# Part-Of-Speech Tagging


## Tutor: Rahmad Mahendra

rahmad.mahendra@cs.ui.ac.id


**Natural Language Processing & Text Mining**

Short Course

Pusat Ilmu Komputer UI

22 – 26 Agustus 2016

pusilkom ui
quality results

# HANDS ON POS TAGGER

# MaxEnt Tagger

- Use NLTK‟s “currently recommended” tokenizers and tagger
  - Maxent Treebank POS tagger trained on Penn
  - Tokenize the document by sentence, then by word; then tag

```
document = '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.'

sentences = nltk.sent_tokenize(document)
for sent in sentences:
     print nltk.pos_tag(nltk.word_tokenize(sent))
```

```
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61',
'CD'), ('years', 'NNS'), ('old', 'JJ'), (',', ','), ('will',
'MD'), ('join', 'VB'), ('the', 'DT'), ('board', 'NN‟)…
```

# NLTK Tagger Class

- Default Tagger
- RegexpTagger
- N-gram taggers
- HMM tagger

# Finding the Tagset

```
>>> from nltk.corpus import brown

>>> brown_news_tagged =
brown.tagged_words(categories='news',
simplify_tags=True)

>>> tag_fd = nltk.FreqDist(tag for (word, tag) in
brown_news_tagged)

>>> tag_fd.keys()

>>> nltk.help.upenn_tagset(‘NN’)
```

# Finding the Verb

```
>>> wsj =
nltk.corpus.treebank.tagged_words(simplify_tags=True)

>>> word_tag_fd = nltk.FreqDist(wsj)

>>> [word + "/" + tag for (word, tag) in word_tag_fd if
tag.startswith('V')]
```

# Default Tagger

- Assigns the same tag to all words
- (Good baseline)

```
tags = [tag for (word, tag) in
brown.tagged_words(categories='news')]
nltk.FreqDist(tags).max()

tagger = nltk.DefaultTagger('NN')
print tagger.tag(sentence)
```

# Regexp Tagger

- Assigns tags based on regular expressions (e.g. morphological structure)
- Processed in a order, the first one that matches is applied

```
patterns =
[(r'^-?[0-9]+(.[0-9]+)?$', 'CD'),     # cardinal numbers
(r'.*able$', 'JJ'),                   # adjectives
(r'.*ness$', 'NN'),                   # nouns formed from
adjectives
(r'.*ly$', 'RB'),                     # adverbs
(r'.*ing$', 'VBG'),                   # gerunds
(r'.*ed$', 'VBD'),                    # past tense verbs
(r'^[A-Z].*s$', 'NNPS'),              # plural proper nouns
(r'.*s$', 'NNS'),                     # plural nouns
(r'^[A-Z].*$', 'NNP'),                # singular proper nouns
(r'.*', 'NN')]                        # singular nouns (default)

tagger = nltk.RegexpTagger(patterns)
print tagger.tag(sentence)
```

# N-Gram Tagger

- Take into account grammatical context of words
- Finds the most likely tag for each word, given the preceding tag
- If we see in training:
  - "They are content"
    - Are/VBP content/JJ
  - The most important part is content "
    - "Is/VBP content/NN
- From the data, "content" would be more likely to be an adjective when it follows a verb
- Then in test data:
  - When we see "He is content" with "is/VBP"
  - Tag "content" as "content/JJ"

# Dataset Partition

```
>>> size = int(len(brown_tagged_sents) * 0.9)

>>> size

>>> size2 = int(len(brown_tagged_sents) * 0.95)

>>> train_sents = brown_tagged_sents[:size]

>>> test_sents = brown_tagged_sents[size:size2]

>>> unigram_tagger = nltk.UnigramTagger(train_sents)

>>> unigram_tagger.evaluate(test_sents)
```

# N-Gram Taggers in NLTK

- General N-gram:
  - `tagger = nltk.NgramTagger(2, training, backoff=backoffTagger)`
- N=2, same as:
  - `tagger = nltk.BigramTagger(training, backoff=backoffTagger)`
- Bigrams, even more so trigrams and up, risk sparse data
  - So they need good backoff taggers

# Combining Tagger

```
>>> t0 = nltk.DefaultTagger('NN')

>>> t1 = nltk.UnigramTagger(train_sents, backoff=t0)

>>> t2 = nltk.BigramTagger(train_sents, backoff=t1)

>>> t2.evaluate(test_sents)
```

# Thank you