Natural Language Processing (NLP)

Hong, ONG Xuan

November 29, 2018

VEF ACADEMY

Contents

Contents

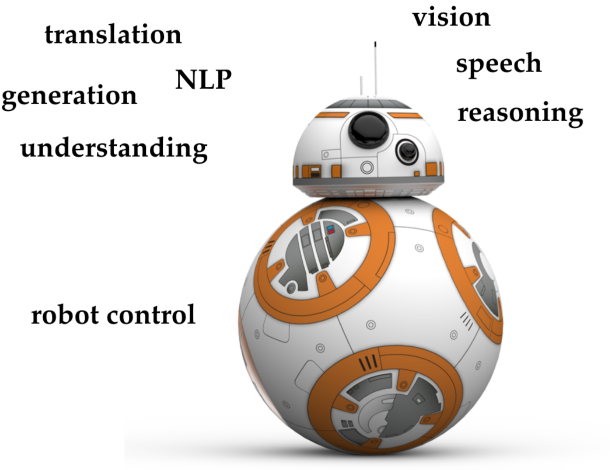

Figure: Android from Star Wars movie.

What is Natural Language Processing

Natural language processing is a field at the intersection of:

- Artificial Intelligence (how to represent knowledge about the world).
- Linguistics (knowledge about language).
- Computer Science (using algorithms for combining knowledge sources).

Goal: "understand" natural language in order to perform tasks that are useful, e.g making appointments, buying things, translation, question answering.

Contents

Simple applications

- Spell checking, keyword search, finding synonyms.
- Extracting information from websites such as product price, dates, location, people or company names.
- Parts-of-speech tagging, name entities recognition, parsing.

Industry applications

- Classifying: spam filtering, sentiment analysis of longer documents for marketing or finance/trading.
- Machine translation.
- Spoken dialog systems: automating customer support, controlling devices, ordering goods.
- Complex question answering.
- Paraphrase, summarization.

Contents

Some brief history

- 1940s - 1950s: Foundation
  - Development of formal language theory (Chomsky, Backus, Naur, Kleene).
  - Probabilities and information theory (Shannon).
- 1957 - 1970s: Pattern-matching with small rule-sets
  - Use of formal grammars as basis for natural language processing (Chomsky, Kaplan).
  - Use of logic and logic based programming (Minsky, Winograd, Colmerauer, Kay).
- 1970s – 1983: Linguistically rich, logic-driven, grounded systems; restricted applications
  - Probabilistic methods for early speech recognition (Jelinek, Mercer)
  - Discourse modeling (Grosz, Sidner, Hobbs)
- 1990s: the statistical revolution in NLP leads to a decrease in NLU work.
- 2010s: NLU returns to center stage, mixing techniques from previous decades areas.

Contents

Contents

NLP levels

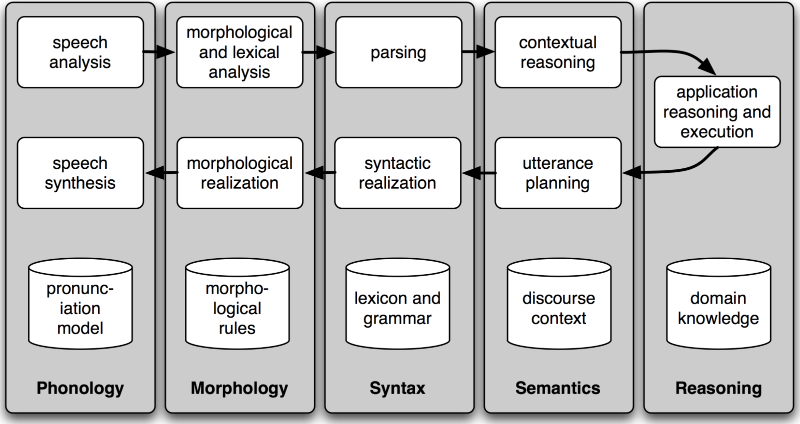

Figure: Simple Pipeline Architecture for a Spoken Dialogue System (nltk Chapter 01)

NLP levels

- Phonetics and Phonology – The study of linguistic **sounds**.
- Morphology – The study of the meaningful components of **words**.
- Syntax – The study of the structural **relationships** between words.
- Semantics – The study of **meaning**.
- Pragmatics – The study of how language is used to accomplish **goals**.
- Discourse – The study of reference to the purposes of **conversation**.

- **Phonetics**: the study of how speech sounds are produced by **articulators** in the mouth.
- **Phonology**: is the area of linguistics that describes the systematic way that sounds are differently realized in different environments, and how this **system of sounds** is related to the rest of the grammar.

Morphology

Concerns how words are constructed from more basic meaning units call morphemes. A morpheme is the primitive **unit of meaning** in language.

Example

uninterested = un + **interest** + ed

cars = **car** + s

giving = **give** + ing

Syntax

If words are the foundation of speech and language processing, syntax is the skeleton. Syntax is the study of formal **relationships between words**.

Example

- The dog bit the boy.
- The boy bit the dog.
- **Bit boy dog the the**.

Semantics

Concerns **what words mean** and how these meaning combine in sentences to form sentence meaning.

Example

- He robbed the **bank** – bank is a financial institution not a river bank
- I want to eat **someplace** that's close to Sai Gon – someplace is location not food.
- I want to eat **Italian food** – Italian stands alone is location.

Pragmatics

Pragmatics is the study of (some parts of) the relation between language and context-of-use. **Context-of-use** includes such things as the identities of people and objects. Context-of-use includes studies of how discourses are structured, and how the listener manages to interpret a conversational partner in a conversation.

Discourse

Concerns how the immediately preceding sentences affect the **interpretation** of the next sentence. For example, interpreting pronouns and interpreting the temporal aspects of the information.

Example

- Marry brought a book for Kelly. **She** didn't like **it**.
  - She refers to Marry or Kelly – possibly Kelly.
  - It refers to book

Contents

Ambiguity

Natural language is extremely rich in form and structure, and very ambiguous.
One input can mean many different things. Ambiguity can be at different levels.

- Lexical (word level) ambiguity – different meanings of words
- Syntactic ambiguity – different ways to parse the sentence
- Interpreting partial information – how to interpret pronouns
- Contextual information – context of the sentence may affect the meaning of that sentence.

Ambiguity example

Some interpretations of : **I made her duck**.

1. I cooked *duck* for her.
2. I cooked *duck* belonging to her.
3. I created a toy *duck* which she owns.
4. I caused her to quickly lower her head or body.
5. I used magic and turned her into a *duck*.

duck – **morphologically** and **syntactically** ambiguous: noun or verb.
her – **syntactically** ambiguous: dative or possessive.
make – **semantically** ambiguous: cook or create.
make – **syntactically** ambiguous:

- Transitive – takes a direct object (exp 2).
- Di-transitive – takes two objects (exp 5).
- Takes a direct object and a verb (exp 4).

Contents

- **Non-standard language**: **(EN)** Great job, SOO PROUD of what youve done! **(VN)** Co' nhung? dieu' that su ko the bik truoc' dc...
- **Segmentation issues**: **(EN)** the New York-New Haven Railroad. **(VN)** Ong gia di nhanh qua ...
- **Idioms**: **(EN)** dark horse, get cold feet, lose face, throw in the towel, **(VN)** quanh di quan lai, cuu kho cuu nan, ...
- **Neologisms**: **(EN)** unfriend, retweet.
- **World knowledge**: Mary and Sue are sisters. Mary and Sue are mothers.
- **Building corpus**: is expensive, need experts in fields for annotation and agreement of humans on the standard.. That's why we call **golden corpus**.

Contents

We could resolve ambiguities at different levels.

- **part-of-speech tagging** – Deciding whether *duck* is verb or noun.
- **word-sense disambiguation** – Deciding whether *make* is create or cook.
- **syntactic ambiguity** – *her duck* is an example of syntactic ambiguity, and can be addressed by probabilistic parsing.

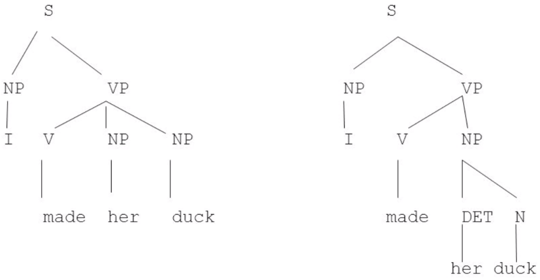

Figure: Example resolve ambiguity by parse-tree.

Rule-based perspective

But it must be recognized that the notion 'probability of a sentence' is an entirely useless one, under any known interpretation of this term. – Noam Chomsky (1969)

Statistical/Corpus-based perspective

Anytime a linguist leaves the group the recognition rate goes up. – Fred Jelinek (then of the IBM speech group) (1988)

Resolve ambiguities

Key theory

- Finite state machines.
- Formal rule systems.
- Probability theory.
- Machine learning tools

Contents

Context-Free Grammar

A context-free grammar (CFG) is a 4-tuple $G = (N, \Sigma, R, S)$ where

- N is a finite set of non-terminal symbols.
- Σ is a finite set of terminal symbols.
- R is a set of rules of the form $X \rightarrow Y_1 Y_2 ... Y_n$ for $n \geq 0, X \in N, Y_i \in (N \cup \Sigma)$.
- $S \in N$ is a distinguished start symbol

CFG - Example

$N = \{S, NP, VP, PP, DT, Vi, Vt, NN, IN\}$
$S = S$
$\Sigma = \{sleeps, saw, man, woman, dog, telescope, the, with, in\}$
$R =$

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

Note: S = Sentence, VP = verb phrase, NP = noun phrase, PP = prepositional phrase, DT = determiner, Vi = intransitive verb, Vt = transitive verb, NN = noun, IN = preposition. The set Σ is the set of possible words in the language.

(Left-most) Derivations

Given a context-free grammar G, a left-most derivation is a sequence of strings $s_1 ... s_n$ where

- $s_1 = S$. i.e., s_1 consists of a single element, the start symbol.
- $s_n \in \Sigma^*$, i.e. s_n is made up of terminal symbols only.
- Each s_i for $i = 2...n$ is derived from s_{i-1} by picking the left-most non-terminal X in s_{i-1} and replacing it by some β where $X \rightarrow \beta$ is a rule in R.

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

- $s_1 = S$.
- $s_2 = NP\ VP$. (We have taken the left-most non-terminal in s_1, namely S, and chosen the rule $S \rightarrow NP\ VP$, thereby replacing S by NP followed by VP.)

(Left-most) Derivations

S	$\rightarrow$	NP	VP
VP	$\rightarrow$	Vi	
VP	$\rightarrow$	Vt	NP
VP	$\rightarrow$	VP	PP
NP	$\rightarrow$	DT	NN
NP	$\rightarrow$	NP	PP
PP	$\rightarrow$	IN	NP

Vi	$\rightarrow$	sleeps
Vt	$\rightarrow$	saw
NN	$\rightarrow$	man
NN	$\rightarrow$	woman
NN	$\rightarrow$	telescope
NN	$\rightarrow$	dog
DT	$\rightarrow$	the
IN	$\rightarrow$	with
IN	$\rightarrow$	in

- $s_3 = DT\ NN\ VP$. (We have used the rule $NP \rightarrow DT\ NN$ to expand the left-most non-terminal, namely NP.)
- $s_4 = the\ NN\ VP$. (We have used the rule $DT \rightarrow the$.)
- $s_5 = the\ man\ VP$. (We have used the rule $NN \rightarrow man$.)
- $s_6 = the\ man\ Vi$. (We have used the rule $VP \rightarrow Vi$.)
- $s_7 = the\ man\ sleeps$. (We have used the rule $Vi \rightarrow sleeps$.)

CFG - Ambiguity

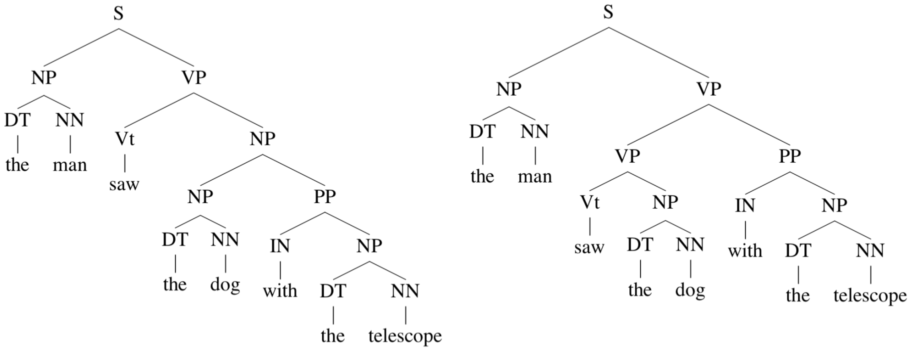

Figure: Two parse trees (derivations) for the sentence under the CFG.

Contents

Probabilistic context-free grammars

Probabilistic context-free grammars (PCFGs) consists of:

1. A context-free grammar $G = (N, \Sigma, R, S)$.
2. A parameter

$$q(\alpha \to \beta)$$

for each rule $\alpha \to \beta \in R$. The parameter $q(\alpha \to \beta)$ can be interpreted as the conditional probabilty of choosing rule $\alpha \to \beta$ in a left-most derivation, given that the non-terminal being expanded is α. For any $X \in N$, we have the constraint

$$\sum_{\alpha \to \beta \in R : \alpha = X} q(\alpha \to \beta) = 1.$$

In addition we have $q(\alpha \to \beta) \geq 0$ for any $\alpha \to \beta \in R$.
Given a parse-tree $t \in T_G$ containing rules $\alpha_1 \to \beta_1, \alpha_2 \to \beta_2, ..., \alpha_n \to \beta_n$, the probability of t under the PCFG is

$$p(t) = \prod_{i=1}^{n} q(\alpha_i \to \beta_i)$$

PCFGs - Example

$N = \{S, NP, VP, PP, DT, Vi, Vt, NN, IN\}$
$S = S$
$\Sigma = \{sleeps, saw, man, woman, dog, telescope, the, with, in\}$
$R, q =$

S	$\rightarrow$	NP	VP	1.0
VP	$\rightarrow$	Vt	NP	0.8
VP	$\rightarrow$	VP	PP	0.2
NP	$\rightarrow$	DT	NN	0.8
NP	$\rightarrow$	NP	PP	0.2
PP	$\rightarrow$	IN	NP	1.0

Vi	$\rightarrow$	sleeps	1.0
Vt	$\rightarrow$	saw	1.0
NN	$\rightarrow$	man	0.1
NN	$\rightarrow$	woman	0.1
NN	$\rightarrow$	telescope	0.3
NN	$\rightarrow$	dog	0.5
DT	$\rightarrow$	the	1.0
IN	$\rightarrow$	with	0.6
IN	$\rightarrow$	in	0.4

A simple probabilistic context-free grammar (PCFG). Our goal is to find parameter that

$$argmax_{t \in T_G(s)} p(t)$$

Input: a sentence $s = x_1 \ldots x_n$, a PCFG $G = (N, \Sigma, S, R, q)$.
Initialization:
For all $i \in \{1 \ldots n\}$, for all $X \in N$,

$$\pi(i, i, X) = \begin{cases} q(X \rightarrow x_i) & \text{if } X \rightarrow x_i \in R \\ 0 & \text{otherwise} \end{cases}$$

Algorithm:

- For $l = 1 \ldots (n-1)$
  - For $i = 1 \ldots (n-l)$
    * Set $j = i + l$
    * For all $X \in N$, calculate

$$\pi(i, j, X) = \underset{\substack{X \rightarrow Y Z \in R, \\ s \in \{i \ldots (j-1)\}}}{\textit{argmax}} \left(q(X \rightarrow Y Z) \times \pi(i, s, Y) \times \pi(s+1, j, Z) \right)$$

Output: Return $\pi(1, n, S) = \sum_{t \in \mathcal{T}(s)} p(t)$

Figure: The CKY parsing algorithm.

CKY Algorithm

```
the - DT (1.0)
man - NN (0.1)
saw - Vt (1.0)
the - DT (1.0)
dog - NN (0.5)
with - IN (0.6)
the - DT (1.0)
telescope - NN (0.3)
the man - NP (0.08000000000000002)
the dog - NP (0.4)
the telescope - NP (0.24)
saw dog - VP (0.32000000000000006)
with telescope - PP (0.144)
the dog - S (0.025600000000000001)
the telescope - NP (0.01152)
saw telescope - VP (0.009216000000000002)
the telescope - S (0.0007372800000000003)

P(t) = 0.0007372800000000003
```

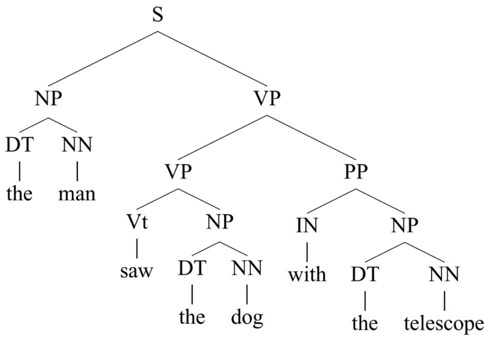

Figure: Parse result from CKY algorithm.

Rule-based vs. Corpus-based NLP

Rule-based grammar

- **Pros**
  - Flexible.
  - Doesn't require a massive training corpus.
  - Understanding of the language phenomenon.
- **Cons**
  - Requires skilled developers and linguists.
  - Slow parser development.

Corpus-based grammar

- **Pros**
  - Easy to scale.
  - "Learnability" without being explicitly programmed.
  - Fast development (if datasets available).
- **Cons**
  - Requires training corpus with annotation.
  - No understanding of the language phenomenon.

Contents

Why do we need language model?

One way to think about it is that the speech recognition model **generates** a large number of **candidate sentences**, together with **probabilities**; the language model is then used to reorder these possibilities based on how likely they are to be a sentence in the language.

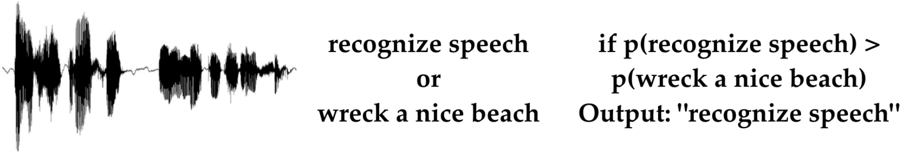

Figure: Language model is useful in speech recognition.

Language model

A language model consists of a finite set $\mathcal{V}$, and a function $p(x_1, x_2, ..., x_n)$ such that:

- For any $\langle x_1 ... x_n \rangle \in \mathcal{V}^\dagger$, $p(x_1, x_2, ..., x_n) \geq 0$.
- In addition,

$$\sum_{\langle x_1 ... x_n \rangle \in \mathcal{V}^\dagger} p(x_1, x_2, ..., x_n) = 1$$

  Hence $p(x_1, x_2, ..., x_n)$ is a probabiluity distribution over the sentences in $\mathcal{V}^\dagger$.

where

- $\mathcal{V}$ is the set of all words in the language.
- $\langle x_1 ... x_n \rangle$ is the sentence in the language (a sequence of words).

We define

$$p(x_1 \ldots x_n) = \frac{c(x_1 \ldots x_n)}{N}$$

where

- $c(x_1 \ldots x_n)$ is the number of times that the sentence $x_1 \ldots x_n$ is seen in our training corpus.
- N is the total number of sentences in the training corpus.

There are $\mathcal{V}^n$ possible sequences of the form $x_1 \ldots x_n$. This is a very poor model: in particular it will assign probability 0 to any sentence not seen in the training corpus.

Markov Models

Chain rule for conditional probability:

$$p(x_1, x_2, ..., x_n) = p(x_1)p(x_2|x_1)p(x_3|x_2, x_1)...p(x_n|x_{n-1}x_{n-2}...x_1)$$

In a second-order Markov process, we make the following assumption which will form the basis of trigram language models, namely that each word depends on the **previous two words** in the sequence:

$$p(x_1, x_2, ..., x_n) = p(x_1)\prod_{i=2}^{n} p(x_i|x_1, ..., x_{i-1}) = p(x_1)\prod_{i=2}^{n} p(x_i|x_{i-2}, x_{i-1})$$

For example, for the sentence

the dog barks STOP

we would have

$$p(\textit{the dog barks STOP}) = p(\textit{the}|*,*) \times p(\textit{dog}|*,\textit{the}) \\ \times p(\textit{barks}|\textit{the},\textit{dog}) \times p(\textit{STOP}|\textit{dog},\textit{barks})$$

where

- $STOP \notin \mathcal{V}$ indicates the end of the sentence.
- $*$ is a special "start" symbol in the sentence.

Contents

Part-of-Speech tagging

INPUT: Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT: Profits/**N** soared/**V** at/**P** Boeing/**N** Co./**N** ,/**,** easily/**ADV** topping/**V** fore- casts/**N** on/**P** Wall/**N** Street/**N** ,/**,** as/**P** their/**POSS** CEO/**N** Alan/**N** Mu- lally/**N** announced/**V** first/**ADJ** quarter/**N** results/**N** ./**.**

KEY:

- N = Noun
- V = Verb
- P = Preposition
- Adv = Adverb
- Adj = Adjective
- . . .

INPUT: Profits soared at Boeing Co., easily topping forecasts on Wall Street, as their CEO Alan Mulally announced first quarter results.

OUTPUT: Profits/**NA** soared/**NA** at/**NA** Boeing/**SC** Co./**CC** ,/**NA** easily/**NA** top- ping/**NA** forecasts/**NA** on/**NA** Wall/**SL** Street/**CL** ,/**NA** as/**NA** their/**NA** CEO/**NA** Alan/**SP** Mulally/**CP** announced/**NA** first/**NA** quarter/**NA** results/**NA** ./**NA**

KEY:

- NA = No entity
- SC = Start Company
- CC = Continue Company
- SL = Start Location
- CL = Continue Location
- ...

Application: flight booking assistant

A trigram HMM consists of a finite set $\mathcal{V}$ of possible words, and a finite set $\mathcal{K}$ of possible tags, together with the following parameters:

- A parameter
$$q(s|u, v)$$
for any trigram (u, v, s) such that $s \in \mathcal{K} \cup \{STOP\}$, and $u, v \in \mathcal{K} \cup \{*\}$. The value for $q(s|u, v)$ can be interpreted as the probability of seeing the tag s immediately after the bigram of tags (u, v).
- A parameter
$$e(x|s)$$
for any $x \in \mathcal{V}, s \in \mathcal{K}$. The value for $e(x|s)$ can be interpreted as the probability of seeing observation x paired with state s.

Trigram HMMs (cont.)

Define $\mathcal{S}$ to be the set of all sequence/tag-sequence pairs $\langle x_1 ... x_n, y_1 ... y_{n+1} \rangle$ such that $n \geq 0, x_i \in \mathcal{V}$ for $i = 1...n$, $y_i \in \mathcal{K}$ for $i = 1...n$, and $y_{n+1} = STOP$. We then define the probability for any $\langle x_1 ... x_n, y_1 ... y_{n+1} \rangle \in \mathcal{S}$ as

$$p(x_1 ... x_n, y_1 ... y_{n+1}) = \prod_{i=1}^{n+1} q(y_i | y_{i-2} y_{i-1}) \prod_{i=1}^{n} e(x_i | y_i)$$

where we have assumed that $y_0 = y_{-1} = *$. Our goal is to find

$$arg\ max_{y_1 ... y_n} p(x_1 ... x_n, y_1 ... y_{n+1})$$

for any input $x_1 ... x_n$.
Note: use Viterbi Algorithm to find the solution.

Trigram HMMs (cont.)

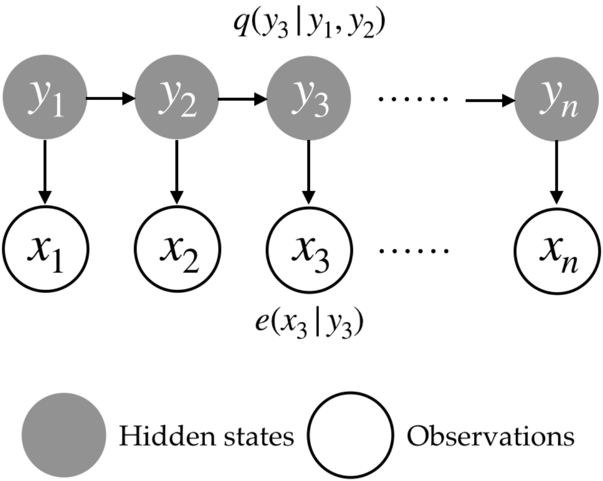

Figure: HMMs illustration

Trigram HMMs (cont.)

Example, if we have $n = 3$, $x_1 ... x_3$ equal to the sentence *the dog laughs*, and $y_1 ... y_4$ equal to the tag sequence $D\ N\ V\ STOP$, then

$$p(x_1...x_n, y_1...y_{n+1}) = q(D|*, *) \times q(N|*, D) \times q(V|D, N) \times q(STOP|N, V) \\ \times e(the|D) \times e(dog|N) \times e(laughs|V)$$

Contents

Word meaning

How do we represent the meaning of a word?
How do we have usable meaning in a computer?

Words as discrete symbols

Encoding features

- **"one-hot"** encoding is popular for categorical features.
- **"bag of words"** is popular for text (token counts).

Example:

- Doc 1: "The cat sat on the hat"
- Doc 2: "The dog chased the cat and the hat"

From dictionary { the, cat, sat, on, hat, dog, chased, and }, we could generate one-hot or bag-of-words vectors

- Doc 1: { 2, 1, 1, 1, 1, 0, 0, 0 }
- Doc 2: { 3, 1, 0, 0, 1, 1, 1, 1 }

Count the frequency of each word type in a large corpus.
List the word types in order of their frequency.
Let f = frequency of a word type, r = its rank in the list.
Zipf's law says: $f \propto 1/r$. In other words, there exists a constant k such that $f \times r = k$.

Word	**f**	**r**
the	2,420,778	1
of	1,045,733	2
to	968,882	3
a	892,429	4
and	865,644	5
in	847,825	6
said	504,593	7
for	363,865	8
that	347,072	9

Zipf's law

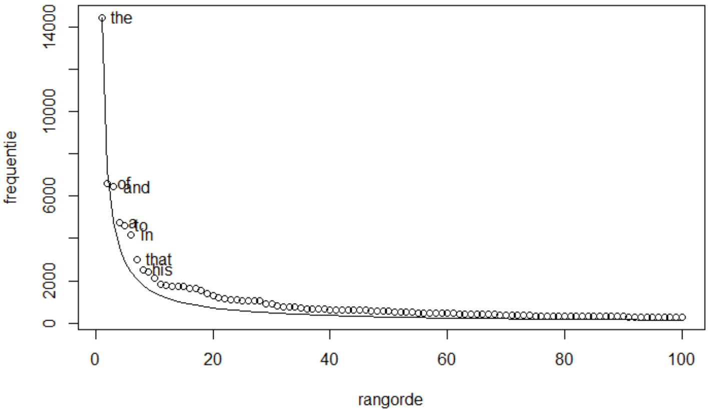

Figure: Zipfian distribution of the frequency (vertical axes) and the rank in the frequency table (horizontal axes) of the first hundred words of Melvilles Moby Dick. The line was predicted by Zipf's law, and the dots depict the actual word frequencies in the text. Credit: Radboud University

TF-IDF

- TF = Term Frequency: if word is repeated in the document, it's probably important.

$$TF_{t,d} = \frac{n_{t,d}}{\sum_{s \in d} n_{s,d}}$$

- IDF = Inverse Document Frequency: rare words more important

$$IDF_t = log \frac{N}{DF_t} + 1$$

- TF-DF = TF x DF: weighted sum for ranking documents.

where

- d: document.
- $t, s \in d$: term (word)
- n: number of occurrences of term t
- N: number of document in corpus.

TF-IDF (cont.)

Example:

- Doc 1: "The cat sat on the hat"
  { 2, 1, 1, 1, 1, 0, 0, 0 }
- Doc 2: "The dog chased the cat and the hat"
  { 3, 1, 0, 0, 1, 1, 1, 1 }

Calculate TF-IDF:

- Doc 1: $[2, 1, 1, 1, 1, 0, 0, 0\,]/6 * (\log(2/[2, 2, 1, 1, 2, 0, 0, 0]) + 1)$
  $= [0.33, 0.16, 0.28, 0.28, 0.16, 0, 0, 0]$
- Doc 2: $[3, 1, 0, 0, 1, 1, 1, 1\,]/8 * (\log(2/[2, 2, 0, 0, 1, 1, 1, 1]) + 1)$
  $= [0.37, 0.12, 0, 0, 0.21, 0.21, 0.21, 0.21]$

Problems

Problem with words as discrete symbols

- There are cases where there is no natural notion of similarity between two words because two vectors are orthogonal even their meaning are close together. Ex: motel $= [0\ 0\ 0\ 1\ 0\ 0\ 0]$, hotel $= [1\ 0\ 0\ 0\ 0\ 0\ 0]$.
- Does not capture position in text, semantics, co-occurrences in different documents, etc. Only useful as a lexical level feature

Core idea: A word's meaning is given by the words that frequently appear close-by.
When a word w appears in a text, its context is the set of words that appear nearby (within a fixed-size window).
Use the many contexts of w to build up a representation of w.

Example

... government debt problems turning into **banking** crises as happened in 2009 ...
... saying that Europe needs unified **banking** regulation to replace the hodgepodge ...
...India has just given its **banking** system a shot in the arm ...

These context words will represent **banking**.

We will build a dense vector for each word, chosen so that it is similar to vectors of words that appear in similar contexts.

Example

linguistics = [0.286, 0.792, -0.177, -0.107, 0.109, -0.542, 0.349, 0.271]

Note: word vectors are sometimes called **word embeddings** or **word representations**.

Word2vec: Overview

Word2vec (Mikolov et al. 2013) is a framework for learning word vectors.
Idea:

- We have a large corpus of text
- Every word in a fixed vocabulary is represented by a vector
- Go through each position t in the text, which has a center word c and context ("outside") words o
- Use the similarity of the word vectors for c and o to calculate the probability of o given c (or vice versa)
- Keep adjusting the word vectors to maximize this probability

Word2vec: Overview

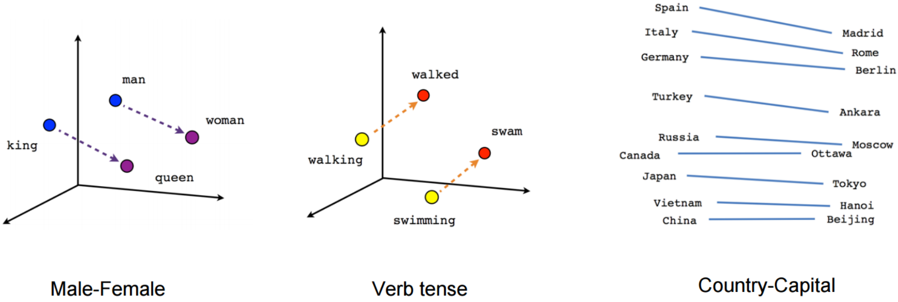

Figure: Projecting word vectors using t-SNE visualization (TensorFlow: Vector Representations of Words)

Representation for all levels

- Word **meaning** as a neural word vector.
- Word **similarities** can be measured. e.g. frogs, toad, lizard, ...
- Every **morpheme** is a vector, a neural network combines two vectors into one vector. e.g. unfortunately = un + fortunate + ly (Luong et al. 2013).
- **Parsing** neural networks can accurately determine the grammatical structure of sentences.
- **Semantics** every word, phrase, logical expression is a vector (Bowman et al. 2014).
- **Application** Sentiment Analysis, Question Answering, Dialogue agents / Response Generation, Machine Translation could representing as vectors for all levels of language.

Contents

Algorithms

Key methods for statistical NLP

- Viterbi
- Embedding methods: word2vec, doc2vec.
- Recurrent Neural Networks (RNN).
- Long Short Term Memory networks (LSTM).

Skills you'll need

- Simple linear algebra (vectors, matrices).
- Basic probability theory.
- Java or Python programming.

Corpus

English

- Brown corpus.
- WSJ corpus.
- Swtichboard corpus.

Vietnamese

- VLSP corpus.
- CLC VTB corpus.
- vnQTag corpus.

NLP conferences

- Association for Computational Linguistics (ACL)
- Empirical Methods in Natural Language Processing (EMNLP)
- International Conference on Computational Linguistics (COLING)
- Conference on Natural Language Learning (CoNLL)
- Special Interest Group on Information Retrieval (SIGIR)
- American Association for Artificial Intelligence (AAAI)
- International Conference on Machine Learning (ICML)
- Neural Information Processing Systems (NIPS)

Contents

References

[1] Daniel Jurafsky and James H. Martin. Speech and Language Processing.
[2] Christopher D. Manning and Hinrich Schütze. Foundations of Statistical Natural Language.
[3] CS224n: Natural Language Processing with Deep Learning.