Unsupervised morpheme segmentation in a non-parametric Bayesian framework

by

SANTA B. BASNET

Presented to the Department of Computer Science of The University of York in Partial Fulfillment of the Requirements for the Degree of

MSc by Research

University of York
Department of Computer Science
Deramore Lane, York, YO10 5GH, UK

February, 2011

Abstract

Learning morphemes from any plain text is an emerging research area in the natural language processing. Knowledge about the process of word formation is helpful in devising automatic segmentation of words into their constituent morphemes. This thesis applies unsupervised morpheme induction method, based on the statistical behavior of words, to induce morphemes for word segmentation. The morpheme cache for the purpose is based on the Dirichlet Process (DP) and stores frequency information of the induced morphemes and their occurrences in a Zipfian distribution.

This thesis uses a number of empirical, morpheme-level grammar models to classify the induced morphemes under the labels prefix, stem and suffix. These grammar models capture the different structural relationships among the morphemes. Furthermore, the morphemic categorization reduces the problems of over segmentation. The output of the strategy demonstrates a significant improvement on the baseline system.

Finally, the thesis measures the performance of the unsupervised morphology learning system for Nepali.

Contents

List of Figures

List of Tables

Acknowledgements

Many people are involved to make this research successful. Especially, my supervisor Dr. Suresh Manandhar, without him I couldn't even imagine this research to be done. I am very thankful to him for his invaluable suggestions, encouragements and guidance to pave the path of the research.

I am very much indebted to 'EURECA Project (www.mrtc.mdh.se/eureca) who has funded me to complete this research degree. I am so grateful to AIG group, Computer Science Department, University of York, for the academic activities such as seminars which helped me a lot to carry out this research.

Special thanks to my mate Matthew Naylor, Burcu Can, Shailesh Pandey, and Suraj Pandey for their significant help and encouragements to complete this work. In addition to this, I would also like to thank Laxmi Khatiwada and Balram Prasain, from Tribhuban University, Nepal for the correction of Nepali test dataset which is used for system evaluation.

Finally, I would like to thank almighty God for giving me sound health, enthusiasm, knowledge and strengths to achieve this degree.

February, 2011

1 Introduction

This thesis presents a work done on unsupervised learning of morphology. It has described an automatic morpheme induction from the plain text by making use of frequency of words. The resulting morphemes are used for segmentation.

1.1 Morphology and Morpheme segmentation

Natural language morphology identifies the internal structure of words. Words contain smaller units of linguistic information called morphemes which are the meaning bearing units in a language. For example the word ‘playing’ consist of two units: the morpheme ‘play’ and the morpheme ‘-ing’. When two or more morphemes are combined into a larger unit such as the word, a structure is formed which contains the morphemes itself and the relationship between them. The purpose of identifying structure of words is to establish the relationship between morphemes and the words that encodes them in a language giving the meaning, the categories it belongs to and its function in a natural language sentence.

Morpheme segmentation is the process of breaking words into their constituents i.e. morphemes. A morpheme segmentation system should be able to tell us the word “playing” is the continuous form of the stem “play”. These morpheme segments fall in two broad classes: stems (or roots) and affixes. Stems are the base form of words and are the main information contributor. Affixes contribute additional information of words such as gender, tense and number. For example, the word “unsegmented” in English can be divided into 3 morphemes “un”, “segment” and “ed” where “segment” is the base form and is a main information contributor. Morpheme “un” acts to negate the meaning of the stem and “ed” places it in the past tense (or adjectiveless). Here, “un” and “ed” are known as affixes. Hence, morphological analysis helps to identify the necessary information such as gender, tense, polarity, number and moods of the word.

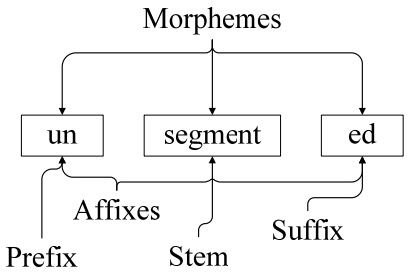


Fig. 1.1 Three segments of the word “unsegmented”.

The affix attached before and after the stem is known as the prefix and the suffix respectively. Some morphemes can appear as complete words. Such morphemes are called free morphemes. Morphemes appearing in a word combined with stems are called bound morphemes. The following Table 1.1 shows some examples of free and bound morphemes.

Table 1.1 Examples of free and bound morphemes.

Language	Free Morphemes	Bound Morphemes
English	do, eat, jump etc.	-s, -ed, -ing, -ness, un-, etc.
Nepali	नाम(name),खा(eat), लेख्(write)etc.	-दै/तै(continuous form),-न(present form), न-(negation), -अक(nominalise)etc.

There are mainly three morphological processes involved in word formation: inflection, derivation and compounding. These processes determine the function of morphemes. In inflection, a stem is combined with a grammatical morpheme to yield a word of the same syntactic class as the original stem, e.g., formation of the word “playing” from the verb “play”. Derivation combines a stem with a grammatical morpheme to yield a word belonging to a different syntactic class, e.g., derivation of the word “player” (noun) from the word “play” (verb), “computation” from the verb “compute” etc. Compounding is

the process of combining two or more words to form a new word, for example, childlike, notebooks and football are the compound words. Daughter-in-law and mass-product are a hyphenated form of compound words. Thus, one way of word variation is induced by attaching the affixes to the stem. A word can have more than one affix attached at the same time. Examples of some words with multiple affixes are shown in Table 1.2.

Table 1.2 Examples of words having single stem and more than one affix.

Language	Word with more than one affixes	Compound Words
English	skill(stem)+full(suffix)+ness(suffix) un(prefix)+segment(stem)+ed(suffix)	sportsman, storekeeper
Nepali	न (prefix)+ टुक्र(stem)+ आइ (suffix)+ एको (suffix) – (un-separated by)खा(stem)+एको(suffix) – (eaten)	खानुहुने – (be eaten), देख्नसके – (succeed to see)

1.2 Introduction to morphology learning

Automatic identification of meaning bearing units or morphemes in a natural language text is an active area of research. Addressing the word variations by analyzing words of a language as a sequence of morphemes is defined as morphology learning. For example, the English word “unpredictable” can be analyzed as a sequence of the three morphemes “un”, “predict” and “able”.

In rule based learning, morphological analysis relies on manually crafting of linguistically motivated rules for morpheme acquisitions and segmentations. In automatic acquisition of morphemes, machine learning techniques are successfully applied for morphology learning. Using machine learning techniques, we can capture certain phenomenon that are involved in word formation and helps to induce morphemes of a language. For example, we can capture regularities among the words such as “chunk”, “chunks”, “chunked” and “chunking”. In these words, we can induce

morphemes “chunk”, “s”, “ed” and “ing” by using pattern regularities. Similarly, we can use machine learning technique to capture the orthographic changes in a word during morpheme attachment to the stem. For example, a character “y” changes to “i” in many English words during suffix (fly→flies) attachment.

To develop the model of morpheme induction and segmentation, I consider the process of adding affixes to the stem as the primary way to induce word variations. In this thesis, I have presented an unsupervised approach to morpheme induction by utilizing the cache of morphemes frequency. The frequency of morphemes is derived from the naturally occurring corpora. It is well established notion that the words behavior in a corpus follows a power law distribution (Zipf’s law) [18]. I examine the statistical distribution of morphemes in a natural language text which are presented in Table 1.3 and 1.4. These morphemes conform to the power law distribution.

Table 1.3 Examples of some common English suffix morphemes and their frequency learnt from Morpho Challenge – 2009 dataset using RePortS algorithm [5].

S.N.	Suffix Morphemes	Frequency
1.	s	2140800
2.	er	964162
3.	es	887832
4.	ed	421481
5.	ing	289236
6.	tion	82304
7.	ment	65294
8.	est	46935
9.	ies	27455
10.	ness	14274

Table 1.4 Examples of some of common English prefixes and their frequency learned from Morpho Challenge – 2009 dataset using RePortS algorithm [5].

S.N.	Prefix Morphemes	Frequency
1.	un	61004
2.	ex	49477
3.	dis	39882
4.	inter	22280
5.	pre	20934
6.	non	13564
7.	post	11251
8.	dia	8651
9.	hyper	485
10.	in-	390

In the thesis statistical property (i.e. frequency) of text is gathered to identify the morphemes of a language and use them to segment words. The algorithm detects the morphemes and classifies them in three categories. They are labeled as prefix, stem and suffix based on their position in the word. In segmentation process, it considers a word to have a single affix, multiple affixes and multiple stems as well. Here, I have populated stems and affixes of a language during learning stage and listed out them for segmentation of words.

1.3 Issues in morphology learning

A commonly asked question regarding how to develop a language independent algorithm that can take a list of words as input and produce a set of morphemes as output is a challenging problem. To address the above question, we have to model a system that can able to implement or encode the human knowledge of morphology i.e. understanding of natural language words to enhance the performance of Natural Language Processing (NLP) tasks. This is not a trivial task.

In practice, language acquisition involves the representation of internal structure which enables us to encode the input words and to generalize it by producing all linguistic forms. Our understanding is that there is no complete computational model for language acquisition. In overall task of NLP, different theories are proposed which help to capture some aspects of language acquisition. Practically, some features of language acquisition are more difficult such as capturing feature of words which does not occur in training corpus. So, these issues have to be resolved while breaking a word into its meaningful units.

In morphology learning, initial focus is on capturing regularities among words. A key challenge in computing the regularities among words is to handle the exceptions present in the sparse data such as the word “singed” is not a variation of the word “sing”. The irregularities present during word formation are a difficult problem and needs to be addressed during morphology learning. For example, the word “sang” is a variation of the word “sing”. Another issue is to identify the grammatical features of segmented morphemes such as tense and number. Generalization may lead to erroneous results for unseen cases and is an issue for a computational model as well. Analysis of words with multiple senses is a context dependent task. We also have to consider what kind of linguistic feature can be encoded in model learning. A language can have unique features that are not observed in other languages which make it more difficult in building a general language model.

I have examined the above issues and developed a computational model that learns the feature of morpheme behavior from a corpus. To model the word structures, this thesis utilizes the linguistically motivated word grammars to encode morphemes relationship during word formation. However, this encoding is empirical which explores the different structures of words. It is capable of understanding the morphological representation and is supported by statistical occurrences.

1.4 Unsupervised morphology

Knowledge-based morphology systems are based on manually constructed rules. These rules are crafted especially to address morphological processes of a particular language by linguistic experts. Building morphological lexicons and rules for all languages is time consuming and requires considerable amounts of human effort. In most of the languages, morphological lexicons are unavailable and building them by humans is a very slow and expensive process. The alternative solution to this problem is to use an unsupervised approach to learn morphological structure of a language.

The unsupervised morphology system is capable of inducing such a structure without making use of human annotated data and is also a language independent system. Words in a natural language can be considered as an open set, i.e. words of a language keeps changing and the number of words grows as well. It gives a new dimension to learning. In similar case of language learning by human, the vocabulary in mind also evolves. A learning system can help to update the domain information periodically. This research focuses on the same learning strategy based on an updated input corpus. It utilizes natural language text gathered from different sources such as news papers and magazines. In this case, the system uses a machine learning approach to induce knowledge of automatic morphology, rather than making use of widespread linguistics annotation.

1.5 Goals and approach

Natural language texts are widely available from different sources such as newspapers, magazines, and blogs. As I discussed earlier, language acquisition from a natural language text has numerous problems and has become a challenging field in NLP. In morphology, morpheme induction and their structure in natural language text must be able to address the process involved in the word formation. The primary goal of this research is to design an unsupervised morphological analyzer of languages. In addition to this goal, I have the following specific goals:

a) Automatic induction of morphemes of a language with the formulation of model through statistical behavior of morphemes and implementation of data-driven learning system.
b) Induced morphemes are used to segment the input words by exploring different word structures (word grammars) without using any external annotated data.
c) Explore the unsupervised morphology learning system in Nepali language.

Natural language applications such as Parts of Speech (POS) induction [29 and 30] and text indexing and retrieval [31] performance can be improved by using unsupervised morphological analyzer. I believe that this research will become a useful system to improve NLP applications statistical machine translations, information retrieval etc. Sample input/output of the system is shown in Table 1.5. This research has focused to capture the statistical proprieties of the natural language texts, i.e. the frequency to learn the morphemes of a language by investigating the different morphological structures of words. After morpheme detection, this algorithm classifies these morphemes in 3 different categories labeled as prefix, stem and suffix. These categories are illustrated and explained in Chapter 4.

Table 1.5 Sample text input in frequency-word format and their corresponding output morphemes.

Frequency	**Word**	**Morphemes**
6	jump	jump
1	jumper	jump + er
3	jumped	jump + ed
1	jumping	jump + ing
4	jumps	jump + s

Furthermore, the categorized morphemes are used to carry out the segmentation of input texts. During segmentation, I employ a linguistically motivated heuristics which

considers a word can have single affix, multiple affixes and multiple stems. Chapter 4 describes this in detail.

1.6 Outline

This thesis is organized as follows. Chapter 2 reviews some of the previous works on unsupervised morphology. More specifically, this discusses different techniques that are applied to capture different morphological phenomena from a plain text corpus. This chapter also talks about some of the language specific heuristics applied on unsupervised morphology.

Chapter 3 describes the mathematical preliminaries of morphological modeling based on a Bayesian inference framework. The process of morpheme induction is based on morpheme cache which stores the frequency information of previously induced morphemes and their occurrences. This chapter also describes the detail formulation of the mathematical model.

Chapter 4 describes four grammar models which I have implemented during the research. These models are based on the formulation of Bayesian statistics which are explained in Chapter 3. These grammar models relate the morphemes and words by different structures i.e. morphological rules. The first model is named as *Baseline*, has a single morpheme cache and the rest three are the extensions of the *Baseline model*. This section also explains the design and implementation of data structure in the system.

Chapter 5 describes the training and testing process with the tabulation of results. At first, it starts with a detailed description of the dataset that are prepared for English and Nepali languages. Next, I describe an evaluation metric which is used to evaluate the results of this system. The results are tabulated in different aspects of evaluation such as segmentation with single split, multiple splits, segmentation results of words having only single stem and single affix.

In Chapter 6, I discuss the results of the system and the sources of error in the results. I also discuss the key issues of the erroneous result with different cases and examples. I present a comparative result of all four models. An idea for further extension and development of the system is also given in the chapter. Finally, Chapter 7 concludes the thesis.

2 Background

Automatic induction of morphemes aims to capture the phenomena involved during generation of word variations through morphological processes. There has been a lot of work in the area of unsupervised morpheme acquisition. Numerous approaches have been developed to address different morphological processes. These approaches help to learn morphemes of a language automatically and are used to segment the words of given language. Some of the current unsupervised approaches of morphology are discussed as below.

2.1 Concatenation phenomena in morphology

Many words in a corpus are formed by adding affixes to the stems. The following describes some of the earlier works on identifying the morpheme boundary.

2.1.1 Letter successor varieties / Conditional entropy between letters

Earlier works [1, 2 and 3] have made a use of statistical properties such as successor and predecessor variety count of words in a corpus to indicate the morpheme boundaries. Calculation of letter successor varieties of some English words READABLE, ABLE, READING, READS, APE, RED, BEATABLE, ROPE, FIXABLE, RIPE and READ is shown in fig. 2.1 and fig. 2.2.

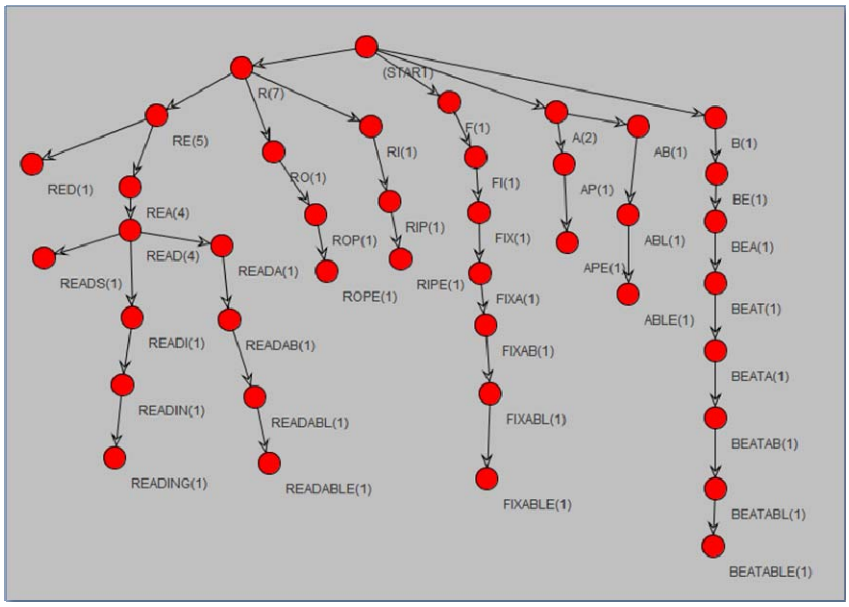


Fig. 2.1 Letter successor varieties recreated from [2].

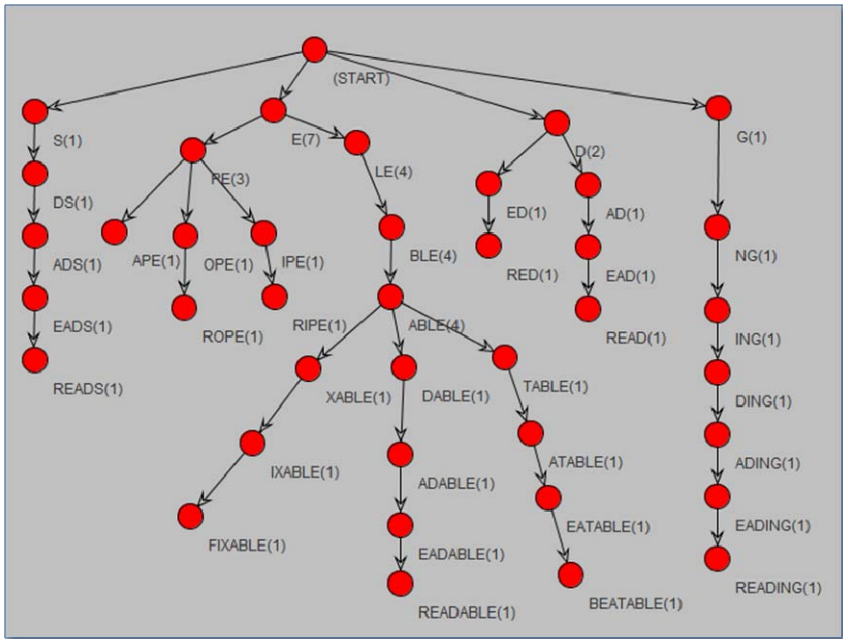


Fig. 2.2 Letter predecessor varieties recreated from [2].

For a test word “READABLE” and the task to find the possible morpheme boundaries the following scenario is observed. If we consider “R” to be a prefix, then 3 distinct characters “E”, “O” and “I” appears immediately after “R”. Hence the successor variety is 3. This process iterates throughout the word “READABLE”. We can see the 3 branches appear after “READ” with characters “S”, “I” and “A” as shown fig. 2.1 i.e. successor variety and also 3 branches appear before “ABLE” with characters “D”, “T”, and “X” i.e. predecessor variety. These successor and predecessor varieties are possible solution to the morpheme discovery of the word “READABLE”. This is shown in Table 2.1.

Table 2.1 Letter successor varieties recreated from [2].

Successor Varieties		Predecessor Varieties	
Prefix	Successor Varieties	Suffix	Predecessor Varieties
R	E, O, I (3)	E	L, P (2)
RE	A, D (2)	LE	B (1)
REA	D (1)	BLE	A (1)
READ	A, I, S (3)*	ABLE	D, T, X (3)*
READA	B (1)	DABLE	A (1)
READAB	L (1)	ADABLE	E (1)
READABL	E (1)	EADABLE	R (1)
READABLE	# (1)*	READABLE	# (1)*

An experimental design has been made by Dang and Chaudri [3] to measure the combined successor and predecessor frequency (SFPF) of the corpus based on the works [1 and 2]. This design considers the previous words seen in the corpus and are used to get the first split before applying SFPF. For example, if a word “abandoned” is previously seen and the current word “abandonedly” has to be processed, then the algorithm first splits the word “abandonedly” by the help of word “abandoned” giving

the result “abandoned + ly”. After this, it applies SFPF to the left word “abandoned” for further segmentation which gives the better results than the earlier works.

In a similar way, the transitional probabilities between the letters and word appearing as a substring of another word are used together to detect the morpheme boundaries [4]. It defines two thresholds during selection of affixes from the candidate list. One is reward-score by 19 and the other is punish-score by -1. The intuition behind these numbers is to select the morphemes as the final candidate only when they pass following tests 5% of the times that they appear.

a) String fragment without suffix should be a valid word.
b) The transitional probability between first letter of suffix and last letter of string fragment should be less than 1.
c) The transitional probability between second last letter and last letter of the string fragment without suffix approximately equal to 1.

The above criteria are also applicable for learning prefixes. The overall scores calculated from rewards and punishments of the morphemes are used not only to identify the candidate morphemes but also to detect the compound morphemes. If a morpheme is composed of two other morphemes both with higher scores then it is accounted as the compound of the two and is removed from the induced morpheme list. These strong hypotheses are unable to capture the morphemes in words whose base forms do not appear in the corpus and are present in the form of orthographic change. For example, during the suffixation “es” to the word “duty”, the character “y” changes to “i” giving the word “duties”. The previously described RePortS algorithm [4] is not capable to induce the suffix “es” from “duties” since the string fragment “duti” is not present in the given corpus as a valid word.

Later, the basic RePortS algorithm [4] is extended by employing better morpheme induction and segmentation strategies [9, 10 and 11]. The equivalence sets of letters is generated with the help of word pairs having small word edit distance and improved the results by 2% in precision without losing recall for German language [9]. In [10 and 11],

language specific heuristics are added for composite suffix detection, incorrect suffix attachments and orthographic changes during model learning. Composite suffixes are formed by combining multiple suffixes. This algorithm detects the composite suffixes by measuring the ratio of words in which both suffixes are attached to the words in which first suffix is attached. Incorrect suffix attachments are identified using the word-root frequency ratio. This ratio is based on the hypothesis that the inflectional or derivational forms of a root word occur less frequent than the root itself. To capture orthographic changes in root, it induces candidate allomorphs[1] with the help of edit distance measure and applies series of filtering techniques to get final root candidate. These heuristics improve the result of PASCAL challenge[2] by about 3% and the algorithm is also tested on the Bengali language. Finally, these algorithms also posses the shortcomings of basic RePortS algorithm and become more language specific due to the implementation of language specific heuristics.

2.1.2 Minimum Description Length (MDL) approach

Some of the well-known approaches in unsupervised learning of morphology are based on application of the Minimum Description Length (MDL) principle [12]. MDL gives the best hypothesis for a given set of data as that one which gives the highest compression of the data. Data is compressed by minimizing the description of model (in bits) and the description of data (in bits) encoded. So, if we are trying to encode the given words of a language, then the MDL technique allow us to represent the words with the smallest set of morpheme segments.

In [5], there are two models implemented to learn the morphemes from the given corpus. The first model uses MDL principle which learns a set of morphemes that minimizes the encoding length of the overall corpus. The second model learns a set of morphemes by utilizing Maximum Likelihood (ML) estimation of the corpus when segmented by the

[1] "An allomorph is one of two or more complementary morphs which manifest a morpheme in its different phonological or morphological environments."(Source-http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsAnAllomorph.htm)

[2]http://research.ics.tkk.fi/events/morphochallenge2005/

set. The MDL principle favors minimum numbers of morphemes to encode the corpus and hence gives the over-segmentation result. Rejection criteria are applied for the rare morphemes and sequence of one letter morphemes. These criteria are manually controlled during segmentation to overcome the over-segmentation problem. Further, the prior distributions of morpheme frequency and morpheme length are used rather than by encoding to measure the goodness of induced morphemes [6]. This gives a better acquisition and segmentation result. Words in a corpus are divided into stems and affixes and lots of heuristics are applied to generate morphological hypotheses [7]. These hypotheses are derived with keeping concern on the morphological structure of Indo-European languages. The stem forms a group called signature and each signature shares a set of possible affixes. The affix signature is similar to the concept of paradigm and is described in section 2.1.3. The algorithm also known as Linguistica uses MDL techniques for model optimization.

2.1.3 Paradigm based approach

In paradigm based morphology, words are grouped together by some means to form a paradigm which relates their constituent morphemes. These morphemes can be categorized in different groups such as stems and suffixes. So, paradigms describe morphemes relationship onto words. Example of some stem-suffix paradigms of English captured by the algorithm [13] is shown below:

i) *Suffixes:* e, ed, es, ing, ion, ions, or
Stems: calibrat,consecrate,decimat,delineat,desecrate,equivocat,postulat, regurgitat

ii) *Suffixes:* ϕ, d, r, r's, rs, s
Stems: analyze, chain-smoke, collide, customize, energize, enquire, naturalize, scuffle

The stem-suffix paradigms for a language are induced by grouping all possible single split stem+suffix pairs of words [13]. These obtained paradigms are filtered out using a

number of heuristic rules. The applied heuristics are aimed to generalize the spurious paradigms and to minimize the numbers of paradigms. The given words are analyzed by making all possible single splits from learned stems and suffixes. If one or more stem-suffix pairs are populated, then those pairs are marked as a possible analysis. Here, ambiguous analyses are also possible for a single word. If no analysis is found this way, it returns either known stems or suffixes as analysis for a word. In [14], word paradigms are learned using the syntactic information i.e., PoS of a word. In each paradigm, morphemes are induced by splitting each word at all single splits and ranked according to their Maximum Likelihood (ML) estimates. It uses expected accuracy between paradigms to merge them and gives the generalized paradigms. These generalized paradigms recover the missing word forms in the corpus. This generalization also minimizes the number of learned paradigms. Word segmentation is carried out either by the paradigm for known words or by the sequence of rules with learned morphemes for unknown words. The compound words are segmented recursively from rightmost end to the left. In case of multiple matches, the system selects the longest one.

In [16], word families (clusters) of morphologically similar words are identified by building a graph where nodes are words and edges are the morphological transformational rules. These rules transform one word into another by substring substitutions and are obtained from the topmost 10,000 frequent words in the given corpus. A rule is selected only when the length of substring matches between words are greater than the predefined threshold. So, a word family includes such words which are tied by same transformational rule. The community detection method described by Newman [15] is used for the identification of word-families. A threshold value in terms of edge-density, ranges from 0 to 1 is used to control the size of family (cluster). Edge-density is the ratio of common edges to the product of nodes between the word families. Varying the density value also helps us to generalize the word families to address the missing words in the corpus. Input words are segmented with the help of these induced word families and the learned morphological transformation rules.

2.2 Addressing word-internal variation in morphology

Some of the morphological works focus on orthographic changes during morpheme attachment to the stem i.e., word formation. The probable morphologically related word pairs are generated from the corpus by using orthographic similarity and semantic similarity [8]. The orthographic similarity is the edit distance score and the semantic similarity is the Mutual Information (MI) measure between the words. This measure gives a list of generated morphologically similar word pairs. This algorithm selects orthographically similar pairs of word according to a preset edit distance threshold value. In case of semantically similar pair, the Mutual Information (MI) measure between two words is calculated. The MI measure gives how the occurrence of a word in a corpus is independent of the occurrence of other and vice versa. Larger MI score implies the words are more dependent on each other. This system selects the common word pairs from both measures and extracts the longest shared substring to parse the word into the stem+suffix form. It also identifies the number of word pairs which are related by a variety of derivational and inflectional processes.

Single character changes in stems (last character only) are captured giving improved result to the morphological analysis in some languages [10]. The orthographically similar candidate list of stems i.e., the allomorphic variations and the orthographic rules are induced from the list of training words of single stem and single suffix only. The list of stems and rules is obtained by altering the last character of a candidate stem. It retains the orthographic rules by employing frequency based filter and those filtered rules are used for final segmentation. Similarly, a single character change between two words is considered to generate equivalence sets of characters in German [9]. This improves the recall of German morphology by 2% without loss of precision.

2.3 Non-parametric Bayesian framework in morphology

The previous sections described morphology learning using various ways such as ML estimate, MDL prior or by using statistical analysis of text along with numerous

heuristics. In [21], a non-parametric Bayesian framework is defined to learn the Probabilistic Context Free Grammar (PCFG) rules through Adaptor Grammars. Adaptor Grammars captures syntax of language (parse tree) by using non-parametric prior. The choice of adaptor specifies the distribution of PCFG rules in non-parametric Bayesian statistics. In morpheme segmentation, these rules generate the morphemes and the adaptor learns the morphemes of a language effectively. It uses the adaptor based on Pitman-Yor process [17] to specify the distributions used in non-parametric Bayesian statistics such as Dirichlet process [17, 19 and 20]. In [22], the adaptor grammar framework is used in unsupervised word segmentation for Sesotho. The word types (lexicon) and tokens are used to model a language [25]. The two-stage framework in [25] uses morpheme-based lexicon generator and Pitman-Yor adaptor to infer language morphology from the word types. It also shows the morphological information learned from types is better than from tokens. Further, this two-stage model framework is also used for word segmentation by taking account of sequential dependencies between words.

This research can be observed in similar direction based on the non-parametric Bayesian framework. However, it has made a use of corpus statistics effectively for morpheme learning by employing a morpheme cache. It captures the morpheme behavior based on word's frequencies and the morpheme cache is used to model the rich-get-richer behaviors i.e., the observed behavior of morphemes in a language and are shown in table 1.3 and 1.4. Furthermore, it has implemented different word grammar models to explore the different structural relationship among the morpheme during word formation.

3 Mathematical preliminaries for non-parametric Bayesian morpheme induction

The main aim of this chapter is to describe mathematical concepts that have been used to infer morphemes of the given language. Before describing different morphological models, we explain how the mathematical framework is used by the system to discover morphemes from a raw text data. To understand the probabilistic implementation of the model, the reader should possess some basic knowledge on probability theory. This chapter proceeds with a brief discussion of basic probability theory and concepts that are central to our approach such as multinomial distributions, Dirichlet processes and Chinese restaurant processes.

3.1 Introduction to probability

3.1.1 Random Experiments and Finite Spaces

Suppose, we perform a random experiment whose possible outcomes is finite. For example, rolling a dice has finite outcomes. Assuming that the rolling a dice is a random event, the chances of getting a single number from 1 to 6 is the same. We can assign the probability value of 1/6 to each outcome. The set of all possible outcomes is known as the *sample space* and is denoted by *S*. In case of rolling a dice, the sample space is: $S = \{1, 2, 3, 4, 5, 6\}$, where the outcomes are *1, 2, 3, 4, 5* and *6*. Any subset *E* of the sample space *S* is an *event* of the experiment. If $E = \{1\}$, then *E* is the event that *1* appears on the roll of the dice. If $E = \{2, 4, 6\}$, then *E* would be the event that an even number appears on the roll. Any set of the outcomes has a probability and is calculated by summing up the probability of its members of the set. So, the probability of getting an even number in rolling a dice is *1/2*.

If *A* and *B* are subsets of outcomes such that $A \subset B$, then the probability of *B* is larger than the probability of *A* i.e. $P(B) > P(A)$. If *A* and *B* are disjoint, then

$$P(A \cup B) = P(A) + P(B).$$

The sum of the probabilities of all outcomes is *1* i.e. $P(S) = 1$. If $A_1, A_2, \ldots, A_n$ are any finite sequence of *n* disjoint events of sample space *S* i.e. $A_i \cap A_j = \phi$, $i \neq j$, $i, j = 1, 2, \ldots, n$ then

$$P\left[\bigcup_{i=1}^{n} A_i\right] = \sum_{i=1}^{n} P(A_i) = 1.$$

3.1.2 Conditional probability and Bayes' rule

When observing an outcome from the random experiment, sometimes we consider outcome that has some additional information. This is the idea behind the *conditional probability*. Suppose, we know that the outcome is in $B \subset S$. Given this information how can we find out the probability of outcome in $A \subset S$? This is described by the conditional probability of *A* given *B* and written as $P(A|B)$.

Consider the roll of a dice and suppose we know that the number is even. Then, what is the probability that the number is six? Let *A* be the event of getting a number from a dice roll. Since the dice in unbiased, the probability of getting any one number is *1/6*. Let *B* be the event of getting even number from a dice roll. The possible outcome is either of *2*, *4* or *6*. Hence the probability of getting an even number is *3/6*. With these information, we can calculate the conditional probability $P(A|B)$ as,

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \tag{3.1}$$

Since the number can be both even and six if and only if it is the number six, $A \cap B = A$ and $P(A|B) = P(A)/P(B)=1/3$. In this case, conditional probability can be interpreted only when $P(B) > 0$.

In this example, the event A (getting a number) and the event B (getting a six) are *dependent* events. If events A and B are *independent* then, $P(AB) = P(A) \times P(B)$. The conditional probability of A given B under this assumption is reduced to just $P(A)$.

Let us introduce another event B^c, read as the complement of B. B and B^c are mutually exclusive events and whose union results in S. Therefore we can write

$$P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A \mid B)P(B) + P(A \mid B^c)P(B^c)}. \tag{3.2}$$

This formula is known as Bayes' rule. If we make partitions of n disjoint events $B_1, B_2, \ldots, B_n$ from the sample space S and let A be the another event, then using the Bayes' rule,

$$P(B_i \mid A) = \frac{P(B_i \cap A)}{P(A)} = \frac{P(A \mid B_i)P(B_i)}{\sum_{j=1}^{n} P(A \mid B_j)P(B_j)}. \tag{3.3}$$

3.1.3 Random variable and distribution

When we perform a random experiment and are interested to know some outcome on the sample space, these quantities of interest are known as the *random variables*. Random variable takes measurable real values. For example, height of the next student to enter the classroom is a random variable. Formally, if the outcome of random experiment is ω, then the value of random variable $X(\omega) \subset \Re$.

A random variable is *discrete* if it takes values in a countable set $\{x_n, n >= 1\} \subset \Re$. Some random variables can take continuous values and are called *continuous* random variable.

The value of the random variable is determined by the outcome of the random experiment so we can assign some probability to possible values of a random variable.

Staying with the discrete case, if X is a discrete random variable associated with values $x_0, x_1, x_2, \ldots$ and the probability of each x_i is $P(X = x_i) = p_i$ then the probability $P(x_i)$, $i = 0, 1, 2, \ldots, \infty$ must satisfy the following conditions.

i) $P(x_i) \geq 0$ for all i, $i = 0, 1, 2, \ldots, \infty$

ii) $\sum_{i=0}^{\infty} P(X_i) = 1$

The function P is called the *probability mass function* (pmf) and the collection $\{x_i, p_n, n \geq 1\}$ is known as the probability distribution of the random variable X. The function $F(x) = P(X \leq x)$ is called *cumulative distribution function* (cdf) of the random variable X. The discrete random variables are studied according to their probability mass function (pmf). In case of continuous random variable, there is a non-negative function $f(x)$ for all $x \in (-\infty, +\infty)$ associated with a random variable X known as the *probability density function* (pdf) which determines the behavior of the random variable.

3.2 The Bayesian method

3.2.1 Terminologies in a Bayesian approach

Statistical inference concerns itself with some unknown parameters that describe distribution of certain random variable of interest. In a Bayesian analysis of the data, we can incorporate *prior information* which helps to strengthen the inference to the partial known data. From the Bayes' rule described in equation 3.3:

$$P(M \mid D) = \frac{P(D \mid M)P(M)}{P(D)}, \tag{3.4}$$

and we define the corresponding terminology:

$$posterior = \frac{likelihood \times prior}{evidence}.$$

The *likelihood* part is the probability of observing data *D* being conditioned on the model *M*. The prior distribution *P(M)* takes any particular value based on the additional information that might be available to the system. The denominator part, *evidence* is a normalization factor and can be taken to be a constant for a given task. So, we can focus on numerator part only, and infer

$$posterior \propto likelihood \times prior \tag{3.5}$$

The *posterior* gives the probability of model based on the new observed data relative to the prior probability. *Maximum Likelihood* (ML) estimation tries to find the parameter that makes the observations most likely i.e. maximizes the value of likelihood, *P(D|M).* A ML estimator can be written as:

$$M_{ML} = \arg\max_{M} P(D \mid M) \tag{3.6}$$

Unlike ML estimation, *Maximum a Posteriori* (MAP) utilizes some prior information through prior distribution, *P(M)*. MAP estimation can be expressed as:

$$M_{MAP} = \arg\max_{M} P(M \mid D) = \arg\max_{M} P(D \mid M)P(M) \tag{3.7}$$

The Bayesian approach extends the MAP estimation by ensuring an exact equality of posterior probability in equation (3.4). The denominator in equation (3.4), the probability of evidence, can be expressed as:

$$P(D) = \int_{M} P(D \mid M)P(M)dM$$

This is only valid assuming the model space is integrable.

Now, Eq. (3.4) becomes,

$$P(M \mid D) = \frac{P(D \mid M)P(M)}{\int_M P(D \mid M)P(M)dM} \tag{3.8}$$

3.2.2 Conjugate priors

In a Bayesian method, the integral of marginal likelihood in equation (3.8) are intractable or are unknown variables. But we can choose the prior belief as additional information to strengthen the inference. A well-known approach to facilitate the model computation is to use a *conjugate prior*. A conjugate prior *P(M)* of likelihood *P(D|M)* is a distribution that results in a posterior *P(M|D)* with same characteristics.

If the likelihood function is *Binomial*, choosing a *Beta distribution* as prior will give the same *Beta distribution* as posterior. The expression of probability density function (*pdf*) of *Binomial distribution* given the *r* success in *n* independent trials with same probability θ in each individual event is:

$$Bionmial(r \mid n, \theta) = \binom{n}{r} \theta^r (1-\theta)^{n-r}, \tag{3.9}$$

where $\binom{n}{r} = \frac{n!}{r!(n-r)!}$ and is called *Binomial coefficient*. In case of *Beta distribution*, the probability density function (*pdf*) is:

$$Beta(\theta \mid p, q) = \frac{1}{\beta(p,q)} \theta^{p-1}(1-\theta)^{q-1} \begin{cases} 0 \le \theta \le 1 \\ p, q > 0 \end{cases},$$

where $\beta(p,q)$ is Beta function and is defined as

$$\beta(p,q) = \int_0^1 \theta^{p-1}(1-\theta)^{q-1} d\theta.$$

This can also be expressed as Gamma function as

$$\beta(p,q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)}.$$

Using Beta as *prior* and Binomial as *likelihood* then the posterior function becomes:

$$p(\theta \mid n,r,p,q) = \frac{\Gamma(p+q+n)}{\Gamma(p+r)\Gamma(n+q-r)}\theta^{p+r-1}(1-\theta)^{n+q-r-1} \tag{3.10}$$

i.e., $p(\theta \mid n,r,p,q) = \text{Beta}(p+r, n+q-r).$

Hence, the posterior distribution is still Beta with $p+r$ and $n-q-r$. Choosing an appropriate prior makes analytic calculations simpler.

3.2 Zipf's Law

Zipfian is an empirical law that characterizes statistical properties of a natural language. It states: given a corpus of natural language, the frequency of any word is inversely proportional (approximately) to the rank in the frequency table. As a consequence, the topmost frequent word will occur approximately twice as often as the next word, three times as often as the third most frequent word and so forth. To generalize, i^{th} ranked word will appear with $1/i$ times the frequency of the most frequent word. An example of Zip's plot of Wikipedia text is shown in fig. 3.1. Such kind of behaviour is not limited to words. The morphemes of a language also conform to Zipf's law. Some common English morphemes (prefixes and suffixes) and their frequencies are listed in Tables 1.3 & 1.4.

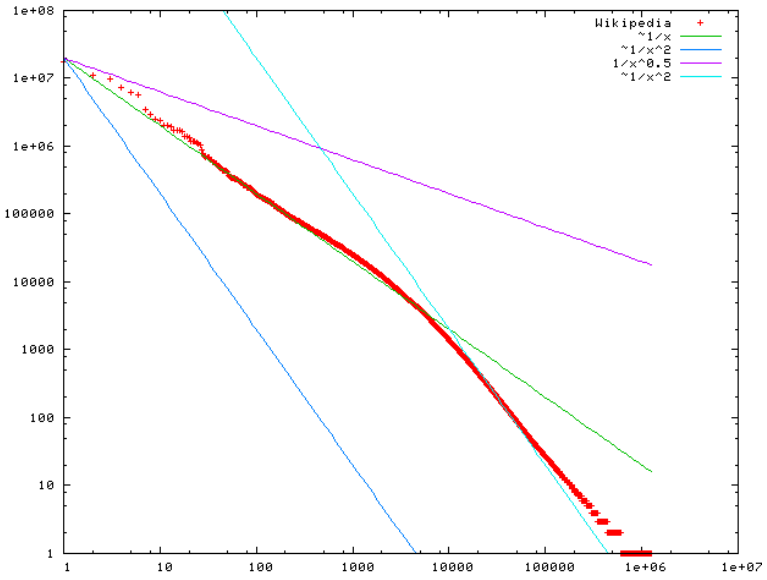


Fig. 3.1 Plot of word frequency in Wikipedia-dump 2006-11-27[3]. The plot is made in log-log coordinates. *X-axis* is rank of a word in the frequency table; *Y-axis* is the total number of the word's occurrences. Most popular words are "the", "of" and "and".

3.3 Chinese Restaurant Process (CRP)

The *Dirichlet Process* (DP) [19 and 20] can be represented in a Chinese Restaurant Process (CRP). Suppose a restaurant has infinite number of round tables 1, 2, . . .,∞. Further assume that infinite number of customers can sit at each table. Customers in the restaurant can sit in the table with the following distribution.

i) First customer must choose first table to sit.

ii) For n^{th} customer, it chooses the new table with probability ($\alpha/\alpha+n-1$) and chooses the occupied table with probability ($c/\alpha+n-1$) where $\alpha>0$, is called concentration parameter of the process and c is the number of customers previously sitting in the table. The equivalent relation for this distribution is also shown in equations (3.9), (3.10) and (3.15).

[3]http://en.wikipedia.org/wiki/Zipf%27s_law

For example, suppose five customers $C_1,\ldots,C_5$are seated according to the following arrangement shown in fig. 3.2. Circles are the table and the list above the circles are the customers. In fig. 3.2, C_1,C_5 are customers sitting in table T_1. $2/\alpha+5$ is the probability distribution for next customer i.e. the 6th customer.

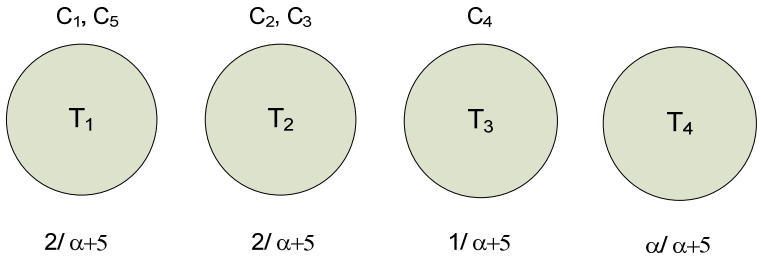


Fig.3.2 Chinese Restaurant Process (CRP) showing a seating arrangement for five customers. Each table (circle) can hold infinite number of customers.

The probability of current sitting arrangement of the five customers in fig. 3.2 is calculated by:

$$P(t_{C1},\ldots,t_{C1}) = P(t_{C1})P(t_{C2}\mid t_{C1})P(t_{C3}\mid t_{C1},t_{C2})P(t_{C4}\mid t_{C1},\ldots,t_{C3})P(t_{C5}\mid t_{C1},\ldots,t_{C4})$$

$$= \frac{\alpha}{\alpha}\times\frac{\alpha}{\alpha+1}\times\frac{1}{\alpha+2}\times\frac{\alpha}{\alpha+3}\times\frac{1}{\alpha+4}$$

t_{C1} is the table where C_1 is seated. In fig. 3.2, t_{C1} is T_1. The above seating arrangement places five customers into three groups $T_1(C_1,C_5)$, $T_2(C_2,C_3)$ and $T_3(C_4)$. If we interchange customers of table T_1 with other table and vice versa, the probability of sitting arrangement of customer's remains constant. This is the *exchangeability property* of CRP. In this way, a distribution of infinitely large customers can be defined by the CRP.

4 Baseline model and its extensions

In this chapter we discuss four models for learning morphemes from a corpus. Latter three models employ different grammatical structure of words and are the extension of the first model which is given the name *Baseline model*. The categorization process of a morpheme places it into one of three groups: prefix, stem and suffix from a single list of morphemes induced from the *Baseline model*. The segmentation task using the induced morphemes is described in section 4.5. First we describe the *Baseline model*.

4.1 Baseline model

This model is particularly simple and assumes that the words are independent units of language. Put differently, the occurrence of a word does not depend on the occurrence/non-occurrence of any other word in the corpus. This is known as the independence assumption. Possible morpheme candidates are generated by splitting the word into all possible single splits. We further make the same independence assumption to the generated morpheme segments. These morphemes and their weights are stored in a cache. The weight of a morpheme is derived from the frequency of parent words. Parent words for a morpheme are words which contains the morpheme at least once in the corpus.

The morpheme segments in this model are represented as menu items as in a restaurant (shown in fig. 4.1). The words are analogous to customers and they are allowed to choose an item from the listed menu or choose a new item. Under this scenario, we first split a word in all possible single splits. If the segment is available in the cache of menu list, the model chooses it from the listed menu. If it is not found, it will label the segment as a new menu item and will add to the menu list. Note that initially there are no items in the cache of the menu list (i.e. morphemes).

Let $m_1,...,m_k$ be the morphemes present in the cache. *Dirichlet Process* (DP) sampler uses the following relation to define the predictive probability of the next morpheme m_{k+1}:

$$P(m_{k+1} \mid m_1,...,m_k,\alpha,G_0) = \begin{cases} \dfrac{N_{k+1}}{\alpha + H}, & \text{If } m_{k+1} \in cache. \\ \dfrac{\alpha}{\alpha + H}, & \text{If } m_{k+1} \notin cache \text{ then } cache := cache \cup \{m_{k+1}\}.\ m_{k+1} \sim G_0. \end{cases} \quad (4.1)$$

Here H is the count of previously processed morphemes, N_{k+1} is the number of parent words contained by the morpheme m_{k+1} in the cache and G_0 is the base distribution. The parent words for a morpheme in a corpus are all words for which the morpheme is a substring. The concentration parameter $\alpha > 0$ determines the variance in the probability of morphemes i.e. if we use $\alpha = 0$, we always use cache. In a corpus, large α means large number of morphemes. This part captures the morphemes frequency behavior in the corpus which follows the power law distribution. Equation (4.1) can be derived using conjugacy of Multinomial-Dirichlet distribution [20]. The final probability of morpheme segment is given by equation (4.2).

$$P(m_{k+1} \mid m_1,...,m_k,\alpha,G_0) = \frac{\alpha \times G_0 + N_{k+1}}{\alpha + H} \quad (4.2)$$

For example, all possible single splits of word "chunked" are: c+hunked, ch+unked, chu+nked, chun+ked, chunk+ed, chunke+d and chunked+$. We can calculate the probability of each split using equation (4.2). For example, the probability of "chunk+ed" is:

$$P(w = chunk + ed \mid m_1,...,m_k,\alpha,G_0) = \frac{\alpha \times G_0 + N_{chunk}}{\alpha + H} \times \frac{\alpha \times G_0 + N_{ed}}{\alpha + H}.$$

The value of α is a constant and G_0 is uniform for all morpheme segments. Initially, $H=0$ and is updated according to the number of words processed during the learning.

For a word that has n possible single splits, we compute the probability of each split $P_1,...,P_n$. For the word "chunked" we get 7 possible split probabilities $P_1,...,P_7$. We then normalize the n probabilities thus obtained for all splits and calculate the weighted value using the relation (4.3).

$$Weighted\ Value = Normalized\ Probability \times Frequency\ of\ word \quad (4.3)$$

Where $Normalized\ Probability = \frac{P_k}{\sum_{j=1}^{n} Pj}$ for all $1 \le k \le n$ and P_k is the probability of k^{th} splits given by the relation (4.2).

Finally, we update the weighted value in the cache for each morpheme segment as a morpheme weight.

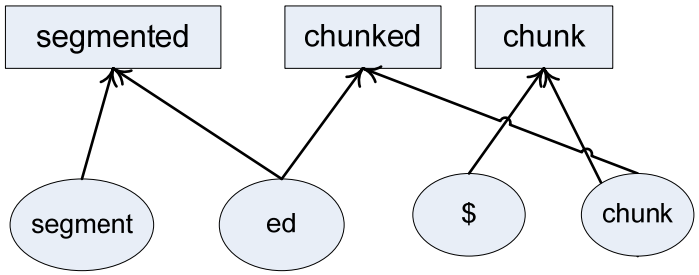


Fig. 4.1 Cache representation of morphemes "$", "chunk", "ed", "un", "segment".

In Fig. 4.1, the boxes represent the words and ovals represent the morpheme segments. These morphemes are the part of the cache. In the oval, a morpheme "ed" is connected with "segmented" and "chunked". Hence, the weight for "ed" is derived from the words "segmented" and "chunked" and stored in the morpheme cache. Similarly, the weights of other morphemes are derived and added in the cache.

4.2 Grammar model– I

In the basic model, we allowed all the generated morphemes to be placed in a single morpheme cache. In this model we extend it by distinguishing prefix, stem and suffix distribution for a word. This gives three morpheme caches. We assume that a word can be made up from one of the three ordering of a prefix, stem and suffix in successive positions. First, a word can have a prefix and a stem. Secondly, a word can have a stem and a stem. Lastly, a word can have a stem and a suffix. The probability of a word w with a single split $s+t$ given their respective models is:

$$\begin{aligned} P(w = s + t \mid M_{prefix}, M_{stem}, M_{suffix}) = \quad & P(s \mid M_{prefix}) \times P(t \mid M_{stem}) \\ & + P(s \mid M_{stem}) \times P(t \mid M_{stem}) \\ & + P(s \mid M_{stem}) \times P(t \mid M_{suffix}). \end{aligned} \tag{4.4}$$

In (4.4), M_{prefix}, M_{stem} and M_{suffix} are the respective morpheme models for prefixes, stems and suffixes. We built the respective three caches for the three morpheme models. Equation (4.4) can be divided into 3 parts and rewritten as,

$$P(w = s + t \mid M_{prefix}, M_{stem}, M_{suffix}) = P_1 + P_2 + P_3$$

where,
$$P_1 = P(s \mid M_{prefix}) \times P(t \mid M_{stem}),$$
$$P_2 = P(s \mid M_{stem}) \times P(t \mid M_{stem})$$
and
$$P_3 = P(s \mid M_{stem}) \times P(t \mid M_{suffix})$$

During model learning, we update the models M_{prefix} and M_{stem} with probability P_1. Similarly we update the respective models with probability P_2 and P_3.

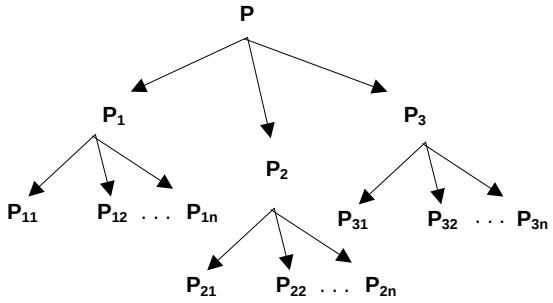


Fig. 4.2 Model updates mechanism of a word having *n* segments set.

In case of the segmentation of the word “chunked” as “chunk+ed”, associated with probability P_1, “chunk” is taken as a prefix and “ed” is taken as a stem. Similarly with probability P_2, “chunk” is considered as a stem and “ed” is considered as a stem as well. Finally, with probability P_3, “chunk” is considered as a stem and “ed” is considered as a suffix. In this way, we update the probabilities of *n* number of single spits in each model for a word and are also shown in fig. 4.2.

4.3 Grammar model– II

Similar to *Grammar model– I*, this also distinguishes the prefix, stem and suffix distribution of a word from the single morpheme list generated from the *Baseline model*. It also assumes that a word can have a prefix, stem and suffix in successive positions. Unlike, *Grammar model– I*, it assumes that a word can have a single stem only in this model. The probability of a word *w* with a single split *s+t* given the respective models is:

$$\begin{aligned} P(w = s + t \mid M_{prefix}, M_{stem}, M_{suffix}) = \quad & P(t \mid M_{stem}) \\ & + P(s \mid M_{prefix}) \times P(t \mid M_{stem}) \\ & + P(s \mid M_{stem}) \times P(t \mid M_{suffix}) \end{aligned} \tag{4.5}$$

Similar to Eq. (4.4), in Eq. (4.5) M_{prefix}, M_{stem} and M_{suffix} are the respective morpheme models for prefixes, stems and suffixes. We also built the respective cache for each model. Eq. (4.5) can also be divided into 3 parts and rewritten as,

$$P(w = s + t \mid M_{prefix}, M_{stem}, M_{suffix}) = P_1 + P_2 + P_3$$

$$\text{where,} \quad P_1 = P(t \mid M_{stem}),$$
$$P_2 = P(s \mid M_{prefix}) \times P(t \mid M_{stem})$$
$$\textit{and}$$
$$P_3 = P(s \mid M_{stem}) \times P(t \mid M_{suffix})$$

In the model learning phase we update these three models in two stages. First, we update the model M_{stem} with probability P_1. Similarly we update the respective models with probability P_2 and P_3. This update mechanism is analogous to the previous *Grammar model–I* described in section 4.2 and we name this update mechanism as *stage–0*. After the iterations in *stage–0*, we further iterate this learning process by allowing a morpheme to be placed in one of these three morpheme tables. This is the *stage–1* process. In this stage, we sample a morpheme from the three tables and select one according to their probability. We allow the given morpheme to be placed in a selected morpheme table only. This process helps to categorize the given morpheme into one of prefix, stem or suffix. The evaluation results for both *stage–0* and *stage–1* are shown in Chapter 5.

4.4 Grammar model– III

This model follows from the same independent assumption that a word appears independent of all others in a corpus. First, these words are clustered in paradigms which are created according to the Chinese Restaurant Process (CRP). A paradigm consists of words that are analogous to customers seated in a restaurant table. This process allows infinite number of words to be placed in a paradigm. The predictive probability of next word is calculated using the equation (4.1). In this case, it uses count of words instead of morphemes. When a word is taken as an input, a paradigm is

assigned to it. A paradigm is a word family that contains some words of a corpus and these words are responsible for learning the morphemes within the paradigm. Put differently, all the paradigms are treated individually during the morphemes learning process. In this way, words are clustered into different paradigms. The number of paradigms learnt is controlled by changing the value of *concentration parameter* (α).

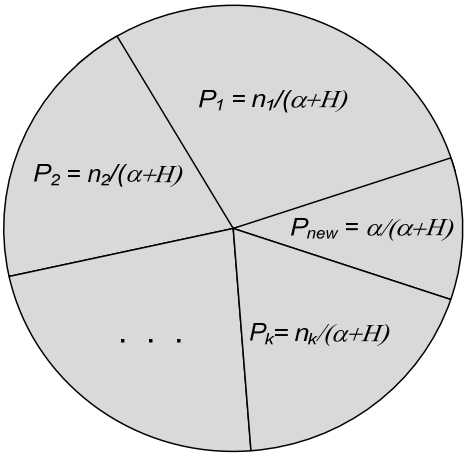


Fig. 4.3 Probability of word w_{m+1} given the previously learnt k tables (paradigms).

Suppose, the input corpus contains n words $w_1,\ldots,w_n$. Initially, w_1 is permitted to sit on the first table (paradigm). Now, the second word w_2 can either sit in the previous table (paradigm) or can select a new table (paradigm) using the relation (4.1). Let there be k tables (paradigms) learned from the n words and $n_1, n_2, \ldots, n_m$ be the number of words in k paradigms respectively. Then, $P_1, P_2,\ldots,P_k$ and P_{new} are the probabilities for $n+1^{th}$ word and is calculated using equation (4.1). This is shown in figure 4.3.

In this way, we induce all the paradigms for a given corpus. After the induction of paradigms, we used *Grammar model−II* described in section 4.3 to learn the morphemes of words individually in all induced paradigms.

4.5 Morpheme segmentation

So far we have learnt all the morphemes from the input corpus. The next task is to segment the given words using induced morphemes from models presented in section 4.1 to 4.4. We have decided to use two techniques for finding the final segments of a word. In the first technique, single splits with the highest weighted values are selected. The weighted values for all splits of a word are calculated using equation (4.3). This is the Maximum Posteriori (MAP) estimation for final segments of a word given the models. Initially, the weights of all single splits for a word are set to zero and are updated in successive iterations by the weighted value. See section 4.1 for the detail calculations of weighted value for a morpheme. After completion of sufficient iterations, we choose the segments set having maximum weight for a word. This method will give us only two segments of a word.

In the second approach, we align the morpheme sequence using dynamic programming, popularly known as the Viterbi alignment. It is used to find the best morpheme sequence according to the weighted values measured after the completion of model learning. All the grammar models described in section 4.1 to 4.4 have used the following criteria to validate the candidate morphemes during the sequence alignment. Let $m_1, m_2, \ldots, m_n$ be the morpheme sequence of a word. Then,

a. At least one of the morpheme m_i should be in stem category.

b. Stems can be alternate with prefixes and suffixes but m_n can't be a prefix and m_1 can't be a suffix of words.

c. For $1 < i \leq k$, if m_i is a prefix, then m_{i+1} must be a stem or a prefix.

d. For $1 < i \leq k$, if m_i is a suffix, then m_{i-1} must be a stem or a suffix.

The same test strategies are applied in [10 & 11] to validate the segmentation of a word among its all possible segmentations. In case of *Grammar model−III* described in section 4.4, we identify the paradigm with the highest count for a given word and select

the paradigm during the word segmentation. We have used the induced morphemes from the selected paradigm to segment a word and apply the above morpheme validation criteria under the selected paradigm.

5 Evaluation

This chapter describes and discusses the performance from the evaluation of the models outlined in Chapter 4. Before presenting the results, a brief description of the dataset and evaluation metrics is provided. The system for English and Nepali language is evaluated as described in the following sections. The results have been compared with the Morpho Challenge [24] results. Finally, result from the segmentation of Turkish words is also included.

5.1 Dataset preparation

For English, we have used the training and test set provided by the Morpho Challenge. The *Baseline model* and *Grammar model – I* are tested on the 2005 dataset provided by the Morpho Challenge. In this dataset, the training set contains 167,377 word types from 24,447,034 words and the test set contains 532 distinct words. In case of *Grammar model – II*, we used both 2005 and 2009 dataset provided by Morpho Challenge. In 2009 dataset, the training set contains 384,903 distinct words from 62,185,728 words. Test set contains 466 distinct words.

For Nepali, we have extracted our corpus from daily newspapers as well as the weekly, half monthly and monthly magazines published during 2009-10. We have pre-processed it to create a word-frequency list given as the input to our system. This corpus contains 819,506 word types from 44,856,552 words. We have tested our model on basic Nepali verbs only. The test set is prepared by manual segmentation of 1,500 randomly chosen distinct words from the training set. We separated out all the verbs as training set from the corpus. The set contains 34,391 distinct words.

We also have tested this system for the Turkish language. Both the training and test dataset for Turkish is provided by the Morpho Challenge.

5.2 Evaluation metric

Evaluation scripts are provided by the Morpho Challenge [24] along with the training and test dataset. We have used the same evaluation metrics described in Morpho Challenge [24]. The Precision (*P*), Recall (*R*) and F-Measure (*F*) are calculated in terms of morpheme boundary. These are calculated using the following formula.

$$P = \frac{H}{H + I} \tag{5.1}$$

$$R = \frac{H}{H + D} \tag{5.2}$$

$$F = \frac{2H}{2H + I + D} \tag{5.3}$$

Here, *H* is the count of correct boundary hits, *I* is the count of boundary markers incorrectly positioned and *D* the count of boundary markers not placed. These counts are based on the comparison with the gold standard. For example, if the word "sprinting" is segmented as "s+print+ing", where '+' is the boundary marker, then the first '+' is counted as an incorrect boundary insertion and the second '+' is counted as the correct boundary hit (when compared with the correct segmentation "sprint+ing"). Similarly, in case of "un+segmented", the first '+' is counted as a correct hit. It also counts the missing boundary in between "segment" and "ed".

In this system, we counted all the insertion, deletion and correct boundary hits of the test words and used equations (5.1), (5.2) and (5.3) to evaluate Precision (*P*), Recall (*R*) and F–Measure (*F*) respectively.

5.3 Results

We have evaluated performance on both English and Nepali dataset described in section 5.1. The four models described in Chapter 4 are run with these dataset. We have set the value of α to 100 for our experiments. This is an empirical value helps to use not only the cache but also the base distribution during learning i.e. allows more morpheme

numbers. The minimum length of a word is set to 4 for segmentation. This threshold controls over-segmentation by limiting the shorter stems of words. In *Grammar model– III*, we set $\alpha = 1$. This small value helps to limit the minimum number of paradigms. We have also evaluated our system on Turkish language. The evaluation metric described in section 5.2 gives the following results.

Table 5.1 Evaluation results of the systems on English assuming two segments only.

Models	Precision (P)	Recall (R)	F-Measure (F)	Stage
Baseline	56.84%	46.18%	50.96%	NA
Grammar Model– I	58.65%	47.60%	52.55%	NA
Grammar Model– II	78.34%	55.02%	64.64%	0
	77.52%	53.82%	63.53%	1
Grammar Model– III	83.16%	51.26%	63.42%	NA

Table 5.2 Evaluation results of the systems on Nepali assuming two segments only.

Models	Precision (P)	Recall (R)	F-Measure (F)	Stage
Baseline	51.25%	37.19%	43.10%	NA
Grammar Model– I	53.16%	38.12%	44.40%	NA
Grammar Model– II	86.99%	68.21%	76.46%	0
	85.29%	66.72%	74.87%	1
Grammar Model– III	82.45%	56.89%	67.33%	NA

Table 5.3 Evaluation results of the systems on English assuming multiple segments.

Models	Precision (P)	Recall (R)	F-Measure (F)	Stage
Baseline	29.36%	43.69%	35.12%	NA
Grammar Model– I	45.81%	22.12%	29.83%	NA
Grammar Model– II	54.42%	66.09%	59.69%	0
	52.96%	65.96%	58.75%	1
Grammar Model– III	40.00%	49.34%	44.18%	NA

Table 5.4 Evaluation results of the systems on Nepali assuming multiple segments.

Models	Precision (P)	Recall (R)	F-Measure (F)	Stage
Baseline	42.08%	6.10%	10.66%	NA
Grammar Model– I	53.92%	43.58%	48.20%	NA
Grammar Model– II	72.85%	56.52%	63.65%	0
	72.84%	56.35%	63.54%	1
Grammar Model– III	70.00%	53.03%	60.34%	NA

Table 5.5 Evaluation results of the systems on Turkish assuming two segments only.

Models	Precision (P)	Recall (R)	F-Measure (F)
Baseline	61.56%	21.74%	32.13%
Grammar Model– I	58.81%	20.22%	30.90%
Grammar Model– II	74.36%	18.18%	30.30%
Grammar Model– III	71.40%	17.02%	27.48%

Table 5.6 Evaluation results of the systems on Turkish assuming multiple segments.

Models	Precision (P)	Recall (R)	F-Measure (F)
Baseline	74.07%	2.14%	4.16%
Grammar Model– I	37.48%	11.80%	17.94%
Grammar Model– II	60.87%	36.19%	45.39%
Grammar Model– III	59.60%	33.96%	43.27%

Table 5.7 Evaluation results of the *Grammar model– II* on English with Morpho Challenge 2009 dataset.

Results of words with two splits			*Results of words with multiple splits*			
Precision (P)	**Recall (R)**	**F-Measure (F)**	**Precision (P)**	**Recall (R)**	**F-Measure (F)**	**Stage**
56.84%	46.18%	50.96%	29.36%	43.69%	35.12%	**0**
51.25%	37.19%	43.10%	42.08%	6.10%	10.66%	**1**

Table 5.8 Evaluation results of the *best systems*[4] on English.

Models	Precision (P)	Recall (R)	F-Measure (F)
Allomorfessor [23]	68.98%	56.82%	62.31%
Morfessor Baseline [5]	74.93%	49.81%	59.84%

[4] http://research.ics.aalto.fi/events/morphochallenge2009/eng1.shtml

5.4 Significance Tests

Results presented in Figure 6.3 and 6.4 shows that the *Grammar Model−II* has the best overall numbers among all the other models. We carry out a statistical significance test between the results of the baseline model and the *Grammar Model−II*. The null hypothesis in our test states that the two approaches are not different. The randomisation tests involves shuffling of the precision and recall scores and reassigning them to one of the two approaches. The assumption is that if the difference in performance is significant, random shuffling will only very infrequently result in a larger performance difference. The relative frequency of this event occurring can be interpreted as the significance level of the difference. We use the *sigf* package [32] developed by Sebastian Pado for randomised significance tests.

We perform the shuffling 100,000 times and thus obtain the p-value (2-tailed) of 9.9999e-6 for both precision and recall. These values shows the baseline mode and the *Grammar Model−II* are significantly different from each other and therefore reject the null hypothesis.

6 Discussion and future work

In earlier chapters, we have analyzed our morpheme learning and segmentation algorithm for different cases of words formation. The result shown in section 5.3 establishes that the algorithm is capable of segmenting words of a corpus in an unsupervised way. However the performance is not as good as the state of the art for English [27]. The best system in Morpho Challenge achieved 62.31% in terms of F–Measure [23] and is also shown in Table 5.8.

One possible explanation of this comes from the Zipfian nature of the words in a corpus. Highly frequent words have greater influences and thereby degrade the quality of segments. The result shown by the grammar model is improved in the MAP estimation where the weights of frequently appeared morphemes are distributed among prefixes, stems and suffixes lists.

In the cache model i.e. CRP, the α parameter lies form 0 to infinity. The response of this model is less sensitive to the changes in the α-value. We test this intuition by varying α from 1 through 7 with a step increment in a sample dataset of size 32,692 Nepali word types. The resulting F–Measure from *Grammar Model−II* is stable and is shown in Figure 6.5. From equation (4.1) we can see that there is always some probability mass missing to choose the new table i.e. when not using cache. This is so because there is always some chance associated to select the new cache for the $(n+1)^{th}$ morpheme. In the case when α is 0, the cache (morpheme cache) is used. When α is very large, for example 10000000, the base distribution is always chosen. In our experiments new morphemes are more likely to be from the data i.e. from the learned morpheme stored in the cache.

For Nepali, there is no benchmark to compare unsupervised approach to morphological segmentation. In our experiment, the *Grammar Model−I* shows some improvement over the *Baseline model*. This is not the same case in English where we can see the degradation in performance (see Table 5.2 and 5.4). This is due to the nature of our test

set for Nepali where we have chosen only verbs with a single stem and multiple suffixes.

The result from *Grammar model– II* shown in Table 5.1, 5.2, 5.3 and 5.6 are much better than the previous *Baseline* and *Grammar Model–I*. In the single split case (Table 5.1 and 5.2), the stems are more accurately identified compared with the previous models. For example, words खोतलिएको and खोतली, खोज्नै and खोज्दैनौ are correctly segmented with stems खोतल, खोज respectively. However, the compound suffixes that are attached to the above stems are not identified at all. Some of the words are also over segmented. For example, the words भगाएकी = भ+गाएकी andचिनिएकी = चिनिए+की are incorrectly segmented. This is due to the influence of highly frequent morphemes. This problem is inherited from the *Baseline model*.

In the next stage, the input corpus is segmented with multiple splits as shown in Table 5.3. As compared with the result of *Grammar model–I*, it improves on precision but suffers on recall due to incorrect segmentations. Words such as “humaneness = human+en+es+s”, “humanize = human+is+e”, “humanlike = human+lik+e” and “clothlike = cloth+lik+e” are over segmented. Words such as “fire-fighter = fire-fight+er”, “firebush = firebush” and “abductions = abduction+s” are under segmented.

To summarize, multiple splits mostly produces over segmentation of words. In 2009 Morpho challenge dataset, many hyphenated words are incorrectly segmented and this is another shortcoming of our model. The results are shown in Table 5.7.

We have also tested our system on words having single stem and single affix. The result is shown in Table 5.7. Some incorrectly segmented words are crise=cris+e, viruses=viru+ses, bodies=bod+ies, sprinting=s+printing, hogging=hog+ging, sharing=s+haring and ceremonies=ceremony+ies. Many words are incorrectly segmented due to changes in spelling during word formation. This model does not have

a mechanism to capture the character changes during the morpheme segmentation process.

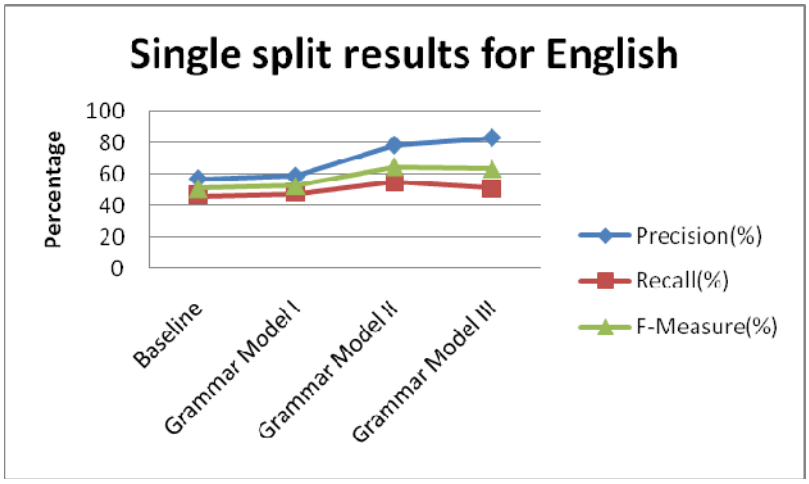


Fig. 6.1 Single split results for English.

Our models utilize word frequencies alone to capture the context which results in an inefficient performance. For example, our *Grammar model– I* leaves out many words unsegmented (i.e. segmented with null suffix). In case of *Grammar model– II*, most of the words with single stem are over segmented. This model favors over segmentation with no under segmentation with single stem words.

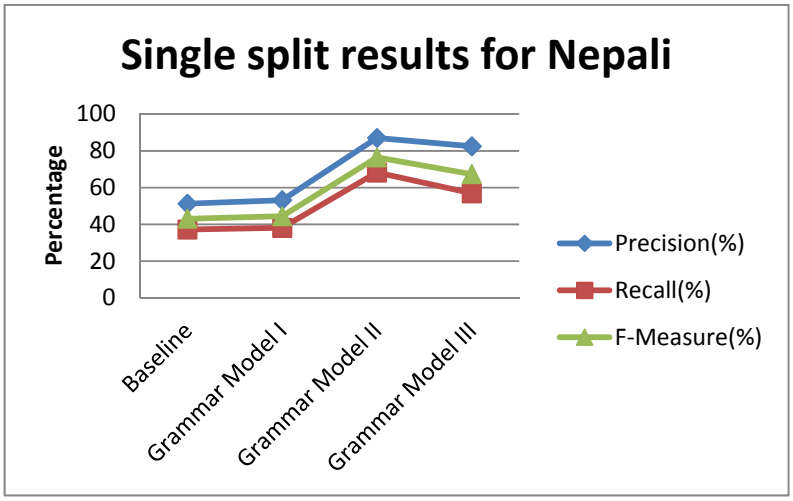


Fig. 6.2 Single split results for Nepali

Furthermore, the over segmentation errors in best sequence alignment occurs due to the influence of most frequent morphemes. We extended the *Grammar model– II* by employing the Hierarchical Dirichlet Process (HDP) [26] to overcome the influence of heavily weighted morphemes. This model is termed as *Grammar model– III*.

In *Grammar model– III*, we got improvement on precision but suffered on recall for English. The result is shown in fig. 6.1. Detail results of *Grammar model– III* assuming two segments only, multiple segments are shown in Table 5.1, 5.2, 5.3 and 5.4.

For the Nepali corpus, we found the best result is given by the *Grammar model– II* and is shown in fig. 6.2 and 6.4. We also tested our system for Turkish and the results are shown in table 5.5 and 5.6. Our systems are motivated by prefix, stem and suffix grammar. Languages like Turkish which is agglutinative in structure, the results are poor due to under segmentation of many words.

Overall, *Grammar model– II* gives the best result in English and Nepali among the four models. We found that the influences of heavily weighted morphemes are inherited in HDP implementation too.

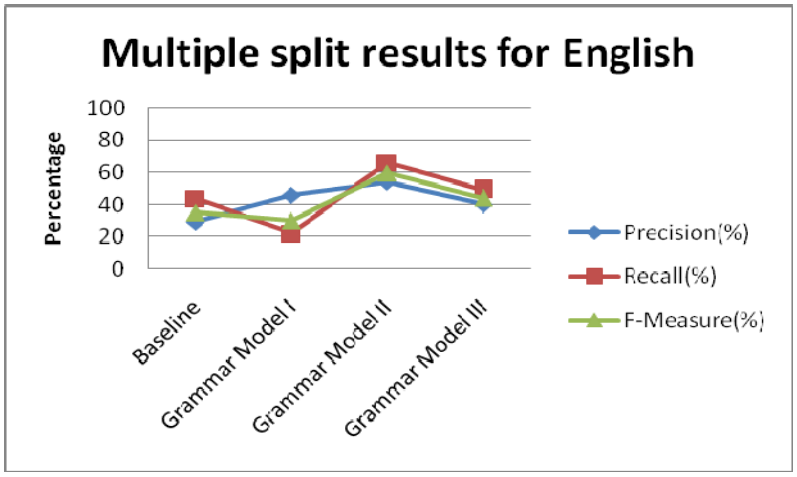


Fig. 6.3 Multiple splits results for English

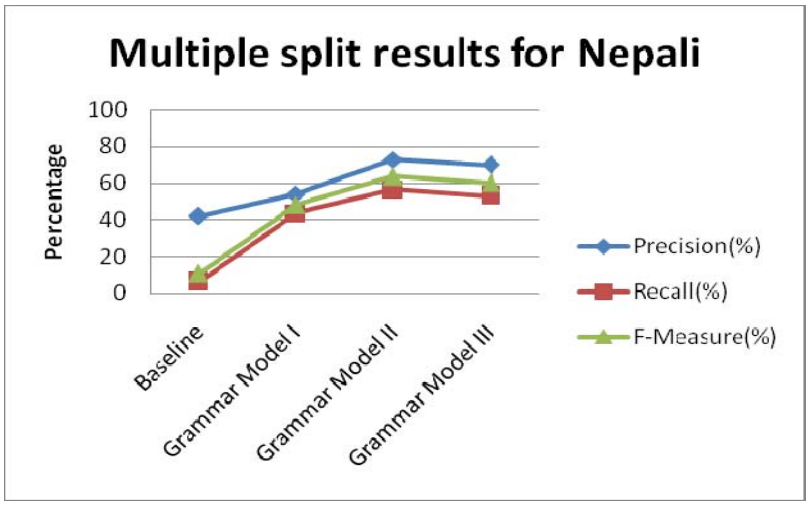


Fig. 6.4 Multiple splits results for Nepali

Many words in English and Nepali are incorrectly segmented due to character changes during morpheme attachment. This is an very general model that uses only the frequency of words for the morpheme learning process and falls short of the state of the art. As mentioned before, we got much erroneous segmentation for hyphenated words.

One possible extension is to develop a grammar model of words that could include the hyphenated words as well. Another future work can be to capture the character changes in words during morpheme attachment [8 and 10].

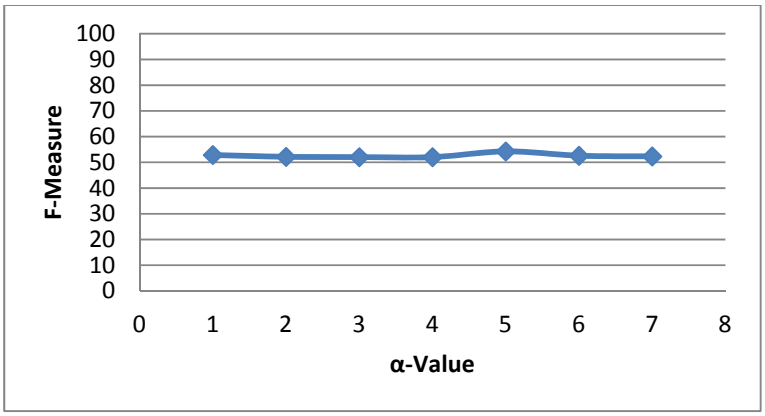


Fig. 6.5 Plot of α-value Vs. F–Measure from sample Nepali dataset.

7 Conclusion

In this thesis, I have described the task of morphological analysis by making use of the statistical properties of words in a corpus and different morpheme grammar structures. This research work has shown that the morpheme behavior follows the power law and this characteristic can be captured to segment the words into their constituent's morphemes. However, the same behavior can lead to erroneous segmentation too. This is described by discussion section in Chapter 6. In addition of our four grammar models, this research has presented the first result on unsupervised morphological segmentation for Nepali language. We have also presented a performance result of the system for Turkish language.

Appendix A: Algorithms

a. Baseline Model

```
1: InputWords←Word_Frequency
2: Morphemes ←ϕ
3: n ←SizeOfInputWords
4: for i = 1 to ndo
5:   Word←InputWords[i]
6:   SegSet←Get_All_Single_Splits (Word)
7:   m ←SizeOfSegmentsSet
8:   for j = 1 to m do
9:        ModelProb[j]←Probability(SegSet[j])
10:       end for
11:       Normalize (ModelProbability)
12:       MaxProb←Find_Max(ModelProbability)
13:       Update(MaxProb)
14:       Freq←Frequency(Word)
15:       for j=1 to Freqdo
16:            Segment ←Sample(SegSet)
17:            Update_Corpus(Segment)
18:            Morphemes ←Morphemes∪ Segment
19:       end for
20:  end for
21:  for i = 1 to ndo
22:       W←InputWords[i]
23:       Segments ←Find_MaxProb_Segments(W)
24:  print Segments
25:  end for
```

b. Grammar Model– I

```
1: InputWords←Word_Frequency
2: Prefix ←ϕ, Stem ←ϕ, Suffix ←ϕ
3: n ←SizeOfInputWords
4: for i = 1 to ndo
5:   Word←InputWords[i]
6:   Set ←Get_All_Single_Splits (Word)
7:   m ←SizeOfSegmentsSet
8:   for j = 1 to m do
9:        P1 = PfxProb(Set[j][0])*StmProb(Set[j][1])
10:            P2 = StmProb(Set[j][0])*StmProb(Set[j][1])
11:            P3 = StmProb(Set[j][0])*SfxProb(Set[j][1])
12:       ModelProb[j]← P1 + P2 + P3
10:       end for
11:       Normalize (ModelProbability)
12:       MaxProb←Find_Max(ModelProbability)
13:       Update(MaxProb)
14:       Freq←Frequency(Word)
15:       for j=1 to Freqdo
16:            Segment ←Sample(SegSet)
17:            Pfx←Sample_Prefix(Segment)
18:            Stm←Sample_Stem(Segment)
19:            Sfx←Sample_Suffix(Segment)
20:            Update_Corpus(Pfx, Stm, Sfx)
21:            Prefix←Prefix∪Pfx
22:            Stem←Stem∪Stm
23:            Suffix←Suffix∪Sfx
24:       end for
25:  end for
26:  for i = 1 to ndo
27:       W←InputWords[i]
28:       Segments ←MaxProb_Segment(W,Prefix,Stem,Suffix)
```

```
29:       print Segments
30:  end for
```

c. Grammar Model– II

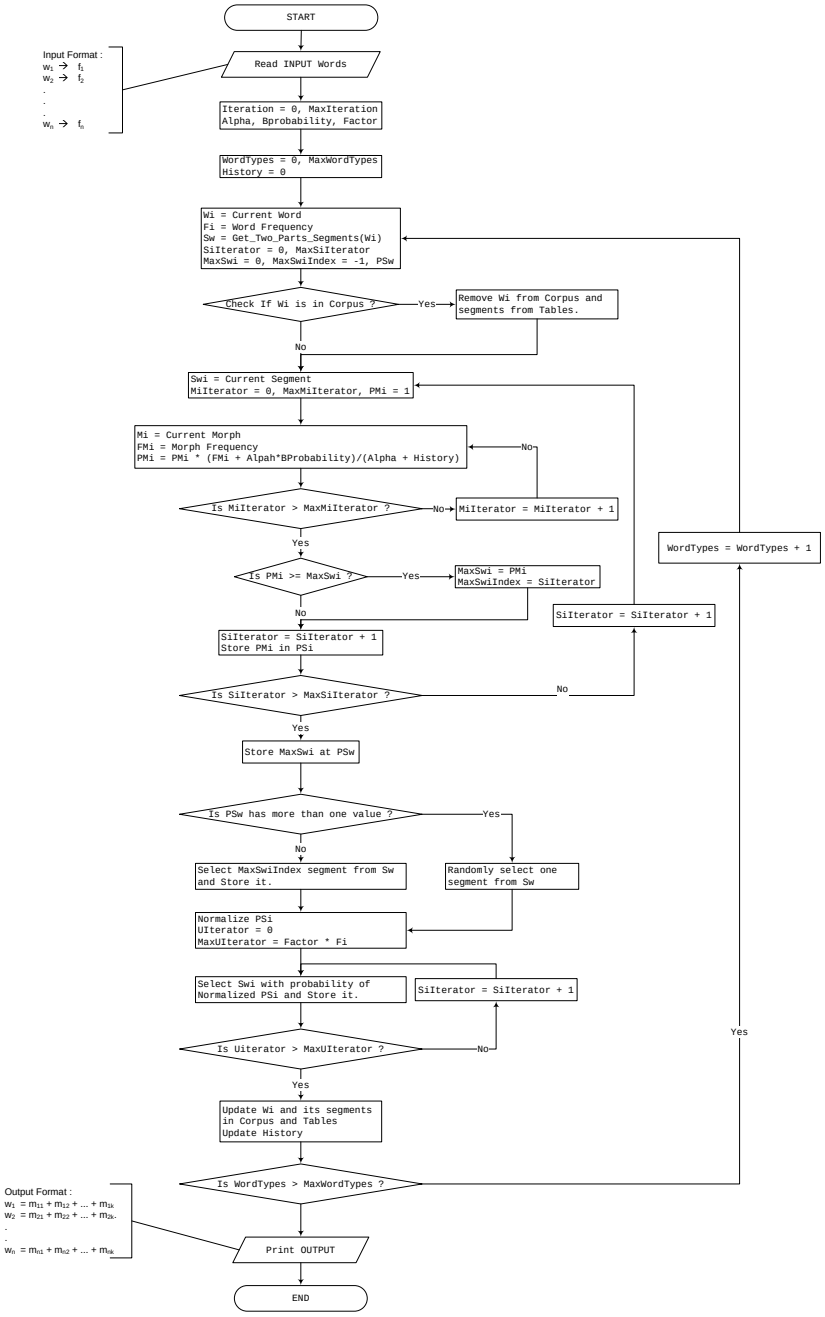
```
1: InputWords←Word_Frequency
2: Prefix ←ϕ, Stem ←ϕ, Suffix ←ϕ
3: n ←SizeOfInputWords
4: for i = 1 to ndo
5:   Word←InputWords[i]
6:   Set ←Get_All_Single_Splits (Word)
7:   m ←SizeOfSegmentsSet
8:   for j = 1 to m do
9:        if Set[j] = Word+$ then
10:            P2 = StmProb(Word)
11:            else
12:            P1= PfxProb(Set[j][0])*StmProb(Set[j][1])

13:            P3 = StmProb(Set[j][0])*SfxProb(Set[j][1])
14:            end if
15:       ModelProb[j]← P1 + P2 + P3
16:       end for
17:       Normalize (ModelProbability)
18:       MaxProb←Find_Max(ModelProbability)
19:       Update(MaxProb)
20:       Freq←Frequency(Word)
21:       for j=1 to Freqdo
22:            Segment ←Sample(SegSet)
23:            if Segment = Word + $ then
24:            Stm←Sample_Stem(Segment)
25:       else
26:                 Pfx←Sample_Prefix(Segment)
27:                 Stm←Sample_Stem(Segment)
28:                 Sfx←Sample_Suffix(Segment)
29:       end if
30:            Update_Corpus(Pfx, Stm, Sfx)
31:            Prefix←Prefix∪Pfx
```

```
32:             Stem←Stem∪Stm
33:             Suffix←Suffix∪Sfx
34:         end for
35:     end for
36:     for i = 1 to ndo
37:         W←InputWords[i]
38:         Segments ←MaxProb_Segment(W,Prefix,Stem,Suffix)
39:         print Segments
40:     end for
```

d. Grammar Model– III

```
1: InputWords←Word_Frequency
2:   Paradigms ←ϕ
3: n ←SizeOfInputWords
4: for i = 1 to ndo
5:   Word←InputWords[i]
6:   Paradigm←Get_Paradigm(Word)
7:   Paradigms ←Paradigms∪ Paradigm
8: end for
9: Cnt←Count(Paradigm)
10:  for i = 1 to Cntdo
11:       Paradigm ← Paradigms[i]
12:       Model(Pfx,Stm,Sfx)[i]←GrammarModel-II(Paradigm)
13:  end for
14:  for i = 1 to ndo
15:       W←InputWords[i]
16:       Segments ←Find_MaxProb_Segments(W, Paradigms)
17:  print Segments
18:  end for
```

Appendix B: Program Flowchart (Baseline Model)



References

1. Harris, Z. S.: From Phoneme to Morpheme. In: Language, 31(2):190-222 (1955)
2. Hafer, M. A., Weiss, S. F.: Word segmentation by letter successor varieties, In: Information Storage & Retrieval, 10:371-385 (1974)
3. Dang, M. T., Chaudri, S.: Simple unsupervised morphology analysis algorithm (SUMAA). In proceedings of the PASCAL Challenges Workshop on Unsupervised Segmentation of Words into Morphemes, Venice, Italy (2006)
4. Keshava, S., Pitler, E.: A simpler, intuitive approach to morpheme induction. In PASCAL Challenge Workshop on Unsupervised Segmentation of Words into Morphemes. (2006)
5. Creutz, M., Lagus, K.: Unsupervised discovery of morphemes. In proceedings of the ACL-02 workshop on Morphological and phonological learning - Volume 6 pages 21-30. (2002)
6. Creutz, M., Lagus, K.: Unsupervised Segmentation of Words Using Prior Distributions of Morph Length and Frequency. In proceeding of the 41st meeting of ACL-03, pages 280-287. (2003)
7. Goldsmith, J.: Unsupervised Learning of the Morphology of a Natural Language. Computational Linguistics, 27:2, pages 153-198. (2001)
8. Baroni, M., Matiasek, J., Trost, H.: Unsupervised discovery of morphologically related words based on orthographic and semantic similarity. In proceedings of the ACL workshop on Morphological and phonological learning Volume 6, pages 48 – 57. (2002)
9. Demberg, V.: A Language-Independent Unsupervised Model for Morphological Segmentation. In proceedings of the 45th Annual Meeting of the ACL, pages 920-927. (2007)
10. Dasgupta, S., Ng, V.: High-Performance, Language-Independent Morphological Segmentation. In proceedings of NAACL HLT 2007, pages 155-163. (2007)
11. Dasgupta, S., Ng, V.: Unsupervised word segmentation for Bangla. In proceedings of ICON, pages 15- 24. (2007)

12. Rissanen, J.: Stochastic Complexity in Statistical Inquiry. Word Scientific Publishing Co, Singapore. (1989)
13. Zeman, D.: Unsupervised Acquiring of Morphological Paradigms from Tokenized Text. In CLEF 2007. LNCS, vol. 5152, pp. 892–899. Springer, Heidelberg. (2007)
14. Can, B., Manandhar, S.: Unsupervised Learning of Morphology by Using Syntactic Categories. In Working Notes for the Cross Language Evaluation Forum (CLEF) Workshop, Corfu, Greece. (2009)
15. Newman, M.: Fast algorithm for detecting community structure in networks. Physical Review. E 69, 0066133. (2004)
16. Bernhard, D.: MorphoNet: Exploring the Use of Community Structure for Unsupervised Morpheme Analysis. Working Notes for the CLEF 2009 Workshop Corfu, Greece. (2009)
17. Wallach, H.M., Jensen, S.T, Dicker, L., Heller, K.A.: An Alternative Prior Process for Nonparametric Bayesian Clustering, Appearing in Proceeding of the 13th International Conference on AI and Statistics(AISTATS), Chia Laguna Resort, Sardinia, Italy. (2010)
18. Zipf, G.K.: Selected Studies of the Principle of Relative Frequency in Language.Cambridge, MA.: Harvard University Press. (1932)
19. Ishwaran, H., James, L.F.: Generalized weighted Chinese restaurant process for species sampling mixture models. StatisticaSinica, 13:1211-1235. (2003)
20. Zhang, X.: A Very Gentle Note on the Construction of Dirichlet Process. The Australian National University, Canberra. (2008)
21. Johnson, M., Griffiths, T.L., Goldwater, S.: Adaptor Grammars: A Framework for Specifying Compositional Nonparametric Bayesian Models. In Advances in neural information processing systems. (Vol. 19). (2007)
22. Johnson, M.: Unsupervised word segmentation for Sesotho using Adaptor. Grammars, Proceedings of the Tenth Meeting of the ACL Special Interest Group on Computational Morphology and Phonology, pages 20-27. (2008)
23. Virpioja, S., Kohonen, O.: Unsupervised Morpheme Discovery with Allomorfessor. In Cross Language Evaluation Forum (CLEF). (2009)

24. Unsupervised segmentation of words into morphemes - Challenge 2005, http://www.cis.hut.fi/morphochallenge2005/evaluation.shtml
25. Goldwater, S.: Nonparametric Bayesian models of lexical acquisition. Ph.D. thesis, Brown University. (2006)
26. Johnson, M., Blunsom, P., Cohn, T., Goldwater, S.: A note on the implementation of Hierarchical Dirichlet Process. Proceedings of the ACL-IJCNLP Conference Short Papers, pages 337-340, Suntec, Singapore. (2009)
27. Unsupervised segmentation of words into morphemes - Challenge 2009, http:// www.cis.hut.fi/morphochallenge2009/eng1.shtml
28. Blackwell D., MacQueen J.B.: Ferguson Distribution VIA Pólya Urn Schemes. The Annals of Statistics. Vol. 1, No. 2, 353–355. (1973)
29. Tseng H., Jurafsky D., Manning C.: Morphological features help pos tagging of unknown words across language varieties. In the Foruth SINGHAN Workshop on Chinese Lanugage Processing. (2005).
30. Shrivastava M., Bhattacharyya P.: Hindi POS Tagger Using Naïve Stemming: Harnessing Morphological Information Without Extensive Linguistic Knowledge. 6th Internal Conference on Natural Language Processing, India. (2008).
31. Schulz S., Honeck M., Hahn U.: Biomedical Text Retrieval in Languages with a Complex Morphology. In Proceedings of the ACL Workshop on Natural Language Processing in the Biomedical Domain, pages 61–68. (2002).
32. Pado S.: User's guide to sigf : Significance testing by approximate randomisation. 2006.