English Morphological Analysis with Machine-learned Rules

Xuri TANG

Dept. Foreign Languages
Wuhan University of Science and Engineering, 430073，Wuhan, P. R. China
tangxuriyz@yahoo.com.cn

Abstract. This paper expounds an algorithm for morphological analysis of English language. The algorithm consists of two closely related components: morphological rule learning and morphological analyzing. The morphological rules are obtained through statistical learning from wordlist, with particular morphological features of English language taken into consideration. The procedure of morphological analysis considers two types of ambiguities: intersectional ambiguity and combinatory ambiguity. The procedure also considers the order of wordform formation in the language. Experiment shows that the algorithm performs distinctively compared to other algorithms.

Keyword: Morphological analysis; statistical learning; intersectional ambiguity; combinatory ambiguity; wordform formation order

1 Introduction

Automatic Morphological Analysis (CMA) is still a much discussed topic in natural language processing.[1] The objective of CMA is to understand the inner mechanism of word form formation in a language. A morphological analyzer can provide valuable information for other computer-based linguistic tasks such as lemmatization, syntactic parsing, machine translation, information retrieving, text clustering and many others.

Goldsmith (2001) classifies the work in automatic morphological analysis into four categories. The classification is done with an emphasis on how the morphological rules are obtained. The first category identifies morpheme boundaries on the basis of the degree of predictability of the n+1st letter given the first n letters. The second employs n-gram grammar and high likelihood to obtain morpheme internal structure. The third category endeavors to discover rules through phonological relationships between pairs of related words. The fourth one seeks for an analysis for a language which is the most concise and consequently boils down to a set of rules for the language.

Two factors are essential to achieve accurate automatic morphological analysis. One factor is the construction of a set of morphological rules and the other is the morphological analysis procedure. The absence or underperformance of either of them impairs the overall ability of the morphological analyzer. Thus in constructing an algorithm for the task, both factors should be addressed.

Machine learning is a promising alternative to obtain morphological rules. This method can avoid problems such as costly human labor, rule inconsistency and can provide additional statistical information which can be used in morphological analysis procedure. Based on information type employed in the machine learning task, we can usefully obtain two classes: supervised learning and unsupervised learning. The class of supervised learning (Bosch, 1999; Wicentowski, 2004) makes use of lexical database with morphological information. A good example of such lexical database is CELEX[2]. The class of unsupervised learning (Goldsmith, 2001; Kurino et al, 2004; Creutz, 2005)

[1] For example, in 2005, a challenge named Unsupervised Segmentation of Words into Morphemes – Challenge 2005 was held in Helsinki University of Technology, Finland.

[2] CELEX is a lexical information database, developed by Burnage et al. The database contains information about lemma, wordform, abbreviation, corpus tagging etc.

uses only a wordlist, with or without information of frequency along the way.

In regard of morphological analysis procedure, there are two popular techniques. One technique is stemming, represented by Porter (1980)[3]. The algorithm given in Porter (1980) consists of two stages: (1) the de-suffixing step which subtracts predefined endings from words, and the recoding step which adds possible ending part to the string obtained in the previous stage. These two stages can be done in sequence or simultaneously. One distinct feature of the algorithm is that it doesn't use a dictionary, which makes it very sufficient in analysis.

The other technique is morphological parsing, represented by Model of Two-level Morphology proposed in Koskenniemi (1983). This model considers morphological analysis and morphological transformation in different phonological situations and encodes the correspondence between surface form and lexical form with finite-state transducer. For example:

Lexical form: ***s p e c i f y + s***

Surface form: ***s p e c i f i e s***

The first morphological analysis system that adopts this model is KIMMO (Karttunen, 1983). The system has two parts: rules and lexical information such as morpheme structure and morphosyntactic constraint.

The morphological analyzer illustrated in this paper falls into the first class of Gold(2001) classification. The system aims at high accuracy of morphological analysis of English language with morphological rules obtained through unsupervised machine learning. The analyzer applies letter transitional probability proposed in Keshava&Pilter(2005) in morphological rule learning and in disambiguation of morphological analysis as well. An initial evaluation of the analyzer shows a promising result with an 88.42% precision, 78.46 recall and 83.14% F-score, which transcends the best results of English language reported in Unsupervised Segmentation of Words into Morphemes – Challenge 2005.

The second part of the paper explains the unsupervised machine learning of morphological rules. The third part introduces the control of morphological analysis procedure, focusing on disambiguation and rule order. The fourth part is a discussion of the result achieved by the system. The paper concluded with a discussion of future work.

2 Morphological rule learning

We have adopted the approach proposed by Kashava&Pilter(2005) in learning affix rules from wordlist and tested the approach using wordlist of different scales. Our experiments show that the affix rules obtained with wordlist of different scales differ over a large range. The difference of coverage and correctness of the rules obtained also result in different performance of the system.

To learn the affix rules, one forward lexicographic tree and one backward lexicographic tree are built. Potential affixes are recognized through a scoring procedure. The scoring procedure consists of two steps. In the first step, a wordform αABβ is considered. If the following three conditions:

1. αA can be found in dictionary;
2. $P(A \mid \alpha) \approx 1$, that is, in the forward lexicographic tree, the transitional probability from α to A approximates 1;
3. $P(B \mid \alpha A) < 1$, that is, in the forward lexicographic tree, the transitional probability from αA to B is less than 1;

are satisfied, Bβ is considered to be a candidate for suffix. Affix can also scored using backward lexicographic tree with symmetric conditions.

The second step scores Bβ by examines all the wordforms ending with the string. The scoring function is given below:

[3] Porter developed a new algorithm for morphological analysis and named it Porter2.

$$score(B\beta) = \begin{cases} (a) + = AwardScore \\ (b) - = PenaltyScore \end{cases}$$

Opton (a) indicates that if the string ending with Bβ meets all the conditions mentioned in the first step, the score of Bβ as suffix grows by AwardScore. Option (b) indicates that if the string ending with Bβ fails to meet all the conditions, the score of Bβ as suffix diminishes by PenaltyScore. After all the strings ending with Bβ are examined, a gold standard is applied. If the socre(Bβ) is larger than 0, the string Bβ is considered as a suffix. Otherwise, it is abandoned. Prefixes are handled in a similar way with backward tree.

Table 1 lists out the experiment results obtained with different dictionary scales. For the forward and backward lexicographic trees we use a corpus of 24,447,034 tokens. As is shown in the list, the dictionary scale used in checking condition a in the first step is a major factor effecting the number of prefix rules and suffix rules and the performance as well. The reason that can account for this difference is not difficult to find. In English language, not all the wordforms can be used as a stem from which derivational words can be formed. For example, the proper name word class，which contains proper names such as "Ann", "Al"，leads to misjudgment of the algorithm.

Table 1. Performance Comaprison with Different Dictionary Scales

index	Dic. Wordform number	Prefix number	Suffix number	Precision (%)	Recall (%)	F-score (%)
1	**167377**	683	1584	87.52	77.14	82.00
2	**57046**	694	1322	88.42	78.46	83.14
3	**14760**	373	989	90.57	72.43	80.49

3. Analysis Procedure Control

A wide-covering and correct set of affix rules is prerequisite for accurate morphological analysis. But it alone does not guarantee a successful analysis. The procedure in which the analysis is done is also crucial. In this paper, two important aspects are dealt with in terms of analysis procedure control. One is disambiguation. The other is affix rule order.

3.1 Disambiguation

Like word segmentation in Chinese, there are ambiguities in morphological analysis. A good understanding of the types of ambiguities certainly helps to solve the ambiguities. In nature, the morphological analysis is analogous to Chinese word segmentation. The categorization of ambiguity in Chinese segmentation may also apply here. Thus we have two types of ambiguity in morphological analysis: intersectional ambiguity and intersectional ambiguity.

3.1.1 Intersectional Ambiguity

Intersectional ambiguity stands for the type of strings which have more than one possible intersectional analysis. For example, given a string ABCD, B, CD, C, and D are all potential affixes.[4] Thus the key of intersectional ambiguity is to decide where the morphological boundary is. Look at the following example:

[4] Please refer to Huang(2002) and Chen(2000) for further explanation about intersectional and combinatory ambiguity in Chinese segmentation.

Exp. 1: (a) *anthropophagous[5] → anthropophagous s
anthropophagous → anthropophag ous
(b) *beneficence → beneficen ce
beneficence → benefi cence
(c) *fieldmice → fieldmi ce
fieldmice → field mice

Intersectional ambiguity is a frequently encountered phenomenon in morphological analysis and is an important factor that causes underperformance of a analyzer.

Keshava&Pilter (2005) proposes to solve the intersectional ambiguity using transitional probabilities between letters. Look at the following examples:

Exp 2: (a)*action → acti on TransProb(i,o)[6] = 0.583511
action → act ion TransProb(t,i) = 0.500998
(b) *aeroplaces → aeroplane s TransProb(e,s) =0.295008
aeroplaces → aeroplan es TransProb(n,e)=0.996983

The real morphological boundaries in Exp 2 generally have a lower transitional probability and are thus correctly identified. In our experiment the filter is set to be 0.40 and wordform with transitional probability less than 0.40 are divided.

But this method doesn't work for words that are not seen in the corpus trained for letter transitional probability. In our experiment, although the corpus is large, there are still some words unencountered:

Exp 3: (a) *pilation → pilati on TransProb(i,o)=0
pilation → pilat ion TransProb(t,i)=0

Transitional Probability can not help in disambiguation in such cases. In our system, we have resorted to an approximation which takes the most frequently used affix as analysis result. Thus, in example 3, the result is "ion"which happens to be right. We imagine that a human speaker, when faced with similar situation, would take the same action.

3.1.2 Combinatory Ambiguity

Combinatory ambiguity in morphological analysis refers to the fact that the analyzer can not decide whether a wordform should have an affix or not. That is, given a wordform AB in which B is a potential suffix, we have to decide whether AB is a word form alone and B is not an affix, or B is a real affix and A is the stem. Thus the key to combinatory ambiguity is to decide whether there is a morphological boundary inside the wordform. Here are some examples:

Exp. 4: (a)*analects → analect s
analects → analects
(b)*potion → pot ion
potion → potion
(c)*thrive → thr ive
thrive → thrive

Combinatory ambiguity is more difficult to solve than intersectional ambiguity. A simple finite automaton surely can not solve the problem, as every rule may have an exception.

To solve combinatory ambiguity, we have also chosen to rely on letter transitional probability. The solution can be explained by observing the following:

Exp. 5: (a)*letter → lett er TransProb(t,e) = 0.944217
(b) alexic → alex ic TransProb(x,i) = 0.0216901
(c) consumer → consum er TransProb(m,i) = 0.516369
(d) encode → en code TransProb(c,n) = 0

The morphological boundary generally has a much lower transitional probability. We have also set the filter to be 0.4 and divide those that have transitional probabilities less than the filter. As for

[5] Wrong analysis is marked with an asterisked (*) and a correct analysis is not marked.
[6] TransProb(l_1,l_2) is the transitional probability between letter l_1 and letter l_2.

example 5d, where the letter transitional probability is 0 and thus lower than the filter, we also consider it a real affix if the string left after subtraction is a word in dictionary. In such cases, a transitional probability of 0 indicates that, though the word is not seen in corpus, it certainly contains a morpheme and a wordform and thus probably a new word.

Experiment shows that the method mentioned above improves the performance to a large extent. However, it can not solve all combinatory ambiguities in analysis. Just like combinatory ambiguity in Chinese segmentation, simple transitional probability can not solve such specific problems. Richer contextual information such as grammatical category and lexical meaning are needed for disambiguation of this type..

3.2 Type and Order

Beard(1995) classifies morphemes into Inflectional Morphemes and Derivational Morphemes and maintains that these two types of morphemes behave differently in word formation. Inflectional morphemes form a closed class instead of an open one; it allows zero forms[7]; it generally does not allow further wordform transformation; and finally, it can't be clustered paradigmatically. Derivational morphemes, however, do not have these features.

The hypothesis mentioned above strongly suggests that a morphological analyzer should treat different morphemes in different ways. At least, the application order of these morphemes should be considered. Different morphemes participate in the formation of a wordform at different times. The sequence in general can be captures by the order given below:

Lexlical Morpheme → Derivational Morpheme → Inflectional Morpheme

In English, the same order applies. Inflectional affixes exist either in zero form, or at the end of a wordform. However, there are three exceptions for the rule. These exceptions are exemplified below:

Exp 6: (a) edly： abstractedly, admittedly, affectedly
(b) ingly: agonizingly, amusingly, lingeringly, movingly
(c) edness: bullheadedness

When doing the analysis, which is the opposite of worform formation, we use the reversed sequence:

Inflectional Morphemes → Derivational Morphemes → Lexical Morphemes

And the exact procedure taken in this system is illustrated in figure 1.

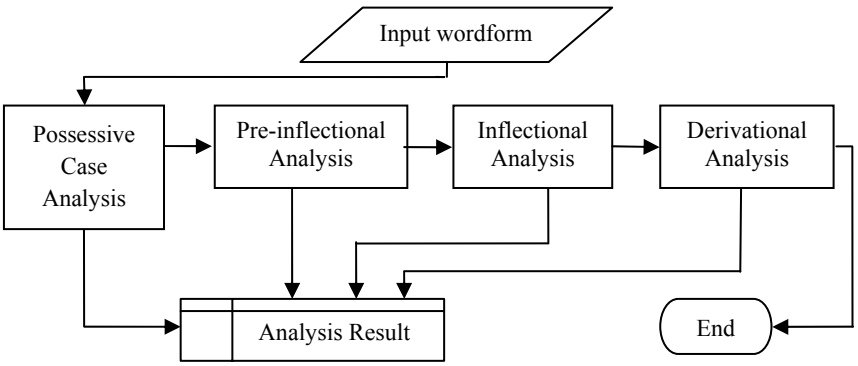

Fig. 1. Flow Chart for morphological analysis, which is the opposite of wordform formation

[7] For example, an English verb "do" may also stand for plural form of the verb.

Table 2. Performance Comarison of Different Morphological Systems

Name	Author	Precision (%)	Recall (%)	F-score (%)
RePortS	Pitler and Keshava, Univ. Yale, USA	76.2	77.4	76.8
Cheat-all	Atwell et al, Leeds and Helsinki	86.0	70.4	77.4
Cheat-top5	Atwell et al, Leeds and Helsink	83.2	74.6	78.6
This paper		88.46	78.61	83.24

4. Experiment Analysis

Experiment has been conducted to evaluate the performance of the analyzer illustrated above. The result of the experiment is compared to the results in Unsupervised Segmentation of Words into Morphemes – Challenge 2005 (Table 2[8]). As is shown in the table, our analyzer has scored a fairly better result than the other algorithms. Noticeably, our algorithm has outperformed the algorithm proposed in Pilter&Keshava(2005) on which the present analyzer is based.

Two reasons can be found to account for such an improvement in performance. The first one is the handling of ambiguity in morphological analysis. We hold that the learning of affix rules and the analysis procedure are both important in the overall performance of a morphological analysis. This belief has led to a careful consideration of ambiguity in morphological analysis and a systematic examination of the phenomenon. The classification of ambiguity into intersectional ambiguity and combinatory ambiguity, attributing to the classification in Chinese segmentation, enables us to treat the ambiguities differently. This is crucial in improving the overall performance of the analyzer. The result obtained in the experiment also proves that our belief is right and both phrases need to be handled with equal effort.

The second reason is the language-specific concern we take when building the analyzer. Because we have aimed to build a morphological analyzer specifically tailored for English, language-specific concern automatically becomes our chosen strategy. Thus many language specific morphological features, such as morpheme application order, rule exceptions and others, are considered. Another manifestation of language specific concern is parameter setting and dictionary scale. As is discussed in statistical language analysis, the performance of a system based on statistical learning relies heavily on training data. We have thus experimented with different parameter settings so as to achieve a better analyzer.

5. Conclusion

On the surface, morphological analysis in Indo-European languages does not seem to be as important as word segmentation in Chinese. But a close look at these languages gives just the opposite answer. The identification of internal structure of wordforms provides very useful information for other language analysis tasks. Such belief has been our motivation for building the analyzer for English.

[8] The same Gold Standard is used for analysis evaluation. Keshava&Pilter(2005) gives a different data from what is given here. The precision, recall and F-score are 82.84%, 79.10% and 80.92% respectively. In the table, the first three rows are the top three given in Cruetz(2005).

In building our analyzer, we have considered both the rule-building phrase and the analysis phrase. Unsupervised learning has been employed to obtain a set of affix transformational rules. Disambiguation measures have been taken to help to make the correct analysis decision. Language-specific concern has also been given priority in the construction of the analyzer. The experiment shows that the analyzer has a satisfactory performance, and the result is fairly higher than many other algorithms.

However, problems remain. The most difficult is combinatory ambiguity. Some wordforms, such as “as”, “this”, can not be correctly analyzed through setting a transitional probability filter. And we doubt that such combinatory ambiguity can be analyzed through letter transitional probabilities. A larger context, such as part of speech, or context between words, is needed for a correct analysis of these words. This can be a research topic our future studies.

References

1. Beard，Robert: Lexeme-Morpheme Base Morphology, a General Theory of Inflection and Word Formation. SUNY Linguistic Series, Albany SUNY Press (1995)
2. Bosch, Antal van den and Daelemans, Walter: Memory-based Morphological Analysis. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, ACL'99, University of Maryland, USA, June 20-26, (1999) 285-292.
3. Creutz, Mathias and Lagus, Krista: Unsupervised Morpheme Segmentation and Morphology Induction from Text Corpora Using Morfessor 1.0. In: Publications in Computer and Information Science, Report A81, Helsinki University of Technology, (2005)
4. Daille, Béatrice, Fabre, Cécile & Sébillot, Pascale: Applications of Computational Morphology. In: Boucher, Paul, Plénat, Marc (eds.) Many Morphologies. Somerville, MA, Cascadilla Press (2002) 210–234
5. Goldsmith, John: Linguistica: An Automatic Morphological Analyzer. In: Okrent, Arika and Boyle, John (eds.) The Proceedings from Main Session of the Chicago Linguistic Society’s Thirty-sixth Meeing. (2000)
6. Goldsmith, John: Unsupervised Learning of the Morphology of a Natural Language. In: Computational Linguistics, Vol 27, number 2, (2001) 153-189
7. Karttunen, Lauri: KIMMO: A general morphological processor. In: Linguistic Forum 22, (1983) 163–186
8. Keshava，Samarth & Pitler, Emily: A Simpler, Intuitive Approach to Morpheme Induction. In: Proceedings of Unsupervised segmentation of words into morphemes -- Challenge 2005, Helsinki University of Technology, Helsinki, Finland (2005)
9. Koskenniemi, Kimmo: Two-level morphology: A general computational model for word-form recognition and production. Ph.D. dissertation, University of Helsinki, (1983)
10. Kurimo, Mikko et al: Unsupervised segmentation of words into morphemes – Challenge 2005 -- An Introduction and Evaluation Report. In: Proceedings of Unsupervised segmentation of words into morphemes -- Challenge 2005. Helsinki University of Technology, Helsinki, Finland, (2005)
11. Porter, M.F.: An algorithm for suffix stripping. In: Program 14, (1980) 130–137.
12. Wicentowski, R: Multilingual Noise-Robust Supervised Morphological Analysis using the WordFrame Model. In: Proceedings of Seventh Meeting of the ACL Special Interest Group on Computational Phonology (SIGPHON) , (2004) 70-77.
13. 陈, 小荷: 现代汉语自动分析. 北京 北京语言文化大学出版社 (2000)
14. 黄, 昌宁, 李, 娟子: 语料库语言学. 北京 商务印书馆 (2002)