis we propose two unsupervised learning algorithms to extract information about the text corpus and the model of the language. The proposed algorithms construct tries that consist of characters and the occurrences of the words as nodes. In the first algorithm, roots of the given words are detected by examining the occurrences in the path of the word. When the root is revealed, the algorithm creates a new trie from the affix parts, left after the root for each word. The algorithm continues recursively until there is no affix left to process. Second algorithm constructs reverse tries to find affixes by examining the occurrences in the path and state the root word of each word in the last step.

This thesis is organized as follows: In Chapter 2, we give some basic concepts and background information about morphology. We also state detailed information about morphological approaches. In Chapter 3, we introduce morpho challenge and present previous work in the field. In Chapter 4, we describe proposed algorithms in detail. In Chapter 5, we give the experimental setup and results. In Chapter 6, we conclude the thesis and give the future work.

Chapter 2
Morphology

2.1. Morphology

In linguistic, morphological analysis or decomposition studies the structure of the words and identifies the morphemes (smallest meaning-bearing elements) of the language. Any word form can be expressed as a combination of morphemes. For instance, the English word "enumeration" can be decomposed as e+number+ate+ion and "interchangeable" as inter+change+able, and the Turkish word "isteyenlerle" as iste+yen+ler+le. Generally words are known as the basic units of the language but morphemes are the smallest syntactic unit and they reveal the relationship between word forms. In this respect, morphological analysis investigates the structure, formation and function of words, and attempts to formulate rules that model the languages.

There are two kinds of languages, polysynthetic and analytic (isolating) languages (Figure 2.1). In polysynthetic languages, words are mostly made up by connecting morphemes. Polysynthetic languages have two sub categories; fusional (inflectual) languages where morphemes are squeezed together and changed during the composition. English, French are good examples of the languages in this type. For example, the English word "sing" becomes "sang" or "sung" according to tense. Modern English is less fusional since languages tend to become less inflected over time due to the intercultural communications. Second type of polysynthetic languages is agglutinative languages in which morphemes do not change in the composition process. Turkish

(Altaic languages generally) and German are two examples of agglutinative languages where morpheme analysis of these type of languages are fairly easier than the fusional ones. Example for this category is the Turkish word "marmaradaki" is decomposed as marmara+da+ki. Analytic languages have stand alone morphemes that remain as independent words. Mandarin, Chinese are good examples of isolating languages. They use free morphemes which depend on tone and word order. Morphology studies all of these different languages and the relationship between them.

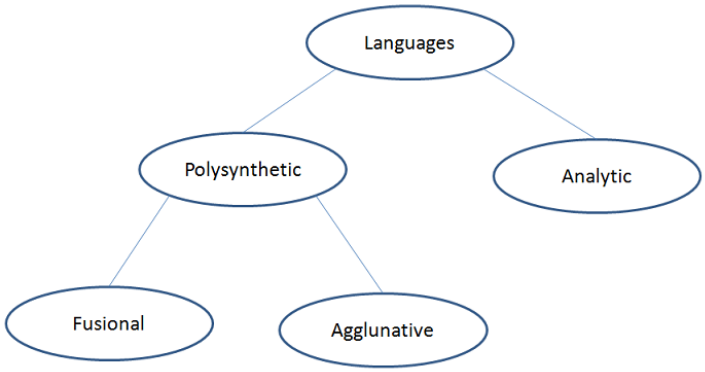

Figure 2.1. Language types

2.2. Lexemes and Word Forms

Morphological analysis basically investigates the word forms and finds out morphemes in a given word. Each word in the language may not refer a different object. For example, in English, words can change form with respect to subject-verb agreement and compound tense rules. For example, verb "try" changes to "tries", "tried", "trying" etc. according to the grammar rules of the sentence. These words mean the same but are presented as different word forms.

A lexeme is an abstract unit that corresponds to a set of word forms taken by a single word which is generally the one written in the dictionary. For example, the

English word “drive” is a lexeme. Lexemes either can change form by inflectional rules, or relate to other lexemes by derivational rules. “drive” becomes “drives” with respect to the tense or becomes “driver” and relate to another lexeme. In polysynthetic languages lexemes and word forms can be decomposed into smaller morphemes. Each word form has at least one root morpheme that expresses the main semantic content. There may also derivational or inflectional morphemes in the word that gives additional semantic or syntactic meanings to the word. Words that consist of root morphemes and derivational morphemes are called stem.

Compounding is another word formation type that combines two different word forms into a new word form. “Football” is an example of compounding where word formation is combined as foot+ball. Word forms that are involved in the compounding process may also have different affixes. In the decomposition of compounds, first two different lexemes are separated and then each lexeme is examined for affixes. In some languages affixes can precede the word called prefix, or can be added after the lexeme body called suffix.

2.3. Inflection and Word Formation

Although rules of constructing word forms are clear, it is not that simple to decide which rules to use during the decomposition process. For some examples even the linguists fail to agree whether a given rule is inflection or word formation. Word formation creates a new word form from two word forms where the grammatical category of the new word form differs from the grammatical category of the original word form. Inflection only changes the forms of the lexeme by adding suffixes with respect to subject verb agreement and tense rules i.e “sing” and “sang”. Moreover, word forms of a lexeme can be categorized into paradigms by conjugations of verbs, declensions of nouns and other inflectual categories. These inflectual categories should be suitable to the syntactic rules of the language. On the other hand, word formation is not restricted

by the syntactic rules of the language.

2.4. Allomorphy

Allomorph is a term for a inflectional change in the morpheme when the meaning stays same but word form is changed phonetically. Allomorphy is one of the complex conditions for morpheme analysis since it is not regular in each word of the language. For example, In English, plural affix -s changes according to last character of the word and become -es by adding one more vowel, on the other hand, some words like ox, goose, and sheep become oxen, geese, and sheep unlike the basic plural pattern. Another example is the change caused by the tense of the sentence. For the past tense in English language -d suffix become -ed according to words but irregular verbs such as go becomes went, gone. Most of the time, if this situation is not considered morphological rules omitted the phonological rules of the language and violate phonotactics.

2.5. Morphological Approaches

There are three principal approaches to morphology,

1. Morpheme-based morphology: Item-and-arrangement approach where words are constructed by adding morphemes repeatedly. Morphemes are smallest meaningful units. Each word is constructed as a collection of morphemes. It has three basics axioms:
    - Baudoin's single morpheme hypothesis: Roots and affixes have the same status in the theory, they are morphemes.
    - Bloomfield's sign base morpheme hypothesis: As morphemes, they are dualistic signs, since they have both (phonological) form and meaning.
    - Bloomfield's lexical morpheme hypothesis: The morphemes, affixes and roots alike, are stored in the lexicon.

2. Lexeme-based morphology: Item-and-process approach that analyzes word forms according to the relations in between. Any word form or stem is produced from another stem by applying a derivational rule and creating a new stem or inflectional rule and changes the word form or a compounding rule that compounds two stems and create a new word formation.
3. Word-based morphology: Word-and-paradigm approach that uses inflectional paradigms to determine the word decomposition. Word-based morphology is generally used for fusional languages. Words can be categorized based on the paradigm they suited. This applies both to existing words and to new ones. Word forms that does not suit any category can create new paradigms.

Generally, for agglutinative languages like Turkish, item and arrangement approach is the preferred one. Fusional languages are more suitable for working with item and process, and word and paradigm approaches. Considering the intercultural effect between languages it is more accurate not to classify approaches and languages one to one. Some languages can be mapped to multiple approaches and complex methodologies can be used to analyze those languages.

Chapter 3
Morpho Challenge

3.1. Morpho Challenge

Morpho challenge is a competition, part of the EU Network of Excellence PASCAL2 Challenge Program. It is organized by Mikko Kurimo, Krista Lagus, Sami Virpioja, and Ville Turunen from Adaptive Informatics Research Centre, Aalto University School of Science and Technology. The challenge was started in 2005 and is arranged in each year except 2006. The objective of the challenge is to design a statistical machine learning algorithm that discovers which morphemes words consist of. Ideally, these are basic vocabulary units suitable for different tasks, such as text understanding, machine translation, information retrieval, and statistical language modeling. The scientific goals are:

- To learn the phenomena underlying word construction in natural languages,
- To discover approaches suitable for a wide range of languages,
- To advance machine learning methodology.

The evaluation of the algorithms are based on the *F-measure*, which is the harmonic mean of *precision* and *recall*. These metrics are calculated by

- Hit (H): A valid cut that means word is cut at the right place.
- Insertion (I): A wrong cut that means word is cut at the wrong place.

- Deletion (D): A missed cut that means a valid cut is ignored.

Based on these possible cuts; precision is the number of hits divided by the sum of the number of hits and insertions, recall is the number of hits divided by the sum of the number of hits and deletions.

In the next part, we will give the brief summary of each algorithm proposed in the morpho challenge year by year.

3.1.1. Morpho Challenge 2005

Creutz and Lagus [2] propose an unsupervised morpheme analysis algorithm 'Morfessor'. The three tested versions of the Morfessor model described throughout the paper are Morfessor Baseline, Morfessor Categories-ML, and Morfessor Categories-MAP respectively. Morfessor Baseline is a context independent splitting algorithm based on minimum description length (MDL) method. The algorithm tries to maximize the product of lexicon probability (product of probability of each letter in the lexicon) and with the probability of each morph token in the corpus string of morphs. Here morphs are the segments found by the algorithm as morpheme candidates. Morfessor Categories-ML introduces morpho categories and corpus segmentations are modeled by Hidden Markov Model. Four categories (prefix, suffix, stem, and non-morpheme (noise)) are used to distinguish morphs. If a morpheme is preceding large number of morphs, it is a prefix. If a morpheme is following large number of morphs, then it is a suffix. If a morpheme is not very short it is likely to be a stem. Morfessor Categories-MAP is a maximum a posteriori algorithm an explicit probability is calculated for both lexicon and corpus. Submorphs are processed until a submorph found in the non-morpheme category, i.e., hierarchical word forms are taken into account. The first algorithm is entirely unsupervised and does not require any initialization parameters. Morfessor categories algorithms need perplexity value to be set for optimal solution.

Bernhard [3], segments words into labeled segments by identifying prefix, suffix, and stem boundaries with segment predictability. Method relies on transitional probabilities of each substring of the word in the lexicon, and distinguishes stems and affixes by examining the differences in lengths and frequencies of words. First the segmentation points are found by looking local minimas of transition probabilities. The segmented morphemes are then categorized according to their length and frequency properties. Longest and less frequent words are treated as stems and shorter and more frequent segments are separated as Suffix or Prefix with respect to their places upon stem. In the segmentation process, words that have the same stem are examined together to find best segmentation. Beyond the classic morpheme categories, Bernhard [3] used a linking element category to overcome non-morpheme segments in the word forms.

Bordag's approach [4], consists of two steps. It is based on letter successor variety method (LSV). The letter successor variety has been introduced as splitting words if the number of distinct letters after a given substring rises significantly or above a certain threshold. LSV method used in the paper is modified in two perspective, first weights are used to decrease noisy result, second related words found to calculate LSV value for each position within the word. Related words are found by looking the contextual similarity and edit distance. After calculating modified LSV values algorithm finds the possible segmentation points, in the next step, a trie-based classificator "Patricia compact tree" (PCT) is used to make generalizations. Segment boundaries are input two classificator with respect to their positions; a prefix classificator and a suffix classificator are trained and final segmentations are done by the results from PCT trees.

Keshava [5] use a simple approach to gather morphemes based on finding substring words and transitional probabilities. The algorithm constructs two trees; a forward tree where each node from top to the leaf corresponds a word in the corpus which words that are substring of other words and a backward tree to find suffix probabilities

easily. In the next step, lists of morphemes are obtained by scoring word fragments on the paths. This list contains all possible morphemes with their score. Morphemes that can be constructed with smaller morphemes are eliminated from the list. In the segmentation part, starting from the last letter each morpheme is splitted until there exist no more morphemes. If there are multiple ways of segmenting word, the one with the lowest transitional probability is used.

Johnsen [6], propose a novel method that assumes each stem has a meaning. The algorithm calculates $beta(a, b)$ density function to find segment boundaries. Here a represents the number of positive i.e. meaningful words if word is segmented from a boundary, and b is the number of negative cases of this segmentation. This study shows that the concept of meaning can be used in the evaluation of morphological analysis. All languages that has enough stem in their corpus permit to use of this approach.

Atwell *et al* [7], propose a majority voting algorithm to get the highest F-measure and Recall from the work done by other algorithms. The algorithm employs different unsupervised morpheme analysis algorithms and decides each segmentation by majority voting.

Swordfish [8] is an n-gram based algorithm for unsupervised morpheme analysis. First of all, the algorithm inputs word list instead of corpus, and calculates n-gram frequencies of each word. These n-grams are considered as the morpheme lexicon and by using this n-gram list, n-gram probabilities are calculated using the maximum likelihood approach. The probabilities are then used to decide on splitting points. The algorithm examines the word and calculates the probability of the term if it is in the n-gram lexicon, if the term can be split into two subterms with higher probability, subterms are chosen to segment word. These procedure runs recursively to segment each word.

SUMAA [9], like the approach in [4] is based on a modified LSV method. The algorithm examines predecessor variety as well as successor variety with this approach the words that can not be segmented via successor variety are easily segmented. This hybrid approach increase the efficiency of the LSV method. However using peak/plateu values of LSV is not giving all the morphemes instead the concatenation of the morphemes, to overcome this situation word that are substring of others split first and recursively put on LSV based algorithm. This modification increased the efficiency further.

Rehman and Hussain [10] basically focused on English. The algorithm assumes that any root morpheme is between 5 to 13 characters and any affix morpheme is at most 3 characters. The first and last 13 characters are potential segments. Each substring from these 13 character lists are examined whether they are in the word list and their frequencies are calculated. Same procedure is done for the trailing 13 characters and the frequencies of the same substrings added. The list of affixes are constructed with the frequencies and the weight of each affix is calculated. For segmentation, the algorithm first tries to split prefixes starting from first character to the last until a prefix found and both segments are valid according to the list. If both segments are valid and less than three characters the trailing part is the suffix. If it is more than three then algorithm starts splitting suffixes according to list until no segmentation found or the stem is less than 5 characters.

3.1.2. Morpho Challenge 2007

Bernhard [11] uses the same algorithm in the morpho challenge 2005 with slight differences. Only the representation of the results are changed since it is required to give the type of the morphemes in the challenge for 2007. Segmented morphemes are flagged as 'B' for stems or bases, 'P' for prefixes, 'S' for suffixes, and 'L' for linking elements. For example, the segmentation of sulking is sulk_B ing_S.

Bordag [12], propose a revised version of [4], i.e. the precision and recall constraints are taken into account. First of all, a compound splitter is added to the algorithm to detect and split compounds in order to gather more accurate LSV values. Also LSV method is employed more than once. According to the results, in each iteration recall of the algorithm is increased whereas precision is slightly decreased. Some modifications are done in the trie clustering part by adding thresholds. A morpheme signature is implemented to show morphemic analysis of the segmented morphs.

Mcnamee and Mayfield [13] propose a method based on n-grams, and 3, 4 or 5-gram with the lowest collection frequency morphemes are reported as the result of the segmentation.

Paramor [14], is another algorithm participated in the challenge for English and German languages. The algorithm examines words and extracts all possible candidate inflectional suffixes. These suffixes are used to construct partial paradigms. Partial paradigms are then merged according to the correlation between. The algorithm filters paradigms with respect to the number of words covered. After clustering paradigms, words are segmented according to the suffixes in the paradigms one by one, and the algorithm gives several results for every possible segmentation.

Zeman [15] proposed a paradigm based approach. All possible suffix-stem pairs are grouped into paradigms. A paradigm consists of a group of suffixes and stems that are suffixed by the suffixes in the paradigm. Since all possible segmentation points are considered, the number of paradigms are huge and need to be filtered. The paradigms that have more suffixes than stems, the paradigms whose border character is same in all suffixes, and the paradigms which have one suffix is filtered. Also the paradigms that are subset of a bigger paradigm is merged with its superset. The segmentation procedure is as follows; each possible segmentation of the word is examined and searched in the paradigms. If the stem and suffix pair are both in the paradigm, the segmentation

is ok. If both of the stem and suffix is not found in any of the paradigms, segmentation is done according to paradigms includes stem or suffix of the pair. If no stem or suffix is found in any paradigms then word is not segmented and algorithm outputs the word as itself.

3.1.3. Morpho Challenge 2008

Goodman [16] propose an algorithm based on linguistic productivity. In first stage, the algorithm finds seed affixes by segmenting each word from all possible partitions. Then affix set is cut down to a subset with respect to the root patterns it is applicable. The algorithm assumed that valid roots take a partially overlapping affix-set, and develops this sets into groups for both feature-set generation and binary clustering. After clustering these sets, the algorithm yields two sets of affixes; a set of possibly valid affixes and a set of spurious affixes. Membership of the affixes are updated one more time by considering quality of shared root distributions. Once affix list is acquired, stemming of word is started from the longest common affix.

Allomorfessor [17], is a modified version of morfessor [2] to solve problems caused by allomorphy. MAP is done by modeling a lexicon of the words in the corpus instead of modeling the original corpus. A new notion *mutation* is added to morfessor to solve allomorphic relations. The mutations are generally occurring in the end of the base form, same mutations should express common orthographical rules, and mutations should be efficient to compute from a pair of strings. The algorithm both inspects the cases with and wihout mutation. Words that has same prefix are examined to detect any allomorphy.

Paramor [18] is a revised version of [14]. Each word is examined by segmenting from every character boundary. When two or more corpus types end in the same word-final string, ParaMor constructs a paradigm seed. Paradigms are then expanded

to full candidate paradigms by adding additional suffixes. Algorithm merges candidate paradigms which likely model the same underlying paradigm of a language with clustering. Finally paradigms filtered to eliminate spurious ones. In the segmentation, word-final strings are searched across the suffixes in the paradigms. When a match found, algorithm controls whether the word is defined in the paradigm and segments word with respect to the boundaries stated in paradigm.

Zeman [19], (a revised version of [15]) propose methods to include prefix identification. Words are reversed to detect prefixes, using rules over all possible prefixes simply yields the prefix candidates. In the segmentation part, stems segmented with a prefix is not taken into account. Segmentations that consist of stem+suffix are examined and if stem has a match in the prefix list is segmented.

3.1.4. Morpho Challenge 2009

Bernhard [20] propose unsupervised morphological analysis relying on community detection algorithms on lexical networks. Unlike the other approaches, the algorithm uses morphological transformation rules to state the relationships between words. For each word w in the list L, the algorithm finds related words from the list by calculating the distance by Ratcliff-Obershelp (computes the similarity of two strings as the doubled number of matching characters divided by the total number of characters in the two strings) algorithm. Then extracts the rule for every word and match pair. Every rule that occurs at least twice is added to the rule list. After transformation rules are acquired, a lexical network is constructed. A lexical network can be represented as a graph $G\ (V, E)$ where V is a set of vertices and E is a set of edges. Each word w_1 in the list is a vertex and if there is a rule R such that $R(w_1) = w_2$ there exist and edge connecting w_1 to w_2. As a next the algorithm finds word families by employing modified Newman's community detection algorithm. The next phase is the morpheme analyses by examining the word families and the transformation rules. Each word in

the family is segmented by selecting the family representative as a root and applying the rules that are connecting the representative to the word. The representative of the family is the smallest word of the family, where there is a tie the most frequent one.

Can and Manandhar [21] propose an algorithm relying on syntactic categories of words. Syntactic categories are gathered by using part-of-speech (PoS) tags that are extracting from an unsupervised PoS tagging algorithm. Each word in the list is clustered with Clark's distributional clustering approach where its context is defined as the previous and next words of the processed word. The number of clusters K should be given beforehand; also there is a spare cluster contains unclustered words. In each step one word in the spare cluster which has minimum KL divergence with the K clusters is chosen to process. After each step clusters which have low KL divergence than the threshold are merged. After constructing syntactic categories, the conditional probability of each morpheme x given its PoS cluster $c, p(x|c)$, is computed. Morphemes are found by splitting each word from all splitting points in each PoS cluster. Found morphemes are ranked with respect to their maximum likelihood estimates. Ranking aids to removing low probability morphemes. Morphemes are chosen for merging if they have common stems in different PoS clusters. Words that are related with the merged morphemes are removed from their PoS clusters and assigned to new paradigms. Words are segmented as stated in their paradigm.

Spiegler *et al* [22], employs minimum description length (MDL) and graph-based unsupervised sequence segmentation (GBUSS) algorithms for finding segmentation points of a given word. It assumes that words are consisting of prefixes, stem and suffixes, where no limitation for number of prefix or suffix exist. In the first step of the algorithm, pseudo stems are found by a window based MDL method. In the initialization of the algorithm window is set as the middle character of the word and extended to right and/or left side according to the MDL window score until the best window is found. After detecting pseudo stems, the algorithm extracts prefixes from

the left side of the stem and suffixes from the right side by an extended version of GBUSS algorithm. The original version of GBUSS extracts the suffix sequences of a given corpus but in the algorithm prefix and suffix sequences are treated as morpheme sequences. The list of morphemes that are residing in the left part of the pseudo stem is called L-corpus and the part residing in the right side is called R-corpus. In an independent manner M-corpus is used to refer these morpheme corpuses in the graph. Position independent n-gram statistics are used to merge letters to morphemes until the stopping criterion is met. Morphemes are represented as nodes in the graph and each directed edge is the concatenation of two morphemes labeled with the frequencies in the M-corpus. In the initialization, each node is a character from M-corpus, then nodes are merged according to an evaluation function based on frequencies. After this step each segment prefixes, stem and suffixes are segmented from the word. The last step of the algorithm is aggregating these results.

Lavalle and Langlais [23] propose an approach based on the notion of formal analogy. The algorithm assumes that, related word pairs according to formal analogy can be segmented using the rules generated from gathered analogies. Paper gives definition of the formal analogy as, a relation between four items noted $[x : y = z : t]$ which reads as "x is to y as z is to t". With a morphological example [*cordially* : *cordial* = *appreciatively* : *appreciative*]. Since computation is very time consuming and not all the analogies are gathered in the paper, as a proof of concept, two families of systems are proposed. In the first family, two sub cofactor-based systems are explained; COF-GAPH and COF-FIRST. The sparsity of the processed analogies is surpassed by using c-rules as cofactors. C-rules are extracted from analogies, to make generalization over the lexicon. The low frequency ones are filtered and to reduce erroneous c-rules, each c-rule R is scored with respect to its productivity, P defined as the ratio $P(R) = V/A$ of the number of time it's application leads to a valid result V over the number of times it could be applied A.Words are placed as nodes into the Word-relation Trees and connected by c-rules as edges from child node to its father so that morphological

complexity is increasing as going down the tree. Placing nodes into the three is depends on an $S(I)$ value which is differs in COF-FIRST and COF-GRAPH systems. Basically a node is connected to a node if $S(I)$ value is more than a threshold. In COF-FIRST $S(I)$ is the productivity of the rule that connects nodes each other, and there can be only one connection from one node to another. In COF-GRAPH there is no limitation and one node can be connected the other with multiple c-rules so the value of $S(I)$ from one not the another is the sum of the productivity of each c-rule that connects nodes. Extraction of morphemes from tree is as follows, if the word is a root node then whole word is the output, if word is somewhere in the middle of the tree than segmentation is union of the morphemes of its father and the morpheme extracted with the c-rule with the highest count that connects node to its father. Second family of systems proposed by the paper is pure analogical systems; ana-seg and ana-pair. These systems use directly analogies and decide segmentation points of the words. Ana-seg calculates the factorizations of words in analogies and selects the one with highest frequency. For example word abolishing is found in 21 different analogies which yields 6 different factorizations. Ana-pair is more complex. In this variant, the analogies are checked for related words. Algorithm checks whether the first and second words or the first and the third one are related in the analogies. Two related pairs are identified by comparing the factors of each word. Then for each pair of words (a, b), the "morpheme" $a + b$ is added in the entries.

Lignos *et al* [24], propose a base and transforms model for morpheme analysis. Three set of words defined in the paper; base set, unmodeled set, and derived set. In the beginning, all words are in unmodeled set and affixes count. Each processed word then put into base set if they are not derived from any other word and affix. Algorithm constructs transforms from base set and derived set relationships. In the learning process, algorithm eliminates all words including a hyphen, but uses the hyphenated segments. Affixes are ranked with respect to their length and the number of types they occurred. Also the words with low frequency are filtered against foreign words. In

the next step, transforms are ranked with respect to the base-derived word count that they occurred and the number of characters they add to the base word. For example, if (s, ing) is occurred in 50 pairs the score would be $50 * (3 - 1) = 100$. Transforms that has segmentation precision less than %1 of the corpus are rejected. Segmentation precision is determined by examining each word from unmodeled set has affix S_2 is exist in the corpus after segmenting the affix. After learning model is stopped, paper deals with the compound words that are in the base or unmodeled set. 4-gram model is used to segment a word with respect to the largest drop in forward transitional probabilities between characters of a word, and if the substrings of that split point are words observed in the corpus, the word is splitted. This process is done recursively to segment the substrings. After all, there are three set of words base, derived and unmodeled and transformations between the words forms that forms set. Words are segmented according to the transforms until a word found in the base set.

Monson *et al* [25] propose an improved version of paramor [14] and [18]. In the previous papers, paramor does not assign a numeric score to its segmentation decisions. A natural language tagger is trained to score each character boundary in each word. Using paramor as a source of labeled data, finite-stage tagger is trained to identify, for each character, c, in a given word, whether or not paramor would place a morpheme boundary immediately before c. The probabilities calculated by the mimic tagger is used to maximize f-measure by adjusting precision and recall values. Threshold for probabilities calculated by the tagger is set to 0.5, the number of probabilities over this threshold found in a word is simply the number of morpheme boundaries proposed by mimic tagger. To adjust recall and precision, a positive factor α is used to decrease or increase the number of morpheme boundaries proposed by the tagger. Then a segmentation score is calculated by consulting a list of sorted probabilities that lie in αk probabilities where k is the number of probabilities over the threshold 0.5. Word is segmented with respect to a segmentation score threshold S. Like [18], [2] is used along with the paramor and propose a single analysis.

Spiegler *et al* [26] present Promodes; a probabilistic generative model approach for unsupervised morphological analysis. The model considers segment boundaries as hidden variables and includes probabilities for letter transitions within segments. Three unsupervised versions is suggested. The first one uses a simple segmenting algorithm 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. A third method is a committee of unsupervised learners where each learner corresponds to different initializations of the expectation maximization method. Then a majority vote algorithm is employed which decides where to segment word.

Tchoukalov *et al* propose a Metamorph algorithm [27], utilizing multiple sequence alignments (MSA). First an ordered list of similar words with respect to Levinstein distance is obtained from the language corpus. Then using a linear gap scoring method words from the list are alignd to the probability distribution of MSA. Then alignment procedure is repeated until MSA remains unchanged or when MSA's end-of-cycle sum of column probability distribution entropy scores increases from the previous cycle's sum. After the realignment finished, some columns of MSA are chosen to segment the alignment where each chunk represents a distinct morpheme of the word in each row. In this step, the segmentation of the words that are in the MSA is done with respect to alignment segmentation. 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.

Virpioja and Kohonen [28] propose a revised version of [17] in morpho challenge 2009. The algorithm present in the paper is almost same with the previous version but Viterbi segmentation is used in training. Unlikely the previous version, the results achieved in this version is closer the Morfessor Baseline [2].

Chapter 4
Proposed Approach

After investigating previous work and the problem, we suggested 2 different approaches for unsupervised morphological analysis. First of all we analyzed the data provided by the Morpho Challenge. The datasets supplied by the challenge includes a wordlist with word frequencies (See Figure 4.1 for an example) and a corpus gathered from different sources. Since character encodings differ in the datasets, some modifications are done. English dataset consists of standard text and all words are lower-cased. Finnish dataset uses ISO Latin 1 (ISO8859-1). The Scandinavian special letters å, ä, ö are rendered as one-byte characters. Turkish dataset is standard text and all words except the letters specific to Turkish are lower-cased. The letters specific to the Turkish language are replaced by capital letters of the standard Latin alphabet, e.g., "açıkgörüşlülügünü" is converted to "aCIkgOrUSlUlUGUnU".

As the first step of our approach, we use the wordlist to construct trie (See Figure 4.2). Since the number of the characters differs in each language, the number of child nodes is set dynamically with respect to the language. Non-alphabet characters such as hypen is omitted in the procedure. As we state in previous chapters, one of the problems in unsupervised morphological analysis is the data sparsity. We observed that given enough large dataset, most of the root words appear in the corpus. For example, in the datasets of the challenge, there exist 15545 root words among 617298 words where total root count for Turkish is 23470 [29], that is %66 of the roots appeared

339 fakir
1 fakirden
4 fakirdi
11 fakirdir
4 fakirdirler
11 fakire
2 fakirhanesine
9 fakiri
2 fakirleSeceGine
4 fakirleSen

Figure 4.1. Sample wordlist from Turkish dataset, words are preceeding with the occurrences in the corpus.

in the dataset. Given a larger set root extraction would be more efficient for our first approach.

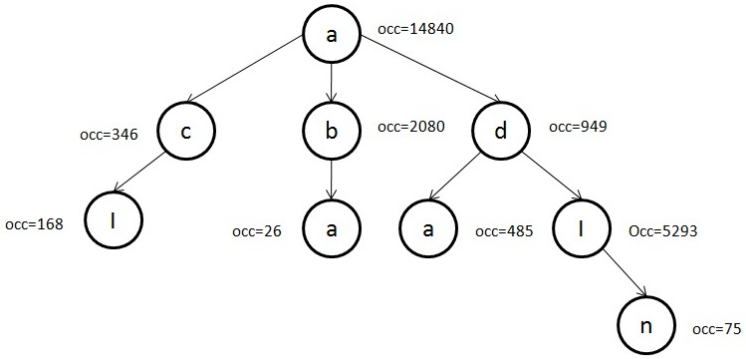

Figure 4.2. Part of a Trie constructed with Turkish dataset.

The pseudo code of our proposed algorithm (REC-TRIE) is given in Figure 4.3. We simply populate word trie with words from the list that are occurred more than 5 times and store the corresponding character, the number of occurrence in the corpus and the number of times that character is used in this path in this depth (Line 3). With the fact explained above, we assume the smallest most occurred word in a path is the root of the words in the path.

```
1  Read words from Wordlist W
2  i = 1
3  Construct word Trie_i
4  Construct word Table
5  Do until no unsegmented words remain
6      For each word in the Table
7          Find boundary for unsegmented part with Trie_i and update Table
8          If the word is not fully segmented
9              Add unsegmented part to Trie_{i+1}
10         End If
11     End For
12 i = i + 1
13 End Do
```

Figure 4.3. The Pseudocode of REC-TRIE.

Once the algorithm finds root morphemes in each word, it saves the corresponding segmentation into a table and continues to the next iteration (Line 7). In each iteration, the rest part of the word is put on a new trie (Line 9) and affix boundaries are found recursively with the same method applied for root extraction. The algorithm continues until no affix candidate is left.

In Figure 4.4 a sample run of REC-TRIE is presented. After initializing word trie the algorithm traverses each path and chooses the most occurred smallest word as root. The word "ada" is segmented as ad+a since the most occurred character is d with 944 occurrence in the path. However the words "adI", "adIn", "adIna", "adInI", and "adInIz" are segmented from adI since the most occurred character in these paths are I with 5293 occurrence (b). Next REC-TRIE constructs affix tries to find affix boundaries. In (c), affixes are inserted in a new trie. Note that "a" is merged from

ab+a and ad+a and occurrence is summed. Again first affixes are found by selecting the most occurred character. This procedure continues until there is no affix candidate left. The final affix is found in (d) and REC-TRIE finish segmentation.

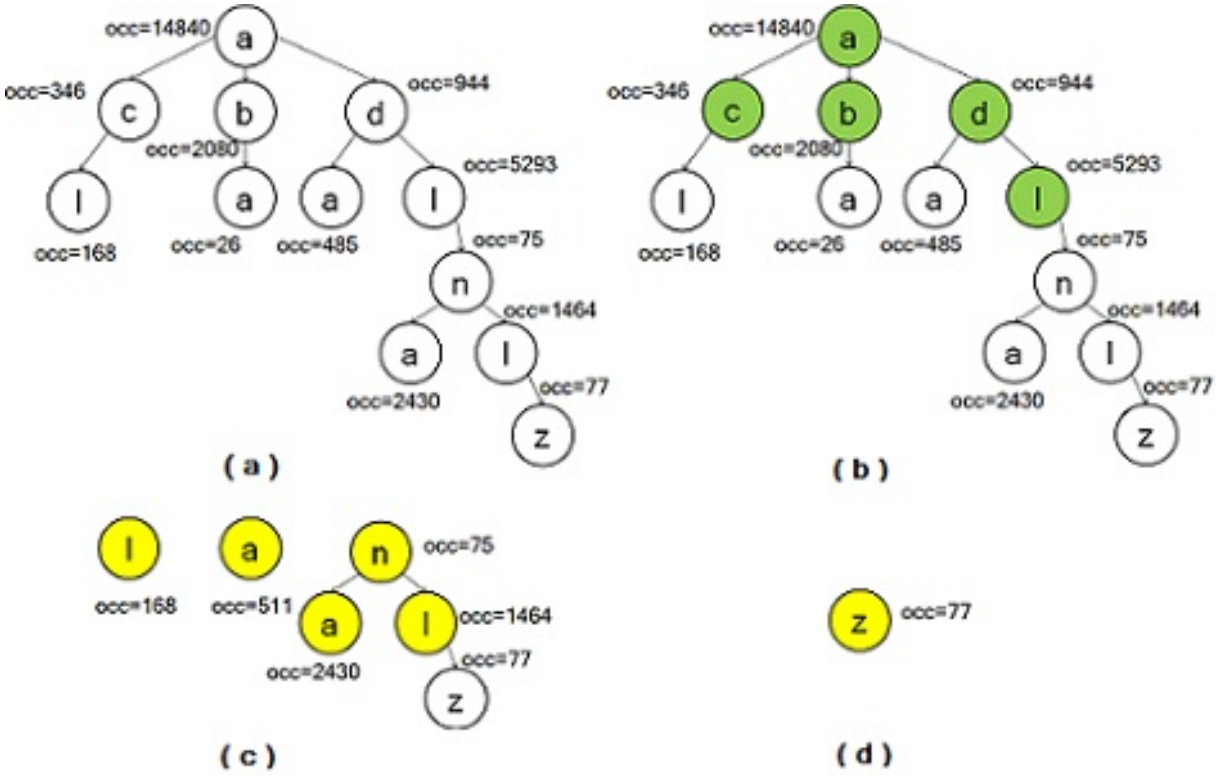

Figure 4.4. Sample run from REC-TRIE (a) A sample trie initialized, (b) After first iteration root words detected, (c) First affix trie is constructed with the extracted affix parts left from roots and first affixes are found, (d) Second iteration REC-TRIE created new affix tree and found the last affix.

The second approach, REVERSE-TRIE is a modified version of the first one. The procedure for finding root and affixes are same but the algorithm finds segmentation boundaries in bottom up fashion by constructing reverse word tries (See Figure 4.5) from the word list. The algorithm tries to exploit common affixes at first and determines the root boundaries at the end of the segmentation.

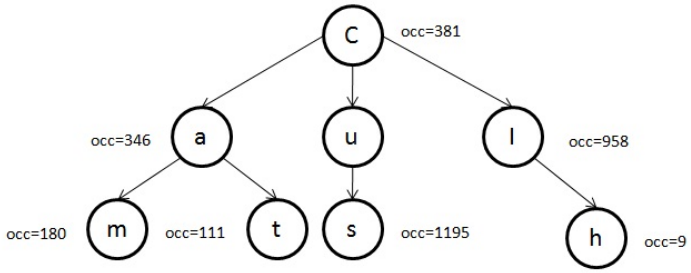

Figure 4.5. Sample reverse trie constructed for second approach.

The pseudo code of the second approach can be seen in the Figure 4.6. Words are reversed before constructing the reverse trie and word table (line 2). The rest of the procedure is the same as the first algorithm. The results of the algorithm are reversed in line 15 to get actual segmentation of the words.

```
1  Read words from Wordlist W
2  Reverse word
3  i = 1
4  Construct word Trie_i
5  Construct word Table
6  Do until no unsegmented words remain
7      For each word in the Table
8          Find boundary for unsegmented part with Trie_i and update Table
9          If the word is not fully segmented
10             Add unsegmented part to Trie_{i+1}
11         End If
12     End For
13 i = i + 1
14 End Do
15 Reverse results
```

Figure 4.6. Pseudo code of REVERSE-TRIE.

Chapter 5
Experiments and Results

Morpho Challenge gives two perl scripts to evaluate algorithms. These scripts simply compare the results against a linguistic gold standard. In the evaluation, only a subset of all words in the corpus is included. For each language, a random subset was picked, and the segmentations of these words were compared to the reference segmentations in the gold standard. The evaluation of an algorithm is based on the *F-measure*, which is the harmonic mean of precision and recall.

As we mentioned before, these metrics are calculated by

- Hit (H): A valid cut that means word is cut at the right place.
- Insertion (I): A wrong cut that means word is cut at the wrong place.
- Deletion (D): A missed cut that means a valid cut is ignored.

Based on these metrics; precision is the number of hits divided by the sum of the number of hits and insertions, and recall is the number of hits divided by the sum of the number of hits and deletions. So the measures are given as

$$\begin{aligned} Precision &= \frac{H}{(H+I)} \\ Recall &= \frac{H}{(H+D)} \\ F-Measure &= \frac{2H}{(2H+I+D)} \end{aligned} \tag{5.1}$$

We have used the dataset of the Morpho Challenge 2009 and evaluate results with the perl scripts provided. Tables 5.1, 5.2, and 5.3 show the results of REC-TRIE and REVERSE-TRIE compared with other algorithms for English, Finnish, and Turkish respectively. REC-TRIE has better F-Measure in Turkish and English than Finnish. This is due to fact that our algorithm finds roots and suffixes by traversing the trie one character at a time so found roots and suffixes are generally short. However, Finnish root words are rather long in the average due to the conservativeness of the language. Especially deletions are fairly more in Finnish since the algorithm oversegments the words. As a result, recall values for Finnish is low and pulls down the F-Measure dramatically.

On the other hand, REVERSE-TRIE has better F-Measure for Finnish than REC-TRIE because REVERSE-TRIE tries to find affixes first and then finds root morphemes. English results for REVERSE-TRIE ranks at the end of the other competitors, this is most probably English is an inflectional language and REVERSE-TRIE works well for agglutinative languages.

As a conclusion, REC-TRIE ranked 9th for English, 11th for Finnish, and 4th for Turkish datasets. The main reason for having low recall values in REC-TRIE is that the algorithm oversegments affixes. Recall values for affixes in the results for all languages has the lowest score with respect to recall values of non-affixes. The algorithm starts to divide affixes character by character if it can not find a valid boundary when there exists less than 3 characters in the last part of the word. This causes algorithm to have more deletions but also precision is rising for languages with small affixes. REVERSE-TRIE on the other hand, ranked last for English, 9th for Finnish, and 13th for Turkish datasets. REVERSE-TRIE unlikely starts with segmenting affixes and in the last step identifies roots of the words. As we mentioned above, the last segments of the words (roots in this case) are again segmented character by character. In REVERSE-TRIE the precision is decreased whereas recall values are increased. Since REVERSE-TRIE

finds affixes with better hits than roots, English and Turkish rankings are worse than REC-TRIE. However, Finnish long affixes are found better than REC-TRIE so that REVERSE-TRIE is above in the Finnish rankings.

Table 5.1. Precision, Recall, and F-Measure of REC-TRIE & REVERSE-TRIE compared with other algorithms in Morpho Challenge 2009 for English.

Author	**Method**	**Precision**	**Recall**	**F-Measure**
Virpioja &Kohonen	Allomorfessor	68.98%	56.82%	62.31%
-	Morfessor Baseline	74.93%	49.81%	59.84%
Monson et al	ParaMor-Morfessor Union	55.68%	62.33%	58.82%
Lignos et al	-	83.49%	45.00%	58.48%
Monson et al	Paramor-Morfessor Mimic	54.80%	60.17%	57.36%
Monson et al	Paramor Mimic	53.13%	59.01%	55.91%
Bernhard	MorphoNet	65.08%	47.82%	55.13%
Lavallée &Langlais	RALI-COF	68.32%	46.45%	55.30%
Our Algorithm	REC-TRIE	50.80%	53.86%	52.29%
Can & Manandhar	-	58.52%	44.82%	50.76%
-	Morfessor CatMAP	84.75%	35.97%	50.50%
Spiegler et al	PROMODES	36.20%	64.81%	46.46%
Lavallée &Langlais	RALI-ANA	64.61%	33.48%	44.10%
Spiegler et al	PROMODES 2	32.24%	61.10%	42.21%
Spiegler et al	PROMODES committee	32.24%	61.10%	42.21%
Tchoukalov et al	MetaMorph	68.41%	27.55%	39.29%
Golénia et al	UNGRADE	28.29%	51.74%	36.58%
Our Algorithm	REVERSE-TRIE	12.62%	82.27%	21.89%
-	Letters	3.82%	99.88%	7.35%

Table 5.2. Precision, Recall, and F-Measure of REC-TRIE & REVERSE-TRIE compared with other algorithms in Morpho Challenge 2009 for Finnish.

Author	**Method**	**Precision**	**Recall**	**F-Measure**
Monson et al	ParaMor-Morfessor Union	47.89%	50.98%	49.39%
Monson et al	Paramor-Morfessor Mimic	51.75%	45.42%	48.38%
-	Morfessor CatMAP	79.01%	31.08%	44.61%
Spiegler et al	PROMODES committee	41.20%	48.22%	44.44%
Monson et al	Paramor Mimic	47.15%	40.50%	43.57%
Spiegler et al	PROMODES 2	33.51%	61.32%	43.34%
Spiegler et al	PROMODES	33.86%	51.41%	42.25%
Lavallée &Langlais	RALI-COF	74.76%	26.20%	38.81%
Our Algorithm	REVERSE-TRIE	25.13%	74.49%	37.58%
Golénia et al	UNGRADE	40.78%	33.02%	36.49%
Our Algorithm	REC-TRIE	45.09%	27.05%	33.81%
Bernhard	MorphoNet	63.35%	22.62%	33.34%
Virpioja &Kohonen	Allomorfessor	86.51%	19.96%	32.44%
-	Morfessor Baseline	89.41%	15.73%	26.75%
Tchoukalov et al	MetaMorph	37.17%	15.15%	21.53%
Lavallée &Langlais	RALI-ANA	60.06%	10.33%	17.63%
-	Letters	5.17%	99.89%	9.83%

Table 5.3. Precision, Recall, and F-Measure of REC-TRIE & REVERSE-TRIE compared with other algorithms in Morpho Challenge 2009 for Turkish.

Author	**Method**	**Precision**	**Recall**	**F-Measure**
Monson et al	Paramor-Morfessor Mimic	48.07%	60.39%	53.53%
Monson et al	ParaMor-Morfessor Union	47.25%	60.01%	52.88%
Monson et al	Paramor Mimic	49.54%	54.77%	52.02%
Our Algorithm	REC-TRIE	53.40%	43.06%	47.68%
Lavallée &Langlais	RALI-COF	48.43%	44.54%	46.40%
-	Morfessor CatMAP	79.38%	31.88%	45.49%
Spiegler et al	PROMODES 2	35.36%	58.70%	44.14%
Spiegler et al	PROMODES	32.22%	66.42%	43.39%
Bernhard	MorphoNet	61.75%	30.90%	41.19
Can & Manandhar	2	41.39%	38.13%	39.70%
Spiegler et al	PROMODES committee	55.30%	28.35%	37.48%
Golénia et al	UNGRADE	46.67%	30.16%	36.64%
Our Algorithm	REVERSE-TRIE	22.61%	72.90%	34.51%
Virpioja &Kohonen	Allomorfessor	85.89%	19.53%	31.82%
-	Morfessor Baseline	89.68%	17.78%	29.67%
Lavallée &Langlais	RALI-ANA	69.52%	12.85%	21.69%
-	Letters	8.66%	99.13%	15.93%
Can & Manandhar	1	73.03%	8.89%	15.86%

Chapter 6
Conclusions

We propose a novel approach for unsupervised morphological analysis, based on trie data structure and word occurrences. The first algorithm (REC-TRIE) proposed constructs a forward word trie and finds root words according to the occurrences of characters in the path. After root detection is completed, remaining affix parts are used to construct affix tries. In each iteration, affix boundaries are detected and results are updated. Similarly, the second algorithm (REVERSE-TRIE) constructs a reverse trie and finds affix boundaries at first and the last morpheme found considered as the root morpheme.

Although our proposed algorithms are simple, the results are encouraging with respect to the other algorithms proposed previously. REC-TRIE ranks 4th in Turkish, REVERSE-TRIE ranks 9th in Finnish when compared with other competitors. The recall values state that we have missed some of the boundaries especially for Finnish in REC-TRIE. Using a bottom up agglutinative approach in REVERSE-TRIE causes to find missegmented boundaries for inflectional languages like English.

These algorithms do not have any methods for prefix detection and there is no control for the irregular changes of the words or umlauts, so we should develop some strategies to cope with these situations. Also merging these two approaches may bring higher hits and increase F-Measure more than the current situation.

Appendix A:
FILES

A.1. Files Attached

A.1.1. Codes of the Algorithms

Code of the REC-TRIE and REVERSE-TRIE are included on the CD attached. Algorithms are implemented in Java and can be run on any java compiler by providing the necessary input files listed below.

A.1.2. Datasets

- Wordlists from the Morpho Challenge are provided in three languages (Finnish, English, and Turkish) in the Dataset folder on the CD attached.
- Although we did not use corpus data in the algorithms we provide them in three languages (Finnish, English, and Turkish).

A.1.3. Evaluation Files

- Evaluation scripts are included on the CD in Evaluation folder. Results gathered from the algorithms can be evaluated by using these scripts. Instructions on how to evaluate scripts is written in the readme.txt file in the same folder.
- Gold standart segmentations to use in the evaluation process are provided in the Evaluation folder.

References

1. Alpaydın, E., *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, MIT Press, 2004.

2. Creutz, M. and K. Lagus, "Morfessor in the Morpho Challenge", *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, 2006.

3. Bernhard, D., "Unsupervised Morphological Segmentation Based on Segment Predictability and Word Segments Alignment", *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, 2006.

4. Bordag, S., "Two-step Approach to Unsupervised Morpheme Segmentation", *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, 2006.

5. Keshava, S., "A simpler, intuitive approach to morpheme induction", *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, 2006.

6. Johnsen, L., "Morphological learning as principled argument", *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, 2006.

7. Atwell, E., A. Roberts, P. Longman, and H. C. Je, "Combinatory Hybrid Elementary Analysis of Text (CHEAT", *Proceedings of the PASCAL Challenge Workshop*

on Unsupervised Segmentation of Words into Morphemes, 2006.

8. Jordan, C., J. Healy, and V. Keselj, "Swordfish: an unsupervised Ngram based approach to morphological analysis", *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 657–658, New York, 2006.

9. Dang, M. T., "Simple Unsupervised Morphology Analysis Algorithm (SUMAA)", *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, 2006.

10. ur Rehman, K. and I. Hussain, "Unsupervised Morphemes Segmentation", *Proceedings of the PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes*, 2006.

11. Bernhard, D., *Simple Morpheme Labelling in Unsupervised Morpheme Analysis*, pp. 873–880, 2008.

12. Bordag, S., *Unsupervised and Knowledge-Free Morpheme Segmentation and Analysis*, pp. 881–891, 2008.

13. McNamee, P. and J. Mayfield, "N-Gram Morphemes for Retrieval", *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access*, Cross-Language Evaluation Forum'08, 2009.

14. Monson, C., J. G. Carbonell, A. Lavie, and L. S. Levin, "ParaMor: Finding Paradigms across Morphology", *Cross-Language Evaluation Forum*, pp. 900–907, 2007.

15. Zeman, D., "Unsupervised Acquiring of Morphological Paradigms from Tokenized Text", *Advances in Multilingual and Multimodal Information Retrieval*, Vol. 5152, pp. 892–899, 2008.

16. Goodman, S. A., "Morphological Induction Through Linguistic Productivity", *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access*, Cross-Language Evaluation Forum'08, 2009.

17. Kohonen, O., S. Virpioja, and M. Klami, "Allomorfessor: Towards Unsupervised Morpheme Analysis", *Evaluating Systems for Multilingual and Multimodal Information Access*, Vol. 5706, pp. 975–982, 2009.

18. Monson, C., J. Carbonell, A. Lavie, and L. Levin, "ParaMor and Morpho Challenge 2008", *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access*, Cross-Language Evaluation Forum'08, pp. 967–974, 2009.

19. Zeman, D., "Using Unsupervised Paradigm Acquisition for Prefixes", *Evaluating Systems for Multilingual and Multimodal Information Access*, Vol. 5706, pp. 983–990, 2009.

20. Bernhard, D., "MorphoNet: Exploring the Use of Community Structure for Unsupervised Morpheme Analysis", *Cross-Language Evaluation Forum (1)*, pp. 598–608, 2009.

21. Can, B. and S. Manandhar, "Unsupervised Learning of Morphology by Using Syntactic Categories", *Working Notes for the CLEF 2009 Workshop*, Corfu, Greece, September 2009.

22. Golenia, B., S. Spiegler, and P. Flach, "UNGRADE: UNsupervised GRAph DEcomposition", *Working Notes for the CLEF 2009 Workshop, Corfu, Greece*, September 2009.

23. Lavallée, J.-F. and P. Langlais, "Morphological acquisition by formal analogy", *Morpho Challenge 2009*, Corfu, Greece, October 2009.

24. Constantine Lignos, M. P. M., Erwin Chanz and C. Yang, "A Rule-Based Unsupervised Morphology Learning Framework", *Working Notes for the CLEF 2009 Workshop*, Corfu, Greece, September 2009.

25. Christian Monson, K. H. and B. Roark, "Probabilistic ParaMor", *Morpho Challenge 2009*, Corfu, Greece, October 2009.

26. Spiegler, S., B. Golenia, and P. Flach, "Promodes: A probabilistic generative model for word decompositions", *Working Notes for the CLEF 2009 Workshop, Corfu, Greece*, September 2009.

27. Tzvetan Tchoukalov, C. M. and B. Roark, "Multiple Sequence Alignment for Morphology Induction", *Morpho Challenge 2009*, Corfu, Greece, October 2009.

28. Virpioja, S. and O. Kohonen, "Unsupervised Morpheme Discovery with Allomorfessor", *Morpho Challenge 2009*, Corfu, Greece, October 2009.

29. Solak, A. and K. Oflazer, "Design and Implementation of a Spelling Checker for Turkish", *Literary and Linguistic Computing*, Vol. 8, 1993.

Curriculum Vitae

Koray Ak was born on 9 September 1985, in İstanbul. He received his B.S. degree in Computer Science & Engineering in 2008 from Işık University. He worked as a research assistant at the department of Computer Engineering of Işık University from 2008. During this time firstly he has been affiliated with the Informatics Research and Development Center. He also worked on the student information and registration systems; Campus-Online and Course-Online as a lead programmer. His research interests include artificial intelligence, machine learning and software engineering.