Introduction to Artificial Intelligence
Natural Language Processing

Janyl Jumadinova
November 14, 2016

Credit: NLP Stanford
Question Answering: IBM’s Watson

Won Jeopardy on February 16, 2011!

WILLIAM WILKINSON’S “AN ACCOUNT OF THE PRINCIPALITIES OF WALLACHIA AND MOLDOVIA” INSPIRED THIS AUTHOR’S MOST FAMOUS NOVEL

Bram Stoker
Information Extraction

Subject: curriculum meeting
Date: November 1, 2016

Hi Janyl, we’ve now scheduled the curriculum meeting. It will be in CC 103 tomorrow from 10:00-11:00.

-Chris

Event: Curriculum mtg
Date: Nov-1-2016
Start: 10:00am
End: 11:00am
Where: CC 103

Create new Calendar entry
Sentiment Extraction

Source: Washington Post
Machine Translation

Translate

English | Spanish | French | English - detected | 
---|---|---|---|

Happy Monday

English | Spanish | Kyrgyz | 
---|---|---|

бактылуу Дүйшөмбү

baktyluu Düyşömbü
Language Technology

mostly solved

Sentiment analysis
- Best roast chicken in San Francisco!
- The waiter ignored us for 20 minutes.

Coreference resolution
- Carter told Mubarak he shouldn’t run again.

Word sense disambiguation (WSD)
- I need new batteries for my mouse.

Part-of-speech (POS) tagging
- ADJ  ADJ  NOUN  VERB  ADV
  Colorless green ideas sleep furiously.

Named entity recognition (NER)
- PERSON  ORG  LOC
  Einstein met with UN officials in Princeton

Machine translation (MT)
- 第13届上海国际电影节开幕...
  The 13th Shanghai International Film Festival...

Information extraction (IE)
- You’re invited to our dinner party, Friday May 27 at 8:30

making good progress

still really hard

Question answering (QA)
- Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?

Paraphrase
- XYZ acquired ABC yesterday
- ABC has been taken over by XYZ

Summarization
- The Dow Jones is up
- The S&P500 jumped
- Housing prices rose
- Economy is good

Dialog
- Where is Citizen Kane playing in SF?
- Castro Theatre at 7:30. Do you want a ticket?
Ambiguity makes NLP hard
Ambiguity makes NLP hard

- Teacher Strikes Idle Kids
- Red Tape Holds Up New Bridges
- Juvenile Court to Try Shooting Defendant
- Local High School Dropouts Cut in Half
Other NLP Difficulties

- **non-standard English**
  Great job @justinbieber! Were SOO PROUD of what you've accomplished! U taught us 2 #neversaynever & you yourself should never give up either❤️

- **segmentation issues**
  - the New York-New Haven Railroad
  - the New York-New Haven Railroad

- **idioms**
  - dark horse
  - get cold feet
  - lose face
  - throw in the towel

- **neologisms**
  - unfriend
  - Retweet
  - bromance

- **world knowledge**
  - Mary and Sue are sisters.
  - Mary and Sue are mothers.

- **tricky entity names**
  - Where is *A Bug’s Life* playing ...
  - *Let It Be* was recorded ...
  - ... a mutation on the for gene ...
Progress

- What tools do we need?
  - Knowledge about language
  - Knowledge about the world
  - A way to combine knowledge sources
What tools do we need?
- Knowledge about language
- Knowledge about the world
- A way to combine knowledge sources

How we generally do this:
- Probabilistic models built from language data
- \( P(\text{"maison"} \rightarrow \text{"house"}) \rightarrow \text{high} \)
- \( P(\text{"L’avocat general"} \rightarrow \text{"the general avocado"}) \rightarrow \text{low} \)
Basic Text Processing

Regular Expressions

- A formal language for specifying text strings.
Basic Text Processing

Regular Expressions

- A formal language for specifying text strings.
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks
Regular Expressions: Disjunctions

**Letters inside square brackets []**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[wW]oodchuck</td>
<td>Woodchuck, woodchuck</td>
</tr>
<tr>
<td>[1234567890]</td>
<td>Any digit</td>
</tr>
</tbody>
</table>

**Ranges [A–Z]**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A–Z]</td>
<td>An upper case letter</td>
</tr>
<tr>
<td></td>
<td>Drenched Blossoms</td>
</tr>
<tr>
<td>[a–z]</td>
<td>A lower case letter</td>
</tr>
<tr>
<td></td>
<td>my beans were impatient</td>
</tr>
<tr>
<td>[0–9]</td>
<td>A single digit</td>
</tr>
<tr>
<td></td>
<td>Chapter 1: Down the Rabbit Hole</td>
</tr>
</tbody>
</table>
Regular Expressions: Negation in Disjunction

- Negations [^Ss]
- Carat means negation only when first in []

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>[^A-Z]</td>
<td>Not an upper case letter</td>
<td>Oyfn priphetchik</td>
</tr>
<tr>
<td>[^Ss]</td>
<td>Neither ‘S’ nor ‘s’</td>
<td>I have no exquisite reason”</td>
</tr>
<tr>
<td>[^e^]</td>
<td>Neither e nor ^</td>
<td>Look here</td>
</tr>
<tr>
<td>a^b</td>
<td>The pattern a carat b</td>
<td>Look up a^b now</td>
</tr>
</tbody>
</table>
Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>groundhog</td>
<td>woodchuck</td>
</tr>
<tr>
<td>yours</td>
<td>mine</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>[gG]roundhog</td>
<td>[Ww]oodchuck</td>
</tr>
</tbody>
</table>

Photo D. Fletcher
### Regular Expressions: \? * + .

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Matches</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>color?r</td>
<td>Optional previous char</td>
<td>color</td>
</tr>
<tr>
<td>oo*h!</td>
<td>0 or more of previous char</td>
<td>oh!</td>
</tr>
<tr>
<td>o+h!</td>
<td>1 or more of previous char</td>
<td>oh!</td>
</tr>
<tr>
<td>baa+</td>
<td></td>
<td>baa</td>
</tr>
<tr>
<td>beg.n</td>
<td></td>
<td>begin igen</td>
</tr>
</tbody>
</table>
Regular Expressions: Example

Find all instances of the word “the” in a text

```
the

[tT]he
```

Misses capitalized examples

```
[^a-zA-Z][tT]he[^a-zA-Z]
```

Incorrectly returns other or theology
Basic Text Processing

Word tokenization

Every NLP task needs to do text normalization:
1. Segmenting/tokenizing words in running text
2. Normalizing word formats
3. Segmenting sentences in running text
How Many Words?

\[ N = \text{number of tokens} \]
\[ V = \text{vocabulary} = \text{set of types} \]

\[ |V| \text{ is the size of the vocabulary} \]

Church and Gale (1990): \[ |V| > O(N^{\frac{1}{2}}) \]

| Tokens = N                  | Types = |V| |
|-----------------------------|---------|---|
| Switchboard phone conversations | 2.4 million | 20 thousand |
| Shakespeare                 | 884,000  | 31 thousand |
| Google N-grams              | 1 trillion | 13 million |
Simple Tokenization in UNIX

- (Inspired by Ken Church’s UNIX for Poets.)
- Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '
' < shakes.txt  
  | sort    
  | uniq --c
```

<table>
<thead>
<tr>
<th>Word</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1945</td>
</tr>
<tr>
<td>AARON</td>
<td>72</td>
</tr>
<tr>
<td>ABBESS</td>
<td>19</td>
</tr>
<tr>
<td>ABBOT</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Aaron</td>
<td>25</td>
</tr>
<tr>
<td>Abate</td>
<td>6</td>
</tr>
<tr>
<td>Abates</td>
<td>1</td>
</tr>
<tr>
<td>Abbess</td>
<td>5</td>
</tr>
<tr>
<td>Abbey</td>
<td>6</td>
</tr>
<tr>
<td>Abbot</td>
<td>3</td>
</tr>
<tr>
<td>....</td>
<td>...</td>
</tr>
</tbody>
</table>
Every NLP task needs to do text normalization:

1. Segmenting/tokenizing words in running text
2. Normalizing word formats
3. Segmenting sentences in running text
Issues in Tokenization

- Finland’s capital → Finland Finlands Finland’s
- what’re, I’m, isn’t → What are, I am, is not
- Hewlett-Packard → Hewlett Packard
- state-of-the-art → state of the art
- Lowercase → lower-case lowercase lower case
- San Francisco → one token or two?
Issues in Tokenization

- Finland’s capital → Finland Finlands Finland’s
- what’re, I’m, isn’t → What are, I am, is not
- Hewlett-Packard → Hewlett Packard
- state-of-the-art → state of the art
- Lowercase → lower-case lowercase lower case
- San Francisco → one token or two?
- **Language Issues**: French, German, Japanese, Chinese,...
Basic Text Processing

Stemming

Every NLP task needs to do text normalization:
1. Segmenting/tokenizing words in running text
2. Normalizing word formats
3. Segmenting sentences in running text
Stemming

- Reduce terms to their stems in information retrieval
- **Stemming** is crude chopping of affixes language dependent
- Example: *automate(s), automatic, automation* all reduced to *automat*.

*for example compressed and compression are both accepted as equivalent to compress.*
Porter's Algorithm

Most common English stemmer.

**Step 1a**

<table>
<thead>
<tr>
<th>Word</th>
<th>Stemmed Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>sses</td>
<td>ss</td>
</tr>
<tr>
<td>ies</td>
<td>i</td>
</tr>
<tr>
<td>ss</td>
<td>ss</td>
</tr>
<tr>
<td>s</td>
<td>Ø</td>
</tr>
<tr>
<td>caresses</td>
<td>caress</td>
</tr>
<tr>
<td>ponies</td>
<td>poni</td>
</tr>
<tr>
<td>caress</td>
<td>caress</td>
</tr>
<tr>
<td>cats</td>
<td>cat</td>
</tr>
</tbody>
</table>

**Step 1b**

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Removed</th>
<th>Word</th>
<th>Stemmed Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>(<em>v</em>)ing</td>
<td>Ø</td>
<td>walking</td>
<td>walk</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sing</td>
<td>sing</td>
</tr>
<tr>
<td>(<em>v</em>)ed</td>
<td>Ø</td>
<td>plastered</td>
<td>plaster</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Step 2 (for long stems)**

<table>
<thead>
<tr>
<th>Word</th>
<th>Stemmed Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>ational</td>
<td>ate</td>
</tr>
<tr>
<td>izer</td>
<td>ize</td>
</tr>
<tr>
<td>ator</td>
<td>ate</td>
</tr>
<tr>
<td>relational</td>
<td>relate</td>
</tr>
<tr>
<td>digitizer</td>
<td>digitize</td>
</tr>
<tr>
<td>operator</td>
<td>operate</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Step 3 (for longer stems)**

<table>
<thead>
<tr>
<th>Word</th>
<th>Stemmed Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>al</td>
<td>Ø</td>
</tr>
<tr>
<td>able</td>
<td>Ø</td>
</tr>
<tr>
<td>ate</td>
<td>Ø</td>
</tr>
<tr>
<td>revival</td>
<td>reviv</td>
</tr>
<tr>
<td>adjustable</td>
<td>adjust</td>
</tr>
<tr>
<td>activate</td>
<td>activ</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Sentence Segmentation

- !, ? are relatively unambiguous
Sentence Segmentation

- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02 or 4.3
Sentence Segmentation

- !, ? are relatively unambiguous
- Period “.” is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02 or 4.3
- Build a binary classifier
  - Classifiers: hand-written rules, regular expressions, or machine-learning
Determining if a word is end-of-sentence: a Decision Tree

Lots of blank lines after me?

- YES: E-O-S
- NO: Final punctuation is ?, !, or :?

Final punctuation is period

- YES: I am “etc” or other abbreviation
- NO: Not E-O-S

I am “etc” or other abbreviation

- YES: Not E-O-S
- NO: E-O-S