A Practical Introduction to Natural Language Processing

Intelligent Processing & Applications Research Cluster Seminar

Dr Lim Lian Tze

5 March 2015
Session 1: Common Tasks and Concepts in NLP

12 March 2015
Session 2: Software Libraries and Resources for NLP

Information Technology Department
School of Science, Engineering and Technology
KDU College Penang

Learning Outcomes

At the end of the seminar, participants will be able to:

- Explain examples of NLP applications and related technical issues.
- Explain the layers of NLP and corresponding processing tasks.
- Use existing libraries to perform common NLP processing tasks.
- Use wordnet-based semantic networks to provide multilingual semantic information in NLP applications.

Contents

NLP and Computational Linguistics

Natural Language = Human Language

- Computers communicating with humans in our own language – a scientific dream!
- Why is there so limited success?
- How is natural language different from computer language?

Natural Language = Human Language

- Dynamic, flexible, ambiguous, changes with time
    - ‘I’m going to the bank’ – what bank?
    - ‘I saw the girl with the telescope’
    - ‘To work we go’, ‘We go to work’, *‘we go work’
    - ‘nice’ means…?

Getting computers to understand natural language is *hard*!

Some issues...

- Humans – inputs often not well-formed (‘ungrammatical’, typos)
- Each language is different: grammar, vocabularies, etc
- SMS/social media talk?
- Special needs – legal, diplomatic, medical...
- Difficult to deal with *all* these concerns at the same time!
- Often customised for each domain or use case scenario

Research Areas

- Computational Linguistics (CL)
    - ‘concerning computational aspects of the human language faculty’
    - ‘statistical or rule-based modeling of natural language from a computational perspective’
    - Linguistics + Cognitive Science + Artificial Intelligence
- Natural Language Processing (NLP)
    - ‘Ability of computer programs to understand and generate human language utterances’ (written text or spoken speech)
    - Application of computational techniques to process natural language utterances
    - Computer Sciences + Artificial Intelligence + Human-Computer Interaction
- Human Language Technology (HLT) – catch-all, more general

How is NLP related to Big Data?

- Big data research: techniques for processing massive ammount of data (terabytes)
- Structured data
  - Databases, records
  - e.g. crime statistics, weather statistics, hospital records, sensor data...
- Unstructured data
  - Natural language corpora
  - e.g. news articles/recordings, interview transcripts, legal case documents, tweets...

Example Applications of NLP

Machine Translation (MT)

- Automatic translation of a text from a *source language* to a *target language* by a computer, preserving the meaning
- Some language pairs have good outputs; some not so good
- (Why?)
- Analyse input $\longrightarrow$ processing $\longrightarrow$ Synthesise output
- Need to ensure meaning is translated correctly
- Need to ensure output is grammatically correct
- 'Translating' by dictionary look up or just translating words individually is *not* MT

However…

Use cases of MT

Somers (2003, ch. 10) pointed out 3 use cases of MT.

- Disemmination
    - Translation output to be distributed for human as-is without changes
    - End users will have high expectations!
    - Output must be more or less perfect and well-formed
    - Hard – except for language pairs with huge amount of training data

Use cases of MT (cont.)

- Example Russian–English translation, suitable for dissemination:

  Russian: 18 февраля 2015 года Аналитическое управление аппарата Совета Федерации совместно с экономическим факультетом МГУ проводят научный семинар «Реалистическое моделирование».

  English: February 18, 2015 Analytical Department of the Federation Council in conjunction with the Faculty of Economics of Moscow State University conducted a scientific seminar "The realistic simulation."

Use cases of MT (cont.)

- Assimilation
    - Just to get a rough idea of the content
    - Output need not be perfect
    - But choice of words should reflect original meaning

Use cases of MT (cont.)

- Example Japanese–English translation, for assimilation:

  Japanese: 世界中の優秀な頭脳を魅了し、研究に集中できるようなサポート体制の整った環境とはどのようなものでしょうか。

  English: Attracts the brightest minds in the world, what What are the well-equipped environment support system, such as can concentrate on research.

Use cases of MT (cont.)

- Interchange
  - Translation in one-to-one communication (telephone or written correspondence).
  - Internet: tweets, blog posts, forums
  - Human translation is out of the question (too slow)!
  - *Any* output (even if poor) is better than *no* output

Some definitions

Utterance	An uninterrupted chain of spoken or written language
Source language	The original language of an utterance
Target language	The language the utterance to be translated to
Language pair	a SL–TL pair for an MT process, in that direction

Text and Corpus Processing

- Given a text or a corpus (a collection of documents)
- Identify the most frequently occurring words; most significant words; group of words ...
- Most frequently occuring: the, a, an...probably not so important!
- Most significant collocations (*n*-grams): finance, investment capital, tax returns...
  $\longrightarrow$ document is probably about Finance or Economy
- Useful for domain identification; document indexing for retrieval (search engine)

Information Extraction (IE)

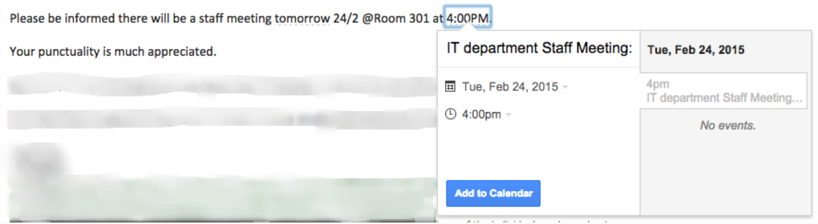
- Extract "interesting" facts to store in a knowledge base
- 'John stays in London. He works there for Polar Bear Design.'

Knowledge Base

$\text{John}_{\text{PER}} \xrightarrow{\text{live-in}} \text{London}_{\text{LOC}}$

$\text{John}_{\text{PER}} \xrightarrow{\text{employee-of}} \text{Polar Bear Design}_{\text{ORG}}$

Another IE Example (Easier?)



NLP applications are often easier to design and implement with a specific use case scenario in mind

Named Entity Recognition (NER)

- Identification of proper nouns in the text
- And classify them into catogeries of interest
- (Typically Person, Location, Organisation, Date, Currency...)
- ‘John$_{\text{PER}}$ stays in London$_{\text{LOC}}$. He works there for Polar Bear Design$_{\text{ORG}}$.’

Co-reference Resolution

- Tracking references to NEs
- John stays in London. He works there for Polar Bear Design.

Question Answering (QA)

- Need to compile, index, extract a knowledge base of facts (re IE)
- Need to analyse and interpret question to identify elements
- Need to search knowledge base
- May need to make inferences
- Need to present answers in a sensible manner

Q: 'Where is Polar Bear Design located?'

A: London

Knowledge Base

$\text{John}_{\text{PER}} \xrightarrow{\text{live-in}} \text{London}_{\text{LOC}}$

$\text{John}_{\text{PER}} \xrightarrow{\text{employee-of}} \text{Polar Bear Design}_{\text{ORG}}$

$\text{Polar Bear Design}_{\text{ORG}} \xrightarrow{\text{based-in}} \text{London}_{\text{LOC}}$

Plagiarism and Paraphrase Detection

- TurnItIn currently just detects plagiarism based on string matching
- What about paraphrasing? Also a form of plagiarism
- Check if several news reports are about the same event/issue
- (Li, McLean, Bandar, O'shea & Crockett, 2006; Pera & Ng, 2011)

Checking the Semantic Similarity

`http://swoogle.umbc.edu/StsService/GetStsSim`

- Inputs:
  - 'Many **dairy** farmers today use **machines** for **operations** from milking to **culturing** cheese.'
  - 'Today many **cow** farmers perform different **tasks** from milking to making **cheese** using **automated devices**.'
- Word order, word substitutions
- $>$ 70% similarity!

Sentiment Analysis & Opinion Mining

- Extract human judgement, evaluation, emotion, polarity from an utterance.
- Blogs, forum posts, tweets, speeches...
- Sentimen Classfication: `http://text-processing.com/demo/sentiment/`
  - 'This movie is overrated – all special effects, no heart.'

Polarity	pos: 0.4; neg: 0.6 (more negative than positive)
Subjectivity	neutral: 0.2; polar: 0.8 (more subjective than objective)

  - Negation: 'It's not bad.' ???
- More targeted:
  - 'The price is rather high, but the material is quite sturdy.'
  - `[price]` -ve; `[material]` +ve

US 2012 Presidential Election Campaign

- 'A system for real-time Twitter sentiment analysis of 2012 US presidential election cycle' (Wang, Can, Kazemzadeh, Bar & Narayanan, 2012)
- Twitter index tracks sentiment on Obama, Romney ▸ Link
- How Social Media Sentiment Impacts the Presidential Campaigns ▸ Link
- Tracking sentiments of a speech ▸ Link

Speech Recognition and Synthesis

- Speech recognition: speech-to-text (STT)
    - Accents, non-native speakers, pauses, filler noises...
- Speech synthesis: text-to-speech (TTS)
    - Easier? (bank teller systems etc)
    - How to simulate natural sounding speech?

Speech Recognition $\neq$ Voice Recognition

- Speech recognition
  - Given a speech sample, what was said? ‘Dubai’ or ‘Good bye’?
  - Involves language modelling (statistical model of valid sentences)
- Voice recognition
  - Given a speech sample, determine the identity of speaker
  - Involves signal processing, voice signatures

Two Approaches

- Signal processing $\longrightarrow$ identify phonemes (sound units)
- Language modelling $\longrightarrow$ likelihood of utterance
    - 'It's fun to recognize speech' or
    - 'It's fun to wreck a nice beach'

NLP Processing Layers

Linguistic Layers in NLP

Morphology	$\Longleftrightarrow$	word formation
Syntax	$\Longleftrightarrow$	sentence structure, grammar
Semantics	$\Longleftrightarrow$	meaning
Pragmatics	$\Longleftrightarrow$	discourse, context
Speech	$\Longleftrightarrow$	phonemes (speech units)

Examples here are for English – other languages may need different approaches

Morphology

Morphology

- How words are formed

  Inflection: plant $\longrightarrow$ plants, planted, planting ...

  Derivation: plant $\longrightarrow$ plantation, implant ...

- For Malay:

  Inflection: sakit $\longrightarrow$ sakitnya; pergi $\longrightarrow$ pergilah

  Derivation: sakit $\longrightarrow$ pesakit, penyakit, sakitan...

- Morphology processing: related to words

Tokenising

- Split input text into processable units
- Just by space characters…?
  - Passers-by | didn’t | go …
- Just by punctuation/word boundaries…?
  - Passers | - | by | didn | ’ | t | go …
- Tokenizers need to consider natural language!
  - Passers-by | did | n’t | go …

Sentence Splitting

- How to identify sentence boundaries?
- “That’s wonderful,’ he said. ‘Have your people call mine. Try to arrange something by 10 a.m. tomorrow.”

Stemming

- **Stem:** reduced form (word stem, base or root form) or a word
- Need not be identical to the morphological root of the word!
- As long as related words map to the same stem
- Usually implemented by stripping prefix/suffix

Stemming (cont.)

- Example stemming:
    - carresses $\rightarrow$ carress
    - ponies $\rightarrow$ poni
    - caress $\rightarrow$ caress
    - cats $\rightarrow$ cat
    - producer $\rightarrow$ produc
    - produced $\rightarrow$ produc
    - producing $\rightarrow$ produc
- Can have phases/sequences of rules (Paice, 1994; Porter, 1980)

Why Stemming?

- Information Retrieval – search for documents based on keywords
- Stem all words in documents and store as index
- Input keyword: producer $\rightarrow$ 'produc'
- Search documents whose indices contain 'produc'
- Results will include documents containing 'produce', 'produced', 'producer' …

Lemmatising

- **Lemma:** base form of a word or term that is used as the *formal dictionary entry* for the term.
- Lemmatising can be seen as a special form of stemming
  - Stemming: outputs do not need to be real words
  - Lemmatising: outputs are genuine words used as headwords in dictionaries

(1)

Input:	*banks*	*raised*	*rates*	*to*	*fight*	*inflation*
Lemmas:	bank	raise	rates	to	fight	inflation

Stemming vs Lemmatising

- Stemming is much faster than lemmatising
- But lemmatising is essential for many NLP tasks

Would lemmatising be required for these languages?

- Malay
- Chinese

Segmentation

- Languages without word boundaries, e.g. Chinese, Thai, Japanese, German...
- Essential for proper understanding!
- Chinese example: 有职称的和尚未有职称的

(2)	有	职称	的	和	尚未	有	职称	的
	with	position	ones	and	not yet	with	position	ones

(3)	有	职称	的	和尚	未有	职称	的
	with	position	ones	monks	without	position	ones

Technological readiness

- For English: libraries exists to perform these tasks
- For other languages: depends – some are still under research and development

Syntax

Syntax

- How words *form phrases and sentences*
- Grammatical rules and structures!
- Syntactic processing: extract structure of phrase/sentences

Part-of-Speech (POS)

- A category assigned to a word based on its grammatical and semantic properties.
- Example: noun, verb, adjective, adverb, determiner, preposition...
- Different languages may have different sets of POS e.g. classifier (penjodoh bilangan)

POS Tagset

- English: Penn Treebank (PTB) tagset is widely adopted (Marcus, Marcinkiewicz & Santorini, 1993)
- `https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

Tag	Description
NN	Noun, singular or mass
NNS	Noun, plural
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
...	...

POS-tagging

- Given an utterance, assign the most likely POS tag to each word token
- Current libraries quite stable now (for English): $\sim 96\%$ accuracy

(4)

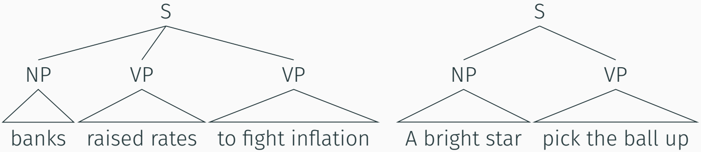
Input:	*banks*	*raised*	*rates*	*to*	*fight*	*inflation*
POS-tags:	NNS	VBD	NNS	TO	VB	NN

Phrase and Sentence Structure

- Sentences/clauses are made up of *phrases* following grammar (syntax) rules
- Some examples:
  - Noun phrase (NP): ‘a bright star’, ‘cats’, ‘stars and moons’
  - Verb phrase (VP): ‘ran’, ‘pick the ball up’
  - Clause/sentence (S): NP VP ‘a bright star pick the ball up’
- (A syntactically correct sentence doesn’t guarantee it makes sense!)

Shallow parsing (chunking)

- Identify the noun phrases, verb phrases etc but do not go into the internal structure

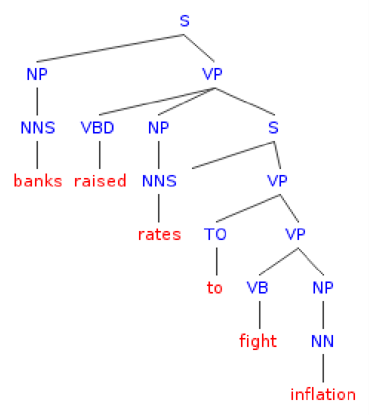


Parsing (deep parsing)

- Fully building the clauses and relations in a sentence
- Syntactic parse tree:

'Banks raised rates to fight inflation'

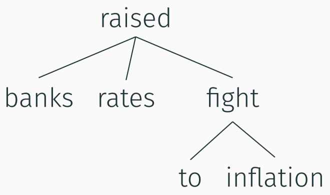
```
(S
  (NP (NNS banks))
  (VP (VBD raised)
    (NP (NNS rates))
    (S
      (VP (TO to)
        (VP (VB fight)
          (NP (NN inflation))))....
```



Dependency parsing

- Find dependency relations in the text

‘Banks raised rates to fight inflation’




```
nsubj(raised, banks)
root(ROOT, raised)
dobj(raised, rates)
aux(fight, to)
vmod(raised, fight)
dobj(fight, inflation)
```

- ‘banks’ is subject of ‘raised’
- ‘rates’ is object of ‘raised’
- …

Technological Readiness

- Parsing is more difficult than POS-tagging
- But largely solved for English
- Varies for other languages (e.g. OK for Chinese, no truly satisfactory one yet for Malay)

Semantic

Semantic

- The meaning conveyed by the text
- Hard!
- How to represent ‘meaning’?
- Still an open question in artificial intelligence, cognitive science, psychology…
- Lots of on-going research

Word Sense

- One of zero to many *meanings or concepts* associated with a given *head word/lemma*, as listed in a specific lexicon
- Lexicon: a machine-readable, structured dictionary
- May also include relations between word senses
    - Synonyms, antonyms, is-a-type-of...

Synonym Expansion

- Example in information retrieval (search engine)
- Search for ‘wizard’ would also retrieve documents containing ‘sorcerer’, ‘magician’

Word Sense Disambiguation (WSD)

- a.k.a. Sense-tagging
- Associating a word occurrence with its most likely sense, with repect to a specific lexicon
- **Stop words:** Words that are ignored in NLP tasks, e.g. function words in a sense-tagging task.

How to identify stop words?

- Open-class words (content words): nouns, verbs, adjectives, adverbs
- Closed-class words (function words): determiners, pronouns, conjunctions, infinitives…

…so WSD needs POS-tagging and lemmatisation first

WSD Example

Senses of bank.n in WordNet

1. sloping land (especially the slope beside a body of water)
2. a financial institution that accepts deposits and channels the money into lending activities
3. a long ridge or pile
4. ...

(5)

Input:	*banks*	*raised*	*rates*	*to*	*fight*	*inflation*
Sense-tags:	bank.n.2	raise.v.13	rates.n.1		fight.v.1	inflation.n.1

Concept Tagging

- Label each sense in the input with a concept tag
  (Example below uses WordNet–SUMO mapping)

(6)

Input:	*banks*	*raised*	*rates*	*to*	*fight*	*inflation*
Sense-tags:	bank.n.2	raise.v.13	rates.n.1		fight.v.1	inflation.n.1
Concept tags:	CORPORATION	INCREASING	TAX		VIOLENTCONTEST	INCREASING

Information Extraction

- Examples as given earlier
- Named entity recognition
- Coreference resolution
    - ‘The cat climbed onto the chair. It yawned and slept.’
    - ‘It’ = ‘the cat’? ‘the chair’?
    - ‘cat’ $\xrightarrow{\text{is-a}}$ ANIMAL $\xrightarrow{\text{is-a}}$ ANIMATE OBJECT
    - ‘chair’ $\xrightarrow{\text{is-a}}$ FURNITURE $\xrightarrow{\text{is-a}}$ INANIMATE OBJECT
    - ANIMATE OBJECT $\xrightarrow{\text{capable-of}}$ ‘yawn’, ‘sleep’
    - $\therefore$ ‘It’ = ‘the cat’

Pragmatics

Pragmatics

- Processing text by inclduing context
- Scenario, behavior, cultural, etc

Q	'Can you pass me the salt?'
Machine	'Yes.'
Human	[picks up salt shaker and hands over]

Teacher	'This is your assignment.'
Student	'What is assignment? Can eat one ah?'
Machine	'An assignment is your homework. It is not edible.'
Teacher	[rolls eyes and ignores comment]

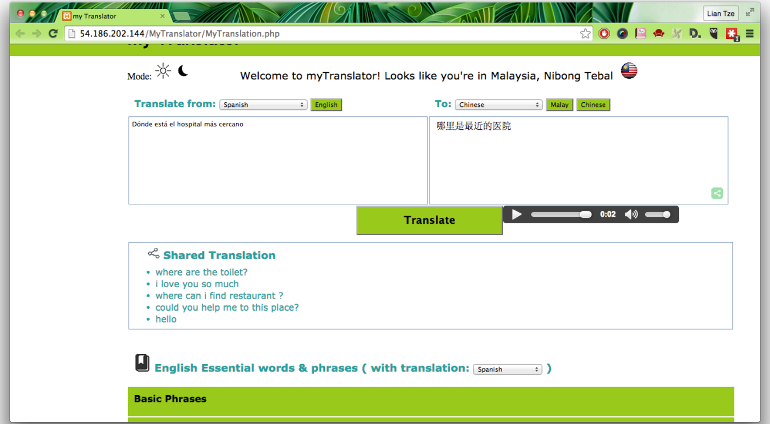
- 'He opened the fridge.' (because he was hungry?)
- VERY HARD!!!

Speech

- Existing libraries: Android, eSpeak, Microsoft SAPI...
- Support for English is satisfactory for FYP purposes
- (Not so good for other languages especially recognition)
- Sounds mechanical!
- Prosody: more natural-sounding, with emotions etc (R&D!)

Example Individual Projects using NLP

Translator Aid for Travellers



Bloom’s Taxonomy Level Categorisation

Analysed Result

Bloom's Taxonomy

No.	Question	Category	Tagged String
1	List 2 server-side programming for web development.		List/NN 2/CD server-side/JJ programming/NN for/IN web/NN development/N
2	List 2 client-side programming language for web develop...		List/NN 2/CD client-side/JJ programming/NN language/NN for/IN web/NN de
3	Security has been a major concern in the current web de...		Security/NN has/VBZ been/VBN a/DT major/JJ concern/NN in/IN the/DT cur
3.a	Explain the term security in software development.	Comprehension	Explain/VB the/DT term/NN security/NN in/IN software/NN development/NN
3.b	Construct a PHP program that will sanitize user input to...	Application	Construct/VB a/DT PHP/NNP program/NN that/WDT will/MD sanitize/VB use
3.c	Evaluate your code written in the above question in te...	Evaluation	Evaluate/VB your/PRP$ code/NN written/VBN in/IN the/DT above/JJ questi
4	There are many ways used by web developers to make t...	Evaluation	Justify/VB your/PRP$ answer/NN ./.

No.	Key Words/Key Sentences	Category
1	Explain	Comprehension
2	Explain the term	Comprehension
3	[Using RegEx Algorithm] Explain the term	Comprehension
4	[Using RegEx Algorithm] Explain the term security in software	Comprehension

More Examples

- Named entity recognition including Malaysian names
- Intelligent meaning lookup for mixed language input with spelling error detection
- *Sentiment analysis of forum posts
- *Information extraction to identify problem parameters
- *Keyword extraction from paper publications

End of Session 1

See you next week!

Code Samples for Common Tasks in NLP

What Programming Languages Can I Use for NLP?

Java Apache OpenNLP, **Stanford NLP**, Lucene, GATE, LingPipe…

Python NLTK (with a nice textbook)

.NET, PHP **Stanford NLP**, Lucene…

Demonstration: Java and PHP, mostly using Stanford's libraries

Stemming

Libraries for Stemming

- Many libraries available
  `http://tartarus.org/martin/PorterStemmer/`
- Or implement your own – nice scope for Individual Project
- Porter (1980) is most famous but there are other algorithms too

Stemming Demo using PorterStemmer.class.php

```php
<?php
    require_once('PorterStemmer.class.php');

    $stem = PorterStemmer::Stem("cats");
    echo "$stem<br/>\n";
    $stem = PorterStemmer::Stem("ponies");
    echo "$stem<br/>\n";
    $stem = PorterStemmer::Stem("produce");
    echo "$stem<br/>\n";
    $stem = PorterStemmer::Stem("producer");
    echo "$stem<br/>\n";
    $stem = PorterStemmer::Stem("producing");
    echo "$stem<br/>\n";
```

Stemming Demo using PorterStemmer.class.php (cont.)

```
?>
```

Stemmer Output

```
cat
poni
produc
produc
produc
```

Stanford Parser

Stanford Parser

- (Klein & Manning, 2003)
- Stanford Parser can POS-tag, lemmatize *and* parse!
- Not always the best results, but widely used 😉

Installing and Setting Up

Java
- Download the Java library from `http://nlp.stanford.edu/software/lex-parser.shtml`
- Unzip and place somewhere on system e.g. in `C:`

PHP
- Download the Java library first
- Download the PHP library from `https://github.com/agentile/PHP-Stanford-NLP`
- Unzip and place in `C:▸xampp▸htdocs`

.NET
- Download the Java library first
- Follow instructions at `http://sergey-tihon.github.io/Stanford.NLP.NET/`
- Class names, function calls etc. exactly same as Java API

POS-Tagging and Lemmatising

Java Code Sample

(Make sure `stanford-parser.jar` and `stanford-parser-version-models.jar` are in the library path)

```java
// Initialise the parser using the English model
String parserModel =
    "edu/stanford/nlp/models/lexparser/englishPCFG.ser.gz";
LexicalizedParser lp = LexicalizedParser.loadModel(parserModel);

// Text to be processed
String text = "26 interested students came to the seminar. "
        + "They signed up quickly.";

// DocumentPreprocessor performs sentence-splitting and tokenising
for (List<HasWord> sentence : new DocumentPreprocessor(
    new StringReader(text))) {
```

Java Code Sample (cont.)

```java
    // Apply the parser on each sentence
    Tree parse = lp.apply(sentence);

    // Just need POS-tag and lemma?
    for (Tree leaf : parse.getLeaves()) {
        String surfaceForm = leaf.value();
        String pos = leaf.parent(parse).value();
        String lemma = Morphology.lemmaStatic(surfaceForm, pos,
            true);
        System.out.print(surfaceForm);
        System.out.print("/");
        System.out.print(lemma);
        System.out.print("/");
        System.out.print(pos);
        System.out.print(" ");
    }
    System.out.println();
}
```

Java Code Sample (cont.)

POS-Tagging and Lemmatising Output

```
23/23/CD interested/interested/JJ students/student/NNS
    came/come/VBD to/to/TO the/the/DT
    seminar/seminar/NN ././.

They/they/PRP signed/sign/VBD up/up/RP
    quickly/quickly/RB ././.
```

PHP Code Sample

```php
<?php
require_once('autoload.php');

// Initialise the parser.
// Put the .jar files somewhere suitable
$parser = new \StanfordNLP\Parser('stanford-parser.jar',
    'stanford-parser-3.5.0-models.jar');

$text = "26 interested students came to the seminar. "
        . "They signed up quickly.";

// parse the text
$result = $parser->parseSentence($text);
```

PHP Code Sample (cont.)

```php
/* var_dump $result and you'll see it's an array with
 * 3 outputs: wordsAndTags, penn, typedDependencies */
var_dump($result);

// If only POS tag and lemma are required:
echo "<ul>";
foreach ($result["wordsAndTags"] as $tagged) {
    // each item is an array of the word and POS
    echo "<li>$tagged[0] ($tagged[1])";
}
echo "</ul>";
?>
```

It doesn't work on my Windows machine!

- Error: `Notice: Undefined offset: 1...`
- Solution: Modify `Parser.php`

```
$output = explode("\n\n", trim($this->getOutput()));
```

to

```
$output = explode("\r\n\r\n", trim($this->getOutput()));
```

The PHP version doesn't return lemmas?

Need to modify `Parser.php` by adding a line:

```
$cmd = $this->getJavaPath()
    . " $options -cp \""
    . $this->getJar()
    . $osSeparator
    . $this->getModelsJar()
    . '" edu.stanford.nlp.parser.lexparser.LexicalizedParser
      -encoding UTF-8 -outputFormat "'
    . $this->getOutputFormat()
    . "\" "
    . '-outputFormatOptions "stem" '
    . $parser
    . " "
    . $tmpfname;
```

POS-Tagging and Lemmatising Output

- 26 (CD)
- interested (JJ)
- student (NNS)
- come (VBD)
- to (TO)
- the (DT)
- seminar (NN)
- . (.)

PHP version returns only the 1st sentence?

- The PHP version only captures the output for 1st sentence
- Possible to modify `Parser.php` to return output for all sentences
- (Try yourself or see me if needed)

Parsing

Which parsing structure to use?

- If you need to use the tree structure of a text – I'd recommend the dependency structure
- Shorter tree; shows parent-child between word/lemmas in text

Java Code Sample

```java
// Continue from earlier Java code
// Use the parsed tree to get the typed dependencies
TreebankLanguagePack tlp = lp.treebankLanguagePack();
GrammaticalStructureFactory gsf = tlp.grammaticalStructureFactory();
GrammaticalStructure gs = gsf.newGrammaticalStructure(parse);
List<TypedDependency> tdl = gs.typedDependenciesCCprocessed();

// Let's just print out each of the parent-child relationship first
for (TypedDependency td : tdl) {
    // parent = "governer"
    IndexedWord parent = td.gov();
    String parentWord = parent.value();
    String parentPOS = parent.tag();
    String parentLemma = Morphology.lemmaStatic(
        parentWord, parentPOS, true);
```

Java Code Sample (cont.)

```
    // child = "dependent"
    IndexedWord child = td.dep();
    String childWord = child.value();
    String childPOS = child.tag();
    String childLemma = Morphology.lemmaStatic(
        childWord, childPOS, true);


    System.out.println(
        "[" + parent.index() + "]" + parentLemma + "/" + parentPOS
        + " <--" + td.reln().getShortName() + "-- "
        + "[" + child.index() + "]" + childLemma + "/" + childPOS);
}
System.out.println();
```

Dependencies as a List

```
[3]student/NNS <--num-- [1]23/CD
[3]student/NNS <--amod-- [2]interested/JJ
[4]come/VBD <--nsubj-- [3]student/NNS
[0]root/null <--root-- [4]come/VBD
[7]seminar/NN <--det-- [6]the/DT
[4]come/VBD <--prep-- [7]seminar/NN

[2]sign/VBD <--nsubj-- [1]they/PRP
[0]root/null <--root-- [2]sign/VBD
[2]sign/VBD <--prt-- [3]up/RP
[2]sign/VBD <--advmod-- [4]quickly/RB
```

Recursively Navigating the Dependency Tree

```java
// recursively go through parent-children links, starting from root
int curParent = 0;
processChildren(curParent, tdl);
System.out.println();

private static void processChildren(int parentID,
     List<TypedDependency> tdl) {
    for (TypedDependency td: tdl) {
        if (td.gov().index() == parentID) {
            IndexedWord childNode = td.dep();
            // do the processing with childNode's values, example:
            // Remember to lemmatise if necessary!!
            System.out.println("Child of node " + parentID + ": ["
                + childNode.index() + "] " +  childNode.word() + "/"
                     + childNode.tag());
            // then process childNode's children...
            processChildren(childNode.index(), tdl);
        }
    }
}
```

Navigating Parent-Child Output

```
Child of node 0: [4] came/VBD
Child of node 4: [3] students/NNS
Child of node 3: [1] 23/CD
Child of node 3: [2] interested/JJ
Child of node 4: [7] seminar/NN
Child of node 7: [6] the/DT

Child of node 0: [2] signed/VBD
Child of node 2: [1] They/PRP
Child of node 2: [3] up/RP
Child of node 2: [4] quickly/RB
```

PHP Code Sample

```
$curParent = 0;
echo "<ul>";
processChildren($curParent, $result["typedDependencies"]);
echo "</ul>";

function processChildren($curParent, $tdl) {
    foreach ($tdl as $td) {
        $parent = explode("/", $td[0]["feature"]);
        $parentLemma = $parent[0];
        $parentPOS = $parent[1];
        $parentIndex = $td[0]["index"];

        $child = explode("/", $td[1]["feature"]);
        $childLemma = $child[0];
        $childPOS = $child[1];
        $childIndex = $td[1]["index"];
```

PHP Code Sample (cont.)

```
        $reln = $td["type"];

        if ($parentIndex == $curParentID) {
            // do the processing with childNode's values, example:
            echo "<li>Child of node $curParentID: [$childIndex] "
                . "$childLemma/$childPOS</li>\n";
            // then process childNode's children...
            processChildren($childIndex, $tdl);
        }
    }
}
```

By default, no POS in typedDependencies?!

Need to modify `Parser.php` by adding another option:

```
$cmd = $this->getJavaPath()
    . " $options -cp \""
    . $this->getJar()
    . $osSeparator
    . $this->getModelsJar()
    . '" edu.stanford.nlp.parser.lexparser.LexicalizedParser
      -encoding UTF-8 -outputFormat "'
    . $this->getOutputFormat()
    . "\" "
    . '-outputFormatOptions "stem,includeTags" '
    . $parser
    . " "
    . $tmpfname;
```

Output

- Child of node 0: [4] come/VBD
- Child of node 4: [3] student/NNS
- Child of node 3: [1] 26/CD
- Child of node 3: [2] interested/JJ
- Child of node 4: [7] seminar/NN
- Child of node 7: [6] the/DT

Speech Synthesis and Recognition

Speech Synthesis and Recognition

- Microsoft Speech Platform
  - English, Japanese, Chinese, French, Spanish...
- Android – Google Speech API
  - English, Spanish, Japanese, Indonesian, French, Italian, Korean, Hindi...
- Read the comprehensive API documentations!

A Word on Language Codes

- ISO-639 Standard
- 2-letter and 3-letter codes

Language	`2-letter`	`3-letter`
English	`en`	`eng`
Malay	`ms`	`msa`, `zsm` (Standard Malay)
Indonesian	`id`	`ind`
Chinese	`zh`	`zho`
Cantonese (Yue)		`yue`
Hokkien (Min Nan)		`nan`
French	`fr`	`fra`
…	…	…

Can also specify Locales

- Can add country code to specify locale

Language code	Language
`en-US`	American English
`en-UK`	British English
`en-AU`	Australian English
`zh-CN`	Mainland China Chinese
`zh-TW`	Taiwanese Chinese
`zh-HK`	Hong Kong Chinese
…	…

- (Sometimes underscore instead of dash; sometimes given as separate arguments…)

Many Others!

Other examples not shown here

- Stanford NER (Java, PHP, .NET)
- If you know Python, do look up NLTK (Bird, Loper & Klein, 2009)
- GATE: Available as GUI workbench and as embedded API
- LingPipe: Some interesting libraries for working with corpora
- ...Many, many more!

WordNets: lexical semantic networks

WordNet: An Electronic Lexical Database

- (Miller, 1995)
- Developed by Princeton University Cognitive Science Laboratory for (American) English
- Lexical entries organised by *meaning* (semantic content)
- Wordnets for many other languages have been developed

Synsets

Synsets = “synonym set”

- Basic unit; represents a word meaning by synonyms, gloss and relations to other synsets
- Different senses of a word (collocation, phrasal verb, etc.) are placed in different synsets according to parts-of-speech
- Each synset contains senses of different words that are considered synonymous

First 3 senses for noun "court"

- 3 synsets, one for each sense (meaning)
- Each synset contain member lemmas with same meaning, same POS
- Each synset has a definition text; may have example sentence

```
<noun.group> court, tribunal, judicature - (an
    assembly (including one or more judges) to conduct
    judicial business)

<noun.group> court, royal_court - (the sovereign and
    his advisers who are the governing power of a state)

<noun.artifact> court - (a specially marked horizontal
    area within which a game is played; "players had to
    reserve a court in advance")
```

Synset Organisation

- 4 synset categories

Category	`POS code`	numerical prefix
noun	`n`	1
verb	`v`	2
adjective	`a, s`	3
adverb	`r`	4

- Primary key: 9-digit synset ID or POS code + 8-digit synset ID
- WN3.0 synset (court, tribunal, judicature) can be identified by 108329453 or n-08329453 or 08329453-n in different systems

Mapping Stanford Parser POS codes to WN POS code

```
/* Ignore all other POS when looking up WN */
if $stanfordPOS starts with 'N' then
   $wnPOS ← 'n'
   $wnPOSnum ← 1
else if $stanfordPOS starts with 'V' then
   $wnPOS ← 'v'
   $wnPOSnum ← 2
else if $stanfordPOS starts with 'J' then
   $wnPOS ← {'a', 's'}
   $wnPOSnum ← 3
else if $stanfordPOS starts with 'R' then
   $wnPOS ← 'r'
   $wnPOSnum ← 4
end if
```

Relations

Noun Synset Relations

hypernymy (court, royal court) *is-a-kind-of* (government, authorities, regime)

holonymy (finger) *is-part-of* (hand, manus, mitt, paw)
(flour) *is-substance-of* (bread), (dough), (pastry)
(jury) *is-member-of* (court, tribunal, judicature)

instance (Mozart, Wolfgang Amadeus Mozart) *is-instance-of* (composer)

(...and their respective inverse relations)

Verb Synset Relations

hypernymy (stroll, saunter) *is-one-way-to* (walk)

troponymy (fear) *has-specific-way* (panic)

cause (teach) *causes* (learn, larn, acquire)

entailment (buy, purchase) *entails* (pay), (choose, take, select, pick out)

verb frames (attack, assail):

```
Somebody --s something
Somebody --s somebody
```

(very simple, without any extra info)

Lexical Relations

antonymy ‘ugliness’ × ‘beauty’, ‘pull’ × ‘push’, ‘difficult’ × ‘easy’, ‘quickly’ × ‘slowly’

attribute ‘strength’ *has-attributes*: ‘delicate’, ‘rugged’, ‘weak’, ‘strong’

derivation ‘maintain’ *is-derivationally-related-to* ‘maintainable’, ‘maintenance’, ‘maintainer’

domain ‘medicine’ *topic-has-terms*: ‘acute’, ‘fulgurating’, ‘gauze’, ...
‘France’ *region-has-terms*: ‘Battle of Valmy’, ‘Bastille’, ‘jeu d’esprit’, ...
‘colloquialism’ *usage-has-terms*: ‘lousy’, ‘humongous’, ‘gobsmacked’, ...

pertainym ‘biannual’ *pertains-to* ‘year’, ‘ancestral’ *pertains-to* ‘ancestor’, ‘Liverpudlian’ *pertains-to* ‘Liverpool’

participle *a* ‘handheld’ *something-participates-in* ‘hold’

Getting the Data

Princeton WordNet Data

- Browse/explore online: `http://wordnet.princeton.edu/`
- Searching WordNet: APIs for many programming languages available
- ...But I recommend downloading WordNet as MySQL data
- Then use whatever programming language you like to query

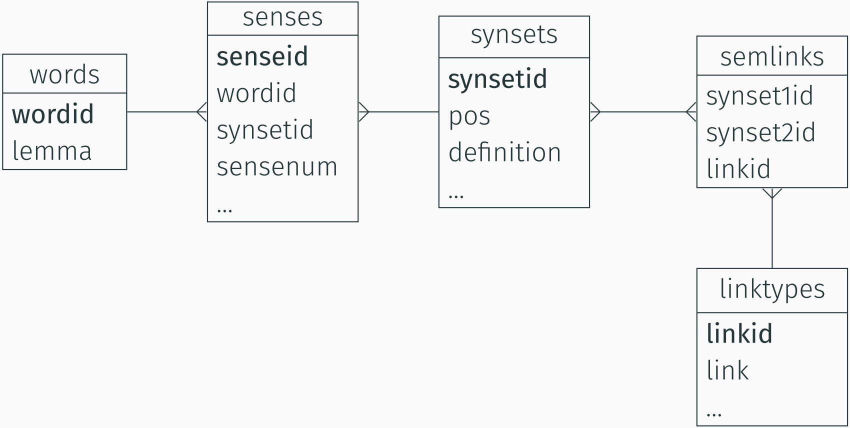
WordNet SQL

- `http://wnsql.sourceforge.net/`
- Download ⟩ wnsql ⟩ mysql ⟩ 3.0 ⟩ mysql-3.0.0-30-wn-30.zip Unzip.
- Create a MySQL database, e.g. `wordnet30`
- Start a Windows command prompt.
  Windows Start ⟩ Run type cmd ↵

```
> cd <folder containing unzipped contents> ↵
> restore.bat ↵
```

- You'll be prompted for the database name, username and password. Wait while the data is copied into tables.
- (May need to add `C:\xampp\mysql\bin` to system path)

Some Tables in WN-MySQL



Querying Synsets (Senses) of a Lemma

```sql
SELECT lemma, synsetid, definition
FROM words INNER JOIN senses USING (wordid)
     INNER JOIN synsets USING (synsetid)
WHERE lemma = 'plant' AND pos = 'n';
```

```
+-------+-----------+-----------------------------------------+
| lemma | synsetid  | definition                              |
+-------+-----------+-----------------------------------------+
| plant | 100017222 | (botany) a living organism ....         |
| plant | 103956922 | buildings for carrying on industrial labor |
| plant | 105906080 | something planted secretly for...       |
| plant | 110438470 | an actor situated in the audience       |
+-------+-----------+-----------------------------------------+
4 rows in set (0.00 sec)
```

Querying Synonyms of a Particular Sense (Synset)

```sql
SELECT lemma
FROM words INNER JOIN senses USING (wordid)
     INNER JOIN synsets USING (synsetid)
WHERE synsetid = 103956922;
```

```
+------------------+
| lemma            |
+------------------+
| industrial plant |
| plant            |
| works            |
+------------------+
3 rows in set (0.00 sec)
```

Looking up Semantic Relations

```sql
SELECT synset2id
FROM semlinks INNER JOIN synsets A
                 ON (A.synsetid = semlinks.synset1id)
              INNER JOIN linktypes USING (linkid)
WHERE A.synsetid = 103956922 AND LINK = 'hypernym';

SELECT lemma
FROM words INNER JOIN senses USING (wordid)
     INNER JOIN synsets USING (synsetid)
WHERE synsetid = 102914991;
```

```
+-----------+
| synset2id |
+-----------+
| 102914991 |
+-----------+
1 row in set (0.00 sec)
```

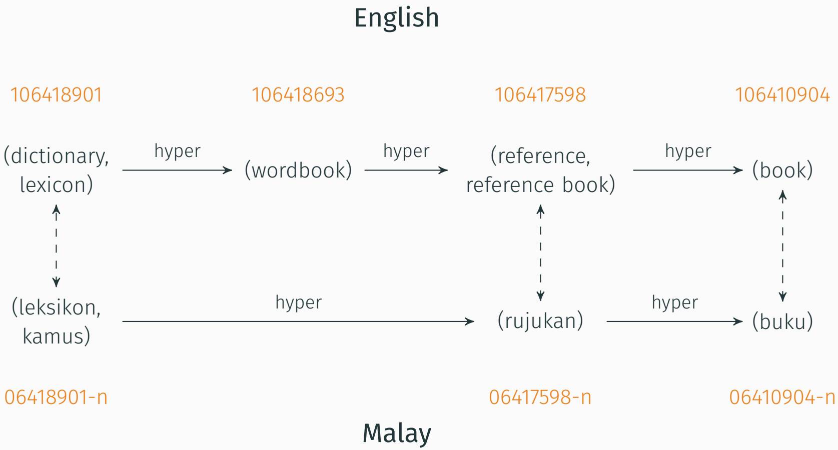
```
+------------------+
| lemma            |
+------------------+
| building complex |
| complex          |
+------------------+
```

Other Lexical Resources Linking to WordNet

Multilingual Wordnets

- Wordnets in different languages – same architecture
- Some free, some not: `http://globalwordnet.org/`
- Almost all are ‘linked’ to PWN (English) by synsetid
- WordNet Bahasa (`http://wn-msa.sourceforge.net/`) (Bond, Lim, Tang & Riza, 2014)
- More languages: Open Multilingual WordNet (`http://compling.hss.ntu.edu.sg/omw/`) (Bond & Paik, 2012)

Looking up Synsets in Other WordNets



Other WordNet-based Resources

- SentiWordNet (Baccianella, Esuli & Sebastiani, 2010)
  - `http://sentiwordnet.isti.cnr.it/`
  - Provides sentiment scores for each synset
  - But see also ML-SentiCon (Cruz, Troyano, Pontes & Ortega, 2014) `http://www.lsi.us.es/~fermin/index.php/Datasets`
- Illustrated WordNet (from Japanese WordNet) (Bond et al., 2009)
  - `http://wn-msa.sourceforge.net/eng/pics.html`
  - Provides a clipart for each synset
- …Many more! Most are OSS.

The End

Thank you!

Bibliography

I am using the APA referencing/citation style in this presentation. *You* should be using Harvard Cite-Them-Right style – do not copy and paste from this list!

- Baccianella, S., Esuli, A. & Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining.. In *LREC* (Vol. 10, pp. 2200–2204).
- Bird, S., Loper, E. & Klein, E. (2009). *Natural language processing with Python*. California: O'Reilly Media.
- Bond, F., Isahara, H., Fujita, S., Uchimoto, K., Kuribayashi, T. & Kanzaki, K. (2009). Enhancing the Japanese Wordnet. In *Proceedings of the 7th Workshop on Asian Language Resources* (pp. 1–8). Association for Computational Linguistics.

Bibliography (cont.)

- Bond, F., Lim, L. T., Tang, E. K. & Riza, H. (2014). The combined Wordnet Bahasa. *NUSA: Linguistic studies of languages in and around Indonesia*, *57*, 83–100. Retrieved from `http://hdl.handle.net/10108/79286`
- Bond, F. & Paik, K. (2012). A survey of wordnets and their licenses. In *Proceedings of the 6th Global WordNet Conference (GWC 2012)* (pp. 64–71). Matsue, Japan.
- Cruz, F. L., Troyano, J. A., Pontes, B. & Ortega, F. J. (2014). Building layered, multilingual sentiment lexicons at synset and lemma levels. *Expert Systems with Applications*, *41*(13), 5984–5994.
- Hutchins, W. J. & Somers, H. L. (1992). *An introduction to machine translation*. Online version: `http://www.hutchinsweb.me.uk/IntroMT-TOC.htm`. London: Academic Press.

Bibliography (cont.)

- Klein, D. & Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics* (pp. 423–430).
- Li, Y., McLean, D., Bandar, Z. A., O'shea, J. D. & Crockett, K. (2006). Sentence similarity based on semantic nets and corpus statistics. *IEEE Transactions on Knowledge and Data Engineering*, *18*(8), 1138–1150.
- Marcus, M. P., Marcinkiewicz, M. A. & Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, *19*(2), 313–330.
- Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, *38*(11), 39–41.

Bibliography (cont.)

- Paice, C. D. (1994). An evaluation method for stemming algorithms. In *Proceedings of the 17th Annual International ACM SIGIR conference on Research and Development in Information Retrieval* (pp. 42–50). Springer-Verlag New York, Inc.
- Pera, M. S. & Ng, Y.-K. (2011). SimPaD: A word-similarity sentence-based plagiarism detection tool on Web documents. *Web Intelligence and Agent Systems*, *9*(1), 27–41.
- Porter, M. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.
- Somers, H. (Ed.). (2003). *Computers and translation: A translator's guide*. John Benjamins Publishing.
- Toutanova, K., Klein, D., Manning, C. & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003* (pp. 252–259). Edmonton, Canada.

Bibliography (cont.)

Wang, H., Can, D., Kazemzadeh, A., Bar, F. & Narayanan, S. (2012). A system for real-time Twitter sentiment analysis of 2012 US presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations* (pp. 115–120). Association for Computational Linguistics.