COMP348 — Document Processing and the Semantic Web

Week 05 Lecture 1: Sequence Labelling

Diego Mollá

Department of Computer Science
Macquarie University

COMP348 2017H1

Programme

Reading

- NLTK Book Chapter 5 "Categorizing and Tagging Words" http://www.nltk.org/book/ch05.html.
- NLTK Chapter 6 "Learning to Classify Text" http://www.nltk.org/book/ch06.html, especially sections 1.4

Programme

Programme

What is Sequence Labelling?

- A sequence labelling problem is one where:
  - the input consists of a sequence $\boldsymbol{X} = (X_1, \ldots, X_n)$, and
  - the output consists of a sequence $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ of labels, where:
  - Y_i is the label for element X_i
- Example: Part-of-speech tagging

$$\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{X} \end{pmatrix} = \begin{pmatrix} \text{Verb,} & \text{Determiner,} & \text{Noun} \\ \text{spread,} & \text{the,} & \text{butter} \end{pmatrix}$$

- Example: Spelling correction

$$\begin{pmatrix} \boldsymbol{Y} \\ \boldsymbol{X} \end{pmatrix} = \begin{pmatrix} \text{write,} & \text{a,} & \text{book} \\ \text{rite,} & \text{a,} & \text{buk} \end{pmatrix}$$

Other applications of sequence labelling

- Named entity recognition and classification (NER) involves finding the named entities in a text and identifying what type of entity they are (e.g., person, location, corporation, dates, etc.).
- Speech transcription can be seen as a sequence labelling task:
  - The input $\boldsymbol{X} = (X_1, \ldots, X_n)$ is a sequence of acoustic frames X_i, where X_i is a set of features extracted from a 50msec window of the speech signal.
  - The output $\boldsymbol{Y}$ is a sequence of words (the transcript of the speech signal).
- Financial applications of sequence labelling:
  - Identifying trends in price movements.
- Biological applications of sequence labelling:
  - Gene-finding in DNA or RNA sequences.

Programme

Sequence Labelling as Classification I

Can we just use a standard classifier?

- Standard classifiers (K-Nearest Neighbours, Naïve Bayes, Support Vector Machine, ...) assume independence between samples:
  - The probability of the label assigned to sample *i* is independent to the probability of the label assigned to sample *j*.
- But in sequence labelling there is interdependence between the labels of different samples.

Modelling Context

Classifier with context features

- A (crude) approach to model interdependence between samples is to add context features.
- For example, we can use features based on previous words and following words.
- We can even incorporate the label of the previous word as a feature.
    - We will see bigram taggers later in this lecture.
- But it is not so easy to incorporate the label of both the previous word and the following word.

Programme

General Principle

- Hidden Markov Models (HMMs) attempt to find the most likely labels given the input sequence using a probabilistic framework.
- In order to estimate the probabilities, HMMs use the Markov Property as a simplifying assumption.
- As it was the case with the Bag of Words approach, this simplifying assumption is not true but the approach can give surprisingly good results.

The Markov Property

The probability of assigning a tag to element i depends on the previous element only.

Why is Finding the Most Probable Label Sequence Hard

- When we use an HMM, we're given data items $\boldsymbol{x}$ and want to return the most probable label sequence.
- If $\boldsymbol{x}$ has n elements and each item has $m = |\mathcal{Y}|$ possible labels, the number of possible label sequences is m^n.
  - the number of possible label sequences grows exponentially with the length of the string.

  $\Longrightarrow$ exhaustive search for the optimal label sequence become impossible once n is large.
- But the Viterbi algorithm finds the most probable label sequence $\widehat{\boldsymbol{y}}(\boldsymbol{x})$ in $O(n)$ (linear) time using dynamic programming over a trellis.
  - the Viterbi algorithm is actually just the shortest path algorithm on a graph structure derived from the Markov assumption.

HMM in NLTK

http://nltk.org/api/nltk.tag.html

```
from nltk.corpus import brown
import nltk
import random
tagged_sents = list(brown.tagged_sents(tagset="universal"))
random.seed(1234)
random.shuffle(tagged_sents)
train = tagged_sents[0:1000]
test = tagged_sents[10000:10200]
untagged_test = [nltk.tag.untag(s) for s in test]
hmm = nltk.tag.HiddenMarkovModelTagger.train(train)
print(hmm.tag(untagged_test[0]))
hmm.test(test)
```

Programme

Diving In

```
>>> text=nltk.wordpunct_tokenize("And now for something completely different")
>>> text
['And', 'now', 'for', 'something', 'completely', 'different']
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
('completely', 'RB'), ('different', 'JJ')]
```

```
>>> text=nltk.word_tokenize("""They refuse to permit us to obtain the
refuse permit""")
>>> text
['They', 'refuse', 'to', 'permit', 'us', 'to', 'obtain', 'the', 'refuse', 'permit']
>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'),
('us', 'PRP'), ('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'),
('refuse', 'NN'), ('permit', 'NN')]
```

Diving In

```
>>> text=nltk.wordpunct_tokenize("And now for something completely different")
>>> text
['And', 'now', 'for', 'something', 'completely', 'different']
>>> nltk.pos_tag(text)
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
('completely', 'RB'), ('different', 'JJ')]
```

```
>>> text=nltk.word_tokenize("""They refuse to permit us to obtain the
refuse permit""")
>>> text
['They', 'refuse', 'to', 'permit', 'us', 'to', 'obtain', 'the', 'refuse', 'permit']
>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'),
('us', 'PRP'), ('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'),
('refuse', 'NN'), ('permit', 'NN')]
```

Programme

1. Sequence Labelling
   - What is Sequence Labelling?
   - Modelling Context
   - Hidden Markov Models
2. Introduction to PoS Tagging
   - English Word Classes
   - PoS Tagging
3. Automatic PoS Tagging
   - Rule-based Tagging
   - N-gram Tagging

Open Class Types

- Continuously updated (new words, borrowed words).
- Types:
  - Nouns
    - Count nouns can be counted "goat/goats", "idea/ideas".
    - Mass nouns cannot be counted "water", "furniture", "peace".
  - Verbs
    - Intransitive verbs do not take objects "John slept".
    - Transitive verbs take one object "John ate sandwiches".
    - Ditransitive verbs take two objects "Mary gave John the book".
  - Adjectives modify nouns: "green", "good".
  - Adverbs modify adjectives or verbs.
    - Locative: "home", "here", "downhill".
    - Degree: "extremely", "very", "somewhat".
    - Manner: "slowly", "delicately".
    - Temporal: "yesterday", "Monday".

Closed Class Types

- Closed class types (functional words): New additions are very rare.
- Types:
  - Prepositions relate a noun with a noun or a verb "on", "under", "over".
  - Determiners specify what the noun is referring to "a", "an", "the".
  - Pronouns refer to a noun or noun phrase "she", "who", "I", "others".
  - Conjunctions join sentences "and", "but", "or".
  - Auxiliary verbs modify verbs "can", "may", "are".
  - Particles for part of a phrasal verb ("eat up", "find out") "up", "down", "on", "off".
  - Numerals quantify the noun "one", "two", "third".

Programme

Part of Speech (PoS) Tagging

- PoS Tagging: Find the part of speech of a word.
  - "saw" can be either a noun or a verb.
- After PoS tagging, every word is given a tag from a predefined tag set.
  - Eg: Penn Treebank tags.
    The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.
- Different tagsets:
  - Penn Treebank (see below);
  - Brown corpus: http://khnt.hit.uib.no/icame/manuals/brown/INDEX.HTM;
  - ... put your favourite tagset here ...

The Penn Treebank Tagset

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NP	Proper noun, singular
NPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PP	Personal pronoun
PP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle

NLTK's "universal" tagset

Tag	Meaning	English Examples
ADJ	adjective	new, good, high, special, big, local
ADP	adposition	on, of, at, with, by, into, under
ADV	adverb	really, already, still, early, now
CONJ	conjunction	and, or, but, if, while, although
DET	determiner, article	the, a, some, most, every, no, which
NOUN	noun	year, home, costs, time, Africa
NUM	numeral	twenty-four, fourth, 1991, 14:24
PRT	particle	at, on, out, over per, that, up, with
PRON	pronoun	he, their, her, its, my, I, us
VERB	verb	is, say, told, given, playing, would
.	punctuation marks	. , ; !
X	other	ersatz, esprit, dunno, gr8, univeristy

Why PoS Tagging?

- Helps disambiguate in a variety of contexts.
  - Word Sense Disambiguation.
  - Machine Translation.
    - "With worldwide translation and documentation services" $\rightarrow$ "Avec traduction mondiale et la documentation entretient"
- PoS tagging is usually done before other NLP tasks (e.g. parsing).
- First stage building block for getting structure of language.

The Penn Treebank

Contains syntactic information and part of speech.

```
w = nltk.corpus.treebank.words()
>>> w[0:20]
['Pierre', 'Vinken', ',', '61', 'years', 'old', ',', 'will', 'join', 'the', 'board', 'as
>>> r=nltk.corpus.treebank.raw()
>>> r[0:100]
'\n( (S \n    (NP-SBJ \n      (NP (NNP Pierre) (NNP Vinken) )\n      (, ,) \n      (ADJP
>>> tw=nltk.corpus.treebank.tagged_words()
>>> tw[0:20]
[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('old
>>> ts=nltk.corpus.treebank.tagged_sents()
>>> ts[0:2]
[[('Pierre', 'NNP'), ('Vinken', 'NNP'), (',', ','), ('61', 'CD'), ('years', 'NNS'), ('ol
>>>
```

The Brown Corpus

- Words annotated with the part of speech.
- Several genres.

```
>>> from nltk.corpus import brown
>>> brown.raw()[0:100]
'\n\n\tThe/at Fulton/np-tl County/nn-tl Grand/jj-tl Jury/nn-tl said/vbd Friday/nr an/at
>>> brown.categories()
['adventure', 'belles_lettres', 'editorial', 'fiction', 'government',
'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion',
'reviews', 'romance', 'science_fiction']
>>> brown.words(categories='news')
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
>>> brown.words(fileids=['cg22'])
['Does', 'our', 'society', 'have', 'a', 'runaway', ',', ...]
>>> brown.sents(categories=['news','editorial','reviews'])
[['The', 'Fulton', 'County', ...], ['The', 'jury', 'further', ...], ...]
>>> brown.tagged_words()
[('The', 'AT'), ('Fulton', 'NP-TL'), ...]
>>> brown.tagged_sents()
[[('The', 'AT'), ('Fulton', 'NP-TL'), ('County', 'NN-TL'), ...], [('The', 'AT'), ('jury'
```

Statistics About Parts of Speech

Exercises

Write Python code that uses NLTK on the Brown corpus to find the answers to these questions:

1. What is the most frequent part of speech?
2. What are the most common verbs in news text?
3. What is the most frequent part of speech before a noun?

Programme

1. Sequence Labelling
   - What is Sequence Labelling?
   - Modelling Context
   - Hidden Markov Models
2. Introduction to PoS Tagging
   - English Word Classes
   - PoS Tagging
3. Automatic PoS Tagging
   - Rule-based Tagging
   - N-gram Tagging

Programme

Rule-Based Approach

- Two stages:
  1. For each word, build a list with its possible parts of speech.
  2. For each word, apply hand-crafted disambiguation rules.
- Types of rules:
  - Constraints: Eliminate impossible tags in the current context.
  - Preferences: Select the most plausible tag in the current context.

Baseline

- tag a word with its most frequent PoS tag.
- 90% accuracy!

NLTK's Default Tagger

Let's assign all words with the same tag: the most frequent tag in the corpus.

```
>>> import nltk
>>> dt = nltk.DefaultTagger('NN')
>>> text = "This is a test"
>>> dt.tag(nltk.word_tokenize(text))
[('This', 'NN'), ('is', 'NN'), ('a', 'NN'), ('test', 'NN
>>>
```

What's the accuracy of this tagger?

```
>>> from nltk.corpus import brown
>>> brown_tagged_sents = brown.tagged_sents(categories='news')
>>> brown_tagged_sents_train = brown_tagged_sents[:3000]
>>> brown_tagged_sents_test = brown_tagged_sents[3000:]
>>> dt.evaluate(brown_tagged_sents_test)
0.12598841741842076
```

NLTK's Default Tagger

Let's assign all words with the same tag: the most frequent tag in the corpus.

```
>>> import nltk
>>> dt = nltk.DefaultTagger('NN')
>>> text = "This is a test"
>>> dt.tag(nltk.word_tokenize(text))
[('This', 'NN'), ('is', 'NN'), ('a', 'NN'), ('test', 'NN
>>>
```

What's the accuracy of this tagger?

```
>>> from nltk.corpus import brown
>>> brown_tagged_sents = brown.tagged_sents(categories='news')
>>> brown_tagged_sents_train = brown_tagged_sents[:3000]
>>> brown_tagged_sents_test = brown_tagged_sents[3000:]
>>> dt.evaluate(brown_tagged_sents_test)
0.12598841741842076
```

NLTK's Regular Expression Tagger

```
>>> import nltk
>>> patterns = [
        (r'^-?[0-9]+(.[0-9]+)?$', 'CD'),
        (r'.*', 'NN')]
>>> num_n_tagger = nltk.RegexpTagger(patterns)
>>> text = "there are 5 words here"
>>> num_n_tagger.tag(nltk.word_tokenize(text))
[('there', 'NN'), ('are', 'NN'), ('5', 'CD'),
 ('words', 'NN'), ('here', 'NN')]
...
>>> num_n_tagger.evaluate(brown_tagged_sents_test)
0.1360118053235327
```

Taking Advantage of Suffix Information

- Suffixes are good clues to determine the part of speech.
- Can you find the most useful suffixes?

```
>>> patterns = [
        (r'.*ing$', 'VBG'),                    # gerunds
        (r'.*ed$', 'VBD'),                     # simple past
        (r'.*es$', 'VBZ'),                     # 3rd singular present
        (r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # cardinal numbers
        (r'.*', 'NN')                          # nouns (default)
    ]
>>> regexp_tagger = nltk.RegexpTagger(patterns)
>>> regexp_tagger.evaluate(brown_tagged_sents_test)
0.15898206927274752
```

Programme

NLTK's Lookup Tagger I

The lookup tagger uses a table (Python dictionary) of words and their recommended tag.

1. Find the n most frequent words (e.g. $n = 100$).
   - How...?
2. For each word, find its most frequent tag and build a dictionary 'likely_tags' with words as the keys and tags as the values.

```
>>> likely_tags['the']
>>> 'AT'
```

   - How...?

NLTK's Lookup Tagger II

3. Create the lookup tagger with this information.

```
>>> lookup_tagger = nltk.UnigramTagger(model=likely_tags)
>>> lookup_tagger.evaluate(brown_tagged_sents_test)
0.4577625570776256
```

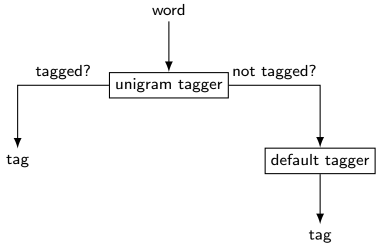
The value reported by the evaluation is the accuracy:

$$\text{accuracy} = \frac{\#\text{ correct tags}}{\text{total }\#\text{ of tokens}}$$

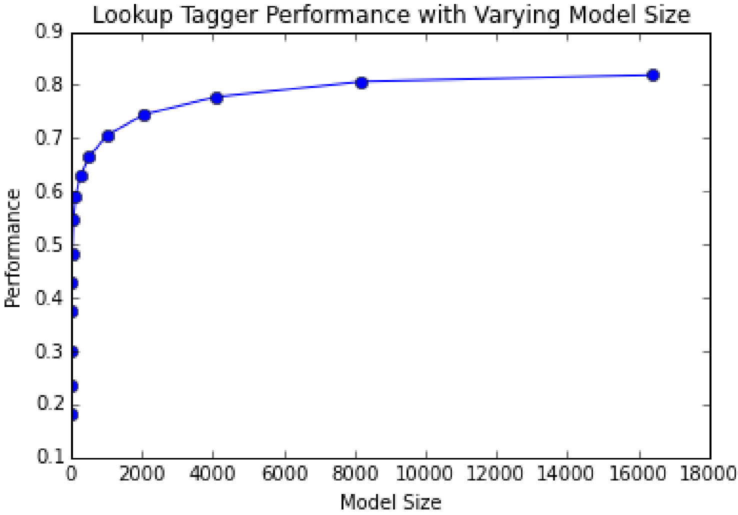
Backoff

- Words that are not in the model won't get a tag.
- Idea: Use a backoff tagger for untagged words.

```
>>> lookup_tagger = nltk.UnigramTagger(model=likely_tags,
                        backoff=nltk.DefaultTagger("NN"))
>>> lookup_tagger.evaluate(brown_tagged_sents)
0.5802427887292572
```



How Many Words Do We Need?



NLTK's Unigram Tagger

- Rather than manually select the list of words, train the system.
- The system gets an annotated corpus and it uses it to find the most likely tags.

Overfitting

- The system must be able to generalise and handle unknown data.
- Overfitting: A system that memorises the complete training corpus will do 100% with the corpus but poorly on new data.
- Always train with a corpus and test with a different corpus.

Training and Testing a Unigram Tagger

```
>>> unigram_tagger = nltk.UnigramTagger(brown_tagged_sents_tr
                                        backoff=nltk.DefaultTagge
>>> unigram_tagger.evaluate(brown_tagged_sents_test)
0.8186880498941975
>>> unigram_tagger.evaluate(brown_tagged_sents_train)
0.9362139917695473
```

We can see that there is overfitting.

Bigram Taggers

An N-gram tagger uses the tags of the previous N-1 words in addition to the current word.

This/DT is/VBZ an/AT example/NN

- 1-grams: unigrams: This is an example.
- 2-grams: bigrams: (None,This) (DT,is) (VBZ,an) (AT,example).
- 3-grams: trigrams: (None,None,This) (None,DT,is) (DT,VBZ,an) (VBZ,AT,example).

N-grams for PoS Tagging should not cross sentence boundaries.

Note about PoS Bigrams

Beware

Note that these bigrams are not the usual NLTK bigrams.

Exercise

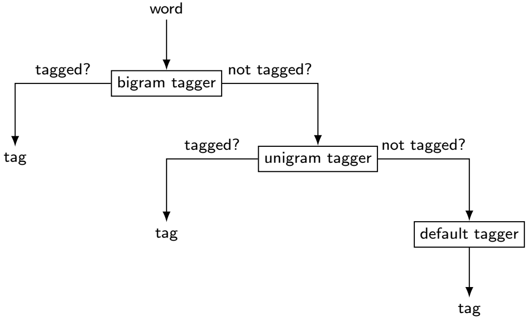
1. What are the differences?
2. How would you obtain the list of bigrams from the Brown corpus?

NLTK's Bigram Tagger

```
>>> bigram_tagger = nltk.BigramTagger(brown_tagged_sents_trai
>>> bigram_tagger.evaluate(brown_tagged_sents_test)
0.0876768014255485
```

The Problem of Data Sparseness

- The longer the N-gram, the more likely a new sentence will contain unknown N-grams.
- Data sparseness: There are not enough training data.
- Solution: Backoff to smaller N-grams.
    - Also called cascading of the taggers.



In Python/NLTK

```
>>> bigram_tagger = nltk.BigramTagger(brown_tagged_sents_trai
>>> bigram_tagger.evaluate(brown_tagged_sents_test)
0.0876768014255485
>>> bigram_tagger = nltk.BigramTagger(brown_tagged_sents_trai
                              backoff = nltk.DefaultTagger
>>> bigram_tagger.evaluate(brown_tagged_sents_test)
0.6936462857779263
>>> bigram_tagger = nltk.BigramTagger(brown_tagged_sents_trai
                              backoff = unigram_tagger)
>>> bigram_tagger.evaluate(brown_tagged_sents_test)
0.8243122842187326
```

Validating the Evaluation

- Sometimes we only get minor improvements; are these improvements significant?
- Perhaps the "improvement" merely reflects minor random variations in the corpus.

Cross-Validation

Also called N-fold cross-validation.

1. Partition the corpus in N folds (typically, $N = 5$ or $N = 10$).
2. Select 1 fold for testing and the rest for training.
3. Repeat the process selecting a different fold for testing and the rest for training.
4. We have now N evaluations; if the results are inconsistent, the improvements are artifacts of random variations in the corpus.

Take-home Messages

1. Explain what a sequence labelling model is, and explain how it can be used in Part-of-Speech tagging (and Named Entity Recognition next week).
2. Evaluate and discuss the results of PoS tagging.
3. Implement N-gram PoS tagging.
4. Explain the concepts of training, testing, cross-validation, data sparseness, overfitting, . . .

What's Next

Given that we have lost one full week of lectures during Easter, the following contents will be covered if there is time in week 6:

Week 6

- Information Extraction.
- Summarisation.