rnlp Documentation

Release 0.3.0

Alexander L. Hayes (@hayesall)

Jan 30, 2020

Getting Started:

Relational NLP Preprocessing: A Python package and tool for converting text into a set of relational facts.

Source Code GitHub

Bugtracker GitHub Issues

CHAPTER 1

Overview

Fig. 1: The U.S. Declaration of Independence is available in the corpus submodule for demonstration. Here it is converted to a set of facts using the imported Python package.

`rnlp` is intended to be a general-purpose tool for converting text into relational facts for use with relational reasoning systems (such as BoostSRL).

Text is converted into relational facts, built around the basic building blocks of *Words*, *Sentences*, and *Blocks*.

Words are individual units of text, such as the words you are currently reading. *Sentences* are a collection of words, often separated by punctuation. *Blocks* are a collection of sentences.

1.1 Environment Setup

The first step is to get Python running on your machine (skip to the next step if you've already done this).

1.1.1 Linux (yum/dnf)

```
$ sudo yum update
$ sudo yum install python
```

1.1.2 Linux (apt-get)

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install python
```

1.1.3 Windows

Download Python from python.org or anaconda.com.

A fairly in-depth guide is available as part of the Conda documentation.

1.2 Installation

Stable builds on PyPi

```
pip install rnlp
```

Some modules in nltk need to be available:

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
```

1.3 Quick Start

1.3.1 Relations

The relations created by `rnlp` include the following:

- Sentence's Relative Position in Block:
  - `earlySentenceInBlock`: Sentence occurs within the first third of a block's length.
  - `midWaySentenceInBlock`: Sentence occurs between the first and last third of a block's length.
  - `lateSentenceInBlock`: Sentence occurs within the last third of a block's length.
- Word's Relative Position in Sentence:
  - `earlyWordInSentence`: Word occurs within the first third of a sentence.
  - `midWayWordInSentence`: Word occurs between a third and two-thirds of a sentence.
  - `lateWordInSentence`: Word occurs within the last third of a sentence.
- Relative Position Between Items:
  - `nextWordInSentence`: Pointer from a word to its neighbor.
  - `nextSentenceInBlock`: Pointer from a sentence to its neighbor.
- Existential Semantics:
  - `sentenceInBlock`: Sentence occurs in a particular block.
  - `wordInSentence`: Word occurs in a particular sentence.
- Low-Level Information about words:
  - `wordString`: A string representation of a word.
  - `partOfSpeechTag`: The word's part of speech (as determined by the nltk part-of-speech tagger).

1.3.2 From text to Relational Facts

Consider the example file `example_files/doi.txt`, the U.S. Declaration of Independence:

```
In Congress, July 4, 1776. The unanimous Declaration of the thirteen united
States of America, When in the Course of human events, it becomes necessary
for one people to dissolve the political bands which have connected them
with another, and to assume among the powers of the earth, the separate and
equal station to which the Laws of Nature and of Nature's God entitle them,
a decent respect to the opinions of mankind requires that they should
declare the causes which impel them to the separation.
...
...
```

`rnlp` can be used either as a commandline tool or as an imported Python Package.

Commandline

```
$ python -m rnlp -f example_files/doi.txt
Reading corpus from file(s)...
Creating background file...
100%|| 18/18 [00:00<00:00, 38it/s]
```

Imported

```
from rnlp.corpus import declaration
import rnlp

doi = declaration()
rnlp.converter(doi)
```

```
nextSentenceInBlock(1,1_1,1_2).
earlySentenceInBlock(1,1_1).
sentenceInBlock(1_1,1).
wordString(1_1_1,'In').
partOfSpeech(1_1_1,"IN").
nextWordInSentence(1_1,1_1_1,1_1_2).
earlyWordInSentence(1_1,1_1_1).
wordInSentence(1_1_1,1_1).
wordString(1_1_2,'Congress').
partOfSpeech(1_1_2,"NNP").
...
...
```

1.4 Learning

This is a brief overview of a learning task. Requirements for a more specific task may vary substantially based on the goals or the data available to you.

We now have `bk.txt` and `facts.txt` as a result of the previous step. In order to get show some results, we will construct a toy data set from predicates easily available in the facts file.

The *Declaration of Independence* contains a set of phrases called the "List of Grievances", where the Founders spell out the 27 violations by King George III.

We can turn these into a text classification task where we learn the structure of the grievances.

1.4.1 Positive and Negative Examples

Labeling data is often a task of its own, but we will take a shortcut and label sentences beginning with **"He"** or **"For"** as being positive examples. Everything else is labeled as negative.

1. Create a 'train' directory for our training data.

```
mkdir train
```

2. This combination of grep, awk, and sort finds all occurances of "He" and "For" in the facts; labels them as a positive example; and adds them to a train_pos.txt file.

```
grep "'He'\|'For'" facts.txt |
  awk '{gsub("wordString","sentenceContainsTarget");
        gsub("_[0-9]*,.*",").");
        print}' |
  sort -u > train/train_pos.txt
```

3. This command does something similar, but returns all sentences *not* containing the example.

```
grep "wordString" facts.txt |
  grep -v "'He'\|'For'" |
  awk '{gsub("wordString","sentenceContainsTarget");
        gsub("_[0-9]*,.*",").");
        print}' |
  sort -u > train/train_neg.txt
```

4. Some sentences have been counted twice–some negative examples are also present in the positive examples. Luckily we can do a set difference to fix this.

```
sort train/train_neg.txt train/train_pos.txt train/train_pos.txt |
  uniq -u > temp; mv temp train/train_neg.txt
```

5. We want to learn about the structure of sentences, so we will replace the default target with our own and move a copy into the train/ directory.

```
awk '{gsub(".*Target.*",
           "mode: sentenceContainsTarget(+SID).");
      print}' bk.txt > train/train_bk.txt
```

6. Finally, move the facts into the same directory.

```
mv facts.txt train/train_facts.txt
```

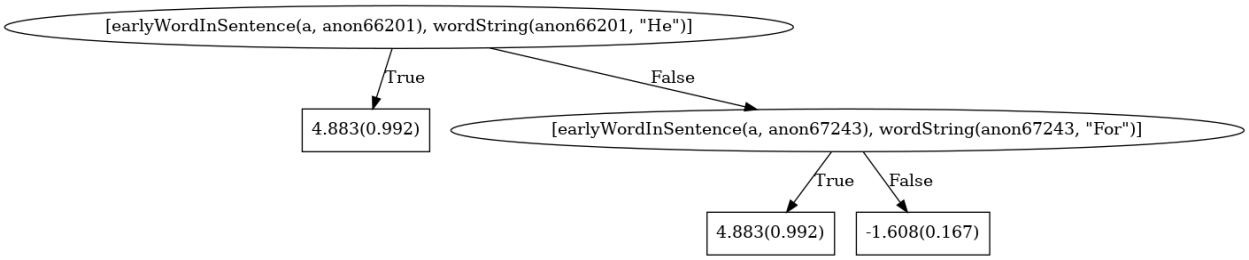
Our train directory should now contain four files, and the 27 positive examples each correspond to one of the grievances.

```
train_bk.txt          18 lines
train_facts.txt     6736 lines
train_neg.txt         17 lines
train_pos.txt         27 lines
```

1.4.2 BoostSRL

Now that our data is organized, we use BoostSRL for learning. Download a copy of the jar file from the website and move it to the base of the repository.

```
java -jar v1-0.jar -l -combine \
     -train train/ -target sentenceContainsTarget \
     -trees 25
```



As expected, the model says that if a word appears early in a sentence, and the string representation of that word is "He": the sentence is likely to be a member of the list of grievances (0.992).

Otherwise, the model makes the same check for the word "For", assigning a high probability if it is (0.992) and a lower probability if not (0.167).

We can interpret this model as saying "If an early word in the sentence is 'He' or 'For', the sentence is part of the list of grievances."

1.5 `rnlp`: Package Overview

1.6 `rnlp.parse` module

1.7 `rnlp.textprocessing` module

1.8 `rnlp.corpus` module