

Subject Overview & Text Pre-processing

COMP90042 Lecture 1



THE UNIVERSITY OF
MELBOURNE

Course overview

- Text processing
- Search & efficient information retrieval
- Machine learning from words and documents
- Structure prediction, words as sequences and trees
- Translation, information extraction, question answering

Prerequisites

- COMP90049 “Knowledge Technologies” or COMP30027 “Machine Learning”
- Python programming experience
- No knowledge of linguistics or advanced mathematics is assumed
- Caveats – Not “vanilla” computer science
 - * Involves some basic **linguistics**, e.g., syntax and morphology
 - * Requires **maths**, e.g., algebra, optimisation, linear algebra, dynamic programming

Expectations and outcomes

- Expectations
 - * develop Python skills
 - * keep up with readings
 - * classroom participation
- Outcomes
 - * Practical familiarity with range of text analysis technologies
 - * Understanding of theoretical models underlying these tools
 - * Competence in reading research literature

Assessment: Assignments and Exam

- Homework (20% total = 6-7% each)
 - * Small activities building on workshop
 - * Released every few weeks, given 2-3 weeks to complete
- Project (30% total)
 - * Released near Easter & due near end of semester
- Exam (50%)
 - * two hour, closed book
 - * covers content from lectures, workshop and prescribed reading
- **Hurdle** >50% exam, and >50% for (homework + project)

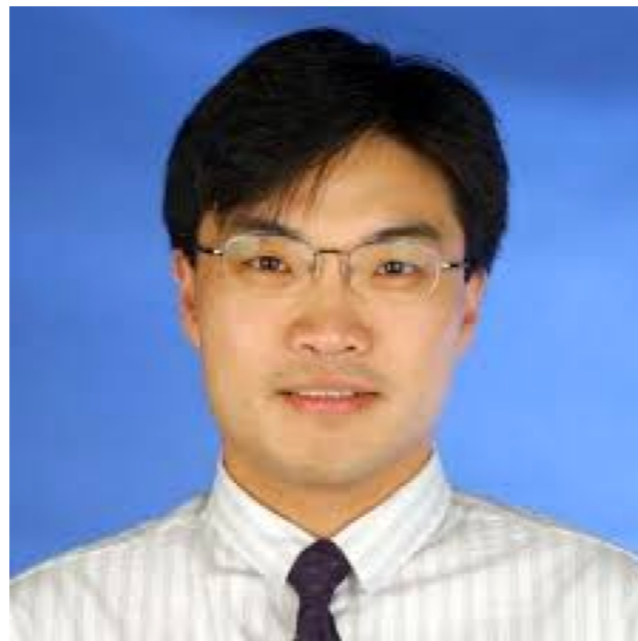
Teaching Staff

- Lecturer



Trevor Cohn

Senior tutors



Winn Chow



Ekaterina
Vylomova

Tutors

- Shivashankar Subramanian
- Andrei Shcherbakov
- Navnita Nandakumar
- Xudong Han
- Zenan Zhai

Course overview

Introduction to text processing

- Text classification, word meaning and document representations

Information Retrieval

- Vector space model, efficient indexing, query expansion

Structure learning

- Sequence tagging, n-gram language modelling, parsing

Larger tasks in Text Analysis

- Information extraction, question answering, translation

Recommended Texts

- Texts:
 - * *Jurafsky and Martin*, Speech and Language Processing, 3rd ed., Prentice Hall. draft ([free online](#)).
 - * *Manning et al*, 2008, Information Retrieval ([free online](#))
 - * *Eisenstein*; Natural Language Processing, Draft 15/10/18. ([free online](#))
- Recommended for learning python:
 - * *Steven Bird, Ewan Klein and Edward Loper*, Natural Language Processing with Python, O'Reilly, 2009. ([free online](#))
- Reading links or PDFs will be posted to website/LMS

Contact hours

- Lectures
 - * Tue 15:15-16:15 Law GM15 (David P. Durham)
 - * Wed 13:00-14:15 Law GM15 (David P. Durham)
- Workshops: several across the week
 - * Bring any questions you have to your tutor, or Senior tutors
 - * May run office hour, if there is sufficient demand
- First method of contact — ask questions on the LMS discussion boards

Python

- Making extensive use of python
 - * workshops feature programming challenges
 - * provided as interactive 'notebooks'
 - * homework and project in python
- Using several great python libraries
 - * NLTK (text processing)
 - * Numpy, Scipy, Matplotlib (maths, plotting)
 - * Scikit-Learn (machine learning tools)

Python

- Python '*Canopy EPD*' installed on workshop machines
 - * Can use this at home (free download, but register with your unimelb email)
 - * Based on Python 3
- New to Python?
 - * Expected to pick this up during the subject, on your own time
 - * Learning resources on worksheet using *edx*
 - * Thursday optional session for python help

Why process text?

- Masses of information ‘trapped’ in unstructured text
 - * How can we find this information?
 - * Let computers automatically reason over this data?
 - * First need to understand the structure, find important elements and relations, etc...
 - * Over 1000s of languages....
- Challenges
 - * Search, displaying results
 - * Information extraction
 - * Translation
 - * Question answering
 - * ...

A Motivating application

- IBM 'Watson' system for Question Answering
 - * QA over large text collections
 - Incorporating speech recognition, speech synthesis and more
 - * <https://www.youtube.com/watch?v=FC3IryWr4c8>
 - * https://www.youtube.com/watch?v=II-M70_bRNq
(from 3:30-4:30)
- Research behind Watson is *not* revolutionary
 - * But this is a transformative result in the history of AI
 - * Combines cutting-edge text processing components with large text collections and high performance computing

Preprocessing

First steps in processing text inputs

```
rules = {  
  RegularExpression["(\\w+) (ss) (es) "] := "$1$2",  
  RegularExpression["(\\w+) (sh) (es) "] := "$1$2",  
  RegularExpression["(\\w+) (ies) "] := "$1" ~~ "y",  
  RegularExpression["(\\w+) (ss) "] := "$1$2",  
  RegularExpression["(\\w+) (us) "] := "$1$2",  
  RegularExpression["(\\w+) (s) "] := "$1"  
};
```

Definitions

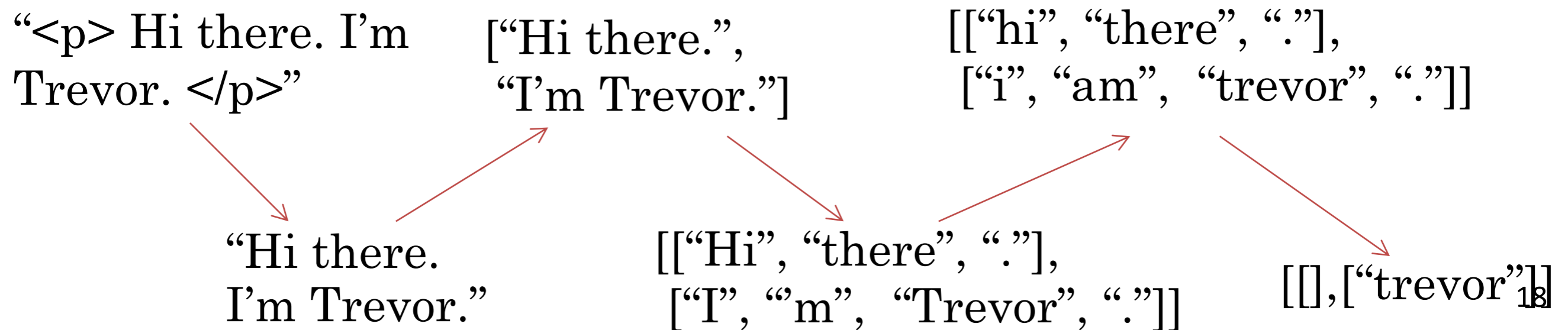
- Words
 - * Sequence of characters with a meaning and/or function
- Sentence
 - * “The student is enrolled at the University of Melbourne.”
- Word **token**: each instance of “the” in the sentence above.
- Word **type**: the distinct word “the”.
 - * Lexicon: a group of word types.
- Document: one or more sentences.
- Corpus: a collection of documents.

Definitions

- Most NLP applications have documents as inputs:
 - * “This movie is so great!!! U should definitely watch it in the theater! Best sci-fi eva!” →
 - * “Eu estive em Melbourne no ano passado.” → “I was in Melbourne last year.”
- **Key point:** language is *compositional*. As humans, we can break these documents into individual components. To understand language, a computer should do the same.
- **Preprocessing** is the first step.

Text Normalisation

- Remove unwanted formatting (e.g. HTML)
- Segment structure (e.g. sentences)
- Tokenise words
- Normalise words
- Remove unwanted words



Sentence segmentation

- Naïve approach: break on sentence punctuation ([.?!])
 - * But periods are used for abbreviations!
(U.S. dollar, ..., Yahoo! as a word)
- Second try: use regex to require capital ([.?!] [A-Z])
 - * But abbreviations often followed by names (Mr. Brown)
- Better yet: have lexicons
 - * But difficult to enumerate all names and abbreviations
- State-of-the-art uses machine learning, not rules

Tokenisation: English

- Naïve approach: separate out alphabetic strings ($\backslash w+$)
- Abbreviations (*U.S.A.*)
- Hyphens (*merry-go-round vs. well-respected vs. yes-but*)
- Numbers (*1,000,00.01*)
- Dates (*3/1/2016*)
- Clitics (*n't* in *can't*)
- Internet language (*http://www.google.com, #metoo, :-)*)
- Multiword units (*New Zealand*)

Tokenisation: Chinese

- Some Asian languages are written without spaces between words
- In Chinese, words often correspond to more than one character

墨大 的 学生 与众不同


Unimelb 's student(s) (are) special

Tokenisation: Chinese

- Standard approach assumes an existing vocabulary
- MaxMatch algorithm
 - * Greedily match longest word in the vocabulary

$V = \{\text{墨, 大, 的, 学, 生, 与, 众, 不, 同, 墨大, 学生, 不同, 与众不同}\}$

墨大的学生与众不同



match 墨大, match 的, match 学生, match 与众不同,
 move to 的 move to 学 move to 与 done

Tokenisation: Chinese

- But how do we know what the vocabulary is
- And doesn't always work

去	买	新西兰	花
go	buy	New Zealand	flowers

去	买	新	西兰花
go	buy	new	broccoli

Word normalisation

- Lower casing (Australia -> australia)
- Removing morphology
- Correcting spelling
- Expanding abbreviations

Inflectional Morphology

- Inflectional morphology creates grammatical variants
- English inflects nouns, verbs, and adjectives
 - * Nouns: *number* of the noun (-s)
 - * Verbs: *number* of the subject (-s), the *aspect* (-ing) of the action and the *tense* (-ed) of the action
 - * Adjectives: *comparatives* (-er) and *superlatives* (-est)
- Many languages have much richer inflectional morphology than English
 - * E.g. French inflects nouns for gender (*un chat, une chatte*)

Lemmatisation

- Lemmatisation means removing any inflection to reach the uninflected form, the *lemma*
 - * *speaking* → *speak*
- In English, there are irregularities that prevent a trivial solution:
 - * *poked* → *poke* (not *pok*)
 - * *stopping* → *stop* (not *stopp*)
 - * *watches* → *watch* (not *watche*)
 - * *was* → *be* (not *wa*)
- A lexicon of lemmas needed for accurate lemmatisation

Derivational morphology

- Derivational morphology creates distinct words
- English derivational *suffixes* often change the lexical category, e.g.
 - * *-ly* (*personal* → *personally*)
 - * *-ise* (*final* → *finalise*)
 - * *-er* (*write* → *writer*)
- English derivational *prefixes* often change the meaning without changing the lexical category
 - * *write* → *rewrite*
 - * *healthy* → *unhealthy*

Stemming

- Stemming strips off all suffixes, leaving a *stem*
 - * E.g. *automate, automatic, automation* → *automat*
 - * Often not an actual lexical item
- Even less lexical sparsity than lemmatisation
- Popular in information retrieval

The Porter stemmer

- Most popular stemmer for English
- Applies rewrite rules in stages
 - * First strip inflectional suffixes,
 - E.g. *-ies* → *-i*
 - * Then derivational suffixes, from right to left
 - E.g. *-isation* → *-ise*; *-ise* →

Fixing Spelling errors

- Why fix them?
 - * Spelling errors create new, rare types
 - * Disrupt various kinds of linguistic analysis
 - * Very common in internet corpora
 - * In web search, particularly important in queries
- How?
 - * String distance (Levenshtein, etc.)
 - * Modelling of error types (phonetic, typing etc.)
 - * Use an n -gram language model

Other word Normalisation

- Normalising spelling variations
 - * Normalize → Normalise (or vice versa)
 - * U r so coool! → *you are so cool*
- Expanding abbreviations
 - * US, U.S. → United States
 - * imho → in my humble opinion

Stop words

- Definition: a list of words to be removed from the document
 - * Typical in bag-of-word (BOW) representations
 - * Not appropriate when sequence is important
- How to choose them?
 - * All *closed-class* or *function* words
 - E.g. *the, a, of, for, he, ...*
 - * Any high frequency words

A final word

- Preprocessing unavoidable in text analysis
- Can have a major effect on downstream applications
- Exact steps may vary depending on corpus, task
- Simple rule-based systems work well, but rarely perfectly

Further Reading

- J&M3 Ch 2. on Normalisation
(includes a review of regex and Levenshtien distance)
- (Optional) details on the Porter Stemmer algorithm
(<http://snowball.tartarus.org/algorithms/porter/stemmer.html>)