MeTA: A Modern C++ Data Sciences Toolkit

2014.11.18

Sean Massung and Chase Geigle

Outline

A Classic Text-Mining Example

Document Classification

Step 1

SvmLight[1]

[1] http://svmlight.joachims.org/

Step 2

SvmPerf[2]

[2] http://www.cs.cornell.edu/people/tj/svm_light/svm_perf.html

Step 3

SvmMulticlass[3]

[3] http://www.cs.cornell.edu/people/tj/svm_light/svm_multiclass.html

Step 4

Data munging

Step 5

Feature generation

Handling unicode

Step 7

Out of memory

Step 8

Incapacitated server, different algorithm?

Step 9

Despair???

Step 0

MeTA!

To the Rescue

META[4] has been created to solve these text miner's headaches once and for all by

1. uniting machine learning and information retrieval algorithms under one roof (no more tool hunting, separate compiling and configuration, etc.),
   - indexing, ranking, searching, topic modeling, classification, parallelization, sequence labeling, (soon) parsing and information network analysis
2. integrating *all steps of text tokenization* into one framework,
3. respecting the Unicode standards,
4. supporting out-of-the-box out-of-core classification,
5. being competitive with existing, highly specific tools, and
6. *letting you write as little code as possible* (often none at all)

[4] `http://meta-toolkit.github.io/meta/`

Components of MeTA

Overview

MeTA can be broken down into the following components:

- Indexes (data storage)
- Analyzers and Filters (text tokenization and processing)
- Search engine (on top of `inverted_index`)
- Classification (on top of `forward_index`)
- Topic modeling (on top of `forward_index`)
- Sequence labeling
- Information networks (in progress)

Text Tokenization

Tokenizers

Tokenizers convert documents to token streams;

Filters

Filters add, remove, or modify tokens;

Analyzers

and **Analyzers** count.

Example

icu-tokenizer $\rightarrow$ lowercase $\rightarrow$ length(2, 35) $\rightarrow$ unigrams

```
[[analyzers]]
method = "ngram-word"
ngram = 1
    [[analyzers.filter]]
    type = "icu-tokenizer"

    [[analyzers.filter]]
    type = "lowercase"

    [[analyzers.filter]]
    type = "length"
    min = 2
    max = 35
```

Porter2 English Stemmer[5]

```
[[analyzers.filter]]
type = "porter2"
```

$$\{waits,\ waited,\ waiting\} \rightarrow wait$$

$$\{consist,\ consisted,\ consistency,\ consists\} \rightarrow consist$$

$$\{run,\ runs,\ running\} \rightarrow run$$

[5] https://github.com/smassung/porter2_stemmer

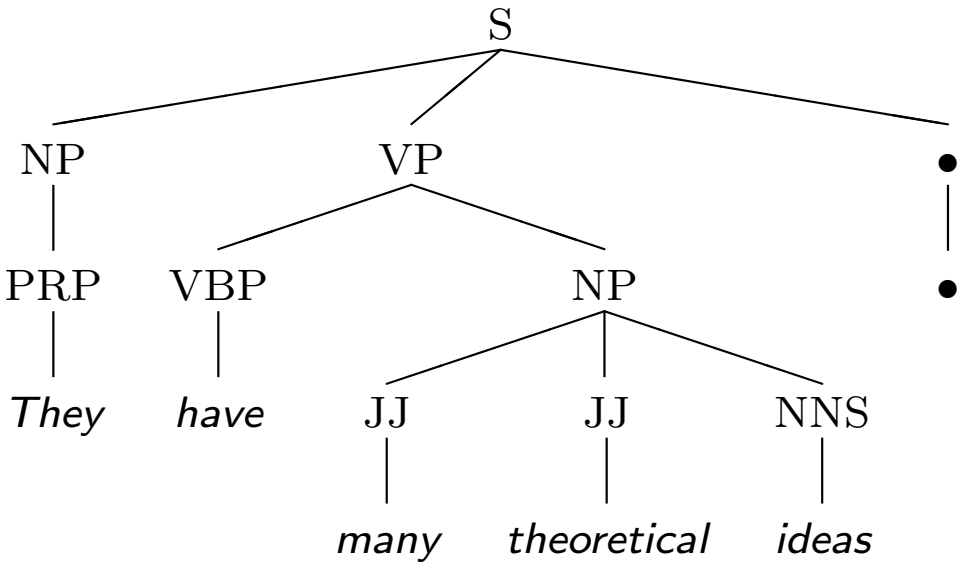
Feature Combining

$$(\text{default-chain} \rightarrow \text{bigrams}) \cup \text{POS-tags}$$

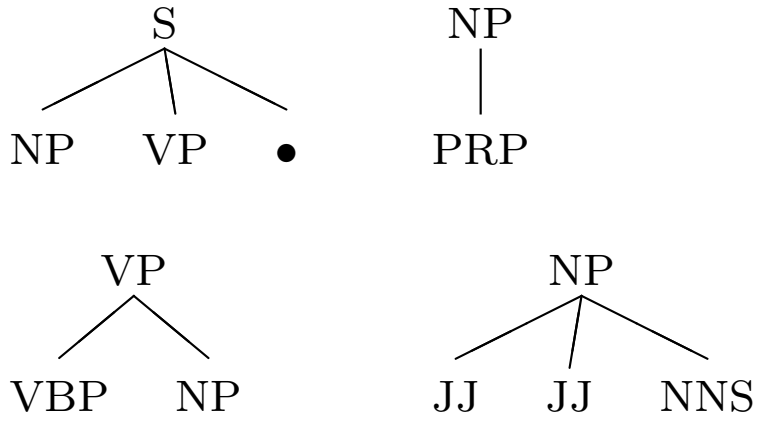
```
[[analyzers]]
method = "ngram-word"
ngram = 2
filter = "default-chain"

[[analyzers]]
method = "ngram-pos"
ngram = 1
```

Features from Parse Trees

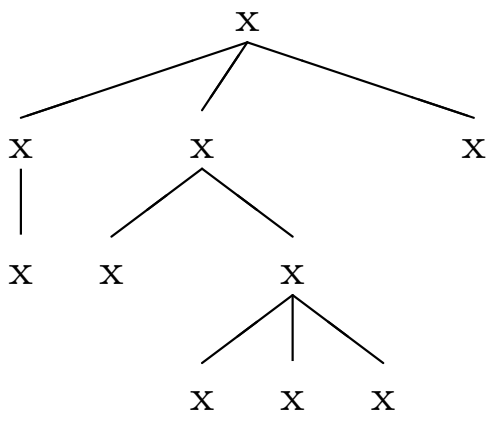
Example parse tree of a sentence

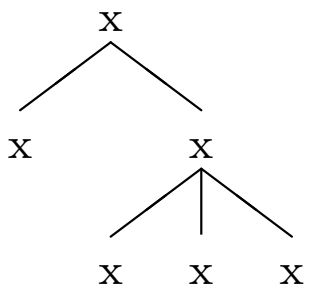


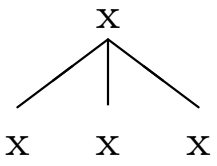
Rewrite rule features



Structural Features from Parse Trees







Search

Indexing Process

corpus $\rightarrow$ documents $\rightarrow$ (analyzer: tokenizer $\rightarrow$ filters) $\rightarrow$ indexer

Corpus variants:

- `line_corpus`: one document per line
- `file_corpus`: one document per file

Ranking Functions

- Configurable, state-of-the-art existing rankers

```
[ranker]
method = "bm25"
k1 = 1.2 # optional
b = 0.75 # optional
k3 = 500 # optional
```

  - Absolute discounting
  - Dirichlet prior
  - Jelinek-Mercer
  - Okapi BM25(L)
  - Pivoted length normalization

- Easy to add new rankers (see `score_data`)

Classification

forward_index

forward_index internally is:

- LIBLINEAR formatted data
- Ancillary structures for efficient access

Ramification: Use your own feature generation, or MeTA's interchangeably!

Classification Methods

Binary classifiers:

- SGD trained:
  - SVM (hinge, smooth hinge, squared hinge)
  - Perceptron
  - Logistic regression

Multiclass adapters:

- One-vs-all
- One-vs-one

Multiclass classifiers:

- Wrapper for LIBSVM/LIBLINEAR
- Naïve Bayes
- Winnow
- Dual perceptron (kernels)
- kNN (using the search engine)

Out-of-core Support

An experiment on **rcv1** with only **100MB** of memory:

LIBLINEAR crashes; META completes in 2 minutes[6]

[6] http://meta-toolkit.github.io/meta/online-learning.html

Topic Modeling

Models

Currently has many flavors of LDA inference:

- Collapsed Gibbs sampling
- Parallel collapsed Gibbs sampling (approximate)
- Collapsed Variational Bayes (0-th order approximation)
- Stochastic Collapsed Variational Bayes (0-th order approximation)

Sequence Labeling

Sequence Labeling

General task on sequence data in NLP, useful for

- Part of speech (POS) tagging
- Chunking/shallow parsing
- Named-entity recognition (NER)
- Information extraction

Supported in MeTA via an implementation of linear-chain conditional random fields (CRFs)

Sequence Labeling Framework

sequences $\rightarrow$ sequence_analyzer $\rightarrow$ model

Sequence analyzer creates observation features for each element:

- Lexical features: $w_t = v_i$, w_t is capitalized, w_t is a number, ...
- Context features: $w_{t+1} = v_i$, $w_{t+2} = v_i$, $w_{t-1} = v_i$, $w_{t-2} = v_i$, ...

User-configurable: specify a function that takes an observation and generates features (represented as strings)

(Ships with a default analyzer for POS tagging)

`http://meta-toolkit.github.io/meta/`

MIT & UIUC/NCSA Licensed
(Liberal; use it at work!)

Questions?