Language, Information and Computation (PACLIC12), 18-20 Feb, 1998, 218-229

Error-Driven Learning of Chinese Word Segmentation

Julia Hockenmaier*
Centre for Cognitive Science
University of Edinburgh
and
Institut für maschinelle Sprachverarbeitung
Universität Stuttgart
julia@cogsci.ed.ac.uk

Chris Brew†
Language Technology Group
Human Communication Research Centre
University of Edinburgh
Chris.Brew@ed.ac.uk

Abstract

Palmer ([4]) demonstrated how Brill's Transformation-based Error-Driven Learning can be applied to word segmentation in various languages. We present experimental results which show that such algorithms can achieve satisfactory performance even with a a very simple initial state annotator We also present two preliminary studies, which suggest that even higher performance might be achieved if simple morphological information is available to the system, and that segmentation performance might actually be improved by combining segmentation with rudimentary part-of-speech tagging.

1 Introduction

Chinese word segmentation is an interesting, but difficult problem. The difficulties include the following:

- "word" is not a very well-defined concept in the context of Chinese: linguists do not have generally accepted guidelines, and in experiments native speakers show only about 75 % agreement on the "correct" segmentation.
- Even if we have guidelines, the problem does not become trivial. The biggest stumbling-blocks for any automatic segmenter are unseen words which occur neither in the lexicon nor in the training data.

Previous work in this area tended to use some combination of stochastic methods and lexical heuristics. Stochastic methods provide robustness and trainability at the expense of compactness and perspicuity in the learnt models, while lexical heuristics tend to be labour intensive and highly corpus-specific But Palmer ([4]) has recently applied Brill's Transformation-based Error-Driven Learning to the segmentation problem, with promising results. The models which are produced are both more compact and easier to interpret than stochastic models, yet produce a performance which is comparable to that of the best hand-crafted systems.

*The first author was supported by a grant from the Studienstiftung des deutschen Volkes.
†The second author was partly supported by a grant from ESRC to the Human Communication Research Centre. Correspondence about this paper should be sent to Chris.Brew@ed.ac.uk.

Our work also uses Brill's technique. We extend Palmer's work by showing some effects of variation in corpus size and rule complexity. We present experiments which achieve satisfactory performance even with a very simple initial state tagger. A major reason for this success is the large size of the training corpus which was employed.

Segmentation-driven retagging Two preliminary studies, reported at the end of this paper, indicate that segmentation performance might actually be improved by simultaneously addressing the task of assigning tags to the text. The benefit of this apparently paradoxical move comes because rules learnt by the system can use the tag layer to describe the context in a richer way than is available in the simpler system. This information can then be exploited by later rules. But while our system assigns part-of-speech tags, it does so only in the service of better segmentation, and the tags which it assigns do not necessarily have any merit beyond their role in assisting segmentation.

2 Transformation-based Error-Driven Learning

Brill's Transformation-based Error-Driven Learning is a symbolic machine learning technique which has been successfully applied, *inter alia* to part-of-speech tagging ([1]). The general scheme of Brill's algorithm is as follows:

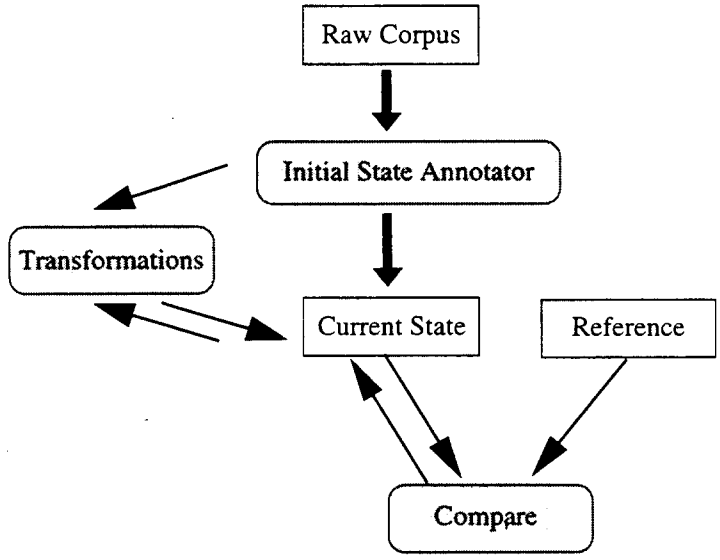

Figure 1: Transformation-based Error-Driven Learning

- Unannotated text is passed through an *initial state annotator* which may be simple (such as assignment of NN or NNP to all words irrespective of context), or relatively complex (a stochastic n-gram tagger).
- The output of the initial state annotator is then compared to a reference corpus, and the system searches for a transformation which gives the greatest local improvement in a user-defined and application specific error function.
- A transformation specifies a triggering environment and a rewrite rule.
- Once a rule has been chosen it is applied to the current annotation of the corpus, generating a new and slightly changed annotation. This is in turn compared to the reference corpus.

- The above process is iterated until no further improvement can be achieved. The sequence of transformations learnt by the system is the final output of the learning process[1].
- We now have a sequence of transformations which can be applied any text which has been passed through the initial state annotator.

This scheme is sufficiently general to apply to many tasks, including Chinese word segmentation, although each task will require the choice of a suitable set of rule templates, which serve to define the space of transformations to be searched by the algorithm. Performance will hinge crucially on the choice of appropriate templates.

3 Transformation-Based Learning applied to word segmentation

Palmer ([4]) reports four experiments on Chinese word segmentation. Adopting Brill's scheme, he used four distinct initial-state annotators:

- a naive character-as-word segmentation
- an initial segmentation obtained by maximum-matching
- an initial segmentation by maximum matching where unknown character sequences were treated by a character-as-word segmentation.
- the output of a high-accuracy word segmenter

For all experiments, Palmer used a set of rule templates with three types of actions:

- `concatenate` two characters.
- `split` two characters
- `slide` a word boundary to the adjacent character.

Rules may or may not refer to the surrounding context. We will call those which do so *context-sensitive*:

- *"concatenate x and y if (not) followed/preceded by z".*

and those which do not *context-insensitive*:

- *"concatenate x, y in all occurrences"*
- *"concatenate x and any following/previous character"*

By their nature the context-sensitive rules are more precise and less general than the context-insensitive ones. We will see the effect of this on the compactness of the models which are induced as we vary the complexity of the rule templates which we employ.

Our experiments show that if the training corpus is large enough, naive character-as-word segmentation and context-insensitive rules can achieve performance similar to Palmer's most sophisticated model.

We also briefly discuss experiments performed on a smaller corpus. They show how different sets of rule templates influence both performance on unseen text and number of rules required to learn the segmentation of the training corpus.

Evaluation For easy comparison, we follow Palmer in using a *balanced* F-measure, a variation of van Rijsbergen's F-measure ([8]) where precision and recall are weighted equally:

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (1)$$

[1] In the simple cases the order of rule application may not matter, but in the general case the applicability of one rule may depend on the prior application of another.

Palmer's experiments on Chinese For his experiments on Chinese, Palmer used a corpus of 2000 sentences (roughly 60,187 words) for training and 560 sentences (roughly 18,783 words) for testing. This is a training data to test data ratio of about 3 to 1. The corpus was taken from the Xinhua news agency. He performed four experiments, differing in the choice of initial state annotator. The results are summarised in table 1. In the first experiment, (called *single-word* in table 1) the initial segmentation was based on the

Setup:	*single-word*	*max-match*	*max-match2*	CHSEG
Transformat'ns:	5903	2897	2450	1755
F-meas. (initial.)	40.3	64.4	82.9	87.9
F-meas. (final)	78.1	84.9	87.7	89.6
Error reduction (%)	63.6	57.8	28.1	14.6

Table 1: The results of Palmer's experiments on the Xinhua corpus

assumption that each character is one word. The second experiment (*max-match*) used Maximum Matching as initial state, based on 57,472 words from the NMSU CHSEG segmenter. The third experiment (*max-match2*) used also Maximum Matching, but treated each character in an unknown character sequence as an individual word. The initial score is almost as high as the result of maximum matching with a rule-based postprocessor. The final performance is nearly as good as that of CHSEG. In a fourth experiment, the initial state was the output of CHSEG.

3.1 Our own experiments

Palmer's experiments provide clear evidence that Brill's algorithm can be successfully applied to the problem of word segmentation, both as a stand-alone system with in-built linguistic knowledge and as a post-processor. But we still need a good understanding of how different rule templates influence performance, both in acquiring the segmentation of the reference corpus and in segmenting unseen text. We also need to know how performance and generalisation scale with corpus size. We will see that training the system on a larger reference corpus can significantly improve performance, even if we restrict ourselves to very simple rule templates. Some of Palmer's rule templates, being conditional on adjacent characters, are highly specific, and run the risk of contributing to over-training. Although highly specific rules are necessary for a completely satisfactory model of word segmentation, we will show that high performance can be achieved by simple and relatively unspecific rules, provided the training corpus is large enough.

For all the experiments reported in this section, we assume a naive initial character-as-word segmentation.

The error function compares word position of the ith character (notated x_i) of the characters in the corpus with the corresponding value in the reference corpus (notated $\overline{x_i}$):

$$E = \sum_{i=1}^{n} |\overline{x_i} - x_i|$$

Implementation We tested on a corpus of roughly 1,700 characters. It became soon clear that efficiency issues would be very important when going to a larger corpus. Our algorithm diverges from Brill only when we could not see a sufficiently efficient implementation for Brill's original idea. We settled on a corpus representation using in-memory associative arrays. For a faithful implementation of Brill's idea, we would need to instantiate each rule template against each bi- and trigram in the corpus. This proved unacceptably costly for large corpora, so we adopted a heuristic approximation to Brill's approach:

- Start by finding all transformations which individually improve the initial segmentation. We call this collection of transformations the initial pool. If one of the rule templates is `t_concat(x y)`, where `x` and `y` are variables ranging over part-of-speech tags, the initial pool might contain *inter alia* the following instantiations of the template:

```
t_concat(N N)
```

```
t_concat(V V)
t_concat(R R)
```

with N = noun, V = verb, R = Roman numerals.

- Use Brill's original algorithm to construct a chain of transformations which reduces the error rate as much as possible.
- Once Brill's algorithm has run to completion on the initial pool, refresh the pool by again searching for the complete set of transformations which will individually reduce error on the current segmentation.
- Run Brill's algorithm to completion on the new pool.

Clearly, this process can be iterated until no further reduction in error rate occurs. Our algorithm, like Brill's is a heuristic search for optimum performance. The differences between the two algorithms will arise only when Brill's algorithm is able to make beneficial use of rules which were not incorporated in our initial pool. In practice the algorithms seem to differ very little.

4 Simple bigram rules

4.1 Experiment 1

In this experiment, an automatically segmented corpus was used as reference. This corpus contains 100,000 words from Guo Jin's Xinhua News Agency corpus [2] We used Jasmine, an online segmentation facility made available by the Chinese University of Hong Kong[3]. The test corpus consisted of 25,000 words from the same source as the training corpus.
In this initial experiment, we used the following rule templates:

concatenate($char_m, char_n$):
Delete the segmentation sign between c_i, c_{i+1} if $c_i = char_m$ and $c_{i+1} = char_n$

con_1($char_m$):
Delete the segmentation sign between c_i, c_{i+1} if $c_i = char_m$

con_2($char_n$):
Delete the segmentation sign between c_i, c_{i+1} if $c_{i+1} = char_n$

split($char_m$, $char_n$):
Insert a segmentation sign between c_i, c_{i+1} if $c_i = char_m$ and $c_{i+1} = char_n$

split_1($char_m$):
Insert a segmentation sign between c_i, c_{i+1} if $c_i = char_m$

split_2($char_n$):
Insert a segmentation sign between c_i, c_{i+1} if $c_{i+1} = char_n$

The results are shown in figure 2. Even for the simplest of our approaches, the score attained (F = 87.10) is considerably higher than Palmer's results with the naive character-as-word initial segmentation (F = 78.1), and comparable with the NMSU-segmenter (F = 87.9), and Palmer's experiment combining maximum-matching with a character-as-word initial segmentation for unrecognised substrings (F = 87.7). Given a large enough corpus, our adaptation of Brill's technique is sufficient to give broadly competitive performance even with very simple rule templates.

A close look at the results of experiment 1 showed us that:

[2]The corpus is available at `http://sunzi.iss.nus.sg:1996/corpora/chinese/recent/PH/`
[3](`http://peflinux0.ie.cuhk.hk/access/`)

Setup:	*bigr*	*bigr-corr*	*trigr*
Transformations:	7635	7523	7442
Errors (initial)	68391	68938	68938
Errors (final)	3215	2618	1353
Error reduction	95.29 %	96.20 %	98.03 %
F-meas. (initial)	42.33	42.02	42.02
F-meas. (final training)	97.33	97.89	98.59
F-meas. (final testing)	87.10	87.39	87.855

Table 2: Our experiments on segmentation

- There are inconsistencies and errors in Jasmine's segmentation In the next section we discuss the consequences of this fact.
- The rule templates which we have used so far are insufficient. We address this in experiments reported later in the paper.

4.2 The impact of Jasmine's inconsistencies

Jasmine's inconsistencies When we compared the reference corpus against the best automatic segmentation of that corpus, as generated in experiment 1, we discovered some inconsistencies in Jasmine's segmentation. We found that the same character string (with the same meaning) in different occurrences was sometimes segmented in different ways. We therefore took a close look at the training corpus, and corrected the inconsistencies which had been detected. During this process errors in the segmentation of personal names were also corrected. It would have been infeasible to hand-correct the entire Jasmine segmentation, but by using Brill's algorithm as a filter we were able to easily locate cases in which Jasmine was clearly performing less than optimally. If we apply our usual error measure to the original corpus and its corrected version, we find a distance of 3081.

Experiment 2: Training on the corrected version We now re-trained the system (using the templates of experiment 1) on the manually corrected version of the corpus. It required 7740 transformations (compared with 7635 for training on the uncorrected corpus) . But the residual error is 2618 (compared to 3215), so we can conclude that the corrected segmentation of the reference corpus is of use to our algorithm. And the very fact that the Brill's algorithm led us to the places where corrections were needed indicates that it is itself correcting some of the original inconsistencies.

4.3 Experiment 3: Trigram rules

In experiment 3 we extended the repertoire of rule templates. We now allowed the following:

conc_3 $(char_m, char_n, char_o)$**:**
Delete the segmentation sign between c_i, c_{i+1}, c_{i+2} if $c_i = char_m$, $c_{i+1} = char_n$ and $c_{i+2} = char_o$

split_3 $(char_m, char_n, char_o)$**:**
Insert segmentation signs between c_i, c_{i+1}, c_{i+2} if $c_i = char_m$, $c_{i+1} = char_n$ and $c_{i+2} = char_o$

This gives the system more potential ability to discriminate among different contexts. It is therefore not surprising that the segmentation of the training corpus was better captured. The residual error against the corrected training corpus was 1353 (compared to 2618 for experiment 2). The learnt segmentation of the training corpus has balanced F-measure of F = 98.59 against the training corpus. When applied to the corrected version of the test corpus, the score is F = 87.855.

4.4 More elaborate rules

This set of rule templates is almost identical to Palmer's except for the `slide`-rules, which weren't included in this experiment, because the same effect can be achieved by a combination of one splitting and concatenation operation.

Due to implementation limitations, it was impossible to investigate the impact of context-sensitive rules when running the experiment on a big corpus. We therefore performed a smaller scale investigation on a corpus of 10.000 words. In this investigation we are much less interested in absolute performance than in understanding the effects of varying the set of templates used. The following context-sensitive rule-templates were added:

conc_foll ($char_m, char_n, char_o$):
Delete the segmentation sign between c_i and c_{i+1} if $c_i = char_m$, $c_{i+1} = char_n$ and $c_{i+2} = char_o$

conc_foll_not ($char_m, char_n, char_o$):
Delete the segmentation sign between c_i and c_{i+1} if $c_i = char_m$, $c_{i+1} = char_n$ and $c_{i+2} \neq char_o$

conc_prev ($char_m, char_n, char_o$):
Delete the segmentation sign between c_i and c_{i+1} if $c_i = char_n$, $c_{i+1} = char_0$ and $c_{i-1} = char_n$

conc_prev_not ($char_m, char_n, char_o$):
Delete the segmentation sign between c_i and c_{i+1} if $c_i = char_n$, $c_{i+1} = char_0$ and $c_{i-1} \neq char_n$

split_foll ($char_m, char_n, char_o$):
Insert a segmentation sign between c_i and c_{i+1} if $c_i = char_m$, $c_{i+1} = char_n$ and $c_{i+2} = char_o$

split_foll_not ($char_m, char_n, char_o$):
Insert a segmentation sign between c_i and c_{i+1} if $c_i = char_m$, $c_{i+1} = char_n$ and $c_{i+2} \neq char_o$

split_prev ($char_m, char_n, char_o$):
Insert a segmentation sign between c_i and c_{i+1} if $c_i = char_n$, $c_{i+1} = char_0$ and $c_{i-1} = char_n$

split_prev_not ($char_m, char_n, char_o$):
Insert a segmentation sign between c_i and c_{i+1} if $c_i = char_n$, $c_{i+1} = char_0$ and $c_{i-1} \neq char_n$

Again, one experiment was conducted using the context-insensitive bigram rule templates as in 4. It acquired 1566 transformations. The final error was 126, which means an error reduction of 98%.
One experiment was performed with the context-insensitive unary, binary and ternary rules. 1517 transformations were learnt, 3.2% fewer than without the ternary rules. This is largely explained by the fact that the concatenation of any 3-character string XYZ requires two binary concatenations `concatenate(X,Y)`, `concatenate(Y,Z)`, but might be captured by the one ternary transformation `conc_3(X,Y,Z)` if none of the binary rules had been applied before. The residual error was 86, 31% smaller than without the ternary rules, and the error reduction was 98.6%. This shows that the limited context-sensitivity inherent to the rule set used in this experiment does already improve performance slightly (at least as far as acquisition of the training data is concerned). Another experiment was performed with the entire set of rule templates. 1566 transformations were acquired from the training corpus, that is exactly as many as in the context-insensitive binary case. But this time, the residual error was only 23, and the error reduction was 99.6%.
Table 3 lists again the results of the different experiments. For evaluation, we used the same test corpus of 25,000 words as in the experiments on the larger corpus, which means that the test corpus is actually 2.5 times as large as the training corpus! Because the training corpus is so small, the results should not be over-interpreted, but $F \approx 74$ is surprisingly high.

The model with the context-insensitive bigram rules achieves a slightly higher performance (74.03) on the test data than the model with the context-insenstive trigrams (73.77), even though the trigram model captures the training data better. The context-sensitive model scores best of the three with $F = 74.16$. Although it needs more rules to capture the training corpus than the context-insensitive trigram model, the residual error on the training corpus is also considerably lower. In other words, the context-sensitive model not only scores slightly higher on unseen test data, but can also deduce more information from a given set of data. All these differences are slight, and need validation on larger corpora.

Setup:	binary	ternary	cont.-sens.
Transformat'ns:	1566	1517	1566
Final error:	126	86	23
F-meas. (train.)	98.83	99.17	99.89
F-meas. (test)	74.03	73.77	74.16

Table 3: Results of experiments on a corpus of 10,000 words

4.5 Summary

If the training corpus is big enough, good performance can be obtained with simple rules. A big asset of this method seems to be its ability to smooth out inconsistencies in the segmentation of the training corpus. Consequently, it gives also a more consistent segmentation of unseen text. But context-insensitive rules are not expressive enough to completely capture the correct segmentation of Chinese words. And when one works with more sophisticated rule templates the combinatorics of the search process become seriously problematic, at least with the implementation techniques of which we are aware.

5 Part-of-speech information for segmentation

5.1 The experiments

In this set of experiments, we looked at the impact of part-of-speech information on word segmentation. The computational demands of the task forced us to use a much smaller corpus. We had to restrict ourselves to a corpus of only 1700 characters (1110 words). The initial state is again a character-as-word segmentation. We make crucial use of a lexicon which lists possible part-of-speech-tags for each character. In addition to rule templates which specified the context in terms of characters, we now also have rule templates which apply to classes of characters, according to their tags. We hoped that these rules would be better able to capture useful generalizations, and we believe this goal has been achieved. As the training data is so small, we based our evaluation of the acquired models on their performance in acquiring the segmentation of the reference corpus and on their complexity (ie. the number of rules). We have no data on performance using unseen text.

The question to be addressed is whether segmentation can be facilitated by the use of part-of-speech information. Our system adopts a strategy of `segmentation-driven retagging`. This strategy interleaves the acquisition of segmentation with a process of retagging which gradually adjusts a naive initial tag assignment in response to errors in the segmentation.

The algorithm has the incidental property that it generates a segmented and tagged corpus. But it is not the role of the system to generate correct part-of-speech tags, and we do not require that the corpus be hand-annotated for part-of-speech. We do of course continue to require a hand-segmentation. The process of retagging and resegmenting is entirely driven by errors in segmentation, without reference to any gold-standard for tagging, and it need not be the case that the final tag assignment stand in any simple relation to linguistic orthodoxy. The goal is to improve segmentation accuracy. If the use of part-of-speech tags as an internal representation aids this process we will deem the approach to have succeeded[4].

5.2 Segmentation-driven retagging

We now explain our algorithm In the previous experiments, conditions on the triggering configuration of a transformation were made on the surrounding characters. In the new experiments the triggering conditions may, but need not, make reference to tags. We now use a **lexicon**, which maps each character to a list of possible parts-of-speech for free occurences of this character.

We used the Concise Oxford English-Chinese Chinese-English Dictionary ([10]) for part-of-speech information. The initial state is again a segmentation of one character per word, but additionally each

[4]Of course, the tag assignments which are produced may turn out to be correct, or otherwise worthy of study in their own right, but that goes beyond the scope of this paper.

character is tagged with its most common part-of-speech.

We now come to the creation of a pool of transformations which might improve the segmentation of the part-of-speech tagged corpus. These transformations affect only the segmentation of the corpus,leaving the tags and the characters unchanged. But where previous experiments allowed reference only to characters, we now add a collection of transformation templates which are sensitive to the presence of particular tags. We instantiate both tag-sensitive and tag-insensitive templates to create an initial pool of transformations. We now use our variant of Brill's algorithm to select a sequence of transformations from this pool. These tranformations will reduce the overall error rate in segmentation, but have so far left the part-of-speech tags untouched.

Once we have done as well as we can in segmenting the corpus, we are ready to revise the initial tag assignment. We do this by changing the tagging in the places where there are deviations from the reference segmentation. Recall that in the initial tag assignment all characters received their most frequent tag. Character in the locations where segmentation has apparently failed are relabelled (in those locations only) with their second most frequent part-of-speech tag. We now generate a new pool of transformations and run these over the retagged corpus. Once these have run to completion we once again adjust the parts-of-speech, iterating the whole process until no further reduction of segmentation error is possible. The error measure is, as before, the sum of the absolute difference between the word positions of the individual characters in the corpus.

The tag-sensitive rules Apart from the context-insensitive unary and binary rules `concatenate`, `con_1`, `con_2`, `split`, `split_1`, `split_2` conditional on the characters involved in the operation, we now have the following rules which are conditional on the tags of the characters:

`t_concat`(t_m, t_n)**:**
Delete the segmentation sign between c_i, c_{i+1} if $Tag(c_i) = t_m$ and $Tag(c_{i+1}) = t_n$

`t_con_1`(t_m)**:**
Delete the segmentation sign between c_i, c_{i+1} if $Tag(c_i) = t_m$

`t_con_2`(t_n)**:**
Delete the segmentation sign between c_i, c_{i+1} if $Tag(c_{i+1}) = t_n$

`t_split`(t_m, t_n)**:**
Insert a segmentation sign between c_i, c_{i+1} if $Tag(c_i) = t_m$ and $Tag(c_{i+1}) = t_n$

`t_split_1`(t_m)**:**
Insert a segmentation sign between c_i, c_{i+1} if $Tag(c_i) = t_m$

`t_split_2`(t_n)**:**
Insert a segmentation sign between c_i, c_{i+1} if $Tag(c_{i+1}) = t_n$

The lexicon The lexicon contains for each character a list of part of speech tags. The tagset is very simple and is taken from the Concise English-Chinese Chinese-English Dictionary ([10]). The corpus contains 534 different characters. 276 of these characters have one possible tag, 258 have two or more. We assumed that any unlisted characters are nouns.

5.3 Results

We conducted three different experiments. All of them acquired the segmentation of the training corpus with complete accuracy (no residual error), but the models differed both in the number of transformations needed to capture the segmentation and the relative frequencies of instances of the different rule templates. Obviously, this corpus is far smaller than we want, so we need to be cautious in our conclusions. Also, because the information is extracted from such a small set of data, it makes little sense to evaluate the performance of acquired segmentation models on unseen text.

In the first experiment, the system scored the transformations by measuring the local error at each site of application, instead of measuring the global error after application of the transformation to the entire

Tag	Explanation
N:	noun
V:	verb
M:	measure word
F:	adverb
Z:	particles and function words
X:	adjective
P:	preposition
C:	conjunction
B:	punctuation mark
PR:	pronoun (and determiner)
S:	numeral
X:	family name
PF,SF	prefix, suffix
R:	roman numbers

Table 4: The Oxford Dictionary tagset

corpus[5]. When rules may be conditional on tags, the simplification might matter more, because rules expressed over the relatively small repertoire of tags are much more likely to collide with each other than are rules expressed over the much larger repertoire of characters.

To see what is going on consider an example. The first rule which was acquired was `t_concat (N N)` (*"concatenate two adjacent characters if both are tagged as noun."*). As concatenation of a pair of characters is defined as assigning the right character the word position value of the left character increased by one, the effect is accumulative , and this operation will increase the word position value of any character tagged as N by the number of immediately adjacent nouns to the left of the character.

Our rule selection heuristic decides purely on the basis of local impact at each site of application, without regard to the accumulative effects described above, so its estimate of the error may differ from the true value, which does include these effects. When actually applied, the transformation `t_concat (N N)` will create very long sequences, especially if applied as first transformation after the initial character-as-word segmentation. Most of these sequences will be errors. This effect can be large, although the heuristic selects it, `t_concat (N N)` actually increases the global error (from 682 to 726).

The corpus now contains many long sequences of characters tagged as N. As far as the rule selection heuristic is concerned, the quickest way of remedying the situation created by the last transformation appears to be the selection of `t_split (N N)`. But when this rule is applied the state of the corpus reverts to the initial segmentation, placing the system in an infinite loop. We need some means of avoiding this impasse.

The most direct solution would be to exclude a transformation from the set of possible transformations once its complement has been applied to the corpus. But not every transformation has a complement, and there would still be problems in pathological cases where two or more transformations conspire to reverse the effect of some other sequence of transformation. For the moment we limit the space of allowable transformations by means of *ad hoc* modifications, which are applied as needed.

Another simple remedy is to score transformations only once their global effect is known. In our implementation this is too costly to use except for small corpora.

A third remedy is to use a different error measure, in this case one which is more lenient with over-long words. We use an error function which simply counts the number of characters for which the current word position is different from the word position in the reference corpus:

$$E = \sum_{i=1}^{n} \begin{cases} 0 & \text{if } \overline{x_i} = x_i \\ 1 & \text{otherwise} \end{cases}$$

Under this regime, the error can never be higher than the number of characters in the corpus.

[5] In earlier experiments we used the same implementation, but in that context the simplification did not matter

We wanted to know which of these alternatives works best in our situation, so we conducted the following three experiments:

Alternative 1: Restrict the transformations In the initial state words are too short rather than too long, so we want to choose restrictions on the transformation set which tend to promote concatenation. We therefore excluded all the splitting rules which are conditional on tags (`t_split`, `t_split_1`, `t_split_2`), leaving in place the corresponding concatentation rules. For this experiment we stay with the local measure of transformation effectiveness.

The first three rules learnt are concatenations of characters with identical tags: `t_concat(N N)`, `t_concat(V V)`, `t_concat(R R)`.

272 transformations capture the segmentation information in the training data, which is less than the 302 rules which are needed if we use only binary rules conditional on characters. and also less than the 286 rules which are learnt with binary and ternary rules conditional on characters.

Alternative 2: Global measure of transformation impact If transformations are scored according to their global effect, and not according to the sum of their local effects, 281 transformations are learnt. We are not sure if it is significant that this is a slightly less compact model than the 272 rule model found using the (somewhat less principled) local version.

Alternative 3: Use a special purpose error measure If the objective function counts the number of characters whose current position within the word is not identical to their position in the reference corpus instead of summing the absolute difference of the values, we get yet another model. 286 rules are required to capture the segmentation of the reference corpus, the same number as with binary and ternary rules conditional on characters.

Conclusion Obviously, it is crucial that these experiments are repeated on larger amounts of data. There does not seem to be much to choose between the various heuristic remedies which we adopted. In all cases the tag sensitive methods outperform those which do not use this information. Given larger corpora we hope it will be possible to develop a better understanding of the trade-offs which operate here.

Using tag information does seem to improve the compactness of the learnt models. We also need to find out how the addition of tags interacts with the more complex rule templates used in our earlier experiments.

In developing the methods reported in the paper we have become aware of a number of more or less orthogonal options in matters of rule representation, search strategy and error measures. We have barely begun to explore the space of possible options. It may be possible to do this in a more systematic and data-driven way by moving to a more explicit Bayesian formulation of the problem, perhaps using heuristics based on description length as described, for example, by Stolcke ([7]).

All our experiments relied on a dictionary as well as a segmented reference corpus. In future we hope to investigate whether it is possible to simultaneously acquire both the segmentation and the tagging of the reference corpus without referring to a dictionary. These experiments would start from an initial state where each word is tagged as a noun. Both the morphological information and the correct segmentation would then be acquired from the information available in the training corpus.

6 Conclusion

Summary of results - Discussion We presented different ways in which Transformation-based Error-Driven Learning can be applied to the problem of Chinese word segmentation. We have shown that this technique, which produces symbolic models by implicitly statistical means, can have advantages. It produces more perspicuous models than stochastic systems, is (relatively) cheap and simple to implement, and shows some ability to generalize in a useful way. Even if we are not interested in developing a state-of-the-art segmentation system, the models which are learned can be investigated and compared for complexity

and performance on the training data. We were driven to this by computational necessity, but it is an important strength of the method that such small-scale studies can provide suggestive preliminary results.

Comparison of our results with [4] gives a clear indication that size of the training corpus is paramount for performance, and that even a linguistically ignorant system with bigram rules can achieve surprisingly high performance if the amount of data it is trained on is large enough.

Open Questions - Directions for future research The nature of the segmentation errors produced by the simplest rules clearly demonstrates that satisfactory language models can only be achieved if more sensitive rule templates are permitted. Controlling the proliferation of these rules is the greatest challenge for the future.

Other open questions also remain, because our tagging experiments could only be performed on very small amounts of training data. If conditions are stated only in terms of characters, there is a risk of overfitting the model to the training corpus. It is not clear whether conventional part-of-speech tags are optimal for this task. At one extreme, an argument can be made for a tagset which makes only the minimal distinction between closed- and open-class words. This would make it almost trivial to prepare an exhaustive dictionary, since there would be no need to record open-class words. On the other hand the fact that our system made such good use of the `t_concat(N,N)` and `t_concat(V,V)` rules suggests that this may be too extreme a move. An intermediate position would be to use a tagset encoding a small number of carefully chosen major-class distinctions.

References

[1] Eric Brill, 1995.
Transformation-based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging Computational Linguistics, Vol 21, Nr 4, 1995.

[2] Keh-Jiann Chen, Shing-Huan Liu, 1992.
Word Identification for Mandarin Chinese Sentences COLING-92 Nantes, p.101-107.

[3] John DeFrancis, 1984
The Chinese Language, University of Hawaii Press, Honolulu, 1984.

[4] David Palmer, 1997.
A Trainable Rule-Based Algorithm for Word Segmentation Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL '97), Madrid, 1997.

[5] Richard Sproat and Chilin Shih, 1993.
A statistical method for finding word boundaries in Chinese text Computer Processing of Chinese and Oriental Languages, 4:336-351, 1993.

[6] Richard Sproat, Chilin Shih, William Gale, Nancy Chang, 1996.
A Stochastic Finite-State Word-Segmentation Algorithm for Chinese Computational Linguistics, Vol 22, Nr 3.

[7] Andreas Stolcke 1994. *Bayesian Learning of Probabilistic Language Models.* Doctoral dissertation, Dept. of Electrical Engineering and Computer Science, University of California at Berkeley.

[8] C.J. van Rijsbergen, 1979.
Information Retrieval. Butterworths, London.

[9] Dekai Wu and Pascale Fung, 1994.
Improving Chinese Tokenization with Linguistic Filters on Statistical Lexical Acquisition ANLP-94, Stuttgart.

[10] *Concise English-Chinese Chinese English Dictionary*
(The Commercial Press and Oxford University Press, 1986.