# EDAN95
# Applied Machine Learning
# http://cs.lth.se/edan95/
## Lecture 8: Word and Segment Categorization

Pierre Nugues

Lund University
Pierre.Nugues@cs.lth.se
http://cs.lth.se/pierre_nugues/

November 26, 2018

# Motivation

The analysis of sentences always involves the analysis of words.
We can divide it in three main tasks:

1. Identify the type of word, for instance noun or verb using the classical grammar;
2. Identify a group or segment, for instance are there three words the name of a person;
3. Identify the relations between two words: for instance is this group the subject of a verb? This corresponds to parsing, semantic analysis, or information extraction.

We will consider the two first tasks.

# Word Categorization: The Parts of Speech

Sentence:

*That round table might collapse*

Annotation:

| Words | Parts of speech | POS tags |
|---|---|---|
| ***that*** | Determiner | DT |
| ***round*** | Adjective | JJ |
| ***table*** | Noun | NN |
| ***might*** | Modal verb | MD |
| ***collapse*** | Verb | VB |

The automatic annotation uses predefined POS tagsets such as the Penn Treebank tagset for English

# Ambiguity

| Words | Possible tags | Example of use |
|---|---|---|
| ***that*** | Subordinating conjunction | *That he can swim is good* |
| | Determiner | *That white table* |
| | Adverb | *It is not that easy* |
| | Pronoun | *That is the table* |
| | Relative pronoun | *The table that collapsed* |
| ***round*** | Verb | *Round up the usual suspects* |
| | Preposition | *Turn round the corner* |
| | Noun | *A big round* |
| | Adjective | *A round box* |
| | Adverb | *He went round* |
| ***table*** | Noun | *That white table* |
| | Verb | *I table that* |
| ***might*** | Noun | *The might of the wind* |
| | Modal verb | *She might come* |
| ***collapse*** | Noun | *The collapse of the empire* |
| | Verb | *The empire can collapse* |

# Segment Recognition

## Group detection – chunking –:

Brackets: [$_{NG}$ The government $_{NG}$] has [$_{NG}$ other agencies and instruments $_{NG}$] for pursuing [$_{NG}$ these other objectives $_{NG}$] .

Tags: *The*/`I` *government*/`I` *has*/`O` *other*/`I` *agencies*/`I` *and*/`I` *instruments*/`I` *for*/`O` *pursuing*/`O` *these*/`I` *other*/`I` *objectives*/`I` *.*/`O`

Brackets: Even [$_{NG}$ Mao Tse-tung $_{NG}$] [$_{NG}$ 's China $_{NG}$] began in [$_{NG}$ 1949 $_{NG}$] with [$_{NG}$ a partnership $_{NG}$] between [$_{NG}$ the communists $_{NG}$] and [$_{NG}$ a number $_{NG}$] of [$_{NG}$ smaller, non-communists parties $_{NG}$] .

Tags: *Even*/`O` *Mao*/`I` *Tse-tung*/`I` *'s*/`B` *China*/`I` *began*/`O` *in*/`O` *1949*/`I` *with*/`O` *a*/`I` *partnership*/`I` *between*/`O` *the*/`I` *communists*/`I` *and*/`O` *a*/`I` *number*/`I` *of*/`O` *smaller*/`I` *,*/`I` *non-communists*/`I` *parties*/`I` *.*/`O`

# Segment Categorization

Tages extendible to any type of chunks: nominal, verbal, etc.
For the IOB scheme, this means tags such as I.Type, O.Type, and B.Type, Types being NG, VG, PG, etc.
In CoNLL 2000, ten types of chunks

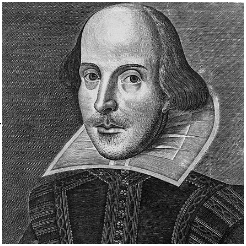
| Word | POS | Group | Word | POS | Group |
|---|---|---|---|---|---|
| *He* | PRP | B-NP | *to* | TO | B-PP |
| *reckons* | VBZ | B-VP | *only* | RB | B-NP |
| *the* | DT | B-NP | *£* | # | I-NP |
| *current* | JJ | I-NP | *1.8* | CD | I-NP |
| *account* | NN | I-NP | *billion* | CD | I-NP |
| *deficit* | NN | I-NP | *in* | IN | B-PP |
| *will* | MD | B-VP | *September* | NNP | B-NP |
| *narrow* | VB | I-VP | *.* | . | O |

Noun groups (NP) are in red and verb groups (VP) are in blue.

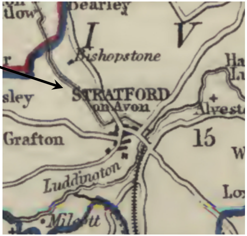
# IOB Annotation for Named Entities

| CoNLL 2002 | | CoNLL 2003 | | | |
|---|---|---|---|---|---|
| Words | Named entities | Words | POS | Groups | Named entities |
| Wolff | B-PER | U.N. | NNP | I-NP | I-ORG |
| , | O | official | NN | I-NP | O |
| currently | O | Ekeus | NNP | I-NP | I-PER |
| a | O | heads | VBZ | I-VP | O |
| journalist | O | for | IN | I-PP | O |
| in | O | Baghdad | NNP | I-NP | I-LOC |
| Argentina | B-LOC | . | . | O | O |
| , | O | | | | |
| played | O | | | | |
| with | O | | | | |
| Del | B-PER | | | | |
| Bosque | I-PER | | | | |
| in | O | | | | |
| the | O | | | | |
| final | O | | | | |
| years | O | | | | |
| of | O | | | | |
| the | O | | | | |
| seventies | O | | | | |
| in | O | | | | |
| Real | B-ORG | | | | |
| Madrid | I-ORG | | | | |
| . | O | | | | |

# Named Entities: Proper Nouns



William Shakespeare was born and brought up in Stratford-upon-Avon



# Others Entities: Common Nouns

Meeting with our guest on the landing at lunchtime

☺

# Evaluation

There are different kinds of measures to evaluate the performance of machine learning techniques, for instance:

- Precision and recall in information retrieval and natural language processing;
- The *receiver operating characteristic* (ROC) in medicine.

| | **Positive examples: $P$** | **Negative examples: $N$** |
|---|---|---|
| Classified as $P$ | True positives: $A$ | False positives: $B$ |
| Classified as $N$ | False negatives: $C$ | True negatives: $D$ |

More on the receiver operating characteristic here: `http://en.wikipedia.org/wiki/Receiver_operating_characteristic`

# Recall, Precision, and the F-Measure

The **accuracy** is $\dfrac{|A \cup D|}{|P \cup N|}$.

**Recall** measures how much relevant examples the system has classified correctly, for $P$:

$$\text{Recall} = \frac{|A|}{|A \cup C|}.$$

**Precision** is the accuracy of what has been returned, for $P$:

$$\text{Precision} = \frac{|A|}{|A \cup B|}.$$

Recall and precision are combined into the **F-measure**, which is defined as the harmonic mean of both numbers:

$$F = \frac{2 \cdot \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

# Evaluation

Accuracy, precision, and recall.
For noun groups with the predicted output:

| Word | POS | Group | Word | POS | Group |
|---|---|---|---|---|---|
| *He* | PRP | B-NP | *to* | TO | B-PP |
| *reckons* | VBZ | B-VP | *only* | RB | B-NP |
| *the* | DT | B-NP | *£* | # | I-NP |
| *current* | JJ | **B-NP** | *1.8* | CD | **B-NP** |
| *account* | NN | I-NP | *billion* | CD | I-NP |
| *deficit* | NN | I-NP | *in* | IN | B-PP |
| *will* | MD | B-VP | *September* | NNP | B-NP |
| *narrow* | VB | I-VP | . | . | O |

Accuracy $= \frac{14}{16}$, recall $= \frac{2}{4} = 0.5$, precision $= \frac{2}{6} = 0.33$
harmonic mean $= 2 \times \frac{0.33 \times 0.5}{0.33 + 0.5} = 0.4$

# Training Sets: The CoNLL Format

The CoNLL format is a tabular format to distribute annotated texts.

This format was created for evaluations carried out by the Conference in natural language learning

Annotation of the Spanish sentence:

> *La reestructuración de los otros bancos checos se está acompañando por la reducción del personal*
> *'The restructuring of Czech banks is accompanied by the reduction of personnel'*

in CoNLL 2006.

# Example of Annotation

| ID | FORM | LEMMA | CPOS | POS | FEATS |
|---|---|---|---|---|---|
| 1 | La | el | d | da | num=s\|gen=f |
| 2 | reestructuración | reestructuración | n | nc | num=s\|gen=f |
| 3 | de | de | s | sp | for=s |
| 4 | los | el | d | da | gen=m\|num=p |
| 5 | otros | otro | d | di | gen=m\|num=p |
| 6 | bancos | banco | n | nc | gen=m\|num=p |
| 7 | checos | checo | a | aq | gen=m\|num=p |
| 8 | se | se | p | p0 | _ |
| 9 | está | estar | v | vm | num=s\|per=3\|mod=i\|tmp=p |
| 10 | acompañando | acompañar | v | vm | mod=g |
| 11 | por | por | s | sp | for=s |
| 12 | la | el | d | da | num=s\|gen=f |
| 13 | reducción | reducción | n | nc | num=s\|gen=f |
| 14 | del | del | s | sp | gen=m\|num=s\|for=c |
| 15 | personal | personal | n | nc | gen=m\|num=s |
| 16 | . | . | F | Fp | _ |

# Another Example

| ID | FORM | LEMMA | PLEMMA | POS | PPOS | FEAT | PFEAT |
|---|---|---|---|---|---|---|---|
| 1 | Battle | battle | battle | NN | NN | _ | _ |
| 2 | - | - | - | HYPH | HYPH | _ | _ |
| 3 | tested | tested | tested | NN | NN | _ | _ |
| 4 | Japanese | japanese | japanese | JJ | JJ | _ | _ |
| 5 | industrial | industrial | industrial | JJ | JJ | _ | _ |
| 6 | managers | manager | manager | NNS | NNS | _ | _ |
| 7 | here | here | here | RB | RB | _ | _ |
| 8 | always | always | always | RB | RB | _ | _ |
| 9 | buck | buck | buck | VBP | VB | _ | _ |
| 10 | up | up | up | RP | RP | _ | _ |
| 11 | nervous | nervous | nervous | JJ | JJ | _ | _ |
| 12 | newcomers | newcomer | newcomer | NNS | NNS | _ | _ |
| 13 | with | with | with | IN | IN | _ | _ |
| 14 | the | the | the | DT | DT | _ | _ |
| 15 | tale | tale | tale | NN | NN | _ | _ |
| 16 | of | of | of | IN | IN | _ | _ |
| 17 | the | the | the | DT | DT | _ | _ |
| 18 | first | first | first | JJ | JJ | _ | _ |
| 19 | of | of | of | IN | IN | _ | _ |
| 20 | their | their | their | PRP$ | PRP$ | _ | _ |
| 21 | countrymen | countryman | countryman | NNS | NNS | _ | _ |
| 22 | to | to | to | TO | TO | _ | _ |
| 23 | visit | visit | visit | VB | VB | _ | _ |
| 24 | Mexico | mexico | mexico | NNP | NNP | _ | _ |
| 25 | , | , | , | , | , | _ | _ |
| 26 | a | a | a | DT | DT | _ | _ |
| 27 | boatload | boatload | boatload | NN | NN | _ | _ |
| 28 | of | of | of | IN | IN | _ | _ |
| 29 | samurai | samurai | samurai | NN | NN | _ | _ |
| 30 | warriors | warrior | warrior | NNS | NNS | _ | _ |

# Features for Part-of-Speech Tagging

The word visit is ambiguous in English:

> *I paid a **visit** to a friend –> noun*
> *I went to **visit** a friend –> verb*

The context of the word enables us to tell, here an article or the infinitive marker

To train and apply the model, the tagger extracts a set of features from the surrounding words, typically a sliding window spanning five words and centered on the current word.

We then associate the feature vector $(w_{i-2}, w_{i-1}, w_i, w_{i+1}, w_{i+2})$ with the part-of-speech tag $t_i$ at index $i$.

# Part-of-Speech Tagging

| ID | FORM | PPOS | |
|---|---|---|---|
| | BOS | BOS | ***Padding*** |
| | BOS | BOS | |
| 1 | Battle | NN | |
| 2 | - | HYPH | |
| 3 | tested | NN | |
| ... | ... | ... | |
| 17 | the | DT | |
| 18 | first | JJ | |
| 19 | of | IN | |
| 20 | their | PRP$ | |
| 21 | countrymen | NNS | ***Input features*** |
| 22 | to | TO | |
| 23 | **visit** | VB | ***Predicted tag*** |
| 24 | Mexico | | ↓ |
| 25 | , | | |
| 26 | a | | |
| 27 | boatload | | |
| ... | ... | ... | |
| 34 | years | | |
| 35 | ago | | |
| 36 | . | | |
| | EOS | | ***Padding*** |
| | EOS | | |

# Feature Vectors

| ID | Feature vectors | | | | | | | PPOS |
|---|---|---|---|---|---|---|---|---|
| | $w_{i-2}$ | $w_{i-1}$ | $w_i$ | $w_{i+1}$ | $w_{i+2}$ | $t_{i-2}$ | $t_{i-1}$ | |
| 1 | BOS | BOS | Battle | - | tested | BOS | BOS | NN |
| 2 | BOS | Battle | - | tested | Japanese | BOS | NN | HYPH |
| 3 | Battle | - | tested | Japanese | industrial | NN | HYPH | JJ |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 19 | the | first | of | their | countrymen | DT | JJ | IN |
| 20 | first | of | their | countrymen | to | JJ | IN | PRP$ |
| 21 | of | their | countrymen | to | visit | IN | PRP$ | NNS |
| 22 | their | countrymen | to | visit | Mexico | PRP$ | NNS | TO |
| 23 | countrymen | to | visit | Mexico | , | NNS | TO | VB |
| 24 | to | visit | Mexico | , | a | TO | VB | NNP |
| 25 | visit | Mexico | , | a | boatload | VB | NNP | , |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 34 | ashore | 375 | years | ago | . | RB | CD | NNS |
| 35 | 375 | years | ago | . | EOS | CD | NNS | RB |
| 36 | years | ago | . | EOS | EOS | NNS | RB | . |

# The Neural Network

We first use a feed-forward architecture corresponding to a logistic regression:

```
np.random.seed(0)

model = models.Sequential()
model.add(layers.Dense(NB_CLASSES,
                       input_dim=X.shape[1],
                       activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=OPTIMIZER,
              metrics=['accuracy'])
model.summary()
model.fit(X, y, epochs=EPOCHS, batch_size=BATCH_SIZE)
model.save('out.model')
```

# Preprocessing

Preprocessing is more complex though: Four steps:

1. Read the corpus

```
train_sentences, dev_sentences, test_sentences, column_name
```

2. Store the rows in dictionaries

```
conll_dict = CoNLLDictorizer(column_names, col_sep='\t')
train_dict = conll_dict.transform(train_sentences)
test_dict = conll_dict.transform(test_sentences)
```

3. Extract the features and store them in dictionaries

```
context_dictorizer = ContextDictorizer()
context_dictorizer.fit(train_dict)
X_dict, y_cat = context_dictorizer.transform(train_dict)
```

4. Vectorize the symbols

```
# We transform the X symbols into numbers
dict_vectorizer = DictVectorizer()
X_num = dict_vectorizer.fit_transform(X_dict)
```

# Code Example

Jupyter Notebook: `4.1-nn-pos-tagger.ipynb`

# Using Embeddings

We replace the one-hot vectors with embeddings: Dense vectors obtained by a principal component analysis or another method.

Here, we will use GloVe

We will add an `Embedding` layer and make it trainable

It would be possible to use a randomly initialized matrix

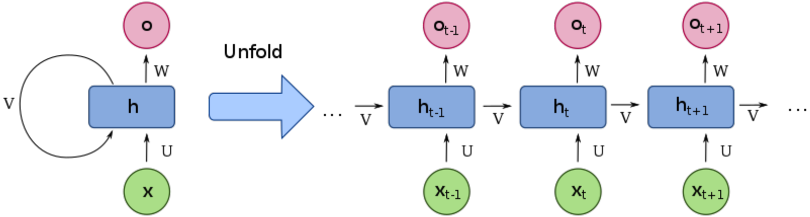
# The Embedding Layer

```
model = models.Sequential()
model.add(layers.Embedding(cnt_uniq, embedding_dim,
                                input_length=2 * w_size + 1))
if embedding_matrix is not None:
    model.layers[0].set_weights([embedding_matrix])
    model.layers[0].trainable = True
model.add(layers.Flatten())
model.add(layers.Dense(nb_classes, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',
                   optimizer=OPTIMIZER,
                   metrics=['accuracy'])
model.summary()
model.fit(X, y, epochs=EPOCHS, batch_size=BATCH_SIZE)
```

# Code Example

Jupyter Notebook: `4.2-nn-pos-tagger-embeddings.ipynb`

# The RNN Architecture




Credit Wikipedia

# Recurrent Neural Networks (RNN)

```
model = models.Sequential()
model.add(layers.Embedding(len(vocabulary_words) + 2,
                           EMBEDDING_DIM,
                           mask_zero=True,
                           input_length=MAX_SEQUENCE_LENGTH))
model.layers[0].set_weights([embedding_matrix])
# The default is True
model.layers[0].trainable = True
model.add(Bidirectional(SimpleRNN(NB_CLASSES + 1,
  return_sequences=True)))
model.add(Dense(lNB_CLASSES + 1, activation='softmax'))
```

# Long Short-Term Memory (LSTM)

```
model = models.Sequential()
model.add(layers.Embedding(len(vocabulary_words) + 2,
                           EMBEDDING_DIM,
                           mask_zero=True,
                           input_length=MAX_SEQUENCE_LENGTH))
model.layers[0].set_weights([embedding_matrix])
# The default is True
model.layers[0].trainable = True
model.add(layers.Bidirectional(layers.LSTM(NB_CLASSES + 1,
  return_sequences=True)))
#model.add(Bidirectional(LSTM(LSTM_UNITS,
  return_sequences=True)))
model.add(Dense(1NB_CLASSES + 1, activation='softmax'))
```

# ???

# ???

# ???

# ???