Chinese Text Segmentation

By
Chung-Yang(Kenneth) Lee
9/10/07

Introduction
• What is a segmenter?
• Why are segmenters important?
• General research questions about segmenters
• Why do computers hate Chinese?
• Why am I studying segmenters?
• My strategy & results so far …
• What’s next?
What is a segmenter?

• A segmenter is a type of tokenizer or something which creates tokens or atomic pieces of data separated by a distinct delimiter.

• They are a tool which allows researchers to do other things with the data.

• A common distinction for segmenters is the language upon which they are applied.

• Segmenters are used widely in computing. Significant fields of research on segmenters are Natural Language Processing, Computational Linguistics, Computer Science, etc.

Why are segmenters important?

• Content / data analysis: Especially in East-Asian languages, there is no spaces between words. Punctuations sometimes exist, sometimes not.

• Wide variety of usages: Computational linguistics, spam, etc.
**General research questions**

- How can we identify words? Punctuation? Places where we can put a space to segment the data
- The representation methods which make sense to a computer and to existing software.
- The efficiency and accuracy of the segmenter

**Why do computers hate Chinese characters?**

- The definition of the token is vague
  - What are the words? Tokens? Where is the punctuation?
  - Without tokens, we can’t do things at the **word-level**, we can only do things at the phrase or sentence level. This sucks if you are a word processor or spam filter or the like, i.e. YOU’RE DUMB AND YOU REALLY LOVE SPAM.
- It takes 2 bytes for a meaningful character

**American president indicated that the economical growth in China is very fast**
Why do computers hate Chinese characters?

- The encoding is different
  - Chinese language belongs to ideographical word system, which usually employs special ways of encoding to represent characters, e.g. gbk, Big5, unicode
- So what do we do?

We think about strategies for parsing the sentence, breaking it up into smaller chunks.

Why am I studying segmenters?

- TREC: Spam track in TREC published Chinese corpus in 2006, which gives new challenges to participants
- Want to try some different strategies
  - Most of the strategies are the same: POS taggers, lexical analyzers, sliding-window …
  - Only a few do extremely well (+95% or higher accuracy)
- Have great applications for other uses (e.g. spam filters, NLP, etc.)

I have to complete my independent studies with Kiduk :-)
My strategy & results so far ...

• Working with Gavin, we’ve devised a strategy based on a left-to-right maximum matching approach using a lookup table based on a robust, but not too lengthy dictionary of common Chinese words.
• The variant of maximum matching
• Testing corpus: Chinese emails from spam track

Segmentation Process

• This sentence is passed through an algorithm which looks at a window of the first \( n \) number of characters and tries to find a match in the dictionary, where \( n \) equals the longest term in the dictionary
• If a match is found, it saves the token(s) to an array and replace the token(s) with double spaces to avoid them being parsed again
• If not, the dictionary is then consulted again for shorter terms to see if a match can be found.
• This process continues until all the shortest terms (2 words) are all looked up in the dictionary
My strategy & results so far …

After the segmentation, all the tokens are recorded as a unique index number.

A new document (email) is then generated with all the index numbers separated by the space for spam detection purpose.

Integrate the segmenter with the spam filter.
My strategy & results so far …

[Result evaluation]

• Compared with segmenter
  • While our segmenter does better at the accuracy, the Mandarin segmenter does better at the efficiency.
  • It’s hard to compare the results since both do a good job at from different perspectives.
  • However, in general it does a great job such that we can integrate it with common Spam filters such as SpamAssassin or Bspam.

What’s next …

• Improve the dictionary. The dictionary is the most important component of this strategy. Too big of a dictionary and the tokenization will be too fine-grained, too small of the dictionary and the tokenization will just be poor.
  • Improve the lookup efficiency which means refactoring the code to use advantages of the programming language syntax as well as memory optimizations.