DIGITAL ASSIGNMENT #1

Apache Software Foundation

Introduction

The Apache Software Foundation (ASF) is an American non-profit corporation founded in June 1999. [1] ASF stems from the Apache group and was incorporated in Delaware, USA. [2]

```
A. Effective date of incorporation

By general consent, it was agreed that the official effective date of
incorporation for the Apache Software Foundation is to be set as June 1,
1999.
```

***Fig. 1*: Section 6, Special Orders from the ASF Board of Directors Meeting Minutes.** **[1]**

ASF is a decentralized network of open source developers that produces Free and open-source software (FOSS) under an Apache License. Furthermore, commercial support provided from the Apache Group is provided without the risk of platform lock-in. Platform lock-ins makes a customer dependent on a vendor for proprietary support which may be but not limited to hardware or software support. Apple and Microsoft are examples of vendors that were criticized for such purposes. [3][4]

Apache OpenNLP

The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text. Developed by Apache Software Foundation, the toolkit was initially released on April 22, 2004. However, did not receive a stable release (v1.8.4) until December 26, 2017. OpenNLP, similar to StanfordNLP, is written in Java and was started by Jason Baldridge and Gann Bierner while they were graduate students in the Division of Informatics at the University of Edinburgh. [5][6]

Despite not being inherently useful as an independent platform, OpenNLP can be integrated with other software to assist in the processing of texts via an API. The toolkit provides functionality to perform sentence detection, tokenization, POS tagging, chunking and parsing, named-entity recognition (NER), and coreference. [7] “While not necessarily state of the art anymore in its approach, it remains a solid choice that is easy to get up and running.” Furthermore, OpenNLP may not necessarily be as “state-of-the art” as Stanford’s Core NLP Suite due to its different underlying approach. [8] However, it does provide basic functionality and very easy installation.

Design

Classifying

The OpenNLP Language Detector classifies a document in ISO-639-3 languages according to the model capabilities. A model can be trained with Maxent, Perceptron or Naive Bayes algorithms. By default, normalizes a text and the context generator extracts n-grams of size 1, 2 and 3. The n-gram sizes, the normalization and the context generator can be customized by extending the *LanguageDetectorFactory*.

Normalizer	Description
EmojiCharSequenceNormalizer	Replaces emojis by blank space
UrlCharSequenceNormalizer	Replaces URLs and E-Mails by a blank space.
TwitterCharSequenceNormalizer	Replaces hashtags and Twitter user names by blank spaces.
NumberCharSequenceNormalizer	Replaces number sequences by blank spaces
ShrinkCharSequenceNormalizer	Shrink characters that repeats three or more times to only two repetitions.

***Table 1*: Normalizers used in classifying in OpenNLP.** **[9]**

Sentence Detection

Sentence segmentation is defined as a process of breaking a stream of input text into sentences for subsequent processing. Most tools define sentences in English as long process units separated by white spaces. Sentences are identified based on boundaries which in this case in punctuation.

A sentence in OpenNLP is defined as, "the longest white space trimmed character sequence between two punctuation marks". The first and last sentences in a paragraph are an exception to this rule. The first non-whitespace character is assumed to be the begin of a sentence, and the last non-whitespace character is assumed to be a sentence end. This is a very trivial and primitive approach as it faces the problem of sentence boundary disambiguation, since, punctuations in English are deemed challenging for tokenizers.

```
Pierre Vinken, 61 years old, will join the board as a nonexecutive director
Nov. 29. Mr. Vinken is
chairman of Elsevier N.V., the Dutch publishing group. Rudolph Agnew, 55
years
old and former chairman of Consolidated Gold Fields PLC, was named a director
of this
British industrial conglomerate.
```

After detecting the sentence boundaries each sentence is written in its own line:

```
Pierre Vinken, 61 years old, will join the board as a nonexecutive director
Nov. 29.
Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group.
Rudolph Agnew, 55 years old and former chairman of Consolidated Gold Fields
PLC,
    was named a director of this British industrial conglomerate.
```

The OpenNLP Sentence Detector cannot identify sentence boundaries based on the contents of the sentence. A prominent example is the first sentence in an article where the title is mistakenly identified to be the first part of the first sentence. Most components in OpenNLP expect input which is segmented into sentences.

Tokenizers

OpenNLP offers three main tokenizers:

- *Whitespace Tokenizer* - A whitespace tokenizer, non-whitespace sequences are identified as tokens
- *Simple Tokenizer* - A character class tokenizer, sequences of the same character class are tokens
- *Learnable Tokenizer* - A maximum entropy tokenizer, detects token boundaries based on probability model

Name Entity Recognition

The Name Finder can detect named entities and numbers in text. To be able to detect entities the Name Finder needs a model. The model is dependent on the language and entity type it was trained for. Thus, making it a little inconvenient [9][10]. However, the OpenNLP projects offers a number of pre-trained name finder models which are trained on various freely available corpora. This further emphasizes the use of OpenNLP as a machine learning toolkit for text analytics rather than just an NLP kit.

Using the English person model available on the downloads page, the Name Finder Tool will output the text with markup for person names as:

```
<START:person> Pierre Vinken <END> , 61 years old , will join the board as
a nonexecutive director Nov. 29 .
Mr . <START:person> Vinken <END> is chairman of Elsevier N.V. , the Dutch
publishing group .
<START:person> Rudolph Agnew <END> , 55 years old and former chairman of
Consolidated Gold Fields PLC ,
    was named a director of this British industrial conglomerate .
```

OpenNLP has a model per Name Entity Recognition tag, however, Stanford Core NLP on the other hand detects all tags with a single Annotator. In terms of named-entity recognition, Stanford CoreNLP works better on general-purpose text. OpenNLP might be a better choice when one wants to extract information from text by using their own models trained on a corpus. [12]

Performance

Dataset Used: *abstract_100.txt*

POS Tagging

Stanford CoreNLP is more efficient than OpenNLP in POS tagging. Though, OpenNLP runs faster than CoreNLP in NER tagging, CoreNLP extracts all NER tags while OpenNLP extracts only location tags.

	OpenNLP	***Stanford CoreNLP***
POS	11.65s	2.69s
NER	11.26s	18.04s

POS Tag Results	
OpenNLP	**Stanford CoreNLP**
26,360	25,919

Table 2: Runtime on a 2016 Macbook Pro with 8 GBs RAM and 256 GBs SSD, including tokenization runtime (left) and POS Tag Results (right) **[12]**

The results produced by these two tools are very similar. The difference between their number of results may be due to their own tokenizer. Stanford CoreNLP tokenizer does a better job in handling punctuations. For example, OpenNLP recognizes [."] as a token and tags it as coordinating conjunction ("CC"), but Stanford NLP would not tag it. This is the main reason that OpenNLP produces more results than Stanford NLP. Therefore, Stanford CoreNLP may have a higher accuracy over OpenNLP because of more accurate tokenization.

Name Entity Recognition

NER Results	
OpenNLP	**Stanford CoreNLP**
150	173

***Table 3: Name Entity Recognition results*[12]**

OpenNLP uses a location NER model only, so we only compare the location NER results. OpenNLP provides results as offsets, e.g., "New York" as a result, while Stanford CoreNLP produces "New" and "York." For easy comparison, the results of OpenNLP are separated word by word.

OpenNLP cannot figure out abbreviations that contain punctuations of a location name while Stanford NLP can. For example, OpenNLP doesn't tag "N.Y." but Stanford NLP does. Also, Stanford NLP can recognize non-English alphabetical-based words, while OpenNLP needs another model to do it. Overall, Stanford CoreNLP tends to be more accurate.

Naïve Bayes

OpenNLP also provides a Naïve Bayes classifier as an unstable release. Usually, labelled data must be provided in order to train the classifier. Furthermore, according to Nigam et al [13], the process of bootstrapping Naive Bayes classifiers over unlabeled data can be used. This is because unlabeled data is usually easier to obtain and cheaper to collect as compared to labelled data. However, this won't be necessary as OpenNLP provides a Naive Bayes classifier that can be used for that purpose.

Appendix C shows the same code used by [11] to evaluate the classifier. The accuracies are as follows:

1. Subjectivity Classification
    - Perceptron: 57.54% (100 iterations)
    - Perceptron: 59.96% (1000 iterations)
    - Maxent: 91.48% (100 iterations)
    - Maxent: 90.68% (1000 iterations)
    - **Naive Bayes: 90.72%**
2. Sentiment Polarity Classification
    - Perceptron: 49.70% (100 iterations)
    - Perceptron: 49.85% (1000 iterations)
    - Maxent: 77.11% (100 iterations)
    - Maxent: 77.55% (1000 iterations)
    - **Naive Bayes: 75.65%**

Dataset Used: *http://www.cs.cornell.edu/people/pabo/movie-review-data/*

Maintenance & Support

Support for OpenNLP is available on multiple websites –

GitHub
https://github.com/apache/opennlp/network
Any issues regarding the source code can be submitted here.

The GitHub repository is fairly active with a total of 645 stars and 83 watchers. Furthermore, *java.libhunt.com* [13] has given OpenNLP a popularity rating of 6.0 and an activity rating of 7.5. It is likely that due to alternative packages such as Spacy, NLTK and Stanford CoreNLP that offer more comprehensive support and simpler packages, OpenNLP has seen a decline in use over the years.

On Apache's GitHub repository, commits occur at a frequency of around once in 2 weeks. Bug fixes are typically fixed by the developers working on the project itself. Two active developers are *kottman* and *kojisekig*.

CoreNLP		Apache OpenNLP
	Repository	
5,204	★ Stars	645
499	Watchers	83
1,904	Forks	282
1 day ago	Last Commit	13 days ago
	More	
L1	Code Quality	L1
Java	</> Language	Java

***Fig. 2*: Comparison between OpenNLP and CoreNLP [14]**

Apache
https://issues.apache.org/jira/secure/Dashboard.jspa
Apache's dedicate forum service for forum support.

In spite of being open source, OpenNLP doesn't receive bug fixes as often as packages such as CoreNLP or NLTK. According to the forum support page, there are several unresolved bugs that have been tagged "major" for over the past two months.

Stanford has a development team specifically dedicated for NLP support. In addition to this, the Apache Software Foundation doesn't receive funding at the same scale as Stanford. Furthermore, after observing their social media presence, most of the lead developers have several other projects they're working on simultaneously. For example, Jörn Kottmann, the team leader in OpenNLP has been working in "Sandstone SA" for the past 6 years as according to his LinkedIn profile [15]. Figure 3 shows the popularity of OpenNLP decreasing over the past year and provides some evidence for the above claims.

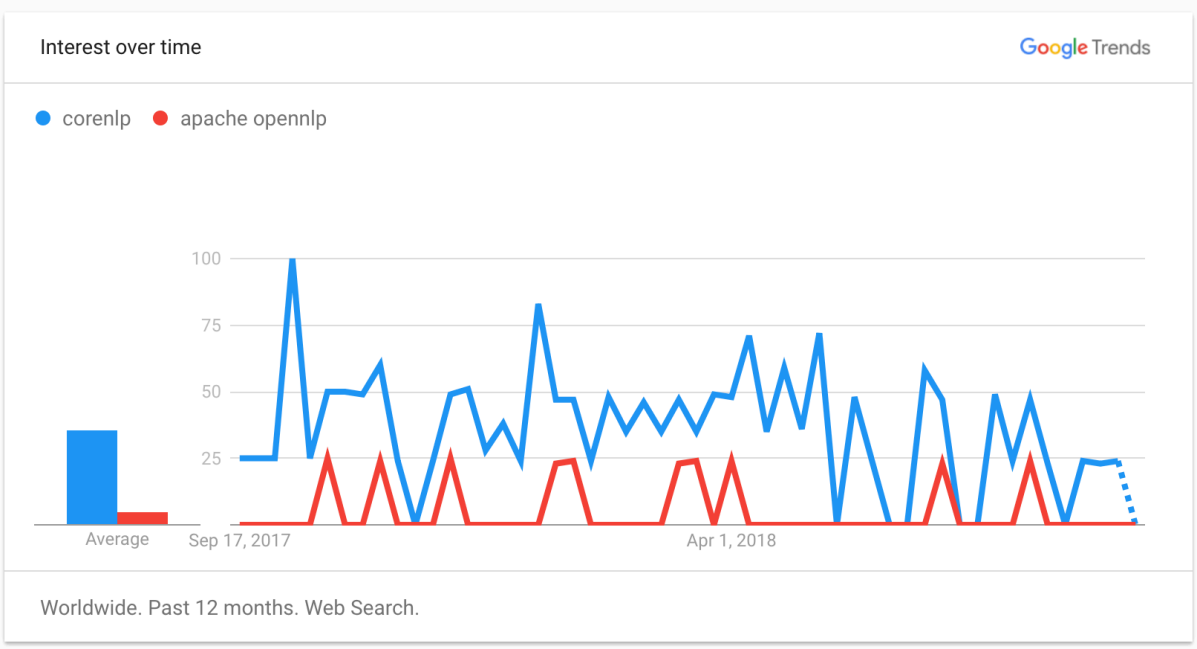

***Fig. 3*: Google Trends for CoreNLP and Apache OpenNLP over the past 12 months.**

Conclusion

Despite the lack of support on its official platform, OpenNLP serves as a great tool for NLP when coupled with machine learning approaches. Its ease of use and simple command line interface make it sufficient for small scale research applications. As an NLP enthusiast, I would recommend OpenNLP for corpuses that tend to deviate from standard written English. One such example would be taking a modern-day transcript from a YouTube video, training it using OpenNLP's training API and then use NER to analyze the text. Another advantage is its simple integration with R. Appendix B provides a sample exercise performed with a free OpenNLP Package on RStudio.

References

[1] https://www.apache.org/foundation/records/minutes/1999/board_minutes_1999_06_01.txt
[2] https://www.apache.org/foundation/records/certificate.html
[3] Baldwin, Roberto. "Why Apple's Custom IPhone Screws Can't Stop the DIY Community." *Wired*, 8 Aug. 2012, www.wired.com/2012/08/if-theres-a-screw-theres-a-way-custom-screws-wont-stop-the-diy-community/.
[4] Robertson, Adi. "How the Antitrust Battles of the 90's Set the Stage for Today's Tech Giants." *The Verge*, 6 Sept. 2018, www.theverge.com/2018/9/6/17827042/antitrust-1990s-microsoft-google-aol-monopoly-lawsuits-history.
[5] https://wiki.apache.org/incubator/OpenNLPProposal
[6] http://www.apache.org/foundation/
[7] http://opennlp.sourceforge.net/README.html
[8] Feed, Grant Ingersoll. "5 Open Source Tools for Taming Text." *Opensource.com*, Opensource.com, 8 July 2015, opensource.com/business/15/7/five-open-source-nlp-tools.
[9] https://opennlp.apache.org/docs/1.9.0/manual/opennlp.html
[10] https://stackoverflow.com/questions/40025981/opennlp-vs-stanford-corenlp
[11] https://aiaioo.wordpress.com/2016/01/13/naive-bayes-classifier-in-opennlp/
[12] https://github.com/Texera/texera/wiki/Evaluating-OpenNLP
[13] Nigam, Kamal, et al. "Using EM to Classify Text from Labeled and Unlabeled Documents." 1998, doi:10.21236/ada350490.
[14] https://java.libhunt.com/compare-corenlp-vs-apache-opennlp
[15] https://www.linkedin.com/in/j%C3%B6rn-kottmann-370a4239/?originalSubdomain=lu

Appendix A: Tutorials

Source	**Link**
Tutorials Point	https://www.tutorialspoint.com/opennlp/
Codeburst	https://codeburst.io/nlp-implementation-using-java-opennlp-guide-and-examples-80d86b02b5b5
Tutorialkart	https://www.tutorialkart.com/opennlp/apache-opennlp-tutorial/
Programcreek	https://www.programcreek.com/2012/05/opennlp-tutorial/
Apache.org	https://cwiki.apache.org/confluence/display/OPENNLP/Apache+OpenNLP+2017+in+Review
YouTube	https://www.youtube.com/watch?v=RggCAXBe6BA

Appendix B: Apache OpenNLP in R

Introduction

An interface to the Apache OpenNLP tools (version 1.5.3). The Apache OpenNLP library is a machine learning based toolkit for the processing of natural language text written in Java. It supports the most common NLP tasks, such as tokenization, sentence segmentation, part-of-speech tagging, named entity extraction, chunking, parsing, and coreference resolution. See OpenNLP for more information

Maxent_Chunk_Annotator

Apache OpenNLP based chunk annotators

Generate an annotator which computes chunk annotations using the Apache OpenNLP Maxent chunker.

```
require(rJava)
```

```
## Loading required package: rJava
```

```
require(NLP)
```

```
## Loading required package: NLP
```

```
require(openNLP)
```

```
## Loading required package: openNLP
```

```
## Requires package 'openNLPmodels.en' from the repository at
## <http://datacube.wu.ac.at>.
## Some text.
s <- paste(c("Pierre Vinken, 61 years old, will join the board as a ",
             "nonexecutive director Nov. 29.\n",
             "Mr. Vinken is chairman of Elsevier N.V., ","the Dutch publishing group."),
           collapse = "")
s <- as.String(s)
## Chunking needs word token annotations with POS tags.
sent_token_annotator <- Maxent_Sent_Token_Annotator()
word_token_annotator <- Maxent_Word_Token_Annotator()
pos_tag_annotator <- Maxent_POS_Tag_Annotator()
a3 <- annotate(s,
               list(sent_token_annotator,
                    word_token_annotator,
                    pos_tag_annotator))
annotate(s, Maxent_Chunk_Annotator(), a3)
```

```
##  id type     start end features
##   1 sentence     1  84 constituents=<<integer,18>>
##   2 sentence    86 153 constituents=<<integer,13>>
##   3 word         1   6 POS=NNP, chunk_tag=B-NP
##   4 word         8  13 POS=NNP, chunk_tag=I-NP
##   5 word        14  14 POS=,, chunk_tag=O
##   6 word        16  17 POS=CD, chunk_tag=B-NP
##   7 word        19  23 POS=NNS, chunk_tag=I-NP
##   8 word        25  27 POS=JJ, chunk_tag=B-ADJP
##   9 word        28  28 POS=,, chunk_tag=O
```

```
##  10 word        30  33 POS=MD, chunk_tag=B-VP
##  11 word        35  38 POS=VB, chunk_tag=I-VP
##  12 word        40  42 POS=DT, chunk_tag=B-NP
##  13 word        44  48 POS=NN, chunk_tag=I-NP
##  14 word        50  51 POS=IN, chunk_tag=B-PP
##  15 word        53  53 POS=DT, chunk_tag=B-NP
##  16 word        55  66 POS=JJ, chunk_tag=I-NP
##  17 word        68  75 POS=NN, chunk_tag=I-NP
##  18 word        77  80 POS=NNP, chunk_tag=B-NP
##  19 word        82  83 POS=CD, chunk_tag=I-NP
##  20 word        84  84 POS=., chunk_tag=O
##  21 word        86  88 POS=NNP, chunk_tag=B-NP
##  22 word        90  95 POS=NNP, chunk_tag=I-NP
##  23 word        97  98 POS=VBZ, chunk_tag=B-VP
##  24 word       100 107 POS=NN, chunk_tag=B-NP
##  25 word       109 110 POS=IN, chunk_tag=B-PP
##  26 word       112 119 POS=NNP, chunk_tag=B-NP
##  27 word       121 124 POS=NNP, chunk_tag=I-NP
##  28 word       125 125 POS=,, chunk_tag=O
##  29 word       127 129 POS=DT, chunk_tag=B-NP
##  30 word       131 135 POS=JJ, chunk_tag=I-NP
##  31 word       137 146 POS=NN, chunk_tag=I-NP
##  32 word       148 152 POS=NN, chunk_tag=I-NP
##  33 word       153 153 POS=., chunk_tag=O
```

```
annotate(s, Maxent_Chunk_Annotator(probs = TRUE), a3)
```

```
##  id type     start end features
##   1 sentence     1  84 constituents=<<integer,18>>
##   2 sentence    86 153 constituents=<<integer,13>>
##   3 word         1   6 POS=NNP, chunk_tag=B-NP, chunk_prob=0.9740431
##   4 word         8  13 POS=NNP, chunk_tag=I-NP, chunk_prob=0.9816025
##   5 word        14  14 POS=,, chunk_tag=O, chunk_prob=0.9863059
##   6 word        16  17 POS=CD, chunk_tag=B-NP, chunk_prob=0.9926662
##   7 word        19  23 POS=NNS, chunk_tag=I-NP, chunk_prob=0.9854421
##   8 word        25  27 POS=JJ, chunk_tag=B-ADJP, chunk_prob=0.9978292
##   9 word        28  28 POS=,, chunk_tag=O, chunk_prob=0.9909762
##  10 word        30  33 POS=MD, chunk_tag=B-VP, chunk_prob=0.979816
##  11 word        35  38 POS=VB, chunk_tag=I-VP, chunk_prob=0.9857121
##  12 word        40  42 POS=DT, chunk_tag=B-NP, chunk_prob=0.9932718
##  13 word        44  48 POS=NN, chunk_tag=I-NP, chunk_prob=0.9947529
##  14 word        50  51 POS=IN, chunk_tag=B-PP, chunk_prob=0.9717558
##  15 word        53  53 POS=DT, chunk_tag=B-NP, chunk_prob=0.9991619
##  16 word        55  66 POS=JJ, chunk_tag=I-NP, chunk_prob=0.9989155
##  17 word        68  75 POS=NN, chunk_tag=I-NP, chunk_prob=0.981308
##  18 word        77  80 POS=NNP, chunk_tag=B-NP, chunk_prob=0.8397682
##  19 word        82  83 POS=CD, chunk_tag=I-NP, chunk_prob=0.9913565
##  20 word        84  84 POS=., chunk_tag=O, chunk_prob=0.992369
##  21 word        86  88 POS=NNP, chunk_tag=B-NP, chunk_prob=0.9910283
##  22 word        90  95 POS=NNP, chunk_tag=I-NP, chunk_prob=0.9902959
##  23 word        97  98 POS=VBZ, chunk_tag=B-VP, chunk_prob=0.9888302
##  24 word       100 107 POS=NN, chunk_tag=B-NP, chunk_prob=0.993464
##  25 word       109 110 POS=IN, chunk_tag=B-PP, chunk_prob=0.9719827
##  26 word       112 119 POS=NNP, chunk_tag=B-NP, chunk_prob=0.9906478
##  27 word       121 124 POS=NNP, chunk_tag=I-NP, chunk_prob=0.9819624
```

```
##  28 word       125 125 POS=,, chunk_tag=O, chunk_prob=0.9897705
##  29 word       127 129 POS=DT, chunk_tag=B-NP, chunk_prob=0.995753
##  30 word       131 135 POS=JJ, chunk_tag=I-NP, chunk_prob=0.9758163
##  31 word       137 146 POS=NN, chunk_tag=I-NP, chunk_prob=0.9990291
##  32 word       148 152 POS=NN, chunk_tag=I-NP, chunk_prob=0.9973766
##  33 word       153 153 POS=., chunk_tag=O, chunk_prob=0.9986785
```

Maxent_Entity_Annotator

Apache OpenNLP based entity annotators

Generate an annotator which computes entity annotations using the Apache OpenNLP Maxent name finder.

```
## Requires package 'openNLPmodels.en' from the repository at
## <http://datacube.wu.ac.at>.
require("NLP")
## Some text.
s <- paste(c("Pierre Vinken, 61 years old, will join the board as a ",
"nonexecutive director Nov. 29.\n",
"Mr. Vinken is chairman of Elsevier N.V., ",
"the Dutch publishing group."),
collapse = "")
s <- as.String(s)
## Need sentence and word token annotations.
sent_token_annotator <- Maxent_Sent_Token_Annotator()
word_token_annotator <- Maxent_Word_Token_Annotator()
a2 <- annotate(s, list(sent_token_annotator, word_token_annotator))
## Entity recognition for persons.
entity_annotator <- Maxent_Entity_Annotator()
entity_annotator
```

```
## An annotator inheriting from classes
##   Simple_Entity_Annotator Annotator
## with description
##   Computes entity annotations using the Apache OpenNLP Maxent name
##   finder employing the default model for language 'en' and kind
##   'person'.
```

```
annotate(s, entity_annotator, a2)
```

```
##  id type     start end features
##   1 sentence     1  84 constituents=<<integer,18>>
##   2 sentence    86 153 constituents=<<integer,13>>
##   3 word         1   6
##   4 word         8  13
##   5 word        14  14
##   6 word        16  17
##   7 word        19  23
##   8 word        25  27
##   9 word        28  28
##  10 word        30  33
##  11 word        35  38
##  12 word        40  42
##  13 word        44  48
##  14 word        50  51
```

```
##  15 word        53  53
##  16 word        55  66
##  17 word        68  75
##  18 word        77  80
##  19 word        82  83
##  20 word        84  84
##  21 word        86  88
##  22 word        90  95
##  23 word        97  98
##  24 word       100 107
##  25 word       109 110
##  26 word       112 119
##  27 word       121 124
##  28 word       125 125
##  29 word       127 129
##  30 word       131 135
##  31 word       137 146
##  32 word       148 152
##  33 word       153 153
##  34 entity       1  13 kind=person
```

```
## Directly:
entity_annotator(s, a2)
```

```
##  id type   start end features
##  34 entity     1  13 kind=person
```

```
## And slice ...
s[entity_annotator(s, a2)]
```

```
## Pierre Vinken
```

```
## Variant with sentence probabilities as features.
annotate(s, Maxent_Entity_Annotator(probs = TRUE), a2)
```

```
##  id type     start end features
##   1 sentence     1  84 constituents=<<integer,18>>
##   2 sentence    86 153 constituents=<<integer,13>>
##   3 word         1   6
##   4 word         8  13
##   5 word        14  14
##   6 word        16  17
##   7 word        19  23
##   8 word        25  27
##   9 word        28  28
##  10 word        30  33
##  11 word        35  38
##  12 word        40  42
##  13 word        44  48
##  14 word        50  51
##  15 word        53  53
##  16 word        55  66
##  17 word        68  75
##  18 word        77  80
##  19 word        82  83
##  20 word        84  84
```

```
##  21 word        86  88
##  22 word        90  95
##  23 word        97  98
##  24 word       100 107
##  25 word       109 110
##  26 word       112 119
##  27 word       121 124
##  28 word       125 125
##  29 word       127 129
##  30 word       131 135
##  31 word       137 146
##  32 word       148 152
##  33 word       153 153
##  34 entity       1  13 kind=person, prob=0.9445758
```

Maxent_POS_Tag_Annotator

Apache OpenNLP based POS tag annotators

Generate an annotator which computes POS tag annotations using the Apache OpenNLP Maxent Part of Speech tagger.

```
require("NLP")
## Some text.
s <- paste(c("Pierre Vinken, 61 years old, will join the board as a ",
"nonexecutive director Nov. 29.\n",
"Mr. Vinken is chairman of Elsevier N.V., ",
"the Dutch publishing group."),
collapse = "")
s <- as.String(s)
## Need sentence and word token annotations.
sent_token_annotator <- Maxent_Sent_Token_Annotator()
word_token_annotator <- Maxent_Word_Token_Annotator()
a2 <- annotate(s, list(sent_token_annotator, word_token_annotator))
pos_tag_annotator <- Maxent_POS_Tag_Annotator()
pos_tag_annotator
```

```
## An annotator inheriting from classes
##   Simple_POS_Tag_Annotator Annotator
## with description
##   Computes POS tag annotations using the Apache OpenNLP Maxent
##   Part of Speech tagger employing the default model for language
##   'en'
```

```
a3 <- annotate(s, pos_tag_annotator, a2)
a3
```

```
##  id type     start end features
##   1 sentence     1  84 constituents=<<integer,18>>
##   2 sentence    86 153 constituents=<<integer,13>>
##   3 word         1   6 POS=NNP
##   4 word         8  13 POS=NNP
##   5 word        14  14 POS=,
##   6 word        16  17 POS=CD
##   7 word        19  23 POS=NNS
```

```
##   8 word        25  27 POS=JJ
##   9 word        28  28 POS=,
##  10 word        30  33 POS=MD
##  11 word        35  38 POS=VB
##  12 word        40  42 POS=DT
##  13 word        44  48 POS=NN
##  14 word        50  51 POS=IN
##  15 word        53  53 POS=DT
##  16 word        55  66 POS=JJ
##  17 word        68  75 POS=NN
##  18 word        77  80 POS=NNP
##  19 word        82  83 POS=CD
##  20 word        84  84 POS=.
##  21 word        86  88 POS=NNP
##  22 word        90  95 POS=NNP
##  23 word        97  98 POS=VBZ
##  24 word       100 107 POS=NN
##  25 word       109 110 POS=IN
##  26 word       112 119 POS=NNP
##  27 word       121 124 POS=NNP
##  28 word       125 125 POS=,
##  29 word       127 129 POS=DT
##  30 word       131 135 POS=JJ
##  31 word       137 146 POS=NN
##  32 word       148 152 POS=NN
##  33 word       153 153 POS=.
```

```
## Variant with POS tag probabilities as (additional) features.
head(annotate(s, Maxent_POS_Tag_Annotator(probs = TRUE), a2))
```

```
##  id type     start end features
##   1 sentence     1  84 constituents=<<integer,18>>
##   2 sentence    86 153 constituents=<<integer,13>>
##   3 word         1   6 POS=NNP, POS_prob=0.9476405
##   4 word         8  13 POS=NNP, POS_prob=0.9692841
##   5 word        14  14 POS=,, POS_prob=0.9884445
##   6 word        16  17 POS=CD, POS_prob=0.9926943
```

```
## Determine the distribution of POS tags for word tokens.
a3w <- subset(a3, type == "word")
tags <- sapply(a3w$features, `[[`, "POS")
tags
```

```
##  [1] "NNP" "NNP" ","   "CD"  "NNS" "JJ"  ","   "MD"  "VB"  "DT"  "NN"
## [12] "IN"  "DT"  "JJ"  "NN"  "NNP" "CD"  "."   "NNP" "NNP" "VBZ" "NN"
## [23] "IN"  "NNP" "NNP" ","   "DT"  "JJ"  "NN"  "NN"  "."
```

```
table(tags)
```

```
## tags
##   ,   .  CD  DT  IN  JJ  MD  NN NNP NNS  VB VBZ
##   3   2   2   3   2   3   1   5   7   1   1   1
```

```
## Extract token/POS pairs (all of them): easy.
sprintf("%s/%s", s[a3w], tags)
```

```
##  [1] "Pierre/NNP"      "Vinken/NNP"      ",/,"
```

```
##  [4] "61/CD"           "years/NNS"       "old/JJ"
##  [7] ",/,"             "will/MD"         "join/VB"
## [10] "the/DT"          "board/NN"        "as/IN"
## [13] "a/DT"            "nonexecutive/JJ" "director/NN"
## [16] "Nov./NNP"        "29/CD"           "./."
## [19] "Mr./NNP"         "Vinken/NNP"      "is/VBZ"
## [22] "chairman/NN"     "of/IN"           "Elsevier/NNP"
## [25] "N.V./NNP"        ",/,"             "the/DT"
## [28] "Dutch/JJ"        "publishing/NN"   "group/NN"
## [31] "./."
```

```
## Extract pairs of word tokens and POS tags for second sentence:
a3ws2 <- annotations_in_spans(subset(a3, type == "word"),
subset(a3, type == "sentence")[2L])[[1L]]
sprintf("%s/%s", s[a3ws2], sapply(a3ws2$features, `[[`, "POS"))
```

```
##  [1] "Mr./NNP"       "Vinken/NNP"    "is/VBZ"        "chairman/NN"
##  [5] "of/IN"         "Elsevier/NNP"  "N.V./NNP"      ",/,"
##  [9] "the/DT"        "Dutch/JJ"      "publishing/NN" "group/NN"
## [13] "./."
```

Maxent_Sent_Token_Annotator

Apache OpenNLP based sentence token annotators

Generate an annotator which computes sentence annotations using the Apache OpenNLP Maxent sentence detector.

```
require("NLP")
## Some text.
s <- paste(c("Pierre Vinken, 61 years old, will join the board as a ",
"nonexecutive director Nov. 29.\n",
"Mr. Vinken is chairman of Elsevier N.V., ",
"the Dutch publishing group."),
collapse = "")
s <- as.String(s)
sent_token_annotator <- Maxent_Sent_Token_Annotator()
sent_token_annotator
```

```
## An annotator inheriting from classes
##   Simple_Sent_Token_Annotator Annotator
## with description
##   Computes sentence annotations using the Apache OpenNLP Maxent
##   sentence detector employing the default model for language 'en'.
```

```
a1 <- annotate(s, sent_token_annotator)
a1
```

```
##  id type     start end features
##   1 sentence     1  84
##   2 sentence    86 153
```

```
## Extract sentences.
s[a1]
```

```
## [1] "Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29."
## [2] "Mr. Vinken is chairman of Elsevier N.V., the Dutch publishing group."
```

```
## Variant with sentence probabilities as features.
annotate(s, Maxent_Sent_Token_Annotator(probs = TRUE))
```

```
##  id type     start end features
##   1 sentence     1  84 prob=0.9998197
##   2 sentence    86 153 prob=0.9968879
```

Maxent_Word_Token_Annotator

Apache OpenNLP based word token annotators

Generate an annotator which computes word token annotations using the Apache OpenNLP Maxent tokenizer

```
require("NLP")
## Some text.
s <- paste(c("Pierre Vinken, 61 years old, will join the board as a ",
"nonexecutive director Nov. 29.\n",
"Mr. Vinken is chairman of Elsevier N.V., ",
"the Dutch publishing group."),
collapse = "")
s <- as.String(s)
## Need sentence token annotations.
sent_token_annotator <- Maxent_Sent_Token_Annotator()
a1 <- annotate(s, sent_token_annotator)
word_token_annotator <- Maxent_Word_Token_Annotator()
word_token_annotator
```

```
## An annotator inheriting from classes
##   Simple_Word_Token_Annotator Annotator
## with description
##   Computes word token annotations using the Apache OpenNLP Maxent
##   tokenizer employing the default model for language 'en'.
```

```
a2 <- annotate(s, word_token_annotator, a1)
a2
```

```
##  id type     start end features
##   1 sentence     1  84 constituents=<<integer,18>>
##   2 sentence    86 153 constituents=<<integer,13>>
##   3 word         1   6
##   4 word         8  13
##   5 word        14  14
##   6 word        16  17
##   7 word        19  23
##   8 word        25  27
##   9 word        28  28
##  10 word        30  33
##  11 word        35  38
##  12 word        40  42
##  13 word        44  48
##  14 word        50  51
##  15 word        53  53
##  16 word        55  66
##  17 word        68  75
##  18 word        77  80
```

```
##  19 word        82  83
##  20 word        84  84
##  21 word        86  88
##  22 word        90  95
##  23 word        97  98
##  24 word       100 107
##  25 word       109 110
##  26 word       112 119
##  27 word       121 124
##  28 word       125 125
##  29 word       127 129
##  30 word       131 135
##  31 word       137 146
##  32 word       148 152
##  33 word       153 153
```

```
## Variant with word token probabilities as features.
head(annotate(s, Maxent_Word_Token_Annotator(probs = TRUE), a1))
```

```
##  id type     start end features
##   1 sentence     1  84 constituents=<<integer,18>>
##   2 sentence    86 153 constituents=<<integer,13>>
##   3 word         1   6 prob=1
##   4 word         8  13 prob=0.9770575
##   5 word        14  14 prob=1
##   6 word        16  17 prob=1
```

```
## Can also perform sentence and word token annotations in a pipeline:
a <- annotate(s, list(sent_token_annotator, word_token_annotator))
head(a)
```

```
##  id type     start end features
##   1 sentence     1  84 constituents=<<integer,18>>
##   2 sentence    86 153 constituents=<<integer,13>>
##   3 word         1   6
##   4 word         8  13
##   5 word        14  14
##   6 word        16  17
```

Parse_Annotator

Apache OpenNLP based parse annotator

Generate an annotator which computes Penn Treebank parse annotations using the Apache OpenNLP chunking parser for English.

```
## Requires package 'openNLPmodels.en' from the repository at
## <http://datacube.wu.ac.at>.
require("NLP")
## Some text.
s <- paste(c("Pierre Vinken, 61 years old, will join the board as a ",
"nonexecutive director Nov. 29.\n",
"Mr. Vinken is chairman of Elsevier N.V., ",
"the Dutch publishing group."),
collapse = "")
s <- as.String(s)
```

```
## Need sentence and word token annotations.
sent_token_annotator <- Maxent_Sent_Token_Annotator()
word_token_annotator <- Maxent_Word_Token_Annotator()
a2 <- annotate(s, list(sent_token_annotator, word_token_annotator))
parse_annotator <- Parse_Annotator()
## Compute the parse annotations only.
p <- parse_annotator(s, a2)
## Extract the formatted parse trees.
ptexts <- sapply(p$features, `[[`, "parse")
ptexts
```

```
## [1] "(TOP (S (NP (NP (NNP Pierre) (NNP Vinken))(, ,) (ADJP (NP (CD 61) (NNS years)) (JJ old)))(, ,)
## [2] "(TOP (S (NP (NNP Mr.) (NNP Vinken)) (VP (VBZ is) (NP (NP (NN chairman)) (PP (IN of) (NP (NP (NNI
```

```
## Read into NLP Tree objects.
ptrees <- lapply(ptexts, Tree_parse)
ptrees
```

```
## [[1]]
## (TOP
##   (S
##     (NP
##       (NP (NNP Pierre) (NNP Vinken))
##       (, ,)
##       (ADJP (NP (CD 61) (NNS years)) (JJ old)))
##     (, ,)
##     (VP
##       (MD will)
##       (VP
##         (VB join)
##         (NP (DT the) (NN board))
##         (PP
##           (IN as)
##           (NP
##             (NP (DT a) (JJ nonexecutive) (NN director))
##             (NP (NNP Nov.) (CD 29))))))
##     (. .)))
##
## [[2]]
## (TOP
##   (S
##     (NP (NNP Mr.) (NNP Vinken))
##     (VP
##       (VBZ is)
##       (NP
##         (NP (NN chairman))
##         (PP
##           (IN of)
##           (NP
##             (NP (NNP Elsevier) (NNP N.V.))
##             (, ,)
##             (NP (DT the) (JJ Dutch) (NN publishing) (NN group))))))
##     (. .)))
```

Appendix C: Naïve Bayes Codes

Source: https://aiaioo.wordpress.com/2016/01/13/naive-bayes-classifier-in-opennlp/

Training a Naive Bayes classifier is a lot like training a maximum entropy classifier. In fact, you still have to use the DocumentCategorizerME class to do it.

But you pass in a special parameter to tell the DocumentCategorizerME class that you want a Naive Bayes classifier instead.

Here is some code for training a classifier (from the OpenNLP manual) in this case, the Maximum Entropy classifier.

```
DoccatModel model = null;
InputStream dataIn = null;
try {
  dataIn = new FileInputStream("en-sentiment.train");
  ObjectStream<String> lineStream =
                    new PlainTextByLineStream(dataIn, "UTF-8");
  ObjectStream<DocumentSample> sampleStream = new DocumentSampleStream(lineStream);

  // Training a maxent model by default!!!
  model = DocumentCategorizerME.train("en", sampleStream);
}
catch (IOException e) {
  // Failed to read or parse training data, training failed
  e.printStackTrace();
}
```

Now, if you want to invoke the new Naive Bayes classifier instead, you just have to pass in a few training parameters, as follows.

```
DoccatModel model = null;
InputStream dataIn = null;
try {
  dataIn = new FileInputStream("en-sentiment.train");
  ObjectStream<String> lineStream =
                    new PlainTextByLineStream(dataIn, "UTF-8");
  ObjectStream<DocumentSample> sampleStream = new DocumentSampleStream(lineStream);

  TrainingParameters params = new TrainingParameters();
  params.put(TrainingParameters.CUTOFF_PARAM, Integer.toString(0));
  params.put(TrainingParameters.ALGORITHM_PARAM,
NaiveBayesTrainer.NAIVE_BAYES_VALUE);

  // Now the parameter TrainingParameters.ALGORITHM_PARAM ensures
  // that we train a Naive Bayes model instead
  model = DocumentCategorizerME.train("en", sampleStream, params);
}
catch (IOException e) {
  // Failed to read or parse training data, training failed
  e.printStackTrace();
}
```