Language and Document Analysis: Motivating Latent variable Models

Wray Buntine
National ICT Australia (NICTA)

MLSS, ANU, Jan., 2009

Part I

Motivation and Background

What a good Statistical NLP Course Needs

Apart from the usual CS background (algorithms, data structures, coding, *etc.*):

- prerequisites or coverage of information theory, and computational probability theory;
- theory of context free grammars, normal forms, parsing theory,*etc.*;
- programming tools: Python!

None of this is presented here!

Outline

Outline

We do a review of the analysis of formal natural language (not a formal analysis of natural language).

What is Formal Natural Language

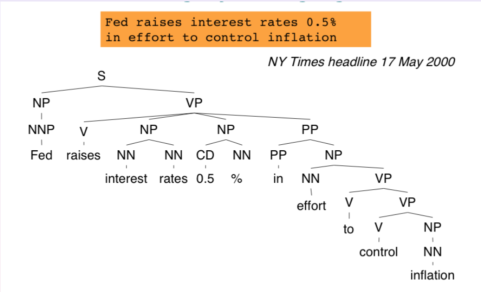
- Formal language is taught in schools (*e.g.*, grammar schools) with correct grammar, punctuation and spelling.
- Most books, more traditional print media, formal business communication, and newspapers use this.
- But errors exist even in the *The Times* and *The New York Times*.
- In contrast, informal language is found in email, people's web pages, chat groups, and "trendy" print media.

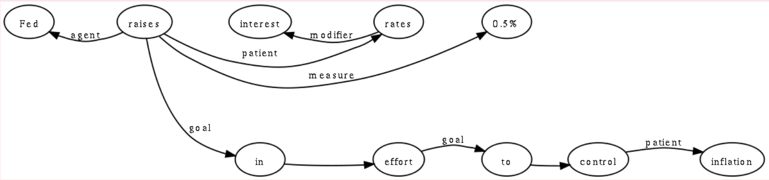
Outline

Analysing Language

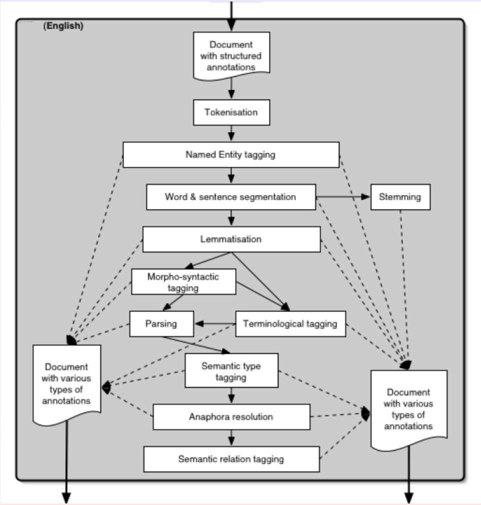
Example from *McCallum's NLP course*

- Left, a traditional parse tree showing constitutent phrases.
- Below, a dependency graph showing *semantic roles*.





Traditional NLP Processing



Full processing pipeline might look like this for English.

- Typical accuracies for various stages might be 90-98%.
- But it can drop down to 60% for the later semantic analysis.

Common Tasks in NLP

Tokenisation: breaking text up into basic tokens such as word, symbol or punctuation.

Chunking: detecting parts in a sentence that correspond to some unit such as "noun phrase" or "named entity".

Part-of-speech tagging: detecting the part-of-speech of words or tokens.

Named entity recognition: detecting proper names.

Parsing: building a tree or graph that fully assigns roles/parts-of-speech to words, and their inter-relationships.

Semantic role labelling: assigning roles such as "actor", "agent", "instrument" to phrases.

NLP in Chinese

Input
A Chinese sentence
我弟弟要买两个足球。
My brother wants to buy two balls.

Output (the word and POS sequence)
我/r (my) 弟弟/n (brother) 要/v (want)
买/v (buy) 两/m (two) 个/q (classifier)
足球/n (football) 。/w (period)

- Tokenisation (segmenting words) is very difficult.
- Easier in Japanese[1] because their foreign words use separate phonetic alphabets.
- Little morphology used.

[1] Japanese writing is based on traditional Chinese.

NLP in Hebrew

Verbs:

רציתי	RATSITI	= I wanted
רצית	RATSITA	= You wanted (male)
רצית	RATSIT	= You wanted (female)
רצה	RATSA	= He wanted
רצתה	RATSTA	= She wanted
רצינו	RATSINU	= We wanted
רציתם	RATSITEM	= You wanted (male, plural)
רציתן	RATSITEN	= You wanted (female, plural)
רצו	RATSU	= They wanted

Lack of vowels:

ספק	SAFEK	= doubt
ספק	SAFAK	= clapped
ספק	SIPEK	= provided
ספק	SUPAK	= has been provided
ספק	SAPAK	= provider

- Has a fairly rich morphology (i.e., modification of words to match case).
- Prepositions attached to words as suffixes.
- Vowels not included in alphabet.

Suffixes:

מדבר	MIDAVAR	= from something
לדבר	LEDAVAR	= to something
כדבר	KEDAVAR	= as something

NLP in Hebrew, cont.

מספר ההרוגים ברעידת האדמה בסין עבר אתמול את ה-13 אלף והוא צפוי לעלות עוד יותר. כלי התקשורת במדינה דיווחו כי כ-19 אלף בני אדם עדיין קבורים מתחת להריסות באחד מאזורי האסון.

- Here is part of a news article about China.
- Underlined words are ambiguous (multiple meanings due to lack of vowels).
- Red parts are attached suffixes.
- Note Hebrew and Arabic share the general features, both are derived from versions of Aramaic.
    - Many Asian and European alphabets are derived from Phoenician, a precursor to Aramaic, but they also have vowels. Phoenician itself is a simplification of Egyptian hieratic.

Translation Difficulties

English: $\boxed{\text{I am in the cafe too.}}$

Finnish: $\boxed{\text{On kahvilassahan.}}$

Finnish, an agglutinating language like Mongolian and Turkish, can express four English words in one!

The translation is: $\text{On}_{\text{I am}}$ $\text{kahvi}_{\text{coffee}}\text{la}_{\text{place}}\text{ssa}_{\text{in}}\text{han}_{\text{emphasis}}$.

This makes statistical machine translation very difficult. For instance, only the base word "kahvila" will be in any dictionary.

Translation Difficulties, cont.

Some languages represent names differently, especially those originating outside of the Latin based alphabets.

Code	Language	Translation
EN	English	Saddam Hussein
LV	Latvian	Sadams Huseins
HU	Hungarian	Szaddám Huszein
ET	Estonian	Saddäm Husayn

Language Ambiguities

An unnamed high-performance commercial parser made the following analysis of a sentence from Reuters Newswire in 1996.

> Clothes made of [hemp and smoking paraphernalia]$_{\text{phrase}}$ were on sale.

The correct analysis is:

> [Clothes made of hemp]$_{\text{phrase}}$ and [smoking paraphernalia]$_{\text{phrase}}$ were on sale.

This misinterpretation is a common semantic problem with current parsing technology.

Language Ambiguities, cont.

- $\boxed{\text{New}_{\text{adjective}} \ \boxed{\text{York Tennis Club}}_{\text{name}} \text{ opening today.}}$ versus
  $\boxed{\boxed{\text{New York Tennis Club}}_{\text{name}} \text{ opening today.}}$
- $\boxed{\boxed{\text{He worked at Yahoo!}}_{\text{sentence}} \ \boxed{\text{Tuesday.}}_{\text{sentence}}}$ versus
  $\boxed{\text{He worked at } \boxed{\text{Yahoo!}}_{\text{name}} \text{ Tuesday.}}_{\text{sentence}}$
- $\boxed{\text{Stolen painting found by } \boxed{\text{tree}}_{\text{location}}.}$ versus
  $\boxed{\text{Stolen painting found by } \boxed{\text{tree}}_{\text{actor}}.}$
- $\boxed{\text{Iraqi } \boxed{\text{head}}_{\text{body part}} \text{ seeks } \boxed{\text{arms}}_{\text{body part}}.}$ versus
  $\boxed{\text{Iraqi } \boxed{\text{head}}_{\text{politician}} \text{ seeks } \boxed{\text{arms}}_{\text{weapons}}.}$

Language Ambiguities, cont.

- Ambiguities arise in all processing steps, due to the tokenisation done, the identification of proper names, the part of speech assigned, the parse, or the semantic role assigned.
- All languages have particular versions of the ambiguity problem. *e.g.*, standard Arabic and Hebrew don't represent vowels in their text!

We resolve ambiguity by appeal to *distributional semantics*, that the meaning of a word is given by its distribution with the words surrounding it, its context.

Handling of ambiguity generally requires that intermediate processing carry uncertainty, for instance, by using latent variables in statistical methods.

Outline

Word Classes (dictionary version of part of speech)

Part of speech	Function	Examples
Verb	action or state	(to) be, have, do, like, work, sing, can, must
Noun	thing or person	pen, dog, work, music, town, London, John
Adjective	describes a noun	a/an, 69, some, good, big, red, well, interesting
Adverb	describes a verb, adjective or adverb	quickly, silently, well, badly, very, really
Pronoun	replaces a noun	I, you, he, she, some
Preposition	links a noun to another word	to, at, after, on, but
Conjunction	joins clauses or sentences or words	and, but, when, because
Interjection	short exclamation, can be in sentence	oh!, ouch!, hi!

Word Forms

Morpheme: Is a semantically meaningful part of a word.

Inflection: A version of the word within the one word class by adding a grammatical morpheme. "walk" to "walks", "walking", and "walked".

Lemma: The base word form without inflections, but no change in word class. "walking" lemmatizers back to "walk", but "redness" (N) does not lemmatise to "red" (A).

Derivation: Adding grammatical morphemes to change the word class. "appoint" (V) to "appointee" (N), "clue" (N) to "clueless" (A). Uses "-ation", "-ness", "-ly" *etc.*

Stemming: Primitive version of lemmatization that strips off grammatical morphemes naively, usually in a context free manner.

Open versus Closed: Nouns, verbs, adjectives, adverbs are considered *open* word classes that continually admit new entries.

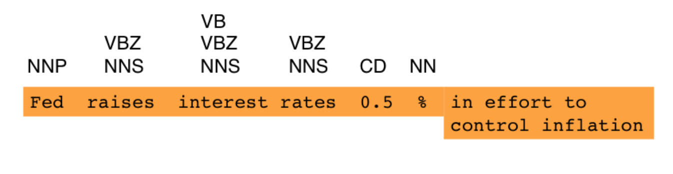
Parts of Speech (computational version)

Example parts of speech from the Tagging Guidelines for the Penn Treebank.

POS	Function	Examples
CC	coordinating conjunction	and, but, either
CD	cardinal number	three, 27
DT	determiner	a, the, those
IN	preposition or subordinating conjunction	out, of, into, by
JJ	adjective	good, tall
JJS	adjective, superlative	best, tallest
MD	modal	he *can* swim
NN	noun, singular or mass	the *ice* is cold
NNS	noun plural	the *iceblocks* are cold
PDT	predeterminer	*all* the boys
SYM	symbol	$, %
VBD	verb, past tense	swam, walked
...	...	...

Parts of Speech (computational version), cont.

- For computational analysis, more detail over the 8 word classes is needed in order to capture inflections and variations supporting a parse.
- With just candidate POS for each word, many different parses can exist. McCallum's initial example is shown again below.



Collocations

Small, usually contiguous, sequence of word that behaves semantically like a single word: "hot dog", "with respect to", "home page", "fourth quarter", "run down",

- Meaning of a collocation is different to the meaning of its parts.
  - The collocation cannot be modified easily without changing the meaning: "kicked the bucket" versus "kicked the tub", "the bucket was kicked".
  - We identify collocations by appeal to distributional semantics.
- Related: multi-word expression/unit, compound, idiom.
- In some languages, collocations replaced by compounds (words are joined with no space or hyphen).
- Important for parsing, dictionaries, terminology extraction, ...

Outline

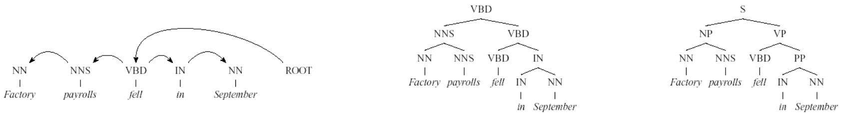
Constituents

A word or a group of words that functions as a single unit within a hierarchical structure.

e.g. noun phrase, prepositional phrase, collocation, *etc.*

- Often can be replaced by a single pronoun and the enclosing sentence is still gramatically valid.
- Serve as a valid answer to some question.
  e.g., How did you get to work? By train.
- Admits standard syntactic manipulations.
  e.g., can be joined with another using "and", can be moved elsewhere in the sentence as a unit.
- Building a parse tree involves building the complete set of constituents for a sentence.

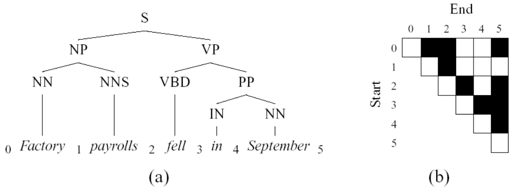
Parsing



(a) Classical Dependency Structure (b) Dependency Structure as CF Tree (c) CFG Structure

- Sometimes we want a dependency tree showing syntactic or semantic relationships, as in (a).
  - Usually, we want the relationships labelled.
    e.g. arc from "fell" to "in" labelled with *time*, arc from "fell" to "payrolls" labelled with *patient*.
- Some formal linguistic theory develops a parse tree, in this case a Context Free Grammar (CFG) is used in (c).
- Figure shows a derivation of the parse tree from the dependency tree.

Shallow Parsing



Span	Label	Constituent	Context
⟨0,5⟩	S	NN NNS VBD IN NN	◇ – ◇
⟨0,2⟩	NP	NN NNS	◇ – VBD
⟨2,5⟩	VP	VBD IN NN	NNS – ◇
⟨3,5⟩	PP	IN NN	VBD – ◇
⟨0,1⟩	NN	NN	◇ – NNS
⟨1,2⟩	NNS	NNS	NN – VBD
⟨2,3⟩	VBD	VBD	NNS – IN
⟨3,4⟩	IN	IN	VBD – NN
⟨4,5⟩	NN	NNS	IN – ◇

(c)

1: (a) Example parse tree with (b) its associated bracketing and (c) the yields and contexts for each constituent span

- A full parse yields many subtrees or constituents, labelled verb phrase (VP), prepositional phrase (PP), *etc.*
- We can also note the labels of a particular type (*e.g.*, all NPs), and build a classifier that recognises just that type.
- Recognising the start and end of a particular type of constituent is called **shallow parsing** or **chunking**.
- Parsing can also be represented as a structured classification problem, recognising the best coherent set of constituents.

Outline

We look beyond the text content to consider applications of document processing.

Processing of Documents

- Documents have a structure with text, links to other documents, citations to publications, images, indexes, and so forth.
- Why do we care about documents?
- What applications can be made?

Outline

Informal Language

Text messages: My smmr hols wr CWOT. B4, we used 2go2 NY 2C my bro, his GF & thr 3 :- kids FTF. ILNY, it's a gr8 plc.

IRC Chat: `Meta-man:` NLP is a little tricky to do over IRC
`Dan_26:` I see no diff
`galamud:` I'm not pissed! I'm flattered! I mean, er... =)
`Meta-man:` hold that thought ...to your checkbook :]
`JonathanA:` HAH! LOL

Web Page Structure

- Web pages have complicated structures and genre, more so than traditional documents (letters, books, *etc.*).
- Example genres: product page, personal home page, FAQ, news item, blog, corporate data sheet, ...
- Much of the content will be template content shared across many similar pages.
- No standard guidelines, so must determine heuristically.

Linguistic Resources

- A large number of different resources now becoming available, due to the Internet and digitisation.
- Included: gazetteers, dictionaries, tagged text (tagged with POS, name entity types, *etc.*), word sense data, case frame and semantic role data (*i.e.*, for verbs), collocations, aligned translations.
- Tagged and marked up linguistic resources are the hardest to get, but are the ones most needed for supervised statistical NLP.

Availability of linguistic resources is a key determining factor in the success of statistical NLP projects.

Unsupervised (or semi-supervised) approaches to statistical NLP are most needed.

Outline

1. Formal Natural Language

2. Document Processing
   - Language in the Electronic Age
   - Information Warfare
   - Why Analyse Documents

3. Document Analysis

The Internet Society

- Primary school students have internet component in coursework are given internet search tasks as assignments.
- Internet news and blogs have overtaken newspapers as primary information source, but the business models are unclear.
- E-government, business and consumer e-services booming.
- Search and internet-based multimedia now a significant form of entertainment.

  e.g. 8 year-old boy with keywords "dinosaur", "meteor".

The Internet Society, cont.

- Advertising on specialist websites, on particular keyword searches, or on your email based on its content, is well focussed.
- Targeted advertising through the web, for instance Google AdSense, is considered the best value for money for advertising.
- Major industry companies track “green” websites and blogs for potential environmental scandals.

Document analysis has taken on a new life due to the internet. Business, government and consumer ramifications still unfolding.

Information Warfare

Definition: "the use and management of information in pursuit of a competitive advantage over an opponent."

- Email spam, link spam, *etc.* Whole websites are now fabricated with fake content in the effort by spammers.
- "More than half of Americans say US news organizations are politically biased, inaccurate, and don't care ...," *Pew Research Center on "news"* (Aug. 2007)
  - "Poll respondents who use the Internet as their main source of news – roughly one quarter of all Americans – were even harsher with their criticism."
  - 80% of the watchers of FOX news had one or more major misconceptions over Iraq war, compared with only 23% for PBS/NPR, *WorldPublicOpinion.ORG survey* (Oct. 2003)

It's an information war out there on the internet (between consumers, companies, not-for-profits, voters, parties, news publishers, ...).

Outline

Bioinformatics: Medline

- *PubMed* is the most popular database in Biology, and the main database MedLine has over 16 million entries.
  - entries are abstracts and metadata in (*MedLine format*, *XML format*, ...
  - 2,000-4,000 new entries/day from 5000 journals in 37 languages.
- The abstract databases are searchable using free text and contolled vocabularies, such as *MeSH* terms.

Tasks in MedLine

- The MeSH terms are generally entered by users and not thorough. Thus subject-specific searching patchy.
- Named entities (genes, proteins) have many different versions so it is difficult to search for them.
- Same problems apply to many technical information resources, such as patent databases.

European Media Monitor: NewsExplorer

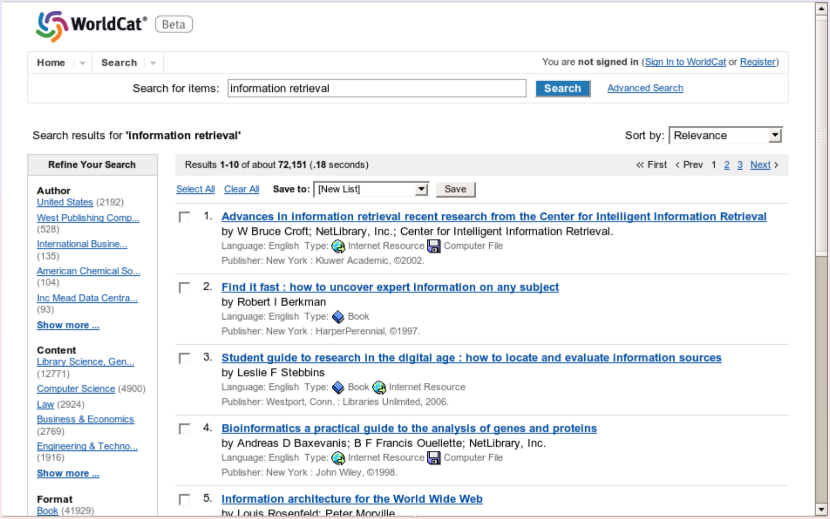
- Developed at the European Commission's Joint Research Center (JRC) in Italy. Online at `http://press.jrc.it/`.
- Completely automated:
  - automatically generate daily news summaries, and provides a *daily briefing*,
  - collect and cluster news events, and *news personalities*,
  - provide geographical, *theme* and time summaries,
  - cross-lingual capabilities.
- Uses relatively simple NLP and SML technology cleverly.
- Widely regarded within the EU Commission and by Google.

Advanced Search Engines

- Clustering output to give a dynamic snapshot of the area, such as *Clusty*.
- Providing a stronger typing of content in terms of area, keyword, genre, document type, such as *Exalead*
- Subject specific areas such as *academic search*, product search and *library catalogue search*.

Advanced Search Engines: Visualisation

World Wide Library



Patent Search: PatentLens

- Started out as a *patent search engine* for Bioinformatics to support patent packaging.
- Software is open source, but largely developed in-house at *Cambia*.
- Many specific facilities to support patents (organisation/company matching, cross-nation support, gene name search ...).
- The patent landscape is changing, see *Open Invention Network*.

Social Bookmarks: Del.icio.us

- *Del.icio.us* is one of the best known social bookmarking sites.
- Uses tagging to provide higher-weighted keywords.
- Uses social bookmarks to get popularity/ "authority" for pages.
- Purchased by Yahoo in 2005.

Opinion: their search returns best pages on fairly general topic areas, *e.g.* *information retrieval*, (*i.e.*, but not "home page" or "lost page" search).

Business Applications

Intelligence: information from the web about consumer trends and opinions, and about competitors.

Summaries: executive reports and overviews based on a large collection of documents input.

Intranet support: search and browse, personalisation, categorization, document management.

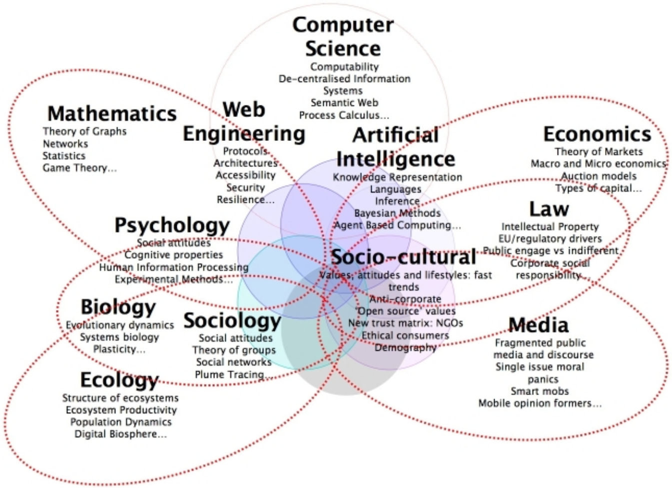
Administration: eGovernment and electronic document processing.

Advertising: many aspects of advertising now running online.

Outline

We sketch out the field of document analysis, with major emphasis on text.

Web Science



From *Web Science*.

Outline

Linguistic Representation

Linguistic aspects:

- basic representations presented previously: morpheme, token, word class, part-of-speech, lemma, collocation, term, named entity, constituent, phrase, parse tree, case frame, semantic role, dependency graph;
- transformations and default processing steps between them;
- differences for different languages;
- sources of ambiguity.

It is important to understand the linguists viewpoints, and their whys and wherefores.

Computational Representation

Computational aspects for the text in documents:

- data formats such as XML and its support tools and representations such as Schema, XQuery, ...;
- data structures and manipulation such as trees, graphs, regular expressions, FSA, ...;
- character processing, UTF8, simplified Chinese, Latin, ...

All of these aspects make a scripting language like Python (or Perl) the best platform for beginning statistical NLP.

Meaning Representation

The layers of processing for the text in documents.

Character level: characters $\longrightarrow$ tokens sentences $\longrightarrow$ paragraphs $\longrightarrow$ documents.

Syntactic level: morphemes $\longrightarrow$ lemmas and parts of speech $\longrightarrow$ collocations, terms and named entities $\longrightarrow$ constituents, phrases $\longrightarrow$ sentences.

Semantic level: case frames and semantic roles, dependencies, topic modelling, genre.

The three levels tend to interact, and the various stages in each level interact as well.

Outline

Part of Speech Data

- Human annotators have taken, say, 20Mb of Wall Street Journal text and carefully assigned POS to tokens.
- There can be some difficulty in assigning POS:
  - "She stepped off/IN the train." `versus` "She pulled off/RP the trick."
  - "We need an armed/JJ guard." *versus* "Armed/VBD with only a knife, ..."
  - "There/EX was a party in progress there/RB."
- POS data laborious to construct, but very useful for statistical methods.

> Most parsers don't require POS tagging beforehand. It is generally done as a pre-processing step for information extraction. or shallow parsing.

Computer Dictionary: CELEX

- CELEX is the Dutch Centre for Lexical Information.
- Provides CDROM with lexical information for English, German and Dutch, called *CELEX2*. Available from LDC.
- Contains orthography (spelling), phonology (sound), morphology (internal structure of words), syntax, and frequency for both lemmas and word-forms.
- Provided for 50,000 lemmata.

Headword	Pronunciation	Morphology	Cl	Type	Freq
celebrant	"sE-lI-br@nt	((celebrate),(ant))	N	sing	6
cellarages	"sE-l@-rIdZIs	((cellar),(age),(s))	N	plu	0
cellular	"sEl-jU-l@r*	((cell),(ular))	A	pos	21

Computer Thesaurus: WordNet

- Developed at Princeton University under the direction of psychology professor George A. Miller from 1985 on.
- Contains over 150,000 words or collocations, *e.g.* see *make*, *red*, *text*.
- Words in a network with link types corresponding to:

hypernym:	generalisation,
hyponym:	specialisation,
holonym:	has as a part,
meronym:	is a part of,
antonym:	contrasting or opposite,
derivationally related:	"textual" is for "text",
word senses:	different semantic use cases identified,
case frames:	case frames for verbs.

- Available free (with an "unencumbered license"), and lots of supporting software.

Gazetteers

- Term originally applies to geographic name databases that might contain auxiliary data such as type (mountain, town, river, *etc.*), location, parent state, *etc.*
- Sometimes extended in NLP to apply to other specialised databases of proper names.
- Proper names treated differently in NLP because:
  - they behave as single tokens and don't inflect,
  - generally are marked with first letter uppercase,
  - are the greatest source of new or unknown words in text, and are not usually in dictionaries.

Good gazetteers and dictionaries are critical for performance in any specialised domain.

Linguistic Data Consortium

- *LDC* is an open consortium initially funded by ARPA.
- Wide *variety of data* including speech and transcripts, news and transcripts, language resources, annotated and parsed data.
- Includes the famous Penn Treebank which has POS tagging and parse trees.

Outline

Important Issues

We've looked at applications, representation and linguistic resources, what about:

- **Software:** many open source tools exist of varying quality, though some of the best tools are commercial and expensive.
- **Evaluation:** a myriad of evaluation tracks exist for every aspect, and these generate some important data sets and resources.
- **Algorithms:** space and time complexity, *etc.*
- **Statistical prerequisites:** the field has prodigious users and creators of statistical techniques.

Recognised Problems

Information retrieval (IR): given query words, retrieve relevant parts from a document collection.

Question answering (QA): similar to IR but return an answer.

Document summarisation: taking a small set of documents on a given theme and preparing a short summary or executive brief.

Topic detection and tracking (TDT): tracking topics, and discovering new ones in information streams.

Semantic web annotation: annotating documents with appropriate semantic mark-up.

Classification: categorising documents into topic hierarchies, or creating hierarchies suited for a collection.

Genre identification: predicting the genre type.

Sentiment analysis: predicting the sentiment (negative, satisfied, happy, ...) of a blog or chat participant or commentary.

Recognised Problems, cont.

Document structure analysis: identifying the parts of a web page or document such as title, index, advertising, body, *etc.*

Linguistic resource development: tagging of text with parse structures, POS, semantic roles, name entities, *etc.*, and development of dictionaries, gazetteers, case frames, *etc.*, especially in specialised subjects.

Recommendation: from user characteristics and prior selections, make recommendations, such as collaborative filtering.

Ranking: given candidate responses for a recommendation or retrieval task, do the fine grained ranking.

Cleaning up Wikipedia: the Wikipedia would be an amazing linguistic resource if only, ...

Recognised Problems, cont.

Machine translation (MT): automatically convert text to another language,

Cross language IR (CLIR): from queries in one language probe document collection in another.

Email spam detection: recognising spam email.

Trust and authority: measures of document/author quality in terms authority and trust based on content, links, citation, history, *etc.*

Communities: analysis and identification of online communities.

Video and Image X: most of the above applied to video and images.

Outline

And so ends Part 1. Next we look at specific problems and algorithms.

Language and Document Analysis: Motivating Latent variable Models

Wray Buntine
National ICT Australia (NICTA)

MLSS, ANU, Jan., 2009

Part II

Problems and Methods

Outline

We review some key problems and key algorithms using latent variables.

1. Part-of-Speech with Hidden Markov Models
   - Markov Model
   - Hidden Markov Model

2. Topics in Text with Discrete Component Analysis
   - Background
   - Algorithms

Outline

We look at the Hidden Markov Model, because its an important base algorithm. We use it to introduce Conditional Random Fields, a recent high performance algorithm.

Parts of Speech, A Useful Example

NNP	VBZ NNS	VB VBZ NNS	VBZ NNS	CD	NN	
`Fed`	`raises`	`interest`	`rates`	`0.5`	`%`	`in effort to control inflation`

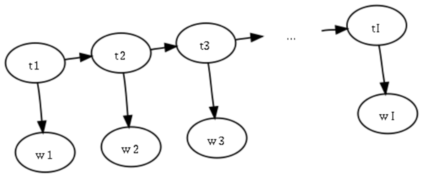
- A set of candidate POS exist for each word. taken from a dictionary or lexicon. Which is the right one in this sentence?
- Lets take some fully tagged data, where the truth is known, and use statistical learning.
- A standard notation for representing tags , in this example, is:

  `Fed/NNP raises/VBZ interest/NNS rates/NNS`
  `0.5/CD %/% ...` *(in effort to control inflation.)*

- We use this to illustrate Markov models and HMMs.
  Reference: Manning and Schütze, chaps 9 and 10.

Outline

Markov Model with Known Tags



- There are I words. w_i = i-th word. t_i = tag for i-th word.
- Our 1st-order Markov model in the figure shows which variables depend on which.
- The $(i+1)$-th tag depends on the i-th tag. The i-th word depends on the i-th tag.
- Resultant formula for $p(t_1, t_2, t_3, \ldots, t_N, w_1, w_2, w_3, \ldots, w_N)$ is

$$p(t_1) \prod_{i=2,\ldots,I} p(t_i | t_{i-1}) \prod_{i=1,\ldots,I} p(w_i | t_i)$$

Fiitting Markov Model with Known Tags

- Have $p(t_1, t_2, t_3, \ldots, t_N, w_1, w_2, w_3, \ldots, w_N)$ is

$$p(t_1) \prod_{i=2,\ldots,I} p(t_i|t_{i-1}) \prod_{i=1,\ldots,I} p(w_i|t_i)$$

- Have K distinct tags and J distinct words.
- Use $p(t_i = k_1|t_{i-1} = k_2) = a_{k_2,k_1}$, $p(t_1 = k) = c_k$, $p(w_i = j|t_i = k) = b_{k,j}$.
- **a** and **b** are probability matrices whose columns sum to one.
- Collecting like terms

$$\prod_k c_k^{S_k} \prod_{k_1,k_2} a_{k_1,k_2}^{T_{k_1,k_2}} \prod_{k,j} b_{k,j}^{W_{k,j}}$$

where T_{k_1,k_2} is count of times tag k_2 follows tag k_1, and
$W_{k,j}$ is count of times tag k assigned to word j, and
S_k is count of times sentence starts with tag k.

Fiitting Markov Model with Known Tags, cont.

- Standard maximum likelihood methods apply, so these parameters **a** and **b** become their observed proportions:
  - a_{k_1,k_2} is proportion of tags of type k_2 when previous was k_1,
  - $b_{k,j}$ is proportion of words of type j when tag was k,
- Thus $a_{k_1,k_2} = \frac{T_{k_1,k_2}}{\sum_{k_2} T_{k_1,k_2}}$, $b_{k,j} = \frac{W_{k,j}}{\sum_j W_{k,j}}$, $c_k = \frac{S_k}{\sum_k S_k}$.
- Note we have many sentences in the training data, and each one has a fresh start, so c_k is estimating from all those initial tags in sentences.
- As is standard when dealing with frequencies, we can smooth these out by adding small amounts to the numerator and denominator to make all quantities non-zero.

Comments

- In practice, the naive estimation of **a** and **b** works poorly because we never have enough data. Most words occur infrequently, so we cannot get good tag statistics for them.
- Kupiec (1992) suggested grouping infrequent words together based on their pattern of candidate POS. This overcomes paucity of data with a reasonable compromise.
  - So "red" and "black" can both be NN or JJ, so they belong to the same *ambiguity class*.
  - Ambiguity classes not used for frequent words.
- Unknown words are also a problem. A first approximation is to assign unknown words with first capitals to NP.

Estimating Tags for New Text

- We now fix the Markov model parameters $\mathbf{a}$, $\mathbf{b}$ and $\vec{c}$.
- We have a new sentence with I words $w_1, w_2, ..., w_I$. How do we estimate its tag set?
- We ignore the lexical contraints for now (*e.g.*, "interest" is VB, VBZ or NNS), and fold them in later.
- Task so described is:

$$\vec{t} \;=\; \text{argmax}_{\vec{t}}\; p\left(\vec{t}, \vec{w} \,\middle|\, \mathbf{a}, \mathbf{b}, \vec{c}\right)$$

where the probability is as before.

Estimating Tags for New Text, cont.

Wish to solve

$$\operatorname{argmax}_{\vec{t}}\ p(t_1) \prod_{i=2,\dots,I} p(t_i|t_{i-1}) \prod_{i=1,\dots,I} p(w_i|t_i)$$

The task is simplified by the fact that knowing the value for tag t_N splits the problem neatly into parts, so define

$$\begin{aligned} m(t_1) &= p(t_1) \\ m(t_N) &= \max_{t_1,\dots,t_{N-1}|t_N}\ p(t_1) \prod_{i=2,\dots,N} p(t_i|t_{i-1}) \prod_{i=1,\dots,N-1} p(w_i|t_i) \end{aligned}$$

We get the recursion for $m(t_{N+1})$:

$$\begin{aligned} &= \max_{t_1,\dots,t_N|t_{N+1}}\ p(t_1) \prod_{i=2,\dots,N+1} p(t_i|t_{i-1}) \prod_{i=1,\dots,N} p(w_i|t_i) \\ &= \max_{t_N|t_{N+1}} \max_{t_1,\dots,t_{N-1}|t_N,t_{N+1}}\ p(t_1) \prod_{i=2,\dots,N+1} p(t_i|t_{i-1}) \prod_{i=1,\dots,N} p(w_i|t_i) \\ &= \max_{t_N}\quad p(t_{N+1}|t_N) p(w_N|t_N) m(t_N) \end{aligned}$$

Estimating Tags for New Text, cont.

We apply this incrementally, building up a contingent solution from left to right. This is called the **Viterbi algorithm**, first developed in 1967.

1. Initialise $m(t_1)$, $m(t_1 = k) = c_k$.
2. For $i = 2, ..., I$, compute $m(t_i)$,

$$m(t_i = k_1) \;=\; \max_{k_2} \left(a_{k_2,k_1} b_{k_2,w_N} m(t_{i-1} = k_2) \right)$$

   then store the backtrace, the k_2 that achieves maximum for each $t_i = k_1$.
3. At the end, I, find the maximum $t_I = \text{argmax}_k m(t_I = k)$, and chain through the backtraces to get the maximum sequence for $t_1, \ldots, t_I$.

This technique is an example of dynamic programming.

Comments

- What about lexical contraints, *e.g.*, our dictionary tells us that "interest" is either VB, VBZ or NNS?
- Thus $p(w_i = \text{'interest'} \mid t_i = \text{'JJS'}) = 0$.
- Thus we would like to enforce zeros in some entries of the **b** matrix.
- Likewise, with the ambiguity classes above, and with the individual words, we just assign some zero's to $b_{k,j}$ for j the index of the word.

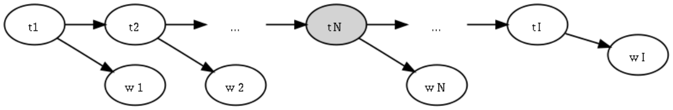
Estimating Tag Probabilities

- We again fix the Markov model parameters **a**, **b** and $\vec{c}$.
- We have a new sentence with I words $w_1, w_2, ..., w_I$. We've got the most likely tag set using the Viterbi algorithm. What's the uncertainty here?
- Task can be described as: find the tag probabilities for each t_N.

$$p(t_N|\vec{w}) \;\propto\; \sum_{\vec{t}/t_N} p\left(\vec{t}, \vec{w} \,|\, \mathbf{a}, \mathbf{b}, \vec{c}\right)$$

where the probability is as before.

Estimating Tag Probabilities, cont.



Wish to compute $p(t_N|\vec{w})$, got by normalising

$$p(t_N,\vec{w}) = \sum_{\vec{t}/t_N} \left(p(t_1) \prod_{i=2,\ldots,I} p(t_i|t_{i-1}) \prod_{i=1,\ldots,I} p(w_i|t_i) \right)$$

Note we have:

$$p(t_N|w_1,\ldots,w_{N-1}) = \sum_{t_1,\ldots,t_{N-1}} \left(p(t_1) \prod_{i=2,\ldots,N} p(t_i|t_{i-1}) \prod_{i=1,\ldots,N-1} p(w_i|t_i) \right)$$

$$p(w_{N+1},\ldots,w_I|t_N) = \sum_{t_{N+1},\ldots,t_I} \left(\prod_{i=N+1,\ldots,I} p(t_i|t_{i-1}) \prod_{i=N+1,\ldots,I} p(w_i|t_i) \right)$$

$$\text{Thus} \quad p(t_N,\vec{w}) = p(t_N|w_1,\ldots,w_{N-1})\,p(w_{N+1},\ldots,w_I|t_N)\,p(w_N|t_N)$$

Estimating Tag Probabilities, cont.

The quantities $p(t_N|w_1,...,w_{N-1})$ and $p(w_{N+1},...,w_I|t_N)$ are traditionally called $\alpha(t_N)$ and $\beta(t_N)$ respectively.

As with the Viterbi, a recursion exists:

$$\begin{aligned} p(t_N|w_1,...,w_{N-1}) &= \sum_{t_{N-1}} p(t_N|t_{N-1})p(w_{N-1}|t_{N-1})p(t_{N-1}|w_1,...,w_{N-2}) \\ p(w_{N+1},...,w_I|t_N) &= \sum_{t_{N+1}} p(t_{N+1}|t_N)p(w_{N+1}|t_{N+1})p(w_{N+2},...,w_I|t_{N+1}) \end{aligned}$$

Compute the first with a forward pass in N, compute the second with a backward pass in N. Hence computing these probabilities is called the *Forward-Backward* algorithm. Complexity is $O(I\,K^2)$.

$$\begin{aligned} \alpha_N(k_1) &= \sum_{k_2} a_{k_2,k_1} b_{k_2,w_{N-1}} \alpha_{N-1}(k_2) \\ \beta_N(k_1) &= \sum_{k_2} a_{k_1,k_2} b_{k_2,w_{N+1}} \beta_{N+1}(k_2) \\ \alpha_1(k) &= c_k \qquad \beta_I(k) = 1 \end{aligned}$$

Outline

1. Part-of-Speech with Hidden Markov Models
   - Markov Model
   - Hidden Markov Model
2. Topics in Text with Discrete Component Analysis

Fitting with Unknown Tags

- We don't always have a large quantity of text tagged with POS. So we would like to try and improve the estimates of the model using untagged or partially tagged data.
- So the problem becomes, estimate $\mathbf{a}$, $\mathbf{b}$ and $\vec{c}$ given the sequence $w_1, w_2, ..., w_I$ but no tags.
- The case with partial tags can be folded in later.
- This problem, where the tags are unknown initially is called a *hidden Markov model* (HMM).

A Little Bit of Magic

- We will use some probability function $q(\vec{t})$ in our solution as a device. This represents *some* valid probability over the tags. NB. it can be represented by a large parameter vector.
- For brevity, refer to **a**, **b** and $\vec{c}$ by a single parameter vector $\vec{\theta}$.
- Consider the function $Q(\vec{\theta}, q())$ given by

$$
\begin{aligned}
&= \log p(\vec{w}|\vec{\theta}) - KL\left(q(\vec{t}) \,||\, p(\vec{t}|\vec{w},\vec{\theta})\right) \\
&= E_{q(\vec{t})}\left(\log p(\vec{t},\vec{w}|\vec{\theta})\right) + I(q(\vec{t}))
\end{aligned}
$$

- A simple expansion of $KL()$ and $I()$ shows the two forms are equal.

A Little Bit of Magic, cont.

- Consider the function $Q(\vec{\theta}, q())$ given by

$$
\begin{aligned}
&= \log p(\vec{w}|\vec{\theta}) - KL\left(q(\vec{t})\,||\,p(\vec{t}|\vec{w},\vec{\theta})\right) \\
&= E_{q(\vec{t})}\left(\log p(\vec{t},\vec{w}|\vec{\theta})\right) + I(q(\vec{t}))
\end{aligned}
$$

- Maximise this w.r.t. $\vec{\theta}$ and $q()$ jointly.
- By the first equation, this holds when $q(\vec{t}) = p(\vec{t}|\vec{w},\vec{\theta})$, and then $Q(\vec{\theta}, q()) = \log p(\vec{w}|\vec{\theta})$.
- By the second equation, this holds if we solve:

$$
\text{argmax}_{\vec{\theta}}\, E_{q(\vec{t})}\left(\log p(\vec{t},\vec{w}|\vec{\theta})\right) \; .
$$

- Thus, iterating these two steps will achieve the maximum likelihood solution $\text{argmax}_{\vec{\theta}}\, \log p(\vec{w}|\vec{\theta})$.

Fitting with Unknown Tags, cont.

The "conceptual" algorithm is to repeatedly re-estimate $\vec{\theta}$.

1. Construct the intermediate distribution $q(\vec{t}) = p(\vec{t}|\vec{w},\vec{\theta})$ from the current $\vec{\theta}$.

   $\longrightarrow$ This maximizes $\left(\log p(\vec{w}|\vec{\theta}) - KL\left(q(\vec{t})\,||\,p(\vec{t}|\vec{w},\vec{\theta})\right)\right)$ w.r.t. $q()$.

2. Use this to evaluate $C(\vec{\theta}) \;=\; E_{q(\vec{t})}\left(\log p(\vec{t},\vec{w}|\vec{\theta})\right)$.

3. Now re-maximise $\vec{\theta}' \;=\; \text{argmax}_{\vec{\theta}}\ C(\vec{\theta})$.

   $\longrightarrow$ This maximizes $\left(E_{q(\vec{t})}\left(\log p(\vec{t},\vec{w}|\vec{\theta})\right) + I(q(\vec{t}))\right)$ w.r.t. $\vec{\theta}$.

This is called the *Expectation-Maximization algorithm*, or EM for short.
It works efficiently when can get the formula for the steps of the conceptual algorithm.

Revision

We wish to evaluate $E_{q(\vec{t})}\left(\log p(\vec{t},\vec{w}|\vec{\theta})\right)$, where:

$$p(\vec{t},\vec{w}|\vec{\theta}) \;=\; \prod_k c_k^{S_k} \prod_{k_1,k_2} a_{k_1,k_2}^{T_{k_1,k_2}} \prod_{k,j} b_{k,j}^{W_{k,j}} \;,$$

where T_{k_1,k_2} is count of times tag k_2 follows tag k_1, and
$W_{k,j}$ is count of times tag k assigned to word j, and
S_k is count of times sentence starts with tag k.

T_{k_1,k_2}, $W_{k,j}$ and S_k are statistics for the tags $\vec{t}$ given words $\vec{w}$.

Fitting with Unknown Tags, cont.

Linearity of expectation gives $E_{q(\vec{t})}\left(\log p(\vec{t}, \vec{w}|\vec{\theta})\right)$

$$
\begin{aligned}
&= E_{q(\vec{t})}\left(\sum_k \log c_k^{S_k} + \sum_{k_1,k_2} \log a_{k_1,k_2}^{T_{k_1,k_2}} + \sum_{k,j} \log b_{k,j}^{W_{k,j}}\right) \\
&= \sum_k E_{q(\vec{t})}(S_k) \log c_k + \sum_{k_1,k_2} E_{q(\vec{t})}(T_{k_1,k_2}) \log a_{k_1,k_2} + \sum_{k,j} E_{q(\vec{t})}(W_{k,j}) \log b_{k,j} ,
\end{aligned}
$$

where the expected values are given by:

$$
\begin{aligned}
E_{q(\vec{t})}(S_k) &= p(t_0{=}k|q(\vec{t})) \\
E_{q(\vec{t})}(T_{k_1,k_2}) &= \sum_i p(t_{i+1}{=}k_2|t_i{=}k_1, q(\vec{t})) \\
E_{q(\vec{t})}(W_{k,j}) &= \sum_i 1_{w_i=j}\, p(t_i{=}k|q(\vec{t}))
\end{aligned}
$$

Fitting with Unknown Tags, cont.

Maximising

$$\sum_k E_{q(\vec{t})}(S_k) \log c_k + \sum_{k_1,k_2} E_{q(\vec{t})}(T_{k_1,k_2}) \log a_{k_1,k_2} + \sum_{k,j} E_{q(\vec{t})}(W_{k,j}) \log b_{k,j}$$

w.r.t. the probability matrices and vectors **a**, **b** and $\vec{c}$ is a standard constrained optimisation problem. Remember the columns of **a**, **b** must add to one.

The solution is:

$$\begin{aligned} c_k &\propto E_{q(\vec{t})}(S_k) \\ a_{k_1,k_2} &\propto E_{q(\vec{t})}(T_{k_1,k_2}) \\ b_{k,j} &\propto E_{q(\vec{t})}(W_{k,j}) \end{aligned}$$

Baum-Welch Algorithm

Putting it all together.

1. From the current solution for **a**, **b** and $\vec{c}$, perform the Forward-Backward algorithm to compute $\alpha_N(\cdot)$ and $\beta_N(\cdot)$.
2. From these, compute

$$p(t_N{=}k|q()) \ \propto \ \alpha_N(k)\beta_N(k)b_{k,W_N} \ ,$$

$$p(t_N{=}k_2|t_{N-1}{=}k_1, q()) \ \propto \ \alpha_{N-1}(k_1)\beta_N(k_2)b_{k_1,W_{N-1}}b_{k_2,W_N}a_{k_1,k_2} \ .$$

3. Hence compute $E_{q(\vec{t})}(S_k)$, $E_{q(\vec{t})}(T_{k_1,k_2})$ and $E_{q(\vec{t})}(W_{k,j})$ using formula on previous page.
4. Now maximise for **a**, **b** and $\vec{c}$ using the proportions on the previous page.

Comments

- This is called the Baum-Welch algorithm, after the original inventors. It is an instance of the so-called EM algorithm.
- Unfortunately this HMM training doesn't work too well for the POS problem. Although, it was for a long time the best method for speech to text recognition.
- Perhaps the poor performance on POS tagging is because we are fitting a joint model $p(\vec{t}, \vec{x}|\vec{\theta})$ rather than a conditional model $p(\vec{t}|\vec{x}, \vec{\theta})$.
- So lets investigate conditional models.

Conditional Fitting with Unknown Tags

- So the problem is to estimate a model for $\vec{t}$ given the sequence $w_1, w_2, ..., w_I$ but no tags.
- We no longer have $p(w_i|t_i)$, rather we want a discriminative model, something like $p(t_i|w_i)$, but also $p(t_i|t_{i-1})$,
- One approach, called the conditional random field (CRF) is to fold them in together to get:

$$p(\vec{t}\,|\,\vec{w}, \mathbf{a},\, \mathbf{b}, \vec{c}) \;\propto\; \exp\left(\sum_i a_{t_{i-1},t_i} + \sum_i b_{t_i,w_i} + \sum_i c_{t_i}\right) \;.$$

- Compare this conditional model with our HMM model, which can be manipulated to

$$p(\vec{t}, \vec{w}|\mathbf{a},\, \mathbf{b}, \vec{c}) \;=\; \exp\left(\sum_i a_{t_{i-1},t_i} + \sum_i b_{t_i,w_i} + \sum_i c_{t_i}\right) \;.$$

Conditional Fitting with Known Tags, cont.

- The so-called *conditional random field* has:

$$p(\vec{t} \,|\, \vec{w}, \mathbf{a}, \mathbf{b}, \vec{c}) \;\propto\; \exp\left(\sum_i a_{t_{i-1},t_i} + \sum_i b_{t_i,w_i} + \sum_i c_{t_i} \right)$$

- We need a normalising constant, Z, a function of $\mathbf{a}$, $\mathbf{b}$ and $\vec{c}$.

$$Z \;=\; \sum_{\vec{t}} p(\vec{t} \,|\, \vec{w}, \mathbf{a}, \mathbf{b}, \vec{c})$$

- Compute this incrementally, rather like a forward pass of the Forward-Backward algorithm.

$$\begin{aligned} Z_1(t_1) &= 1 \\ Z_N(t_N) &= \sum_{t_{N-1}} Z_{N-1}(t_{N-1}) \exp\left(a_{t_{N-1},t_N} + b_{t_{N-1},w_{N-1}} + c_{t_{N-1}}\right) \\ Z &= \sum_{t_N} Z_N(t_N) \exp\left(b_{t_N,w_N} + c_{t_N}\right) \end{aligned}$$

Conditional Fitting with Known Tags, cont.

We have to use gradient based algorithms to fit this as there are no closed forms.

- Lets look at the likelihood to maximise $\log p(\vec{t}|\vec{w},\vec{\theta})$.

$$\sum_k S_k c_k + \sum_{k_1,k_2} T_{k_1,k_2} a_{k_1,k_2} + \sum_{k,j} W_{k,j} b_{k,j} - \log Z$$

- Note **a**, **b** and $\vec{c}$ are no longer probability matrices and vectors.
- Now it happens that

$$\begin{aligned} \frac{\partial \log Z}{\partial a_{k_1,k_2}} &= E_{p(\vec{t}|\vec{w},a,b,\vec{c})}\left(T_{k_1,k_2}\right) , \\ \frac{\partial \log Z}{\partial b_{k,j}} &= E_{p(\vec{t}|\vec{w},a,b,\vec{c})}\left(W_{k,j}\right) . \end{aligned}$$

  These expected values can be computed by a variant of the forward-backward algorithm, as before.

- Thus we have all the derivatives of the likelihood.

Comments

- Training slower than for a HMM.
- Conditional training with some unknown tags also works, but is more complicated again.
- In principle. you can now use any features, not just the words $\vec{w}$. People use:
  - capitalisation, all-caps, use of non-alphabetic letters,
  - presence of prefixes and suffixes,
  - properties of surrounding words,
  - match of words to different gazetteers.
- In this case, the performance is very dependent on the choice of features!

Outline

Methods for discovering hiiden components or topics in semi-structured data.
Reference: Buntine and Jakulin, 2006.

Outline

Motivation

- Web industry players are exploring the use of topic models for text. *e.g.*, Microsoft, Yahoo, various startups.
- Large amounts of text in different context available (blogs, news, corporate, Wikipedia, language, ...).
- Current processing performance is of the order of one million documents on a multi-core system in a few days.

Motivation

- We start with a collection of documents in some area.
- We'd like to discover the topics in the collection automatically, using *unsupervised learning*.
- A document is modelled as having multiple topics, for instance one sports article can have three component topics: Argentina, Soccer, and Crowd Behaviour.
- A topic is modelled as a set of words that frequently occur together.

Viewing Topics at the Word Level (Blei, Ng, and Jordan, 2003)

"Arts"	"Budgets"	"Children"	"Education"
NEW	MILLION	CHILDREN	SCHOOL
FILM	TAX	WOMEN	STUDENTS
SHOW	PROGRAM	PEOPLE	SCHOOLS
MUSIC	BUDGET	CHILD	EDUCATION
MOVIE	BILLION	YEARS	TEACHERS
PLAY	FEDERAL	FAMILIES	HIGH
MUSICAL	YEAR	WORK	PUBLIC
BEST	SPENDING	PARENTS	TEACHER
ACTOR	NEW	SAYS	BENNETT
FIRST	STATE	FAMILY	MANIGAT
YORK	PLAN	WELFARE	NAMPHY
OPERA	MONEY	MEN	STATE
THEATER	PROGRAMS	PERCENT	PRESIDENT
ACTRESS	GOVERNMENT	CARE	ELEMENTARY
LOVE	CONGRESS	LIFE	HAITI

The William Randolph Hearst Foundation will give $1.25 million to Lincoln Center, Metropolitan Opera Co., New York Philharmonic and Juilliard School. "Our board felt that we had a real opportunity to make a mark on the future of the performing arts with these grants an act every bit as important as our traditional areas of support in health, medical research, education and the social services," Hearst Foundation President Randolph A. Hearst said Monday in announcing the grants. Lincoln Center's share will be $200,000 for its new building, which will house young artists and provide new public facilities. The Metropolitan Opera Co. and New York Philharmonic will receive $400,000 each. The Juilliard School, where music and the performing arts are taught, will get $250,000. The Hearst Foundation, a leading supporter of the Lincoln Center Consolidated Corporate Fund, will make its usual annual $100,000 donation, too.

Figure 8: An example article from the AP corpus. Each color codes a different factor from which the word is putatively generated.

Example: Topics in the Wikipedia

- We take 1 million documents from the Wikipedia, and tokenise the text in each document, without linguistic processing.
- This yields about half a gigabyte of binary data.
- We train the topic models and then look at the topics.

Example Topic: Mythology

NOUNS					
mythology	0.03337	God	0.02048	name	0.014747
goddess	0.012911	spirit	0.012639	legend	0.0087992
myth	0.0070882	demons	0.006807	Sun	0.0060099
Temple	0.0054717	deity	0.0054247	Bull	0.0051629
Dragon	0.0051379	Maya	0.0051243	King	0.00512
Sea	0.0049453	Norse	0.0044707	horse	0.0044592
symbol	0.0042196	animals	0.0040112	fire	0.0039879
hero	0.0038755	Romans	0.0038696	Apollo	0.0037588
VERBS					
called	0.034078	said	0.031081	see	0.029521
given	0.0269	associated	0.024591	according	0.021724
represented	0.020964	known	0.018896	could	0.017499
made	0.016952	depicted	0.01524	appeared	0.014662
ADJECTIVES					
Greek	0.091163	ancient	0.055393	great	0.02853
Egyptian	0.028071	Roman	0.025783	sacred	0.020446

Historical Background

Long history of component models before the discrete topics models we consider:

- Principal Components Analysis (PCA), dimensionality reduction tool, invented by Karl Pearson in 1901, theoretical relationship to least squares and Gaussians.
- Independent Components Analysis (ICA), invented by Herault and Jutten in 1986, for *blind source separation* of image and signal data, usually used with PCA.
- Latent Semantic Indexing (LSI), intended for text in IR, but of mixed benefit, and difficult to interpret, a variant of PCA.

Gaussian and least squares models fail for the smaller counts data we are considering. Need Poisson or multinomial modelling instead.

Discrete Topic Models, a Short History

- Soft clustering, "grade of membership", Woodbury & Manton, 1982.
- Admixture modelling in statistics, 1980s.
- Hidden facets in image interpretation, Non-negative Matrix Factorization (NMF), Seung and Lee, 1999.
- Probabilistic Latent Semantic Analysis (PLSI), topics in text, Hofmann, 1999.
- Admixture modelling, fully Bayesian, population structure from genotype data, Pritchard, Stephens and Donnelly, 2000.
- Latent Dirichlet Allocation (LDA) Blei, Ng and Jordan, 2001. Variant of Pritchard *et al.* Introduced mean-field algorithm.
- Collapsed Gibbs sampler, Griffiths and Steyvers, 2004.
- Gamma-Poisson model (GaP), Canny 2004 (extension of NMF).

... variants, extensions, adaptations, ..., 2001-2008

Bag of words to represent text

A page out of Dr. Zeuss's *The Cat in The Hat*:

> *So, as fast as I could, I went after my net. And I said, "With my net I can bet them I bet, I bet, with my net, I can get those Things yet!"*

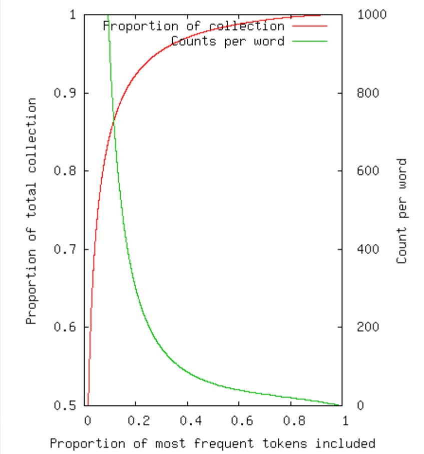
In the *bag of words* representation as *word* (*count*):

> *after(1) and(1) as(2) bet(3) can(2) could(1) fast(1) get(1) I(7) my(3) net(3) said(1) so(1) them(1) things(1) those(1) went(1) with(2) yet(1)* .

Notes:

- For the Reuters RCV1 collection from 2000: $I \approx 800k$ documents, $J \approx 400k$ different words (excluding those occurring few times), $S \approx 300M$ words total.
- Represent as sparse matrix/vector form with integer entries.
- Compresses to about 2 bytes per token (*e.g.* $2S$ bytes) total storage.

Document-word tradeoffs



Data from NY Times collection from UCI.

- Deleting about 50% of the most infrequent words from the dictionary decreases the collection size by only about 3%.
- We can train on a subset of the dictionary as a way of boot-strapping.
- Shows that compression of various word matrices and vectors can be significant.
- Should also ignore words occurring in, say, 30% or more of documents as "stop" words.

Issues in text representation

- The basic semantic units in text are not words but, most commonly, compound words.
  - *e.g.*, "New York Times", "George Bush"
  - most common are single words.
  - occasionally compound words are *not* contiguous.
- Web pages full of "cruft", HTML junk, adverts, company fluff, navigation aids, boilerplate, ...
- Different "styles" of topics exists:

  genre: *e.g.*, product page, blog, news, corporate info.,

  library style categorisation: as done by Dewey Decimal, and DMOZ

  opinion and sentiment: *e.g.*, positive, anti-Microsoft, "green", ...

Outline

Basic Model

- Everything tokenised, so have I documents, J words in the dictionary, K different topics/components.
- The model of how frequent words are for a topic given by the *topic by word* matrix of proportions, Θ of dimension $J \times K$.
- The model of how topics are distributed in a given document given by a Dirichlet of dimension K with parameters $\vec{\alpha}$.
- For a given document i, we'll sample the topics proportions, a *latent or hidden* variable $\vec{m}_i$ as

$$\vec{m}_i \ \sim \ \text{Dirichlet}_K(\vec{\alpha})$$

- Words in a document i generated independently, proportion given by J-dimensional vector $\vec{m}_i^\dagger \Theta$. For sequence $l = 1, ..., L$

$$p(j_l \,|\, \Theta, \vec{m}_i) \ = \ \sum_k m_{i,k} \theta_{k,j_l} \ .$$

Basic Model

The model has the following *parameters*:

K: number of topics,

$\vec{\alpha}$: used to generate topics for each document,

Θ: word proportions for each topic.

The *sampling model* acts as follows:

1. For each document indexed by i:
   1. Generate the topic proportions for the document $\vec{m}_i \sim \text{Dirichlet}_K(\vec{\alpha})$.
   2. For each word indexed by l in the document i:
      1. generate the topic of the word $k_l \sim \text{Discrete}_K(\vec{m}_i)$,
      2. take the k_l-th column from Θ, which is $\vec{\theta}_{k_l}$, generate the word $j_l \sim \text{Discrete}_J(\vec{\theta}_{k_l})$.

Each document has *hidden (or latent) variables* $\vec{m}_i$ and $\vec{k}_i$.

Revision

The K-dimensional Dirichlet distribution is a function on a proportions $\vec{m}$, of the form

$$p(\vec{m} \,|\, \vec{\alpha}, K, \text{Dirichlet}) \;=\; \frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_k^{\alpha_k - 1} \;.$$

The normalising constant $Z_K(\vec{\alpha})$ evaluates as

$$\prod_k \Gamma(\alpha_k) \bigg/ \Gamma\left(\sum_k \alpha_k\right)$$

Means are given by

$$\text{E}(m_k) \;=\; \alpha_k \bigg/ \sum_k \alpha_k$$

Document Likelihoods

The likelihood including all the latent variables, $p(\vec{j}_i, \vec{k}_i, \vec{m}_i \,|\, \text{for doc } i, \Theta, \alpha)$:

$$\left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \prod_{l \in WordSequence_i} m_{i,k_{i,l}} \theta_{k_{i,l}, j_{i,l}} \;.$$

Marginalising out the latent topic assignments, $\vec{k}_i$, giving $p(\vec{j}_i, \vec{m}_i \,|\, \text{for doc } i, \Theta, \alpha)$:

$$\left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \prod_{l \in WordSequence_i} \sum_{k \in Topics} m_{i,k} \theta_{k, j_{i,l}} \;.$$

Marginalising out instead the topic proportions $\vec{m}_i$, giving $p(\vec{j}_i, \vec{k}_i \,|\, \text{for doc } i, \Theta, \alpha)$, where $C_{i,k}$ is the count of topic k in document i,

$$\frac{Z_K(\vec{\alpha} + \vec{C}_i)}{Z_K(\vec{\alpha})} \prod_{l \in WordSequence_i} \theta_{k_{i,l}, j_{i,l}} \;.$$

Estimating Test Likelihoods

- The likelihood we would like to report in testing is, $p(\vec{j}_i \,|\, \text{for doc } i, \Theta, \alpha)$. We have no computable form for this.
- The previous likelihoods are almost certainly bad over-estimates, unless the latent/hidden variables used in evaluating them or sampled uniformly, in which case they are very poor estimates and useless.
- If we sample the topic assignments $\vec{k}_i$ proportionally too $p(\vec{j}, \vec{k} \,|\, \text{for doc } i, \Theta, \alpha)$, the document likelihood can be approximated as

$$\frac{1}{N} \Big/ \sum_{n=1}^{N} \frac{1}{\prod_{l \in WordSequence_i} \theta_{k_{n,l}, j_l}}$$

where we have N sample vectors $\vec{k}_n$.

See (Carlin and Chib, 1995).

Variational EM Algorithm: Rough Outline

Seeks to maximise the likelihood, $p(\vec{j}_i, \vec{m}_i \text{ for } i \in Docs\,|\Theta, \alpha)$, where $\vec{j}_i$ are the words for a document, $Z_k()$ is Dirichlet normaliser:

$$\prod_{i \in Docs} \left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \left(\prod_{l \in WordSequence} \sum_{k \in Topics} m_{i,k} \theta_{k,j_{i,l}} \right) .$$

Typically consists of a few hundred cycles in the form

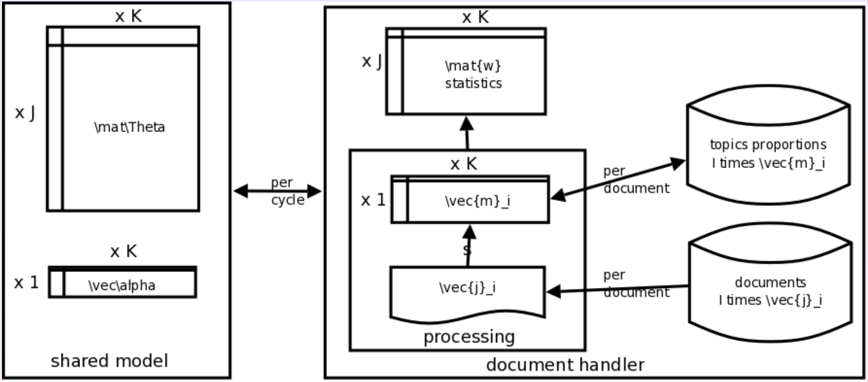
1. For each document i, re-estimate/improve values for $\vec{m}_i$, based on the factored approximation

$$\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \left(\prod_{l \in WordSequence} \prod_{k \in Topics} m_{i,k}^{\mu_{k,l}} \right) .$$

2. Re-assign values for Θ based on statistics collected in step (1), based on the factored approximation

$$\prod_i \prod_l \prod_k \theta_{k,j_{i,l}}^{m_{i,k}} .$$

Parallel Variational EM



Parallel Variational EM, notes

- Distribute documents to different document handlers.
- The documents $\vec{j}_i$, and the document proportions $\vec{m}_i$ can be streamed, so are not a significant memory cost.
- $\vec{m}_i$ will need to be compressed when K is large.
- Need to communicate Θ and α with each major cycle: collect statistics, then distribute update; efficient primitives should be used for communiciation.

Collapsed Gibbs Algorithm: Derivation

Take the likelihood, $p(\vec{j}_i, \vec{m}_i \text{ for } i \in Docs \,|\Theta, \alpha)$:

$$\prod_{i \in Docs} \left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \left(\prod_{l \in WordSequence_i} \sum_{k \in Topics} m_{i,k} \theta_{k,j_{i,l}} \right) .$$

Introduce the topics per word, $p(\vec{j}_i, \vec{k}_i, \vec{m}_i \text{ for } i \in Docs \,|\Theta, \alpha)$

$$\prod_{i \in Docs} \left(\frac{1}{Z_K(\vec{\alpha})} \prod_{k \in Topics} m_{i,k}^{\alpha_k - 1} \right) \left(\prod_{l \in WordSequence_i} m_{i,k_{i,l}} \theta_{k_{i,l},j_{i,l}} \right) .$$

Collect terms in Θ and $\vec{m}_i$, with statistics $\vec{W}$ and $\vec{C}_i$ respectively, and integrate/marginalise $\vec{m}_i$, giving $p(\vec{j}_i, \vec{k}_i \text{ for } i \in Docs \,|\Theta, \alpha)$

$$\prod_{k \in Topics} \prod_{j \in Words} \theta_{k,j}^{W_{k,j}} \prod_{i \in Docs} \frac{Z_K(\vec{\alpha} + \vec{C}_i)}{Z_K(\vec{\alpha})} .$$

Collapsed Gibbs Algorithm: Derivation, cont.

Finally, integrate/marginalise Θ (by adding prior for Θ of $\vec{\gamma}$)
$p(\vec{j}_i, \vec{k}_i \text{ for } i \in Docs \,|\vec{\alpha}, \vec{\gamma})$

$$\prod_{k \in Topics} \frac{Z_J(\vec{\gamma} + \vec{W}_k)}{Z_J(\vec{\gamma})} \prod_{i \in Docs} \frac{Z_K(\vec{\alpha} + \vec{C}_i)}{Z_K(\vec{\alpha})} \,.$$

Substituting the normalising constant $Z(\cdot)$ yields

$$\prod_{k \in Topics} \left(\frac{\Gamma\left(\sum_j \gamma_j\right)}{\Gamma\left(\sum_j (\gamma_j + W_{k,j})\right)} \prod_{j \in Words} \frac{\Gamma(\gamma_j + W_{k,j})}{\Gamma(\gamma_j)} \right)$$

$$\prod_{i \in Docs} \left(\frac{\Gamma\left(\sum_k \alpha_k\right)}{\Gamma\left(\sum_k (\alpha_k + C_{i,k})\right)} \prod_{k \in Topics} \frac{\Gamma(\alpha_k + C_{i,k})}{\Gamma(\alpha_k)} \right)$$

Collapsed Gibbs Algorithm: Rough Outline

Probability $p(\vec{k}_i \text{ for } i \in Docs \,|\vec{j}_i \text{ for } i \in Docs, \vec{\alpha}, \vec{\gamma})$

$$\propto \prod_{k \in Topics} \left(\frac{\Gamma\left(\sum_j \gamma_j\right)}{\Gamma\left(\sum_j \left(\gamma_j + W_{k,j}\right)\right)} \prod_{j \in Words} \frac{\Gamma(\gamma_j + W_{k,j})}{\Gamma(\gamma_j)} \right)$$

$$\prod_{i \in Docs} \left(\frac{\Gamma\left(\sum_k \alpha_k\right)}{\Gamma\left(\sum_k \left(\alpha_k + C_{i,k}\right)\right)} \prod_{k \in Topics} \frac{\Gamma(\alpha_k + C_{i,k})}{\Gamma(\alpha_k)} \right)$$

Change the topic assignment for one word, $k_{i,l}$, gives simple product formula for a Gibbs update on $k_{i,l}$. See Griffiths and Steyvers 2004.

$$p(k_{i,l} = k \,|\, j_{i,l} = j, \vec{W}, \vec{C}, \vec{\alpha}, \vec{\gamma}) \;\propto\; (C_{i,k} + \alpha_k) \frac{W_{k,j} + \gamma_j}{\sum_j \left(W_{k,j} + \gamma_j\right)}$$

where $\vec{C}_i$ is the topic totals for document i, and $\vec{W}$ is the topic totals by word.

Collapsed Gibbs Algorithm: Rough Outline, cont.

The formula for a Gibbs update on $k_{i,l}$:

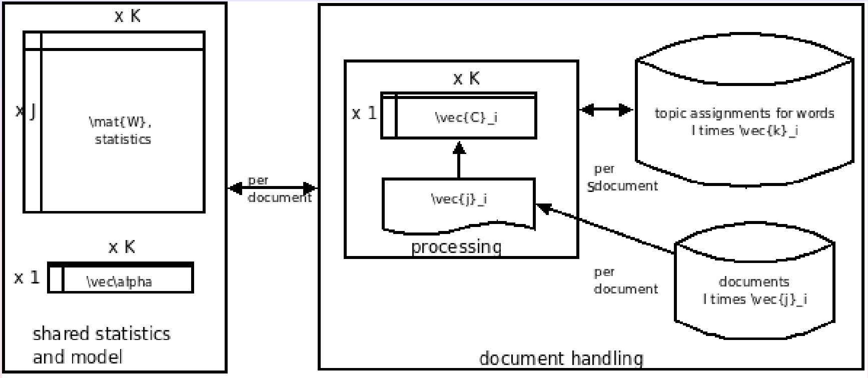
$$p(k_{i,l} = k \,|\, j_{i,l} = j, \vec{W}, \vec{C}, \vec{\alpha}, \vec{\gamma}) \;\propto\; (C_{i,k} + \alpha_k) \frac{W_{k,j} + \gamma_j}{\sum_j (W_{k,j} + \gamma_j)}$$

where $\vec{C}_i$ is the topic totals for document i, and $\vec{W}$ is the topic totals by word.

Algorithm consists of, say, thousand cycles in the form:

1. For each document i,
    1. Recompute topic totals $\vec{C}_i$ from stored topic assignments $\vec{k}_i$.
    2. For sequence $l = 1, ..., L$ in document,
        1. for word $j_{i,l}$, re-sample its topic assignment $k_{i,l}$ using statistics $\vec{W}$ and $\vec{C}_i$,
        2. update $\vec{W}$ and $\vec{C}_i$.

Parallel Collapsed Gibbs EM



Parallel Collapsed Gibbs, notes

- Distribute documents to different document handlers.
- The documents $\vec{j}_i$, and their topic assignment $\vec{k}_i$ can be streamed, again. Both about the same size.
- Need to update statistics W continuously! Best batch it, find the difference, compress, and communicate.
- W compressible by factor of 2-20, or more if many document handlers involved.

Various papers from UCI group on distributed LDA, and the `ParallelTopicModel.java` code of Mallet, by Mimno and McCallum.

Comments

- Many of the published models, NMF with K-L metric, GaP, LDA and PLSI are all variations of one another if we ignore hyperparameters, and the statistical and optimisation methods used.
  NB. Bregman divergence variations also exist.
- Different kinds of algorithms used: variational EM, maximum likelihood, Gibbs sampling, collapsed Gibbs sampling, ...
- Lots of extensions exist:
  - using N-th (usually 2nd) order Markov models on words,
  - hierarchical extensions,
  - correlated topics, *e.g.*, Pachinko,
  - sparse matrices,
  - time dependent or otherwise conditional topics.