LNMIIT

Natural Language Processing

Some screenshots are taken from NLP course by Jufrasky
— Used only for educational purpose

Developments

- IBM Watson wins Jeopardy Challenge (2011)
- IBM Deep Blue beats Gary Kasparov (1997)
- DeepMind's AlphaZero (AI system that learns chess in 24 hours) beats the best chess engine Stockfish (2017)
- AlphaGo defeated two best in players of Go (2015-16)

Problems

- Information Extraction
- Machine Translation
- Conversational Agent / Dialogue System
- Question and Answer Systems

We need to look into…

- Phonetics
- Morphology
- Syntax
- Semantics

✓

- Pragmatics
- Discourse

LNMIIT

Progress

- Almost Done:
    - Spam versus Ham - 99% accuracy
    - PoS - 97%
    - NER - 97%

- Good Progress:
    - Sentiment Analysis
    - Wordsense disambiguation
    - Parsing
    - Machine Translation
    - Information Extraction

- Hard problems:
    - Question Answering systems
    - Paraphrase
    - Summarisation
    - Dialogue

Crash Blossoms and Garden Path Sentences

- "*Dutch military plane carrying bodies from Malaysian Airlines Flight 17 crash lands in Eindhoven*" (July 23, 2014)
- *"I went to bank"*
- *"Fed raises interest rates"*
- *"The old man the boat"*

Issues

- ambiguity
- non-standard text (example: tweets)
- segmentation problems
- idioms
- neologisms
- world knowledge
- tricky entity names - bio-names

State of the Art & History

- Foundation Insights: 1940s and 1950s
  - Automaton
  - Probabilistic or Information Theoretic
    - McCulloch-Pitts Neuron (1943)
    - Chomsky (1950) — Finite State Machines and CFG
      - Backus (1959) & Naur et al (1960) — ALGOL
    - Probabilistic algorithms for speech and language processing

State of the Art & History

- Two Camps: 1957 to 1970
  - Symbolic — Chomsky's related works
  - Stochastic — AI — McCarthy, Minsky, Shannon and others
    - Stochastic and Statistics — Bayesian models (Mosteller and Wallace (1964)
    - Logic and General problem Solving — Newell and Simon
  - Brown Corpus — one-million word corpus from Newspaper, Novels, non-fiction, academics etc.,

State of the Art & History

Unified for LUNAR QA system

- Four Paradigms: 1970 to 1983
  - Stochastic — speech recognition, HMM
  - Logic-based — functional grammar
  - Natural language understanding — SHRDLU systems
  - Discourse modeling — BDI (Belief-Desire-Intention)

State of the Art & History

- Empiricism and Finite State Machine Revisited
  - FSM:
    - Finite state phonology and morphology by Kaplan and Kay (1981)
    - Finite state models of syntax by Church (1980)
  - Empiricism:
    - IBM Watson Research Center's work on probabilistic models of speech recognition: parsing, PoS tagging, addressing ambiguities and semantics

State of the Art & History

- All branches come together: 1994 to 1999
  - Algorithms for parsing, PoS tagging, reference resolution and discourse processing through probabilistic models
  - Commercial exploitation of speech and language processing

LNMIIT

State of the Art & History

- The Rise of ML (2000 to 2008)
  - Linguistics Data Consortium (LDC) — large amounts of spoken and written materials available
    - All has syntactic, semantics and pragmatic annotations
    - Parsing and semantic analysis problems became a set of problems in supervised learning
  - Learning models brought statistical & probabilistical models closer
  - High Performance Computing enabled ML in NLP
  - At last works of Brown et al (1990), Och and Ney (2003) [Machine translation] and Biel (2003) [Topic modeling] showed that we can even work with unannotated text data

Regular Expression

Men are all alike.
IN WHAT WAY
They're always bugging us about something or other.
CAN YOU THINK OF A SPECIFIC EXAMPLE
Well, my boyfriend made me come here.
YOUR BOYFRIEND MADE YOU COME HERE
He says I'm depressed much of the time.
I AM SORRY TO HEAR YOU ARE DEPRESSED

— Weizenbaum (1966) ELIZA — *A computer program for the study of natural language communication between man and machine*

Regular Expression

- First developed by Kleene (1956)
- Regular Expression (RE) is a formula in a special language that specifies simple classes of strings
- Alternatively, RE is an algebraic notation for characterising a set of strings
- For any RE we can build an equivalent finite state automata (FSA)
- RE search requires a pattern that we want to search for and a corpus of texts to search through

LNMIIT

Disjunction

- *[Ww]* matches either *W* or *w*
- *[A-Z]* matches any one of the alphabet from *A* to *Z*
- *[a-z]* matches any one of the alphabet from *a* to *z*
- *[A-Za-z]* matches any one of the alphabet from *A* to *Z* or from *a* to *z*
- *[0-9]* matches any one of the digit from *0* to *9*
- *[!]* what this will match?

Negation in Disjunction

- [^Tt] matches characters other than T or t
- [^A-Z] matches all characters except A to Z
- [^A-Za-z] matches all characters other than the alphabets
- *Ram*|*Sita* represents either Ram or Sita

Special Characters

- **?** matches exactly zero or one occurrence of the previous character or expression

- ***** matches exactly zero or more occurrences of the previous character or expression

- + matches exactly one or more occurrences of the previous character or expression

- {n} matches n occurrences of the previous character or expression

- {n,m} matches n to m occurrences of the previous character or expression

- {n,} matches at least n occurrences of the previous character or expression

Anchors

- ^ is used to show that expression to be matched at the starting of new line
- $ is used to show that expression to be matched at the end of new line

Basic Text Processing

- *“The cat in the hat”*
- *“The other one there, the blithe one”*

LNMIIT

Basic Text Processing

- *“The cat in the hat”*
- *“The other one there, the blithe one”*
- Search for
  - *[Tt]he*
  - *[Th]he[^A-Za-z]*
  - *[^A-Za-z][Th]he[^A-Za-z]*
- False Positive: *‘blithe’*
- False Negative: *‘The’*

Basic Text Processing

- Tokenisation
- How many tokens are there?
  - San Francisco, New Delhi
  - Speech — uh…, main….mainly
  - Cat, Cats, cat, cats, I’m, They’re, India’s capital, Ph.D and so on
- How many types/unique tokens?

Basic Text Processing

- N = Number of tokens
- V = Vocabulary = Set of types
- Phone conversations:
    - N = 2.4 million
    - |V| = 20000
- Shakespeare:
    - N = 834000
    - |V| = 31000
- Google N-grams
    - N = 1 trillion
    - |V| = 13 million

$|V| > O(N^{1/2})$ from Church and Gale 1990)

Word Segmentation — Maximum Matching algorithm

- “The cat in the hat”
  - apply maximum matching algorithm for the above
- “The table down there”
  - apply maximum matching algorithm for the same!!

Word Segmentation — Maximum Matching algorithm

- Maximum Matching doesn’t work well in English
- It works well for Chinese where the average word length is just 2.3
- It works well with words of less length

LNMIIT

Normalising Tokens

- U.S.A & USA
- asymmetric expansions — Window, Window(s)
- case folding — make all lower case letters
  - Exceptions — General Motors, Congress
  - Sentiments analysis caps or lower is important

Lemmatization

- Reduce the variant forms to base forms
  - am, are, is —> be
  - cat, cats, Cat, Cats —> cat
  - Morphemes are the small meaning full units that make words
  - Morphemes can be words, affixes-prefixes or suffixes. Examples of Morpheme: -ed = turns a verb into the past tense. un- = prefix that means not.

Lemmatization

- Stems: The core meaning-bearing unit
- Affixes: that is attached to the stems — prefix or suffix — according to a grammatical rule
- Stemming: It is a crude chopping of affixes
    - automate, automation, automates, automatic, automated all converted to automat

Stemming Process

- Porter’s algorithm for English Stemmer:

LNMIIT

Minimum Edit Distance

- Word’s / String’s similarity
  - Spell correction — graet with either great/grate/target/rage/raget
  - Computational Biology — Aligning two nucleotides / amino-acids sequence
  - Machine Translation, Information Extraction, Speech Recognition

Minimum Edit Distance

- Minimum edit distance between two strings
    - Operations:
        - Insertion
        - Deletion
        - Substitution
- Minimise the number of operations

I N T E * N T I O N

* E X E C U T I O N

1 delete
3 substitution
1 insertion

5 operations

8 operations

Levenshtein Distance

Computational Biology

```
AGGCTATCACCTGACCTCCAGGCCGATGCCC
TAGCTATCACGACCGCGGTCGATTTGCCCGAC
```

```
-AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC
```

Other Applications

- Evaluating Similarity of Sentences
    - Spokesman said the senior advisor was killed
    - Spokesman confirmed that the senior advisor was dead
- Named entity extraction and entity coreference — IBM and IBM Ltd

Minimum Edit Distance

- Searching for a sequence of edits / paths from the starting string to the final string
  - Given: Word which is to be transformed
  - Operations: Insertion, Deletion and Substitution
  - Output: The word we are trying to get
  - Path Cost: Minimise the cost / edits / operations

Algorithm

- Sample space is HUGE! (If we do it exhaustively)
- Minimising number of edits for two strings depends on minimising the number of edits for its substrings!
- Problem is recursive in nature but subproblems are depended!
- Dynamic Programming is the appropriate one here

Algorithm

- Two strings X and Y: X of length n and Y of length n
  - X to be transformed to Y through I, D and S operations
  - D(i,j) is the edit distance between X[1..i] and Y[1..j]
  - D(n,m) is the edit distance of X to Y

Algorithm

- Computing D(n,m) using D(i,j) where i and j are smaller values than n and m, respectively
- Combine the values D(i,j) to get D(n,m)

Algorithm

$D(0,j) = j$ (Insert)

$D(i,0) = i$ (delete)

For all i,j

$D(i,j) = \mathrm{Min}\{D(i-1,j) + 1, D(i,j-1) + 1, D(i-1,j-1) + 2\}$

$D(n,m)$ will be the output

LNMIIT

Algorithm

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

LNMIIT

Algorithm

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3	4								
N	2	3	4	5						
I	1	2	3	4						
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Algorithm

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

Backtrace in Minimum edit distance

- Ultimately if we need to find the optimal alignment then we need to do a backtrace
- Every time we enter a new cell in the table we note down from where we came from (minimum one)
- After we reach the end of the table we back trace by recalling the previous cell we came from we shall be able to get the optimal alignment

Backtrace in Minimum edit distance

n	9	↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙←↓ 12	↓ 11	↓ 10	↓ 9	↙ 8
o	8	↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↓ 10	↓ 9	↙ 8	← 9
i	7	↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	↙ 8	← 9	← 10
t	6	↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙ 8	← 9	← 10	←↓ 11
n	5	↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙↓ 10
e	4	↙ 3	← 4	↙← 5	← 6	← 7	←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9
t	3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙ 7	←↓ 8	↙←↓ 9	↓ 8
n	2	↙←↓ 3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↓ 7	↙←↓ 8	↙ 7
i	1	↙←↓ 2	↙←↓ 3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙ 6	← 7	← 8
#	0	1	2	3	4	5	6	7	8	9
	#	e	x	e	c	u	t	i	o	n

I N T E * N T I O N

* E X E C U T I O N

Backtrace in Minimum edit distance

- Time complexity: O(mn)
- Space complexity: O(mn)
- Backtrace: O(m+n)