Technical Report of LA Manager

Huajie Zhang, MyungHyun Lee (Kent)

April 21, 2010

Abstract

This document presents the technical report for the project *Language Analyzer Manager* (*LA Manager*). LA Manager is an essential part of the SF-1 system. Its aim is to parse terms from a given text and expand the terms by various analyzing methods including morpheme analyzing. It can be used as basic text pre-processing tool for many information retrieval and machine learning applications. This document will go over the features of the modified LA Manager based on the originial verion from SF-1 project.

Contents

1 Document History

Date	Author	Description
2009-06-30	Huajie Zhang	Initial version

2 Overview

The overview of LA Manager is shown in Figure 1.

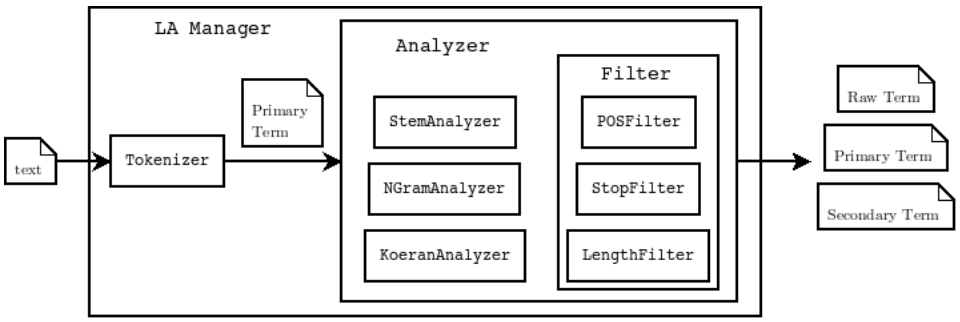

Figure 1: LA Manager at a glance

2.1 Output

Generally, LA Manager generates the following three kinds of outputs:

- *Raw Term*: These are the raw terms that completely represent the original raw text.
- *Primary Term*: Within the Raw Terms, the terms that consists either alphabetic or numeric characters fall into this category.
- *Secondary Term*: The terms that are analyzed from the Primary Terms by analyzers fall into this category.

Raw Terms are saved as an array in the order of the word offset value, and is used to generate the original text out of a sequence of term IDs. Primary Term and Secondary Term are indexed.

2.2 Module

The components of LA Manager consists of two groups: Tokenizer and Analyzers (Filter, StemAnalyzer, NGramAnalyzer). The input text is first passed through the Tokenizer and then processed using different analyzers, which can be pipelined as follows:

1. *Tokenizer*: split the input text into a list of tokens, the tokenizer can be configured by different options.
2. *Filter* (*StopFilter*, *NounFilter* or *POSFilter*): filter out the stop words or analyze the Part-Of-Speech (POS) of words and filter the words of user specified types, currently not available.
3. *StemAnalyzer*: perform stemming on words.
4. *NGramAnalyzer*: extract the char-based ngram or word-based ngram.

The Tokenizer is necessary part and the analyzers are optional. The Tokenizer ouputs Raw Term and Primary Term. The Primary Term are input into the analyzers (2,3,4), which will generate Secondary Term as output.

3 Tokenizer

3.1 Introduction

The tokenizing is done by categorizing characters into two basic categories: delimters (white space and special characters by default) and content characters (alphabetic, numeric characters). Continuous characters in the same category are grouped into a single term. For example, SF-1 Revolution! will return tokens SF, 1, and Revolution.

In LA Manager, each output term is assigned a word offset which is calculated by the term's relative position within the original text. When a text "It's like A@B" is given to the Tokenizer, it is parsed like the following:

Table 1: Raw Term Example

TERM	WORD OFFSET
It	0
'	1
s	2
like	3
A	4
@	5
B	6

Notice, the white space doesn't occupy a position. Thus the white spaces that are between "s" and "like" and between "like" and "A" are ignored.

As one can see, the terms are assigned with word offsets which show their relative position in the original text. These terms are the Raw Terms. They are saved in a sequence sorted by their word offset values. The Raw Terms are used to rebuild the original text, since they exactly represent the original text when put together.

From these Raw Terms the alphabetic + numeric characters are selected to be the Primary Terms (see Table 2). These terms are usually the ones that are indexed or searched with, along with the Secondary Terms. It can be noticed that the Primary Terms do not have their own word offset sequence, but follow the word offset number of the Raw Terms. This way the terms can be easily found in the Raw Term list when needed.

Table 2: Primary Term Example

TERM	WORD OFFSET
It	0
s	2
like	3
A	4
B	6

3.2 Configuration

Tokenizer uses several options to change how the terms are parsed from the original text. A single character is set to be one of the following options: Allow, Divide, Unite.

The examples assume that text "A@B" is given, and the charcter '@' is set to each behavior. The default Tokenizer will parse the text "A@B" into tokens "A" and "B".

Method	Description
Allow	The characters set as *allow* will be removed from the list of delimiters. E.g. "A@B" => "A@B"
Divide	The characters will be set to be a delimiter. E.g. "A@B" => "A", "B"
Unite	The character will be used to concatenate the two adjacent tokens on either side of the character. E.g. "A@B" => "AB"

The following shows an example of how the list of Primary Terms change as the characters ['] and [@] are set.

- Allow: The character is recognized as an alphabetic character.

TERM	WORD OFFSET
It's	0
like	1
A@B	2

- Divide: The character is recognized as a delimiter. Acts like the default Tokenizer.

TERM	WORD OFFSET
It	0
s	2
like	3
A	4
B	6

- Unite: The character is used to concatenate two terms on both side.

TERM	WORD OFFSET
Its	0
like	1
AB	2

There are two things to note here.

- Primary Terms are the terms that are indexed, and not the Raw Terms, and they may not be the same Term as the Raw Term.
- Primary Terms have their word offsets set to the word offset value of the Raw Terms that they correspond to. In some cases, there may be more than one corresponding Raw Term.

4 Filter and Analyzer

When text are tokenized into a list of Raw Terms and Primary Terms, the latter can be further processed by several Filters and Analyzers. The terms created during this process

are Secondary Terms. Secondary Terms have the same word offsets as the Primary Terms that they expanded from.

4.1 Filter

Filters are used to remove the words of specific type from the primary list. StopFilter will filter out the stop words in the Primary Term list. NounFilter will only remain noun words in the list. LengthFilter will filter out word with length smaller (or larger) than a threshold. These Filters are useful to remove the noise from raw text according to the users' specific need.

4.2 Analyzer

StemAnalyzer is used to stem the original words through morphological analysis. NGramAnalyzer expands the term into N gram terms. The N value can be specified by user. Two types of N gram can be extracted: character-based approach and term-based approach. Character-based approach will use extract continuous N characters in a word, for example, "win", "ind" in "windows", while term-based approach extract continious N words as N gram in a sentance.