Technical Report of *Chinese Morphological Analyzer(Chen)*

Vernkin Smith

April 8, 2009

Abstract

Practical results show that performance of statistic segmentation system outperforms that of hand-crafted rule-based systems. And the evaluation shows than the accuracy drop caused by out-of-vocabulary (OOV) words is at least five times greater than that of segmentation ambiguities. The better performance of OOV recognition the higher accuracy of the segmentation system in whole, and the accuracy of statistic segmentation systems with character-based tagging approach outperforms any other word-based system. This Report is about a supervised machine-learning approach to Character-based Chinese word segmentation. A maximum entropy tagger is trained on manually annotated data to automatically assign to Chinese characters, or *hanzi*, tags that indicate the position of a *hanzi* within a word.

Contents

Chapter 1

Overview

1.1 A Decade Review of Chinese Word Segmentation

During the last decade, especially since the First International Chinese Word Segmentation Bakeoff was held in July 2003, the study is automatic Chinese word segmentation has been greatly improved. Those improvements could be summarized as following: (1) on the computation sense Chinese words in read text that have been well-defined by ”segmentation guidelines + lexicon + segmented corpus”; (2) practical results show that performance of statistic segmentation system outperforms that of hand-crafted rule-based systems; (3) the evaluation shows than the accuracy drop caused by out-of-vocabulary (OOV) words is at least five times greater than that of segmentation ambiguities; (4) the better performance of OOV recognition the higher accuracy of the segmentation system in whole, and the accuracy of statistic segmentation systems with character-based tagging approach outperforms any other word-based system.

1.2 Main Difficulties in Chinese Segmentation

It is generally agreed among researchers that word segmentation is a neccessary first step in Chinese language processing. However, unlike Engish text in which sentences are sequences of words delimited by white spaces, in Chinese text, sentences are represented as strings of Chinese characters or *hanzi* without similar natural delimiters. Therefor, the first step in a Chinese language processing task is to identify the sequence of words in a sentence and mark boundaries in appropriate places. This may sound simple enough but in reality identifying words in Chinese is a non-trivial problem that has drawn a large body of research in the Chinese language processing community.

It is easy to demonstrate that the lack of natural delimiters itself is not the heart of the problem. In a hypothetical language where all words are represented with a finite set of symbols, if one subset of the symbols always start a word and another subset, mutually exclusive from the previous subset, always end a word, identifying words would be a trivial exercise. Nor can the problem be attributed to the lack of inflectional morphology. Although it is true in Indo-European languages inflectional affixes can generally be used to signal word boundaries, it is conceivable that a hypothetical language can use symbols other than inflectional morphemes to serve the same purpose. Therefore the issue is neither the lack of natural word delimiters nor the lack of inflectional morphemes in a language, rather it is whether the language has a way of unambiguously signaling the boundaries of a word.

The real difficulty in automatic Chinese word segmentation is the lack of such unambiguous word boundary indicators. In fact, most hanzi can occur in different positions within different words. The examples in Table 1 show how the Chinese character 产(“produce”) can occur in four different positions. This state of affairs makes it impossible to simply list mutually exclusive subsets of hanzi that have distinct distributions, even though the number of hanzi in the Chinese writing system is in fact finite. As long as a hanzi can occur in different word-internal positions, it cannot be relied upon to determine word boundaries as they could be if their positions were more or less fixed.

Table 1. A hanzi can occur in multiple word-internal positions

Position	Example
Left	产生 ’to come up with’
Word by itself	产小麦 ’to grow wheat’
Middle	生产线 ’assemby line’
Right	生产 ’to produce’

The fact that a hanzi can occur in multiple word-internal positions leads to ambiguities of various kinds. For example, 文 can occur in both word-initial and word-final positions. It occurs in the word-final position in 日文(“Japanese”) but in the word-initial position in 文章(“article”). In a sentence that has a string “日文章”, as in (1a), an automatic segmenter would face the dilemma whether to insert a word boundary marker between 日 and 文, thus grouping 文章 as a word, or to mark 日文 as a word, to the exclusion of 章. The same scenario also applies to 章, since like 文, it can also occur in both word-initial and word-final positions.

1. (a): Segmentation 1
日文 章魚 怎么 说?
Japanese octopus how say
“How to say octopus in Japanese?”
1. (b): Segmentation 2
日 文章 魚 怎么 说?

Japan article fish how say

Ambiguity also arises because some hanzi should be considered to be just word components in certain contexts and words by themselves in others. For example, 魚 can be considered to be just a word component in 章魚. It can also be a word by itself in other contexts. Presented with the string 章魚 in a Chinese sentence, a human or automatic segmenter would have to decide whether 魚 should be a word by itself or form another word with the previous hanzi. Given that 日, 文章, 章魚, 魚 are all possible words in Chinese, how does one decide that 日文 章魚 is the right segmentation for the sentence in (1) while 日 文章 魚 is not? Obviously it is not enough to know just what words are in the lexicon. In this specific case, a human segmenter can resort to world knowledge to resolve this ambiguity, knowing that 日 文章 魚 would not make any kind of real-world sense.

Chapter 2

Introduce to Character-based Chinese Segmentation

Follow the *Overview* and *Abstract*, this chapter would introduce Character-based Chinese Segmentation. The Maximum Entropy Modeling (see chapter-3) used in the segmentation and more detail see the following chapters. In this charper, we first formalize the idea of tagging *hanzi*(Chinese Character) based on their word-internal positions and describe the tag set we used.

First we convert the manually segmented words in the corpus into a tagged sequence of Chinese characters. To do this, we tag each character with one of the four tags, L, R, M and R depending on its position within a word. It is tagged L if it occurs on the left boundary of a word, and forms a word with the character(s) on its right. It is tagged R if it occurs on the right boundary of a word, and forms a word with the character(s) on its left. It is tagged M if it occurs in the middle of a word. It is tagged R if it forms a word by itself. We call such tags position-of-character (POC) tags to differentiate them from the more familiar part-of-speech (POS) tags. For example, the manually segmented string in (2a) will be tagged as (2b):

Example Senetence 2:

(a) 上海 计划 到 本 世纪 末 实现 人均 国内 生产 总值 五千 美元

(b) 上/L 海/R 计/L 划/R 到/R 本/R 世/L 纪/R 末/R 实/L 现/R 人/L 均/R 国/L 内/R 生/L 产/R 总/L 值/R 五/L 千/R 美/L 元/R

(c) Shanghai plans to reach the goal of 5,000 dollars in per capita GDP by the end of the century.

Given a manually segmented corpus, a POC-tagged corpus can be derived trivially with perfect accuracy. The reason why use such POC-tagged sequences of characters instead of applying n-gram rules to segmented corpus directly[Palmer, 1997[5]; Hockenmaier and Brew, 1998[3]; Xue, 2001[6]] is that they are much easier to manipulate in the training process. In addition, the POC tags reflect our observation that the ambiguity problem is due to the fact that a hanzi can occur in different word-internal positions and it can be resolved in context. Naturally, while some characters have only one POC tag, most characters will receive multiple POC tags, in the same way that words can have multiple POS tags. Table 2 shows how all four of the POC tags can be assigned to the character 产 (“produce”):

Table 2. A character can receive as many as four tags

Position	Tag	Example
Left	L	产生 ’to come up with’
Word by itself	R	产小麦 ’to grow wheat’
Middle	M	生产线 ’assemby line’
Right	R	生产 ’to produce’

If there is ambiguity in segmenting a sentence or any string of hanzi, then there must be some hanzi in the sentence that can receive multiple tags. For example, each of the first four characters of the sentence in (1) would have two tags. The task of the word segmentation is to choose the correct tag for each of the hanzi in the sentence. The eight possible tag sequences for (1) are shown in (3a), and the correct tag sequence is (3b).

Example Senetence 4:

(a) 日/L,R 文/R,L 章/L,R 鱼/R,R 怎/L 么/R 说/R ?

(b) 日/L 文/R 章/L 鱼/R 怎/L 么/R 说/R ?

Also like POS tags, how a character is POC-tagged in naturally occurring text is affected by the context in which it occurs. For example, if the preceding character is tagged R or R, then the next character can only be tagged L or R. How a character is tagged is also affected by the surrounding characters. For example, 关(”close”) should be tagged R if the previous character is 开 (”open”) and neither of them forms a word with other characters, while it should be tagged L if the next character is 心 (”heart”) and neither of them forms a word with other characters. This state of affairs closely mimics the familiar POS tagging problem and lends itself naturally to a solution similar to that of POS tagging. The task is one of ambiguity resolution in which the correct POC tag is determined among several possible POC tags in a specific context. Our next step is to train a maximum entropy model on the perfectly POC-tagged data derived from a manually segmented corpus to automatically POC-tag unseen text.

Chapter 3

Maximum Entropy Modeling

This chapter provides a brief introduction to Maximum Entropy Modeling. The Advantage of maximum entropy model includes: Based on features, allows and supports feature induction and feature selection; offers a generic framework for incorporating unlabeled data; only makes weak assumptions; gives flexibility in incorporating side information; natural multi-class classification.

Maximum Entropy (ME or maxent for short) model is a general purpose machine learning framework that has been successfully applied in various fields including spatial physics, computer vision, and Natural Language Processing (NLP). This introduction will focus on the application of maxent model to NLP tasks. However, it is straightforward to extend the technique described here to other domains.

3.1 The Modeling Problem

The goal of statistical modeling is to construct a model that best accounts for some training data. More specific, for a given empirical probability distribution $\tilde{p}$, we want to build a model p as close to $\tilde{p}$ as possible.

Of course, given a set of training data, there are numerous ways to choose a model p that accounts for the data. It can be shown that the probability distribution of the form 3.1 is the one that is closest to $\tilde{p}$ in the sense of Kullback-Leibler divergence, when subjected to a set of feature constraints:

$$p(y \mid x) = \frac{1}{Z(x)} \exp\left[\sum_{i=1}^{k} \lambda_i f_i(x,y)\right] \tag{3.1}$$

Here $p(y \mid x)$ denotes the conditional probability of predicting an *outcome* y on seeing the *context* x. $f_i(x,y)$'s are feature functions (described in detail later), λ_i's are the weighting parameters for $f_i(x,y)$'s. k is the number of features and Z(x) is a normalization factor (often called partition function) to ensure that $\sum_y p(y|x) = 1$.

ME model represents evidence with binary functions1 known as *contextual predicates* in the form:

$$f_{cp,y'}(x,y) = \begin{cases} 1 & \text{if } y = y' \text{ and } cp(x) = true \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Where cp is the contextual predicate that maps a pair of *outcome* y and *context* x to $\{true, false\}$.

The modeler can choose arbitrary feature functions in order to reflect the characteristic of the problem domain as faithfully as possible. The ability of freely incorporating various problem-specific knowledge in terms of feature functions gives ME models the obvious advantage over other learn paradigms, which often suffer from strong feature independence assumption (such as naive bayes classifier).

For instance, in part-of-speech tagging, a process that assigns part-of-speech tags to words in a sentence, a useful feature may be (DET is DETERMINER for short):

$$f_{prev\_tag\_DET,NOUN}(x,y) = \begin{cases} 1 & \text{if } y = NOUN \text{ and } prev\_tag\_DET(x) = true \\ 0 & \text{otherwise} \end{cases}$$

which is *activated* when previous tag is DETERMINER and current word's tag is NOUN. In Text Categorization task, a feature may look like (RO is ROMANTIC for short):

$$f_{doc\_has\_RO,love\_story}(x,y) = \begin{cases} 1 & \text{if } y = love_story \text{ and } doc\_has\_RO(x) = true \\ 0 & \text{otherwise} \end{cases}$$

which is *activated* when the term ROMANTIC is found in a document labeled as type:love_story.

Once a set of features is chosen by the modeler, we can construct the corresponding maxent model by adding features as constraints to the model and adjust weights of these features. Formally, We require that:

$$E_{\tilde{p}} < f_i >= E_p < f_i >$$

Where $E_{\tilde{p}} < f_i >= \sum_x \tilde{p}(x,y) f_i(x,y)$ is the empirical expectation of feature $f_i(x,y)$ in the training data and E$E_p < f_i >= \sum_x p(x,y) f_i(x,y)$ is the feature expectation with respect to the model distribution p. Among all the models subjected to these constraints there is one with the Maximum Entropy, usually called the Maximum Entropy Solution.

3.2 Parameter Estimation

Given an exponential model with n features and a set of training data (empirical distribution), we need to find the associated real-value weight for each of the n feature which maximize the model's log-likelihood:

$$L(p) = \sum_{x,y} \tilde{p}(x,y) \log p(y \mid x) \tag{3.3}$$

Selecting an optimal model subjected to given contains from the exponential (log-linear) family is not a trivial task. There are two popular iterative scaling algorithms specially designed to estimate parameters of ME models of the form 3.1: *Generalized Iterative Scaling* [Darroch and Ratcliff, 1972][1] and *Improved Iterative Scaling* [Della Pietra et al.,1997][2].

Recently, another general purpose optimize method *Limited-Memory Variable Metric* (L-BFGS for short) method has been found to be especially effective for maxent parameters estimating problem [Malouf, 2003][4]. L-BFGS is the default parameter estimating method in the implementation.

3.3 Usage of the MaxEnt Model

This section covers the basic steps required to build and use a Conditional Maximum Entropy Model.

3.3.1 Representing Features

The mathematical representation of a feature used in a Conditional Maximum Entropy Model can be written as:

$$f_{cp,y'}(x,y) = \begin{cases} 1 & \text{if } y = y' \text{ and } cp(x) = true \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

where cp is the $contextualpredicate$ which maps a pair of $outcomey$ and $contextx$ into $\{true, false\}$.

This kind of math notation must be expressed as features of literal string in order to be used in this toolkit. So a feature in part-of-speech tagger which has the form (DET is DETERMINER for short):

$$f_{prev\_tag\_DET,NOUN}(x,y) = \begin{cases} 1 & \text{if } y = NOUN \text{ and } prev\_tag\_DET(x) = true \\ 0 & \text{otherwise} \end{cases}$$

can be written as a literal string: "tag-1=DETERMINER_NOUN".

3.3.2 Create a Maxent Model Instance

A *maxent* instance can be created by calling its constructor:

```
#include <maxent/maxentmodel.hpp>
using namespace maxent;
MaxentModel m;
```

This will create an instance of MaxentModel class called m. Please note that all classes and functions are in the namespace maxent. For illustration purpose, the include and using statements will be ignored intentionally in the rest of this section.

3.3.3 Adding Events to Model

Typically, training data consists of a set of events (samples). Each event has a *context*, an *outcome*, and a *count* indicating how many times this event occurs in training data.

Remember that a *context* is just a group of *contextpredicates*. Thus an event will have the form:

$$[(predicate_1, predicate_2, \ldots, predicate_n), outcome, count]$$

Suppose we want to add the following event to our model:

$$[(predicate_1, predicate_2, predicate_3), outcome1, 1]$$

We need to first create a context:

```
std::vector<std::string> context;
context.append("predicate1");
context.append("predicate2");
context.append("predicate3");
. . .
```

It's possible to specify feature value (must be non-negative) in creating a context:

```
std::vector<pair<std::string, float> > context;
context.append(make_pair("predicate1",1.0));
context.append(make_pair("predicate2",2.0));
context.append(make_pair("predicate3",3.0));
. . .
```

Before any event can be added, one must call begin_add_event() to inform the model the beginning of training.

```
m.begin_add_event();
```

Now we are ready to add events:

```
m.add_event(context, "outcome1", 1);
```

The third argument of add_event() is the count of the event and can be ignored if the count is 1.

One can repeatedly call add_event() until all events are added to the model.

After adding the last event, end_add_event() must be called to inform the model the ending of adding events.

```
m.end_add_event();
```

3.3.4 Training the Model

Train a Maximum Entropy Model is relatively easy. Here are some examples:

```
// train the model with default training method
m.train();
// train the model with 30 iterations of L-BFGS method
m.train(30, "lbfgs");
// train the model with 100 iterations of GIS method and apply
// Gaussian Prior smoothing with a global variance of 2
m.train(100, "gis", 2);
// set terminate tolerance to 1E-03
m.train(30, "lbfgs", 2, 1E-03);
```

The training methods can be either “gis” or “lbfgs” (default). Also, if m.verbose is set to 1 (default is 0), training progress will be printed to stdout.

You can save a trained model to a file and load it back later:

```
m.save("new_model");
m.load("new_model");
```

A file named new_model will be created. The model contains the definition of context predicates, outcomes, mapping between features and feature ids and the optimal parameter weight for each feature.

If the optional parameter binary is true and the library is compiled with zlib support, a compressed binary model file will be saved which is much faster and smaller than plain text model. The format of model file will be detected automatically when loading:

```
m.save("new_model", true); //save a (compressed) binary model
m.load("new_model");       //load it from disk
```

3.3.5 Using the Model

The use of the model is straightforward. The *eval*() function will return the probability $p(y|x)$ of an *outcomey* given some *contextx*:

```
m.eval(context, outcome);
```

eval_all() is useful if we want to get the whole conditional distribution for a given context:

```
std::vector<pair<std::string, double> > probs;
m.eval_all(context, probs);
```

eval_all() will put the probability distribution into the vector probs. The items in probs are the outcome labels paired with their corresponding probabilities. If the third parameter sort_result is true (default) *eval_all*() will automatically sort the output distribution in descendant order: the first item will have the highest probability in the distribution.

Chapter 4

POS Tagger

This Chapter discusses the steps involved in building a Part-of-Speech (POS) tagger using Maximum Entropy Model in detail.

4.1 The Tagging Model

The task of POS tag assignment is to assign correct POS tags to a word stream (typically a sentence). The following table lists a word sequence and its corresponding tags:

To attack this problem with the Maximum Entropy Model, we can build a conditional model that calculates the probability of a tag y, given some contextual information x:

$$p(y|x) = \frac{1}{Z(x)} \exp\left[\sum_{i=1}^{k} \lambda_i f_i(x,y)\right]$$

Thus the possibility of a tag sequence $\{t_1, t_2, \ldots, t_n\}$ over a sentence $\{w_1, w_2, \ldots, w_n\}$ can be represented as the product of each $p(y|x)$ with the assumption that the probability of each tag y depends only on a limited context information x:

$$p(t_1, t_2, \ldots, t_n | w_1, w_2, \ldots, w_n) \approx \prod_{i=1}^{n} p(y_i|x_i)$$

Given a sentence $\{w_1, w_2, \ldots, w_n\}$, we can generate K highest probability tag sequence candidates up to that point in the sentence and finally select the highest candidate as our tagging result.

4.2 Feature Selection

We select features used in the tagging model by applying a set of feature templates to the training data. The features are:

1. Prefix / suffix characters of the word;

2. Whether it is numeric;
3. Whether it is hyphen;
4. The word and the POS of the last/previous word;
5. The word and the POS of the last/previous 2th word;
6. the words and the POSs combination of the last/previous 1st the 2th words.

The following table is the features which are selected from the actual sentence:

Table 4.1 The Features Example

curword=years
tag-1=CD
word-2=,
tag-1,2=,CD
word+1=old
curword=old
word-1=years
tag-1=NNS
tag-1,2=CD,NNS
word+2=will
word-1=old
prefix=E
prefix=El
suffix=r
suffix=er
suffix=ier

Table 4.2 The Contextual Predicates Example

Condition	Contextual Predicates
w_i is not rare	$w_i = X$
w_i is rare	X is prefix of w_i, $\|X\| \leq 4$
	X is suffix of w_i, $\|X\| \leq 4$
	X contains number
	X contains uppercase character
	X contains hyphen
$\forall w_i$	$t_{i-1} = X$
	$t_{i-w}t_{i-1} = XY$
	$w_{i-1} = X$
	$w_{i-2} = X$
	$w_{i+1} = X$
	$w_{i+2} = X$

Please note that if a word is rare (occurs less than 5 times in the training set) several additional contextual predicates are used to help predict the tag based on the word's form. A useful feature might be:

$$f(x,y) = \begin{cases} 1 & \text{if y=VBG and } current\_suffix\_is\_ing(x) = true \\ 0 & \text{otherwise} \end{cases}$$

and is represented as a literal string: "suffix=ing_VBG".

Chapter 5

Segment Tagger

This Chapter discusses the steps involved in building a Segment Tagger tagger using Maximum Entropy Model in detail. Compared with POS, Segment Tagger uses POC (Position of Character)

5.1 POC (Position of Character)

There are four POC tags: L (Left position of the word), M (Middle position of the word), R (Right position of the word) and I (Independent or Word by itself). See the following example:

Table 5.1: The four POC tags of the Character 产.

Tag	Example
L	产生 'to come up with'
R	产小麦 'to grow wheat'
M	生产线 'assemby line'
I	产 'to produce'

5.2 POC Tagger

The POC tagger here uses the same probability model as the POS tagger. The probability model is defined over $H \times T$, where H is the set of possible *contexts* or "*histories*" and T is the set of possible tags. The model's joint probability of a history h and a tag t is defined as

$$p(h,t) = \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h,t)} \tag{5.1}$$

where π is a normalization constant, $\{\mu, \alpha_1, ..., \alpha_k\}$ are the model parameters and $\{f_1, ..., f_k\}$ are known as features, where $f_j(h,t) \in \{0,1\}$. Each feature f_j has a corresponding parameter α_j, hat effectively serves as a "*weight*" of this feature. In the training process, given a sequence of characters $\{c_1, ..., c_k\}$ and their POC tags $\{t_1, ..., t_k\}$ as training data, the purpose is to determine the parameters $\{\mu, \alpha_1, ..., \alpha_k\}$ that maximize the likelihood of the training data using p:

$$L(P) = \prod_{i=1}^{n} P(h_i, t_i) = \prod_{i=1}^{n} \pi\mu \prod_{j=1}^{k} \alpha_j^{f_j(h_i,t_i)} \tag{5.2}$$

The success of the model in tagging depends to a large extent on the selection of suitable features. Given (h, t), a feature must encode information that helps to predict t . The features used are instantiations of the feature templates. Feature templates (2) to (4) represent character features while (5) represents tag features. $C_{-3}...C_3$ are characters and $T_{-3}...T_3$ are POC tags. Feature template (1) represents the default feature.

Feature templates:

1. Default feature
2. The current character (C_0)
3. The previous (next) two characters $(C_{-2}, C_{-1}, C_1, C_2)$
4. The previous (next) character and the current character (C_{-1}, C_0, C_1), the previous two characters (C_{-2}, C_{-1}), and the next two characters $C_1, C_2)$.
5. The previous and the next character $C_{-1}, C_1)$.
6. The tag of the previous character T_{-1}, and the tag of the character two before the current character T_{-2}

In general, given (h, t), these features are in the form of co-occurrence relations between t and some type of context h , or between t and some properties of the current character. For example,

$$f_i(h_i, t_i) = \begin{cases} 1 & \text{if } t_{i-1} = L \ \& \ t_i = R \\ 0 & otherwise \end{cases}$$

This feature will map to 1 and contribute towards $p(h_i, t_i)$ if c_{i-1} is tagged L and c_i is tagged R.

The feature templates encode three types of contexts. First, features based on the current and surrounding characters $(2, 3, 4, 5)$ are extracted. Given a character in a sentence, this model will look at the current character, the previous two and next two characters. For example, if the current character is 们 (plural marker), it is very likely that it will occur as a suffix in a word, thus receiving the tag R. On the other hand, for other characters, they might be equally likely to appear on the left, on the right or in the middle. In those cases where it occurs within a word depends on its surrounding characters. For example, if the current character is 爱 (“love”), it should perhaps be tagged L if the next character is 护 (“protect”). However, if the previous character is 热 (“warm”), then it should perhaps be tagged R. Second, features based on the previous tags (5) are extracted. Information like this is useful in predicting the POC tag for the current character just as the POS tags are useful in predicting the POS tag of the current word in a similar context. For example, if the previous character is tagged I or R, this means that the current character must start a word, and should be tagged either L or I. Finally, a default feature (1) is used to capture cases where no other features are available. When the training is complete, the features and their corresponding parameters will be used to calculate the probability of the tag sequence of a sentence when the tagger tags unseen data. Given a sequence of characters $c_1, ..., c_n$, the tagger searches for the tag sequence $t_1, ..., t_n$ with the highest probability

$$P(t_1, ..., t_n \mid C_1, ..., C_n) = \prod_{i=1}^{n} P(t_i \mid h_i) \tag{5.3}$$

and the conditional probability of for each POC tag t given its history h is calculated as

$$P(t \mid h) = \frac{p(h,t)}{\sum_{t' \in T}^{P(h,t')}} \tag{5.4}$$

5.3 Pre-processing

In the Chinese articles, besides Chinese Characters, they are usually contain letters, digits and punctuations characters (we call them *special string*). And these characters are have two forms: English Form and the Chinese Form. For Example, word ”APPLE” is English Form (each character occupies 1 byte) and ”ＡＰＰＬＥ” is Chinese Form (each character occupies more than 1 byte). Although they are different from the displaying forms, they should be treated the same as an independent word, that is, it is wrong to divide ”APPLE” into more than 2 parts.

The Pro-processing algorithm aims to avoid such situations. For convenient, all the encodings (except UTF-8) are transformed to UTF-8 encoding for processing, and the program can detect all the meaningful special strings.

Table 5.2: The types of Characters

INIT	the initial type
DIGIT	the digit character, like "0" and "1"
PUNC	the punctuation character, like "." and ","
HYPHEN	the hyphen character, like "-"
LETTER	the letter character, like "a" and "D"
OTHER	other character, like Chinese character "汉"

Appendix A

Appendix - Project schedules and milestones

	Milestone	**Start**	**Finish**	**In Charge**	**Description**	**Status**
1	Survey on the project	2008-02-01	2008-02-13	Vernkin	1. Have a general overview of current Chinese Segmentation techniques and their differences. 2. Select a suitable one (Character-based) and do more research on it.	Finished
2	Design the Architecture of CMA	2008-02-16	2008-02-20	Vernkin	1. Basen on WISE KMA Orange, design the Architecture for the MA Systems (CJK). 2. Moreover, design the common Architecture for the CMA. 3. Finally focus on the Architecture for the approach I selected.	Finished
3	Implement CMA and Unit Test	2008-02-23	2008-03-25	Vernkin	1. Implement the Maximum Entropy Model. 2. Implement character-based Segmentation. 3. Integrate all the components into CMA.	Finished
4	Optimization the Segmentation Logic	2008-03-26	2008-04-10	Vernkin	1. Try other features sets; 2. Optimize the code logic to reduce the execute time; 3. List out the error segmentations, classify the reasons and do extra post-segmentation optimization.	65% Finished

Bibliography

[1] J.N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, Vol. 43:pp 1470–1480, 1972.

[2] Stephen Della Pietra, Vincent J. Della Pietra, and John D. Laf ferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.

[3] Julia Hockenmaier and Chris Brew. Error-driven segmentation of chinese. *Communications of COLIPS*, 1(1):69–84, 1998.

[4] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation, 2003.

[5] David Palmer. A trainable rule-based algorithm to word segmentation. *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics*, 1997.

[6] Nianwen Xue. Defining and automatically identifying words in chinese. *Ph.D. thesis, University of Delaware*, 2001.