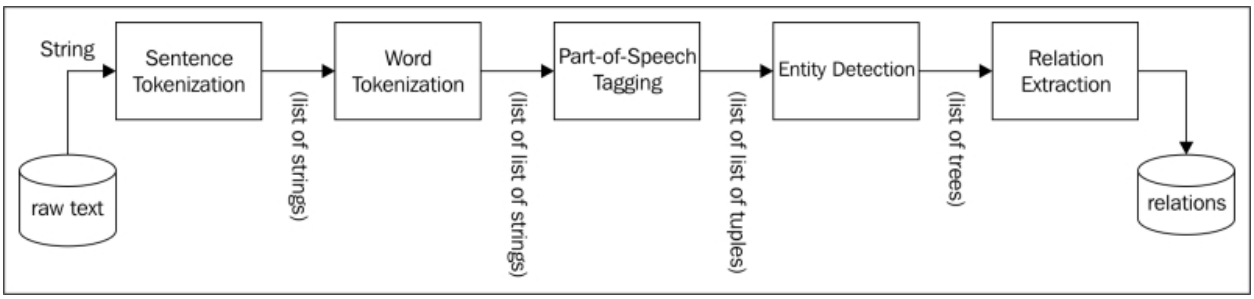
CS410: Tech Review
Topic: NLTK for Information Extraction
Subhasish Bose (sbose4@illinois.edu)

What is Information Extraction:

Most of the data that exists today (mostly on web) are unstructured and obtaining information from this unstructured data is difficult. Information extraction is the process of extracting structured information from unstructured text data. Information Extraction has many applications, including business intelligence, sentiment detection, email scanning etc.

Information extraction pipeline:

A typical information extraction pipeline looks like to the following figure:



What is NLTK:

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).
It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

NLTK for Information Extraction:

To perform the first three tasks of Information extraction pipeline (which is commonly known as preprocessing), NLTK can be used very easily.

Sample code:

```
>>> import nltk
>>> def preprocess(doc):
... sentence = nltk.sent_tokenize(doc) # Sentence Tokenization
... sentence = [nltk.word_tokenize(s) for s in sentence] # Word Tokenization
... sentence = [nltk.pos_tag(s) for s in sentence] #Part-of-Speech Tagging
```

Sample output:

```
>>> preprocess("Stoicism is a school of Hellenistic philosophy founded by Zeno of Citium
in Athens in the early 3rd century BC. Stoicism is a philosophy of personal ethics
informed by its system of logic and its views on the natural world.")
```

```
#sentence tokenizing
['Stoicism is a school of Hellenistic philosophy founded by Zeno of Citium in Athens in
the early 3rd century BC.', 'Stoicism is a philosophy of personal ethics informed by its
system of logic and its views on the natural world.']

#word tokenizing
[['Stoicism', 'is', 'a', 'school', 'of', 'Hellenistic', 'philosophy', 'founded', 'by',
'Zeno', 'of', 'Citium', 'in', 'Athens', 'in', 'the', 'early', '3rd', 'century', 'BC',
'.'], ['Stoicism', 'is', 'a', 'philosophy', 'of', 'personal', 'ethics', 'informed', 'by',
'its', 'system', 'of', 'logic', 'and', 'its', 'views', 'on', 'the', 'natural', 'world',
'.']]

#pos tagging
[[('Stoicism', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('school', 'NN'), ('of', 'IN'),
('Hellenistic', 'NNP'), ('philosophy', 'NN'), ('founded', 'VBN'), ('by', 'IN'), ('Zeno',
'NNP'), ('of', 'IN'), ('Citium', 'NNP'), ('in', 'IN'), ('Athens', 'NNP'), ('in', 'IN'),
('the', 'DT'), ('early', 'JJ'), ('3rd', 'JJ'), ('century', 'NN'), ('BC', 'NNP'), ('.',
'.')], [('Stoicism', 'NN'), ('is', 'VBZ'), ('a', 'DT'), ('philosophy', 'NN'), ('of',
'IN'), ('personal', 'JJ'), ('ethics', 'NNS'), ('informed', 'VBN'), ('by', 'IN'), ('its',
'PRP$'), ('system', 'NN'), ('of', 'IN'), ('logic', 'NN'), ('and', 'CC'), ('its', 'PRP$'),
('views', 'NNS'), ('on', 'IN'), ('the', 'DT'), ('natural', 'JJ'), ('world', 'NN'), ('.',
'.')]]
```

Next comes entity detection. The basic technique for entity detection is **Chunking**.
Chunking works on top of POS tagging and it chunks together set of tokens like Verb phrase or Noun. Chunker segments and labels multi-token sequences as one group. Each of these multi-token sequences are called a chunk. Each chunk has a particular grammatical function.

Sample Noun Phrase Chunker code:

```
import nltk
gram = ("NP: {<DT>?<JJ>*<NN>}")
sent = "last night i saw a black dog barking at a kid"
>>> chunking = nltk.RegexpParser(gram)
>>> sent_token = nltk.word_tokenize(sent)
>>> tagging = nltk.pos_tag(sent_token)
>>> tree = chunking.parse(tagging)>>> treeTree('S', [Tree('NP', [('last', 'JJ'), ('night',
'NN')]), Tree('NP', [('i', 'NN')]), ('saw', 'VBD'), Tree('NP', [('a', 'DT'), ('black',
'JJ'), ('dog', 'NN')]), Tree('NP', [('barking', 'NN')]), ('at', 'IN'), Tree('NP', [('a',
'DT'), ('kid', 'NN')])])
```

There is another technique – **Chinking** which is somewhat similar to chunking but instead of taking chunks as a whole, a chunk gets defined which we want to remove. This piece of chunk is called a chink and the process is known as chinking.

Sample Chinking Code:

```
gram = r"""
... NP:
... {<.*>+} #this line chunks everything
... }<DT|NN>+{ #it chinks sequence of DT and NN
... """
sent = "last night i saw a black dog barking at a kid"
>>> sent_token = nltk.word_tokenize(sent)
>>> tagging = nltk.pos_tag(sent_token)
>>> tree = chunking.parse(tagging)
>>> treeTree('S', [Tree('NP', [('last', 'JJ')]), ('night', 'NN'), ('i', 'NN'), Tree('NP',
```

```
[('saw', 'VBD')]), ('a', 'DT'), Tree('NP', [('black', 'JJ')]), ('dog', 'NN'), ('barking',
'NN'), Tree('NP', [('at', 'IN')]), ('a', 'DT'), ('kid', 'NN')])
```

Chunks can be presented/seen both using tags and trees. However, the **IOB** tags are most common representation: I (Inside), O (Outside) and B (Begin).

Once entities have been identified in a text, next step is to extract the relations that exist between them.

Sample Code:

```
>>> IN = re.compile(r'.*\bin\b(?!\b.+ing)')
>>> for doc in nltk.corpus.ieer.parsed_docs('NYT_19980315'):
...     for rel in nltk.sem.extract_rels('ORG', 'LOC', doc,
...                                      corpus='ieer', pattern = IN):
...         print(nltk.sem.rtuple(rel))
[ORG: 'WHYY'] 'in' [LOC: 'Philadelphia']
[ORG: 'McGlashan &AMP; Sarrail'] 'firm in' [LOC: 'San Mateo']
[ORG: 'Freedom Forum'] 'in' [LOC: 'Arlington']
[ORG: 'Brookings Institution'] ', the research group in' [LOC: 'Washington']
[ORG: 'Idealab'] ', a self-described business incubator based in' [LOC: 'Los Angeles']
[ORG: 'Open Text'] ', based in' [LOC: 'Waterloo']
[ORG: 'WGBH'] 'in' [LOC: 'Boston']
[ORG: 'Bastille Opera'] 'in' [LOC: 'Paris']
[ORG: 'Omnicom'] 'in' [LOC: 'New York']
[ORG: 'DDB Needham'] 'in' [LOC: 'New York']
[ORG: 'Kaplan Thaler Group'] 'in' [LOC: 'New York']
[ORG: 'BBDO South'] 'in' [LOC: 'Atlanta']
[ORG: 'Georgia-Pacific'] 'in' [LOC: 'Atlanta']
```

References:

https://www.nltk.org/book/ch07.html#ref-ie-tokenize

http://pars.ie/content/02-publications/2-teaching/2-text_mining/_lecture_11_chapter_7_information_extraction_part_1.pdf

https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781784396909/4/ch04lvl1sec34/information-extraction

https://ilmoirfan.com/information-extraction-from-text-in-python-using-nltk/

https://medium.com/@jeffysam02/chunking-and-extracting-information-using-nltk-part-6-5ecceeb4aac4

https://www.techopedia.com/definition/30343/natural-language-toolkit-nltk