Natural Language Processing

Lecture III. Preprocessing
– because not all corpora are just words you can find in Merriam-Webster

Forrest Sheng Bao, Ph.D.

Dept. of Computer Science
Iowa State University
Ames, IA 50011

Sept. 7, 2021

Outline

Outline

Not everyone speaks American English or ASCII

Although in Hollywood movies, aliens all speak English the moment they hit Early (and always in NYC), this is the reality:

```
In [20]: s
Out[20]: 'Universitäten'

In [21]: s.encode()
Out[21]: b'Universit\xc3\xa4ten'

In [22]: s.encode().decode('utf-8')
Out[22]: 'Universitäten'

In [23]: s.encode().decode('latin-1')
Out[23]: 'UniversitÃ¤ten'

In [24]: s.encode().decode('GBK')
Out[24]: 'Universit盲ten'
```

Text encoding for plain text

- How does computers store text information?
- Digital computers represent data using a sequence of numbers (e.g., 1s and 0s).
- Natural languages have basic building blocks: letters, symbols, characters, etc.
- A piece of text is just a character *string*.
- Use one-to-one mapping between binary sequences and the building blocks, e.g., 1100 for “A”, 1001 for “B”, etc.
- ASCII
- Local encodings in Europe and Asia
- UTF8
- Many languages are not yet digitized. Actually many languages still do not have scripts (i.e., writing system).

Text encoding for plain text

- How does computers store text information?
- Digital computers represent data using a sequence of numbers (e.g., 1s and 0s).
- Natural languages have basic building blocks: letters, symbols, characters, etc.
- A piece of text is just a character *string*.
- Use one-to-one mapping between binary sequences and the building blocks, e.g., 1100 for “A”, 1001 for “B”, etc.
- ASCII
- Local encodings in Europe and Asia
- UTF8
- Many languages are not yet digitized. Actually many languages still do not have scripts (i.e., writing system).

Text encoding for plain text

- How does computers store text information?
- Digital computers represent data using a sequence of numbers (e.g., 1s and 0s).
- Natural languages have basic building blocks: letters, symbols, characters, etc.
- A piece of text is just a character *string*.
- Use one-to-one mapping between binary sequences and the building blocks, e.g., 1100 for “A”, 1001 for “B”, etc.
- ASCII
- Local encodings in Europe and Asia
- UTF8
- Many languages are not yet digitized. Actually many languages still do not have scripts (i.e., writing system).

Text encoding for plain text

- How does computers store text information?
- Digital computers represent data using a sequence of numbers (e.g., 1s and 0s).
- Natural languages have basic building blocks: letters, symbols, characters, etc.
- A piece of text is just a character *string*.
- Use one-to-one mapping between binary sequences and the building blocks, e.g., 1100 for “A”, 1001 for “B”, etc.
- ASCII
- Local encodings in Europe and Asia
- UTF8
- Many languages are not yet digitized. Actually many languages still do not have scripts (i.e., writing system).

Text encoding for plain text

- How does computers store text information?
- Digital computers represent data using a sequence of numbers (e.g., 1s and 0s).
- Natural languages have basic building blocks: letters, symbols, characters, etc.
- A piece of text is just a character *string*.
- Use one-to-one mapping between binary sequences and the building blocks, e.g., 1100 for "A", 1001 for "B", etc.
- ASCII
- Local encodings in Europe and Asia
- UTF8
- Many languages are not yet digitized. Actually many languages still do not have scripts (i.e., writing system).

Text encoding for plain text

- How does computers store text information?
- Digital computers represent data using a sequence of numbers (e.g., 1s and 0s).
- Natural languages have basic building blocks: letters, symbols, characters, etc.
- A piece of text is just a character *string*.
- Use one-to-one mapping between binary sequences and the building blocks, e.g., 1100 for “A”, 1001 for “B”, etc.
- ASCII
- Local encodings in Europe and Asia
- UTF8
- Many languages are not yet digitized. Actually many languages still do not have scripts (i.e., writing system).

Text encoding for plain text

- How does computers store text information?
- Digital computers represent data using a sequence of numbers (e.g., 1s and 0s).
- Natural languages have basic building blocks: letters, symbols, characters, etc.
- A piece of text is just a character *string*.
- Use one-to-one mapping between binary sequences and the building blocks, e.g., 1100 for “A”, 1001 for “B”, etc.
- ASCII
- Local encodings in Europe and Asia
- UTF8
- Many languages are not yet digitized. Actually many languages still do not have scripts (i.e., writing system).

Text encoding for plain text

- How does computers store text information?
- Digital computers represent data using a sequence of numbers (e.g., 1s and 0s).
- Natural languages have basic building blocks: letters, symbols, characters, etc.
- A piece of text is just a character *string*.
- Use one-to-one mapping between binary sequences and the building blocks, e.g., 1100 for “A”, 1001 for “B”, etc.
- ASCII
- Local encodings in Europe and Asia
- UTF8
- Many languages are not yet digitized. Actually many languages still do not have scripts (i.e., writing system).

Text encoding for plain text

- How does computers store text information?
- Digital computers represent data using a sequence of numbers (e.g., 1s and 0s).
- Natural languages have basic building blocks: letters, symbols, characters, etc.
- A piece of text is just a character *string*.
- Use one-to-one mapping between binary sequences and the building blocks, e.g., 1100 for “A”, 1001 for “B”, etc.
- ASCII
- Local encodings in Europe and Asia
- UTF8
- Many languages are not yet digitized. Actually many languages still do not have scripts (i.e., writing system).

Unicode – too big to install

For CJK, Unicode has 48 strokes (U+31C0..U+31EF), 224 radicals (U+2F00..U+2FDF) and 12 ideogrpahic description characters (U+2FF0..U+2FFF). But it still has each character!

2FF0 **Ideographic Description Characters** **2FFF**

	2FF
0	⿰ 2FF0
1	⿱ 2FF1
2	⿲ 2FF2
3	⿳ 2FF3
4	⿴ 2FF4
5	⿵ 2FF5
6	⿶ 2FF6
7	⿷ 2FF7
8	⿸ 2FF8
9	⿹ 2FF9

Ideographic description characters

These are visibly displayed graphic characters, not invisible composition controls.

2FF0 ⿰ IDEOGRAPHIC DESCRIPTION CHARACTER LEFT TO RIGHT
2FF1 ⿱ IDEOGRAPHIC DESCRIPTION CHARACTER ABOVE TO BELOW
2FF2 ⿲ IDEOGRAPHIC DESCRIPTION CHARACTER LEFT TO MIDDLE AND RIGHT
2FF3 ⿳ IDEOGRAPHIC DESCRIPTION CHARACTER ABOVE TO MIDDLE AND BELOW
2FF4 ⿴ IDEOGRAPHIC DESCRIPTION CHARACTER FULL SURROUND
2FF5 ⿵ IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM ABOVE
2FF6 ⿶ IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM BELOW
2FF7 ⿷ IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM LEFT
2FF8 ⿸ IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM UPPER LEFT
2FF9 ⿹ IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM UPPER RIGHT
2FFA ⿺ IDEOGRAPHIC DESCRIPTION CHARACTER SURROUND FROM LOWER LEFT
2FFB ⿻ IDEOGRAPHIC DESCRIPTION CHARACTER OVERLAID

2F00 **Kangxi Radicals** **2FDF**

	2F0	2F1	2F2	2F3	2F4	2F5	2F6	2F7	2F8	2F9	2FA	2FB	2FC	2FD
0	⼀ 2F00	⼐ 2F10	⼠ 2F20	⼰ 2F30	⽀ 2F40	⽐ 2F50	⽠ 2F60	⽰ 2F70	⾀ 2F80	⾐ 2F90	⾠ 2FA0	⾰ 2FB0	⿀ 2FC0	⿐ 2FD0
1	⼁ 2F01	⼑ 2F11	⼡ 2F21	⼱ 2F31	⽁ 2F41	⽑ 2F51	⽡ 2F61	⽱ 2F71	⾁ 2F81	⾑ 2F91	⾡ 2FA1	⾱ 2FB1	⿁ 2FC1	⿑ 2FD1
2	⼂ 2F02	⼒ 2F12	⼢ 2F22	⼲ 2F32	⽂ 2F42	⽒ 2F52	⽢ 2F62	⽲ 2F72	⾂ 2F82	⾒ 2F92	⾢ 2FA2	⾲ 2FB2	⿂ 2FC2	⿒ 2FD2
3	⼃ 2F03	⼓ 2F13	⼣ 2F23	⼳ 2F33	⽃ 2F43	⽓ 2F53	⽣ 2F63	⽳ 2F73	⾃ 2F83	⾓ 2F93	⾣ 2FA3	⾳ 2FB3	⿃ 2FC3	⿓ 2FD3
4	⼄ 2F04	⼔ 2F14	⼤ 2F24	⼴ 2F34	⽄ 2F44	⽔ 2F54	⽤ 2F64	⽴ 2F74	⾄ 2F84	⾔ 2F94	⾤ 2FA4	⾴ 2FB4	⿄ 2FC4	⿔ 2FD4
5	⼅ 2F05	⼕ 2F15	⼥ 2F25	⼵ 2F35	⽅ 2F45	⽕ 2F55	⽥ 2F65	⽵ 2F75	⾅ 2F85	⾕ 2F95	⾥ 2FA5	⾵ 2FB5	⿅ 2FC5	⿕ 2FD5
6	⼆ 2F06	⼖ 2F16	⼦ 2F26	⼶ 2F36	⽆ 2F46	⽖ 2F56	⽦ 2F66	⽶ 2F76	⾆ 2F86	⾖ 2F96	⾦ 2FA6	⾶ 2FB6	⿆ 2FC6	
7	⼇ 2F07	⼗ 2F17	⼧ 2F27	⼷ 2F37	⽇ 2F47	⽗ 2F57	⽧ 2F67	⽷ 2F77	⾇ 2F87	⾗ 2F97	⾧ 2FA7	⾷ 2FB7	⿇ 2FC7	
8	⼈ 2F08	⼘ 2F18	⼨ 2F28	⼸ 2F38	⽈ 2F48	⽘ 2F58	⽨ 2F68	⽸ 2F78	⾈ 2F88	⾘ 2F98	⾨ 2FA8	⾸ 2FB8	⿈ 2FC8	
9	⼉ 2F09	⼙ 2F19	⼩ 2F29	⼹ 2F39	⽉ 2F49	⽙ 2F59	⽩ 2F69	⽹ 2F79	⾉ 2F89	⾙ 2F99	⾩ 2FA9	⾹ 2FB9	⿉ 2FC9	
A	⼊	⼚	⼪	⼺	⽊	⽚	⽪	⽺	⾊	⾚	⾪	⾺	⿊	

Unicode can make NLP throny

- Lots of duplicated characters or glyphs, e.g., there is a cross (U+07d9) in Nko, while there many other crosses in Unicode.
- Invisible characters.
- Potential adversarial attack by using code in different parts of Unicode.
-
    - Bad Characters: Imperceptible NLP Attacks, Boucher et al., 2021
    - Hey, AI software developers, you are taking Unicode into account, right … right?

Unicode can make NLP throny

- Lots of duplicated characters or glyphs, e.g., there is a cross (U+07d9) in Nko, while there many other crosses in Unicode.
- Invisible characters.
- Potential adversarial attack by using code in different parts of Unicode.
-
  - Bad Characters: Imperceptible NLP Attacks, Boucher et al., 2021 https://arxiv.org/pdf/2106.09898.pdf
  - Hey, AI software developers, you are taking Unicode into account, right ... right? https://www.theregister.com/2021/08/06/unicode_ai_bug/

Unicode can make NLP throny

- Lots of duplicated characters or glyphs, e.g., there is a cross (U+07d9) in Nko, while there many other crosses in Unicode.
- Invisible characters.
- Potential adverserial attack by using code in different parts of Unicode.
-
  - Bad Characters: Imperceptible NLP Attacks, Boucher et al., 2021 https://arxiv.org/pdf/2106.09898.pdf

Unicode can make NLP throny

- Lots of duplicated characters or glyphs, e.g., there is a cross (U+07d9) in Nko, while there many other crosses in Unicode.
- Invisible characters.
- Potential adverserial attack by using code in different parts of Unicode.
-
  - Bad Characters: Imperceptible NLP Attacks, Boucher et al., 2021 `https://arxiv.org/pdf/2106.09898.pdf`
  - Hey, AI software developers, you are taking Unicode into account, right ... right? `https://www.theregister.com/2021/08/06/unicode_ai_bug/`

Unicode can make NLP throny

- Lots of duplicated characters or glyphs, e.g., there is a cross (U+07d9) in Nko, while there many other crosses in Unicode.
- Invisible characters.
- Potential adversarial attack by using code in different parts of Unicode.
-
    - Bad Characters: Imperceptible NLP Attacks, Boucher et al., 2021 https://arxiv.org/pdf/2106.09898.pdf
    - Hey, AI software developers, you are taking Unicode into account, right ... right? https://www.theregister.com/2021/08/06/unicode_ai_bug/

Unicode can make NLP throny

- Lots of duplicated characters or glyphs, e.g., there is a cross (U+07d9) in Nko, while there many other crosses in Unicode.
- Invisible characters.
- Potential adverserial attack by using code in different parts of Unicode.
-
    - Bad Characters: Imperceptible NLP Attacks, Boucher et al., 2021 https://arxiv.org/pdf/2106.09898.pdf
    - Hey, AI software developers, you are taking Unicode into account, right ... right? https://www.theregister.com/2021/08/06/unicode_ai_bug/

Text encoding in Python3

- In Python 3, a string is by default in UTF-8.
- https://docs.python.org/3/howto/unicode.html
- There is a separate type for strings, bytes.
- When opening a file, ensure the encoding:

```
In [1]: with open("BIG5.example", 'r') as f:
   ...:     print (list(f))
   ...:
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
<ipython-input-1-91b059321606> in <module>()
      1 with open("BIG5.example", 'r') as f:
----> 2     print (list(f))
      3

/usr/lib/python3.6/codecs.py in decode(self, input, final)
    319         # decode input (taking the buffer into account)
    320         data = self.buffer + input
--> 321         (result, consumed) = self._buffer_decode(data, self.errors, final)
    322         # keep undecoded input until the next call
    323         self.buffer = data[consumed:]

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa4 in position 0: invalid start byte

In [2]: with open("BIG5.example", 'r', encoding="BIG5") as f:
   ...:     print (list(f))
   ...:
['反攻大陸\n', '消滅共匪\n', '\n']
```

Text encoding in Python3

- In Python 3, a string is by default in UTF-8.
- https://docs.python.org/3/howto/unicode.html
- There is a separate type for strings, bytes.
- When opening a file, ensure the encoding:

```
In [1]: with open("BIG5.example", 'r') as f:
   ...:     print (list(f))
   ...:
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
<ipython-input-1-91b059321606> in <module>()
      1 with open("BIG5.example", 'r') as f:
----> 2     print (list(f))
      3

/usr/lib/python3.6/codecs.py in decode(self, input, final)
    319         # decode input (taking the buffer into account)
    320         data = self.buffer + input
--> 321         (result, consumed) = self._buffer_decode(data, self.errors, final)
    322         # keep undecoded input until the next call
    323         self.buffer = data[consumed:]

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa4 in position 0: invalid start byte

In [2]: with open("BIG5.example", 'r', encoding="BIG5") as f:
   ...:     print (list(f))
   ...:
['反攻大陸\n', '消滅共匪\n', '\n']
```

Text encoding in Python3

- In Python 3, a string is by default in UTF-8.
- https://docs.python.org/3/howto/unicode.html
- There is a separate type for strings, bytes.
- When opening a file, ensure the encoding:

```
In [1]: with open("BIG5.example", 'r') as f:
   ...:     print  (list(f))
   ...:
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
<ipython-input-1-91b059321606> in <module>()
      1 with open("BIG5.example", 'r') as f:
----> 2     print  (list(f))
      3

/usr/lib/python3.6/codecs.py in decode(self, input, final)
    319         # decode input (taking the buffer into account)
    320         data = self.buffer + input
--> 321         (result, consumed) = self._buffer_decode(data, self.errors, final)
    322         # keep undecoded input until the next call
    323         self.buffer = data[consumed:]

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa4 in position 0: invalid start byte

In [2]: with open("BIG5.example", 'r', encoding="BIG5") as f:
   ...:     print  (list(f))
   ...:
['反攻大陸\n', '消滅共匪\n', '\n']
```

Text encoding in Python3

- In Python 3, a string is by default in UTF-8.
- https://docs.python.org/3/howto/unicode.html
- There is a separate type for strings, bytes.
- When opening a file, ensure the encoding:

```
In [1]: with open("BIG5.example", 'r') as f:
   ...:     print  (list(f))
   ...:
---------------------------------------------------------------------------
UnicodeDecodeError                        Traceback (most recent call last)
<ipython-input-1-91b059321606> in <module>()
      1 with open("BIG5.example", 'r') as f:
----> 2     print  (list(f))
      3

/usr/lib/python3.6/codecs.py in decode(self, input, final)
    319         # decode input (taking the buffer into account)
    320         data = self.buffer + input
--> 321         (result, consumed) = self._buffer_decode(data, self.errors, final)
    322         # keep undecoded input until the next call
    323         self.buffer = data[consumed:]

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xa4 in position 0: invalid start byte

In [2]: with open("BIG5.example", 'r', encoding="BIG5") as f:
   ...:     print  (list(f))
   ...:
['反攻大陸\n', '消滅共匪\n', '\n']
```

Beyond plain text

Some text data are more than *plain text*, including information about formatting and field/structure.

- PDF: font and coordinate for each character – no strings
- (Encapsulated) Postscript
- Markup languages: HTML, XML (including Microsoft Office XML). Easier but not that easy: e.g., the same type of information is in different tags or tags of different attributes, e.g.,

```
<a href="a.html" id="first_link">Go back</a>
<a href="b.html" id="second_link">Next</a>
```

- (The evil) Microsoft formats (prior to Office XML)
- Dictionary-like data: YAML, JSON

Outline

Pipeline for NLP preprocessing

You need to have the right data! (Although a lot of research is "garbage in, garbage out")

1. Download the corpus or Crawl the web or other ways to get the text (e.g., OCR)
2. If the data is (semi-)structured, parse it, e.g., parsing the HTML.
3. Clean up: handling errors (e.g., introduced by OCR or text extraction from Powerpoint slides), and others...
4. Tokenization
5. normalization: turn all tokens to their canonical form (including stemming/lemmatization)
6. Optionally, you wanna index your text so you can locate them easily later. String search will take very long.

Some text data are more difficult to handle than others, e.g., Twitter vs. NYTimes.

Pipeline for NLP preprocessing

You need to have the right data! (Although a lot of research is "garbage in, garbage out")

1. Download the corpus or Crawl the web or other ways to get the text (e.g., OCR)
2. If the data is (semi-)structured, parse it, e.g., parsing the HTML.
3. Clean up: handling errors (e.g., introduced by OCR or text extraction from Powerpoint slides), and others...
4. Tokenization
5. normalization: turn all tokens to their canonical form (including stemming/lemmatization)
6. Optionally, you wanna index your text so you can locate them easily later. String search will take very long.

Some text data are more difficult to handle than others, e.g., Twitter vs. NYTimes.

Pipeline for NLP preprocessing

You need to have the right data! (Although a lot of research is "garbage in, garbage out")

1. Download the corpus or Crawl the web or other ways to get the text (e.g., OCR)
2. If the data is (semi-)structured, parse it, e.g., parsing the HTML.
3. Clean up: handling errors (e.g., introduced by OCR or text extraction from Powerpoint slides), and others...
4. Tokenization
5. normalization: turn all tokens to their canonical form (including stemming/lemmatization)
6. Optionally, you wanna index your text so you can locate them easily later. String search will take very long.

Some text data are more difficult to handle than others, e.g., Twitter vs. NYTimes.

Pipeline for NLP preprocessing

You need to have the right data! (Although a lot of research is “garbage in, garbage out”)

1. Download the corpus or Crawl the web or other ways to get the text (e.g., OCR)
2. If the data is (semi-)structured, parse it, e.g., parsing the HTML.
3. Clean up: handling errors (e.g., introduced by OCR or text extraction from Powerpoint slides), and others...
4. Tokenization
5. normalization: turn all tokens to their canonical form (including stemming/lemmatization)
6. Optionally, you wanna index your text so you can locate them easily later. String search will take very long.

Some text data are more difficult to handle than others, e.g., Twitter vs. NYTimes.

Pipeline for NLP preprocessing

You need to have the right data! (Although a lot of research is "garbage in, garbage out")

1. Download the corpus or Crawl the web or other ways to get the text (e.g., OCR)
2. If the data is (semi-)structured, parse it, e.g., parsing the HTML.
3. Clean up: handling errors (e.g., introduced by OCR or text extraction from Powerpoint slides), and others...
4. Tokenization
5. normalization: turn all tokens to their canonical form (including stemming/lemmatization)
6. Optionally, you wanna index your text so you can locate them easily later. String search will take very long.

Some text data are more difficult to handle than others, e.g., Twitter vs. NYTimes.

Pipeline for NLP preprocessing

You need to have the right data! (Although a lot of research is "garbage in, garbage out")

1. Download the corpus or Crawl the web or other ways to get the text (e.g., OCR)
2. If the data is (semi-)structured, parse it, e.g., parsing the HTML.
3. Clean up: handling errors (e.g., introduced by OCR or text extraction from Powerpoint slides), and others...
4. Tokenization
5. normalization: turn all tokens to their canonical form (including stemming/lemmatization)
6. Optionally, you wanna index your text so you can locate them easily later. String search will take very long.

Some text data are more difficult to handle than others, e.g., Twitter vs. NYTimes.

Outline

Corpus

- The body of text to be processed. This class focuses on English.
- E.g., all articles on Wikipedia form a corpus.
- E.g., all news from a newspaper form another corpus.
- There are many corpora, e.g.,

```
    nltk.corpus.AlignedCorpusReader
nltk.corpus.AlpinoCorpusReader
nltk.corpus.BNCCorpusReader
nltk.corpus.BracketParseCorpusReader
nltk.corpus.CHILDESCorpusReader
nltk.corpus.CMUDictCorpusReader
nltk.corpus.CategorizedBracketParseCorpusReader
nltk.corpus.CategorizedCorpusReader
nltk.corpus.CategorizedPlaintextCorpusReader
nltk.corpus.CategorizedSentencesCorpusReader
nltk.corpus.CategorizedTaggedCorpusReader
nltk.corpus.ChasenCorpusReader
nltk.corpus.ChunkedCorpusReader
nltk.corpus.ComparativeSentencesCorpusReader
nltk.corpus.ConllChunkCorpusReader
nltk.corpus.ConllCorpusReader
nltk.corpus.CorpusReader
nltk.corpus.CrubadanCorpusReader
nltk.corpus.DependencyCorpusReader
nltk.corpus.EuroparlCorpusReader
nltk.corpus.FramenetCorpusReader
nltk.corpus.IEERCorpusReader
nltk.corpus.IPIPANCorpusReader
nltk.corpus.IndianCorpusReader
nltk.corpus.KNBCorpusReader
```

```
    nltk.corpus.PropbankCorpusRe
nltk.corpus.ProsConsCorpusReade
nltk.corpus.RTECorpusReader
nltk.corpus.RegexpTokenizer
nltk.corpus.ReviewsCorpusReader
nltk.corpus.SemcorCorpusReader
nltk.corpus.SensevalCorpusReade
nltk.corpus.SentiSynset
nltk.corpus.SentiWordNetCorpusRe
nltk.corpus.SinicaTreebankCorpu
nltk.corpus.StringCategoryCorpu
nltk.corpus.SwadeshCorpusReader
nltk.corpus.SwitchboardCorpusRe
nltk.corpus.SyntaxCorpusReader
nltk.corpus.TEICorpusView
nltk.corpus.TaggedCorpusReader
nltk.corpus.TimitCorpusReader
nltk.corpus.TimitTaggedCorpusRe
nltk.corpus.ToolboxCorpusReader
nltk.corpus.TwitterCorpusReader
nltk.corpus.UdhrCorpusReader
nltk.corpus.UnicharsCorpusReade
nltk.corpus.VerbnetCorpusReader
nltk.corpus.WordListCorpusReade
nltk.corpus.WordNetCorpusReader
```

Another dataset catalog: Tensorflow datasets

https://www.tensorflow.org/datasets/catalog/overview

A corpus ?= a set of sentences

- Building a corpus is not that easy when your input data is not plain-text sentences only.
- Should you include non-sentence text (table content, headings, external links, etc.) from Wikipedia into the corpus when training word embeddings?
- What about style information? Should "*this* is not right" as a line in a movie be treated as "this is not right"?
- What about equations in a paper? (Suppose it is in LaTeX/MathML)

The lack of labeled corpora is limiting NLP research

- Having the text alone is not enough for many NLP (research) tasks.
- NLP tasks are far more complicated and diverse than those in other AI areas, such as CV (where most of the tasks are related to object detection/recognition for which anyone can be a human annotator).
- For example, for years, there is no dataset for using supervised approach to text summarization – no corpus has sentence-level binary labels.
- As another example, QA research was a chaos before Stanford SQUAD dataset.
- Today, lots of research has no common ground truth. People publish papers on their own small-scale datasets, e.g., REALSumm and SummEval have only 100 articles annotated.
- It is also wrong to justify: “because everyone else used this corpus, so do I.” (“IBM does this xxx. We don’t do this xxx.” –Steve Jobs)

The lack of labeled corpora is limiting NLP research

- Having the text alone is not enough for many NLP (research) tasks.
- NLP tasks are far more complicated and diverse than those in other AI areas, such as CV (where most of the tasks are related to object detection/recognition for which anyone can be a human annotator).
- For example, for years, there is no dataset for using supervised approach to text summarization – no corpus has sentence-level binary labels.
- As another example, QA research was a chaos before Stanford SQUAD dataset.
- Today, lots of research has no common ground truth. People publish papers on their own small-scale datasets, e.g., REALSumm and SummEval have only 100 articles annotated.
- It is also wrong to justify: “because everyone else used this corpus, so do I.” (“IBM does this xxx. We don’t do this xxx.” –Steve Jobs)

The lack of labeled corpora is limiting NLP research

- Having the text alone is not enough for many NLP (research) tasks.
- NLP tasks are far more complicated and diverse than those in other AI areas, such as CV (where most of the tasks are related to object detection/recognition for which anyone can be a human annotator).
- For example, for years, there is no dataset for using supervised approach to text summarization – no corpus has sentence-level binary labels.
- As another example, QA research was a chaos before Stanford SQUAD dataset.
- Today, lots of research has no common ground truth. People publish papers on their own small-scale datasets, e.g., REALSumm and SummEval have only 100 articles annotated.
- It is also wrong to justify: "because everyone else used this corpus, so do I." ("IBM does this xxx. We don't do this xxx." –Steve Jobs)

The lack of labeled corpora is limiting NLP research

- Having the text alone is not enough for many NLP (research) tasks.
- NLP tasks are far more complicated and diverse than those in other AI areas, such as CV (where most of the tasks are related to object detection/recognition for which anyone can be a human annotator).
- For example, for years, there is no dataset for using supervised approach to text summarization – no corpus has sentence-level binary labels.
- As another example, QA research was a chaos before Stanford SQUAD dataset.
- Today, lots of research has no common ground truth. People publish papers on their own small-scale datasets, e.g., REALSumm and SummEval have only 100 articles annotated.
- It is also wrong to justify: "because everyone else used this corpus, so do I." ("IBM does this xxx. We don't do this xxx." –Steve Jobs)

The lack of labeled corpora is limiting NLP research

- Having the text alone is not enough for many NLP (research) tasks.
- NLP tasks are far more complicated and diverse than those in other AI areas, such as CV (where most of the tasks are related to object detection/recognition for which anyone can be a human annotator).
- For example, for years, there is no dataset for using supervised approach to text summarization – no corpus has sentence-level binary labels.
- As another example, QA research was a chaos before Stanford SQUAD dataset.
- Today, lots of research has no common ground truth. People publish papers on their own small-scale datasets, e.g., REALSumm and SummEval have only 100 articles annotated.
- It is also wrong to justify: "because everyone else used this corpus, so do I." ("IBM does this xxx. We don't do this xxx." –Steve Jobs)

The lack of labeled corpora is limiting NLP research

- Having the text alone is not enough for many NLP (research) tasks.
- NLP tasks are far more complicated and diverse than those in other AI areas, such as CV (where most of the tasks are related to object detection/recognition for which anyone can be a human annotator).
- For example, for years, there is no dataset for using supervised approach to text summarization – no corpus has sentence-level binary labels.
- As another example, QA research was a chaos before Stanford SQUAD dataset.
- Today, lots of research has no common ground truth. People publish papers on their own small-scale datasets, e.g., REALSumm and SummEval have only 100 articles annotated.
- It is also wrong to justify: "because everyone else used this corpus, so do I." ("IBM does this xxx. We don't do this xxx." –Steve Jobs)

Corpora are very diverse

- We are in a very open domain.
- But datasets differ in the way that they are stored (e.g., JSON), organized and shared.
- Corpora are floating in the homepages of researchers in various ways.
- Example 1: `http://jmcauley.ucsd.edu/data/amazon/`
- Example 2: `https://webscope.sandbox.yahoo.com/catalog.php?datatype=l`
- Example 3: `https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets`
- Example 4: `https://catalog.ldc.upenn.edu/ldc2008t19`
- Lots of subjectivity in annotated data.

Corpora are very diverse

- We are in a very open domain.
- But datasets differ in the way that they are stored (e.g., JSON), organized and shared.
- Corpora are floating in the homepages of researchers in various ways.
- Example 1: `http://jmcauley.ucsd.edu/data/amazon/`
- Example 2: `https://webscope.sandbox.yahoo.com/catalog.php?datatype=l`
- Example 3: `https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets`
- Example 4: `https://catalog.ldc.upenn.edu/ldc2008t19`
- Lots of subjectivity in annotated data.

Corpora are very diverse

- We are in a very open domain.
- But datasets differ in the way that they are stored (e.g., JSON), organized and shared.
- Corpora are floating in the homepages of researchers in various ways.
- Example 1: `http://jmcauley.ucsd.edu/data/amazon/`
- Example 2: `https://webscope.sandbox.yahoo.com/catalog.php?datatype=l`
- Example 3: `https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets`
- Example 4: `https://catalog.ldc.upenn.edu/ldc2008t19`
- Lots of subjectivity in annotated data.

Corpora are very diverse

- We are in a very open domain.
- But datasets differ in the way that they are stored (e.g., JSON), organized and shared.
- Corpora are floating in the homepages of researchers in various ways.
- Example 1: `http://jmcauley.ucsd.edu/data/amazon/`
- Example 2: `https://webscope.sandbox.yahoo.com/catalog.php?datatype=l`
- Example 3: `https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets`
- Example 4: `https://catalog.ldc.upenn.edu/ldc2008t19`
- Lots of subjectivity in annotated data.

Corpora are very diverse

- We are in a very open domain.
- But datasets differ in the way that they are stored (e.g., JSON), organized and shared.
- Corpora are floating in the homepages of researchers in various ways.
- Example 1: `http://jmcauley.ucsd.edu/data/amazon/`
- Example 2: `https://webscope.sandbox.yahoo.com/catalog.php?datatype=l`
- Example 3: `https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets`
- Example 4: `https://catalog.ldc.upenn.edu/ldc2008t19`
- Lots of subjectivity in annotated data.

Corpora are very diverse

- We are in a very open domain.
- But datasets differ in the way that they are stored (e.g., JSON), organized and shared.
- Corpora are floating in the homepages of researchers in various ways.
- Example 1: http://jmcauley.ucsd.edu/data/amazon/
- Example 2: https://webscope.sandbox.yahoo.com/catalog.php?datatype=l
- Example 3: https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets
- Example 4: https://catalog.ldc.upenn.edu/ldc2008t19
- Lots of subjectivity in annotated data.

Corpora are very diverse

- We are in a very open domain.
- But datasets differ in the way that they are stored (e.g., JSON), organized and shared.
- Corpora are floating in the homepages of researchers in various ways.
- Example 1: `http://jmcauley.ucsd.edu/data/amazon/`
- Example 2: `https://webscope.sandbox.yahoo.com/catalog.php?datatype=l`
- Example 3: `https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets`
- Example 4: `https://catalog.ldc.upenn.edu/ldc2008t19`
- Lots of subjectivity in annotated data.

Corpora are very diverse

- We are in a very open domain.
- But datasets differ in the way that they are stored (e.g., JSON), organized and shared.
- Corpora are floating in the homepages of researchers in various ways.
- Example 1: http://jmcauley.ucsd.edu/data/amazon/
- Example 2: https://webscope.sandbox.yahoo.com/catalog.php?datatype=l
- Example 3: https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets
- Example 4: https://catalog.ldc.upenn.edu/ldc2008t19
- Lots of subjectivity in annotated data.

Crawling data from the Web (Python 2.x)

- Often times, the data is not available. So you crawl. (So do Google, Bing, and many others.)
- In Python, you may use `urllib2`:

```
import urllib2
response = urllib2.urlopen('http://gozips.
    uakron.edu/~fbao5/')
html=response.read()
```

- What if the webpage is dynamic? With the help of another library `urllib`. Example below for POST method.

```
import urllib
url="http://10.24.47.178/search.py"
values = {"single":"landing"}
data=urllib.urlencode(values)
response=urllib2.urlopen(urllib2.Request(
    url, data))
html=response.read()
```

Make your crawler automatically cool down

An example for using `sysvinit` back in 2015, placed under `/etc/init`. Recent Linux distros use `systemd`.

```
description     "Airbnb review fetcher"

setuid forrest
setgid forrest

start on (net-device-up
          and local-filesystems
          and runlevel [2345])
stop on runlevel [016]

respawn
respawn limit 10 5
limit nofile 500000 500000

env LANG=en_US.UTF-8
env LC_CTYPE=en_US.UTF-8

script
  exec /home/forrest/Apps/airbnb/bin/python\
  /home/forrest/Apps/airbnb-reviews/tools/app 2>&1 >> \
  /home/forrest/airbnb_review_fetcher.log
end script
```

Thanks for Wu Jiang, head of robotics research at Boxed Labs.

Parse HTML

See iPython notebook notes.

Index your document (if applicable)

- To allow easy localization of string in documents, you wanna build an inversed index.
- The very basic searching or information retrieval.
- See a different iPython notebook notes.

Outline

Why data clean up is needed

Run the code below and compare the sample text from three summarization datasets.

```
import tensorflow_datasets as tfds

for piece in tfds.load("cnn_dailymail",split="test"):
    print (piece['article'])
    break

for piece in tfds.load("billsum",split="test"):
    print (piece['text'])
    break

for piece in tfds.load("big_patent",split="test"):
    print (piece['description'])
    break
```

Clean up the text is not that easy

- `''we are happy''.split()`
- The ambiguity of non-alphanumeric symbols, e.g., the dash in "low-hanging fruit" vs. "ex-
press", or the spaces between "New York City" ("New York" + "City" or "New" + "York" + "City"?)
- You cannot change all words to lowercase, e.g., "It's a CAT 4 hurricane." Kitty or category?
- Numbers make things complicated, e.g., CO2.
- Some words shouldn't be separated, e.g., in vitro, de facto, et al.
- A common practice is dropping all non-alphabets – but it won't work for technical or scientific documents.

Help the downstream tasks

- Suppose you are processing a post from StackOverflow: "Apply `get` onto the method `cat`."
- Downstream tasks like POS tagging may fail on "get" and "cat".
- Some string substitution is needed.

Help the downstream tasks

- Suppose you are processing a post from StackOverflow: "Apply `get` onto the method `cat`."
- Downstream tasks like POS tagging may fail on "get" and "cat".
- Some string substitution is needed.

Help the downstream tasks

- Suppose you are processing a post from StackOverflow: "Apply `get` onto the method `cat`."
- Downstream tasks like POS tagging may fail on "get" and "cat".
- Some string substitution is needed.

Outline

Word Tokenization

- Carrying an concrete meaning, a token is the basic element to construct a sentence – at least linguistics say so.
- Things are not as easy as

```
"This is a class about NLP".split()
```

- What are acceptable tokens?
  - Should contractions like "n't" in "he isn't happy" be split?
  - "We see 2 people in the park after 3pm" Should we reject all digits? Are "2" and "3" semantically connected here?
  - Shall we break "New York"?
- Some tokenization is fairly simple. https://web.archive.org/web/19970614072242if_/http://www.cis.upenn.edu:80/~treebank/tokenization.html

Word Tokenization

- Carrying an concrete meaning, a token is the basic element to construct a sentence – at least linguistics say so.
- Things are not as easy as

```
"This is a class about NLP".split()
```

- What are acceptable tokens?
  - Should contractions like "n't" in "he isn't happy" be split?
  - [illegible]
  - [illegible]
- Some tokenization is fairly simple.
  https://web.archive.org/web/19970614072242if_/http://www.cis.upenn.edu:80/~treebank/tokenization.html

Word Tokenization

- Carrying an concrete meaning, a token is the basic element to construct a sentence – at least linguistics say so.
- Things are not as easy as

```
"This is a class about NLP".split()
```

- What are acceptable tokens?
  - Should contractions like “n’t” in “he isn’t happy” be split?
  - “We see 2 people in the park after 3pm” Should we reject all digits? Are “2” and “3” semantically connected here?
  - Shall we break “New York”?
- Some tokenization is fairly simple.
  https://web.archive.org/web/19970614072242if_/http://www.cis.upenn.edu:80/~treebank/tokenization.html

Word Tokenization

- Carrying an concrete meaning, a token is the basic element to construct a sentence – at least linguistics say so.
- Things are not as easy as

```
"This is a class about NLP".split()
```

- What are acceptable tokens?
  - Should contractions like “n’t” in “he isn’t happy” be split?
  - “We see 2 people in the park after 3pm” Should we reject all digits? Are “2” and “3” semantically connected here?
  - Shall we break “New York”?
- Some tokenization is fairly simple. `https://web.archive.org/web/19970614072242if_/http://www.cis.upenn.edu:80/~treebank/tokenization.html`

Word Tokenization

- Carrying an concrete meaning, a token is the basic element to construct a sentence – at least linguistics say so.
- Things are not as easy as

```
"This is a class about NLP".split()
```

- What are acceptable tokens?
  - Should contractions like “n’t” in “he isn’t happy” be split?
  - “We see 2 people in the park after 3pm” Should we reject all digits? Are “2” and “3” semantically connected here?
  - Shall we break “New York”?
- Some tokenization is fairly simple. https://web.archive.org/web/19970614072242if_/http://www.cis.upenn.edu:80/~treebank/tokenization.html

Word Tokenization

- Carrying an concrete meaning, a token is the basic element to construct a sentence – at least linguistics say so.
- Things are not as easy as

```
"This is a class about NLP".split()
```

- What are acceptable tokens?
  - Should contractions like “n’t” in “he isn’t happy” be split?
  - “We see 2 people in the park after 3pm” Should we reject all digits? Are “2” and “3” semantically connected here?
  - Shall we break “New York”?
- Some tokenization is fairly simple.
  `https://web.archive.org/web/19970614072242if_/http://www.cis.upenn.edu:80/~treebank/tokenization.html`

Word Tokenization

- Carrying an concrete meaning, a token is the basic element to construct a sentence – at least linguistics say so.
- Things are not as easy as

```
"This is a class about NLP".split()
```

- What are acceptable tokens?
  - Should contractions like “n’t” in “he isn’t happy” be split?
  - “We see 2 people in the park after 3pm” Should we reject all digits? Are “2” and “3” semantically connected here?
  - Shall we break “New York”?
- Some tokenization is fairly simple. https://web.archive.org/web/19970614072242if_/http://www.cis.upenn.edu:80/~treebank/tokenization.html

Regular expression

- A regular expression defines how to generate a string by drawing characters from alphabets. You can draw characters from different sets at different "steps", like the example $(abc)*(123)?$ below.
- $(a)*$ means { ϵ, a, aa, aaa, aaaa, ... }, where ϵ is the empty string.
- $(ab)*$ means { ϵ, ab, abab, ababab, abababab, ... }
- $(a|b)*$ means { ϵ, a, b, ab, ba, aaa, aab, abb, bbb, bba, baa, bab, ... } where $|$ means "or".
- $(a)+$ means { a, aa, aaa, } because $+$ means repeating at least once.
- $(ab)?$ means { ab, ϵ} because ? means once or none.
- $(abc)*(123)?$ means {ϵ, abc, abc123, abcabc, abcabc123, ... } any string that begins with the string "abc" and ends with or without the string "123".
- $0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)+$ means any natural number in common writing format. Note that it does not allow an integer to start with 0.

Regular expression

- A regular expression defines how to generate a string by drawing characters from alphabets. You can draw characters from different sets at different “steps”, like the example $(abc)*(123)?$ below.
- $(a)*$ means { ϵ, a, aa, aaa, aaaa, ... }, where ϵ is the empty string.
- $(ab)*$ means { ϵ, ab, abab, ababab, abababab, ... }
- $(a|b)*$ means { ϵ, a, b, ab, ba, aaa, aab, abb, bbb, bba, baa, bab, ... } where $|$ means “or”.
- $(a)+$ means { a, aa, aaa, } because + means repeating at least once.
- $(ab)?$ means { ab, ϵ} because ? means once or none.
- $(abc)*(123)?$ means {ϵ, abc, abc123, abcabc, abcabc123, ... } any string that begins with the string “abc” and ends with or without the string “123”.
- $0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)+$ means any natural number in common writing format. Note that it does not allow an integer to start with 0.

Regular expression

- A regular expression defines how to generate a string by drawing characters from alphabets. You can draw characters from different sets at different "steps", like the example $(abc)*(123)?$ below.
- $(a)*$ means { ϵ, a, aa, aaa, aaaa, ... }, where ϵ is the empty string.
- $(ab)*$ means { ϵ, ab, abab, ababab, abababab, ... }
- $(a|b)*$ means { ϵ, a, b, ab, ba, aaa, aab, abb, bbb, bba, baa, bab, ... } where $|$ means "or".
- $(a)+$ means { a, aa, aaa, } because $+$ means repeating at least once.
- $(ab)?$ means { ab, ϵ} because ? means once or none.
- $(abc)*(123)?$ means {ϵ, abc, abc123, abcabc, abcabc123, ... } any string that begins with the string "abc" and ends with or without the string "123".
- $0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)+$ means any natural number in common writing format. Note that it does not allow an integer to start with 0.

Regular expression

- A regular expression defines how to generate a string by drawing characters from alphabets. You can draw characters from different sets at different “steps”, like the example $(abc)*(123)?$ below.
- $(a)*$ means { ϵ, a, aa, aaa, aaaa, ... }, where ϵ is the empty string.
- $(ab)*$ means { ϵ, ab, abab, ababab, abababab, ... }
- $(a|b)*$ means { ϵ, a, b, ab, ba, aaa, aab, abb, bbb, bba, baa, bab, ... } where $|$ means “or”.
- $(a)+$ means { a, aa, aaa, } because + means repeating at least once.
- $(ab)?$ means { ab, ϵ} because ? means once or none.
- $(abc)*(123)?$ means {ϵ, abc, abc123, abcabc, abcabc123, ... } any string that begins with the string “abc” and ends with or without the string “123”.
- $0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)+$ means any natural number in common writing format. Note that it does not allow an integer to start with 0.

Regular expression

- A regular expression defines how to generate a string by drawing characters from alphabets. You can draw characters from different sets at different "steps", like the example $(abc)*(123)?$ below.
- $(a)*$ means { ϵ, a, aa, aaa, aaaa, ... }, where ϵ is the empty string.
- $(ab)*$ means { ϵ, ab, abab, ababab, abababab, ... }
- $(a|b)*$ means { ϵ, a, b, ab, ba, aaa, aab, abb, bbb, bba, baa, bab, ... } where $|$ means "or".
- $(a)+$ means { a, aa, aaa, } because + means repeating at least once.
- $(ab)?$ means { ab, ϵ} because ? means once or none.
- $(abc)*(123)?$ means {ϵ, abc, abc123, abcabc, abcabc123, ... } any string that begins with the string "abc" and ends with or without the string "123".
- $0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)+$ means any natural number in common writing format. Note that it does not allow an integer to start with 0.

Regular expression

- A regular expression defines how to generate a string by drawing characters from alphabets. You can draw characters from different sets at different "steps", like the example $(abc)*(123)?$ below.
- $(a)*$ means { ϵ, a, aa, aaa, aaaa, ... }, where ϵ is the empty string.
- $(ab)*$ means { ϵ, ab, abab, ababab, abababab, ... }
- $(a|b)*$ means { ϵ, a, b, ab, ba, aaa, aab, abb, bbb, bba, baa, bab, ... } where $|$ means "or".
- $(a)+$ means { a, aa, aaa, } because + means repeating at least once.
- $(ab)?$ means { ab, ϵ} because ? means once or none.
- $(abc)*(123)?$ means {ϵ, abc, abc123, abcabc, abcabc123, ... } any string that begins with the string "abc" and ends with or without the string "123".
- $0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)+$ means any natural number in common writing format. Note that it does not allow an integer to start with 0.

Regular expression

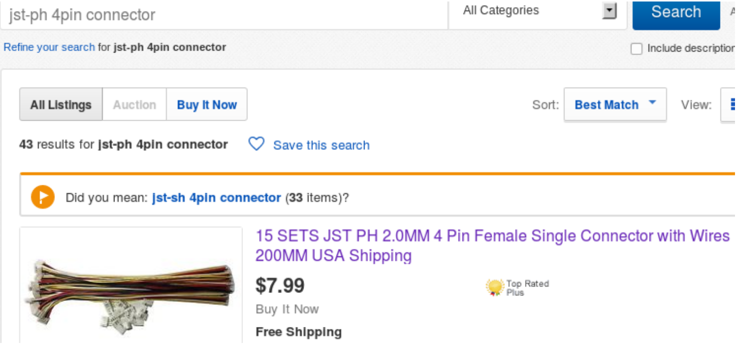
- A regular expression defines how to generate a string by drawing characters from alphabets. You can draw characters from different sets at different "steps", like the example $(abc)*(123)?$ below.
- $(a)*$ means { ϵ, a, aa, aaa, aaaa, ... }, where ϵ is the empty string.
- $(ab)*$ means { ϵ, ab, abab, ababab, abababab, ... }
- $(a|b)*$ means { ϵ, a, b, ab, ba, aaa, aab, abb, bbb, bba, baa, bab, ... } where $|$ means "or".
- $(a)+$ means { a, aa, aaa, } because + means repeating at least once.
- $(ab)?$ means { ab, ϵ} because ? means once or none.
- $(abc)*(123)?$ means {ϵ, abc, abc123, abcabc, abcabc123, ... } any string that begins with the string "abc" and ends with or without the string "123".
- $0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)+$ means any natural number in common writing format. Note that it does not allow an integer to start with 0.

Regular expression

- A regular expression defines how to generate a string by drawing characters from alphabets. You can draw characters from different sets at different "steps", like the example $(abc)*(123)?$ below.
- $(a)*$ means { ϵ, a, aa, aaa, aaaa, ... }, where ϵ is the empty string.
- $(ab)*$ means { ϵ, ab, abab, ababab, abababab, ... }
- $(a|b)*$ means { ϵ, a, b, ab, ba, aaa, aab, abb, bbb, bba, baa, bab, ... } where $|$ means "or".
- $(a)+$ means { a, aa, aaa, } because + means repeating at least once.
- $(ab)?$ means { ab, ϵ} because ? means once or none.
- $(abc)*(123)?$ means {ϵ, abc, abc123, abcabc, abcabc123, ... } any string that begins with the string "abc" and ends with or without the string "123".
- $0|(1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)+$ means any natural number in common writing format. Note that it does not allow an integer to start with 0.

Use regular expressions

- See iPython notebook notes: the syntax of regex for most UNIX-like OSes and C-style languages may be different from the syntax we just saw but the principles are the same.
- Do not tokenize stupidly.



- You may need to craft your own tokenizer. https://stackoverflow.com/questions/15929233/writing-a-tokenizer-in-python

Regexp-based feature engineering

https://github.com/forrestbao/autoscholar/commit/b732ae5fa14d65558cd0eff7a7a9e103ea7cd937

Get our hands dirty!

- NLTK: Natural Language Toolkit, classical NLP toolkit in Python
- SpaCy: Modern, fancy, claimed to be industry-level, friendly APIs, leveraging GPUs (not for all tasks)
- Stanza: from Stanford NLP group, derived from Core (also model-based), native in Python, models trained in PyTorch, leveraging GPU

Tokenization and Sentence segmentation in NLTK

Using default ones:

```
In [1]: import nltk

In [2]: nltk.tokenize.word_tokenize \
...: ("I am happy. mr. Wang is happy")
Out[2]: ['I', 'am', 'happy', '.', 'mr.', 'Wang', 'is',
        'happy']

In [3]: nltk.tokenize.sent_tokenize \
   ...:  ("I am happy. mr. Wang is happy")
Out[3]: ['I am happy.', 'mr. Wang is also happy']

In [4]: nltk.tokenize.sent_tokenize \
   ...:  ("I am happy, mr. wang is happy")
Out[4]: ['I am happy, mr. wang is also happy']
```

As you can see, sentence segmentation heavily depends on puncuations. The default sentencizer in NLTK is Punkt.
Many other varieties at
https://www.nltk.org/api/nltk.tokenize.html

Tokenization and sentence segmentation in Spacy

Pipeline-based NLP.
https://spacy.io/usage/spacy-101#pipelines

```
In [32]: nlp=spacy.load("en_core_web_sm", \
    ...: exclude=["tok2vec",'tagger','parser','ner', 'attribute_ruler', 'lemmatizer'])
    ...: nlp.add_pipe"sentencizer")
Out[32]: <spacy.pipeline.sentencizer.Sentencizer at 0x7f487e80fdc0>

In [33]: [[sent.text for sent in doc.sents ]\
    ...:             for doc in nlp.pipe(\
    ...: ['today is monday. it is the first day.', 'soup is yammy. pizza sucks.'])]
Out[33]:
         [['today is monday.', 'it is the first day.'],
          ['soup is yammy.', 'pizza sucks.']]

In [34]: [[word.text for word in doc ]
    ...:             for doc in nlp.pipe(\
    ...: ['today is monday. it is the first day.', 'soup is yammy. pizza sucks.'])]
Out[34]: [['today', 'is', 'monday', '.', 'it', 'is', 'the', 'first', 'day', '.'],
          ['soup', 'is', 'yammy', '.', 'pizza', 'sucks', '.']]
```

In Stanza (model-based, slower)

Like SpaCy, Stanza is also pipelined. But it takes one string rather than a collection of strings as the input.

```
In [1]: import stanza

In [2]: nlp = stanza.Pipeline(lang='en', processors='tokenize')
2021-08-26 14:15:27 INFO: Loading these models for language: en (English):
=======================
| Processor | Package |
-----------------------
| tokenize  | ewt     |
=======================

In [7]: sentences = 'i am very very happy. the weather is very good.'

In [8]: [[token.text for token in sentence.tokens] for sentence in nlp(sentences).sentences]
Out[8]:
 [['i', 'am', 'very', 'very', 'happy', '.'],
  ['the', 'weather', 'is', 'very', 'good', '.']]

In [9]: [sentence.text for sentence in nlp(sentences).sentences]
Out[9]: ['i am very very happy.', 'the weather is very good.']
```

Other tokenizers

- WordPiece, used in BERT, supports subword
- SentPiece, used in XLnet,

Outline

Stemming and lemmatization

- The English language has inflexions, *inflectional morphology*. We modify words for different grammatical purposes. “I give him a book” vs. “He gave me a book.”
- A language of very little inflexion is called an *analytic language* or an *isolating language*.
- Most mainland southeast Asian and Oceanic languages (Chinese, Vietnamese, Thai, etc.) are isolating language.
- Nevertheless, English is almost the most isolating language among *synthetic languages*, such as High German or Scottish Gaelic.

Stemming and lemmatization

- The English language has inflexions, *inflectional morphology*. We modify words for different grammatical purposes. “I give him a book” vs. “He gave me a book.”
- A language of very little inflexion is called an *analytic language* or an *isolating language*.
- Most mainland southeast Asian and Oceanic languages (Chinese, Vietnamese, Thai, etc.) are isolating language.
- Nevertheless, English is almost the most isolating language among *synthetic languages*, such as High German or Scottish Gaelic.

Stemming and lemmatization

- The English language has inflexions, *inflectional morphology*. We modify words for different grammatical purposes. “I give him a book” vs. “He gave me a book.”
- A language of very little inflexion is called an *analytic language* or an *isolating language*.
- Most mainland southeast Asian and Oceanic languages (Chinese, Vietnamese, Thai, etc.) are isolating language.
- Nevertheless, English is almost the most isolating language among *synthetic languages*, such as High German or Scottish Gaelic.

Stemming and lemmatization

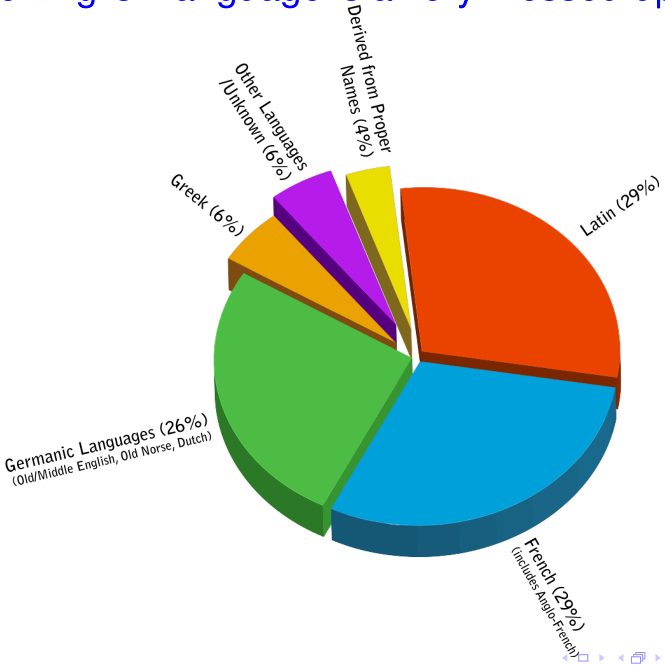
- The English language has inflexions, *inflectional morphology*. We modify words for different grammatical purposes. “I give him a book” vs. “He gave me a book.”
- A language of very little inflexion is called an *analytic language* or an *isolating language*.
- Most mainland southeast Asian and Oceanic languages (Chinese, Vietnamese, Thai, etc.) are isolating language.
- Nevertheless, English is almost the most isolating language among *synthetic languages*, such as High German or Scottish Gaelic.

The English language is a very messed-up one. I

1. Ðan ſhe ðiſſe Old Anȝliſh Tunȝan eode ſpecan
2. Than she ȝan to-spaken ðis Middle Englysshe Tonȝue
3. Then she wente to spake this Early Modern Englysh Tongue
4. Then she went to speak this Late Modern English Tongue

“English irregular verbs” are actually “regular verbs” in Germanic languages.

The English language is a very messed-up one. II



Stemming and lemmatization

- For many tasks, we want the words “dog” and “dogs” to be considered the same.
- Stemming: get the stem, e.g., removing the affix.
- Stemming may not be enough:
  - “I saw you” should become “I see you”
- Lemmatization: get the lemma, the item word in a dictionary.

seek
/sēk/

verb
verb: **seek**; 3rd person present: **seeks**; past tense: **sought**; past participle: **sought**; gerund or present participle: **seeking**

- How to build a lemmatizer?
  - Take advantage of the dictionary
  - What about words beyond? E.g., “transformative” (NSF-style) vs “transforming” (a real English word in dictionaries)
- Don’t take things for granted: NLTK’s “The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.” which means that “lying” will not be restored back to “lie” by it.

Stemming and lemmatization

- For many tasks, we want the words “dog” and “dogs” to be considered the same.
- Stemming: get the stem, e.g., removing the affix.
- Stemming may not be enough:
  - “I saw you” should become “I see you”
- Lemmatization: get the lemma, the item word in a dictionary.

seek

/sēk/

verb

verb: **seek**; 3rd person present: **seeks**; past tense: **sought**; past participle: **sought**; gerund or present participle: **seeking**

- How to build a lemmatizer?
  - Take advantage of the dictionary
- Don’t take things for granted: NLTK’s “The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.” which means that “lying” will not be restored back to “lie” by it.

Stemming and lemmatization

- For many tasks, we want the words “dog” and “dogs” to be considered the same.
- Stemming: get the stem, e.g., removing the affix.
- Stemming may not be enough:
  - “I saw you” should become “I see you”
- Lemmatization: get the lemma, the item word in a dictionary.

seek

/sēk/

verb

verb: **seek**; 3rd person present: **seeks**; past tense: **sought**; past participle: **sought**; gerund or present participle: **seeking**

- How to build a lemmatizer?
  - Take advantage of the dictionary
- Don’t take things for granted: NLTK’s “The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.” which means that “lying” will not be restored back to “lie” by it.

Stemming and lemmatization

- For many tasks, we want the words “dog” and “dogs” to be considered the same.
- Stemming: get the stem, e.g., removing the affix.
- Stemming may not be enough:
  - “I saw you” should become “I see you”
- Lemmatization: get the lemma, the item word in a dictionary.

seek
/sēk/

verb
verb: **seek**; 3rd person present: **seeks**; past tense: **sought**; past participle: **sought**; gerund or present participle: **seeking**

- How to build a lemmatizer?
  - Take advantage of the dictionary
  - What about words beyond? E.g., “transformative” (NSF style) vs “transforming” (a real English word in dictionaries.)
- Don’t take things for granted: NLTK’s “The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.” which means that “lying” will not be restored back to “lie” by it.

Stemming and lemmatization

- For many tasks, we want the words “dog” and “dogs” to be considered the same.
- Stemming: get the stem, e.g., removing the affix.
- Stemming may not be enough:
  - “I saw you” should become “I see you”
- Lemmatization: get the lemma, the item word in a dictionary.

seek

/sēk/

verb

verb: **seek**; 3rd person present: **seeks**; past tense: **sought**; past participle: **sought**; gerund or present participle: **seeking**

- How to build a lemmatizer?
  - Take advantage of the dictionary
  - What about words beyond? E.g., “transformative” (NSF-style) vs “transforming” (a real English word in dictionaries)
- Don’t take things for granted: NLTK’s “The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.” which means that “lying” will not be restored back to “lie” by it.

Stemming and lemmatization

- For many tasks, we want the words “dog” and “dogs” to be considered the same.
- Stemming: get the stem, e.g., removing the affix.
- Stemming may not be enough:
  - “I saw you” should become “I see you”
- Lemmatization: get the lemma, the item word in a dictionary.

seek

/sēk/

verb

verb: **seek**; 3rd person present: **seeks**; past tense: **sought**; past participle: **sought**; gerund or present participle: **seeking**

- How to build a lemmatizer?
  - Take advantage of the dictionary
  - What about words beyond? E.g., “transformative” (NSF style) vs “transforming” (a real English word in dictionaries.)
- Don’t take things for granted: NLTK’s “The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.” which means that “lying” will not be restored back to “lie” by it.

Stemming and lemmatization

- For many tasks, we want the words “dog” and “dogs” to be considered the same.
- Stemming: get the stem, e.g., removing the affix.
- Stemming may not be enough:
  - “I saw you” should become “I see you”
- Lemmatization: get the lemma, the item word in a dictionary.

seek
/sēk/

verb
verb: **seek**; 3rd person present: **seeks**; past tense: **sought**; past participle: **sought**; gerund or present participle: **seeking**

- How to build a lemmatizer?
  - Take advantage of the dictionary
  - What about words beyond? E.g., “transformative” (NSF style) vs “transforming” (a real English word in dictionaries.)
- Don’t take things for granted: NLTK’s “The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.” which means that “lying” will not be restored back to “lie” by it.

Stemming and lemmatization

- For many tasks, we want the words “dog” and “dogs” to be considered the same.
- Stemming: get the stem, e.g., removing the affix.
- Stemming may not be enough:
  - “I saw you” should become “I see you”
- Lemmatization: get the lemma, the item word in a dictionary.

seek
/sēk/

verb
verb: **seek**; 3rd person present: **seeks**; past tense: **sought**; past participle: **sought**; gerund or present participle: **seeking**

- How to build a lemmatizer?
  - Take advantage of the dictionary
  - What about words beyond? E.g., “transformative” (NSF style) vs “transforming” (a real English word in dictionaries.)
- Don’t take things for granted: NLTK’s “The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.” which means that “lying” will not be restored back to “lie” by it.

Stemming and lemmatization

- For many tasks, we want the words “dog” and “dogs” to be considered the same.
- Stemming: get the stem, e.g., removing the affix.
- Stemming may not be enough:
  - “I saw you” should become “I see you”
- Lemmatization: get the lemma, the item word in a dictionary.

seek
/sēk/

verb
verb: **seek**; 3rd person present: **seeks**; past tense: **sought**; past participle: **sought**; gerund or present participle: **seeking**

- How to build a lemmatizer?
  - Take advantage of the dictionary
  - What about words beyond? E.g., “transformative” (NSF style) vs “transforming” (a real English word in dictionaries.)
- Don’t take things for granted: NLTK’s “The WordNet lemmatizer only removes affixes if the resulting word is in its dictionary.” which means that “lying” will not be restored back to “lie” by it.

Downside of stemming and lemmatization

- By converting, e.g.,“are”, “is”, “am” and “be” into “be”, we lose information that may help use on other tasks.
- Some combinations are meaningless when tokenized or lemmatized, e.g., in “I should have quited grad school when Sequoia gave me $2M”, “should have quited” together mean something.
- Accurate lemmatization requires understanding the role of each word, e.g., the “saw” in “I saw the table (into half)” should be kept instead of being changed to “see.”

Outline

Normalization

- Normalization means restoring a word to its canonical form, e.g., “saw” to “see”.
- Stemming and lemmatization are just one side of normalization.
- Other common tasks of normalization:
  - Removing stop words: words that are too frequency to become background noise, e.g., “the”, “of”, etc.
  - Preserving entities, e.g., “New York”.
  - Expanding abbreviations, e.g., “PhD” as “Piled Higher and Deeper” not “Philosophiae Doctor”.
  - Changing characters into lowercase. Not for all!
  - Question: Do you wanna treat “lower case” and “lowercase” the same? How about “E-mail” and “email”?

Normalization

- Normalization means restoring a word to its canonical form, e.g., “saw” to “see”.
- Stemming and lemmatization are just one side of normalization.
- Other common tasks of normalization:
  - Removing *stop words*, words that are too frequency to become background noise, e.g., “the”, “of”, etc.
  - Preserving entities, e.g., “New York”.
  - Expanding abbreviations, e.g., “PhD” as “Piled Higher and Deeper” not “Philosophiae Doctor”.
  - Changing characters into lowercase. Not for all!
  - Question: Do you wanna treat “lower case” and “lowercase” the same? How about “E-mail” and “email”?

Normalization

- Normalization means restoring a word to its canonical form, e.g., “saw” to “see”.
- Stemming and lemmatization are just one side of normalization.
- Other common tasks of normalization:
  - Removing *stop words*, words that are too frequency to become background noise, e.g., “the”, “of”, etc.
  - Preserving entities, e.g., “New York”.
  - Expanding abbreviations, e.g., “PhD” as “Piled Higher and Deeper” not “Philosophiae Doctor”.
  - Changing characters into lowercase. Not for all!
  - Question: Do you wanna treat “lower case” and “lowercase” the same? How about “E-mail” and “email”?

Normalization

- Normalization means restoring a word to its canonical form, e.g., “saw” to “see”.
- Stemming and lemmatization are just one side of normalization.
- Other common tasks of normalization:
  - Removing *stop words*, words that are too frequency to become background noise, e.g., “the”, “of”, etc.
  - Preserving entities, e.g., “New York”.
  - Expanding abbreviations, e.g., “PhD” as “Piled Higher and Deeper” not “Philosophiae Doctor”.
  - Changing characters into lowercase. Not for all!
  - Question: Do you wanna treat “lower case” and “lowercase” the same? How about “E-mail” and “email”?

Normalization

- Normalization means restoring a word to its canonical form, e.g., “saw” to “see”.
- Stemming and lemmatization are just one side of normalization.
- Other common tasks of normalization:
  - Removing *stop words*, words that are too frequency to become background noise, e.g., “the”, “of”, etc.
  - Preserving entities, e.g., “New York”.
  - Expanding abbreviations, e.g., “PhD” as “Piled Higher and Deeper” not “Philosophiae Doctor”.
  - Changing characters into lowercase. Not for all!
  - Question: Do you wanna treat “lower case” and “lowercase” the same? How about “E-mail” and “email”?

Normalization

- Normalization means restoring a word to its canonical form, e.g., "saw" to "see".
- Stemming and lemmatization are just one side of normalization.
- Other common tasks of normalization:
  - Removing *stop words*, words that are too frequency to become background noise, e.g., "the", "of", etc.
  - Preserving entities, e.g., "New York".
  - Expanding abbreviations, e.g., "PhD" as "Piled Higher and Deeper" not "Philosophiae Doctor".
  - Changing characters into lowercase. Not for all!
  - Question: Do you wanna treat "lower case" and "lowercase" the same? How about "E-mail" and "email"?

Normalization

- Normalization means restoring a word to its canonical form, e.g., “saw” to “see”.
- Stemming and lemmatization are just one side of normalization.
- Other common tasks of normalization:
  - Removing *stop words*, words that are too frequency to become background noise, e.g., “the”, “of”, etc.
  - Preserving entities, e.g., “New York”.
  - Expanding abbreviations, e.g., “PhD” as “Piled Higher and Deeper” not “Philosophiae Doctor”.
  - Changing characters into lowercase. Not for all!
  - Question: Do you wanna treat “lower case” and “lowercase” the same? How about “E-mail” and “email”?

Normalization

- Normalization means restoring a word to its canonical form, e.g., “saw” to “see”.
- Stemming and lemmatization are just one side of normalization.
- Other common tasks of normalization:
  - Removing *stop words*, words that are too frequency to become background noise, e.g., “the”, “of”, etc.
  - Preserving entities, e.g., “New York”.
  - Expanding abbreviations, e.g., “PhD” as “Piled Higher and Deeper” not “Philosophiae Doctor”.
  - Changing characters into lowercase. Not for all!
  - Question: Do you wanna treat “lower case” and “lowercase” the same? How about “E-mail” and “email”?

PDF is not a mark-up language

Increasing demands for petroleum have stimulated sustainable ways to produce chemicals and biofuels. Specifically, fatty acids of varying chain lengths (C_6–C_{16}) naturally synthesized in many organisms are promising starting points for the catalytic production of industrial chemicals and diesel-like biofuels. However, bio-production of fatty acids from plants and other microbial production hosts relies heavily on manipulating tightly regulated fatty acid biosynthetic pathways. In addition, precursors for fatty acids are used along other central metabolic pathways for the production of amino acids and biomass, which further complicates the engineering of microbial hosts for higher yields. Here, we demonstrate an iterative metabolic engineering effort that integrates computationally driven predictions and metabolic flux analysis techniques to meet this challenge. The OptForce procedure was used for suggesting and prioritizing genetic manipulations that overproduce fatty acids of different chain lengths from C_6 to C_{16} starting with wild-type *E. coli*. We identified some common but mostly chain-specific genetic interventions alluding to the possibility of fine-tuning overproduction for specific fatty acid chain lengths. In accordance with the OptForce prioritization of interventions, *fabZ* and acyl-ACP thioesterase were upregulated and *fadD* was deleted to arrive at a strain that produces 1.70 g/L and 0.14 g fatty acid/g glucose (~39% maximum theoretical yield) of C_{14-16} fatty acids in minimal M9 medium. These results highlight the benefit of using computational strain design and flux analysis tools in the design of recombinant strains of *E. coli* to produce free fatty acids.

Increasing demands for petroleum have stimulated sustainable ways to produce chemicals and biofuels. Speci?cally, fatty acids of varying chain lengths (C6C16) naturally synthesized in many organisms are promising starting points for the catalytic production of industrial chemicals and diesel- like biofuels. However, bio-production of fatty acids from plants and other microbial production hosts relies heavily on manipulating tightly regulated fatty acid biosynthetic pathways. In addition, precursors for fatty acids are used along other central metabolic pathways for the production of amino acids and biomass, which further complicates the engineering of microbial hosts for higher yields. Here, we demonstrate an iterative metabolic engineering effort that integrates computationally driven predictions and metabolic ?ux analysis techniques to meet this challenge. The OptForce procedure was used for suggesting and prioritizing genetic manipulations that overproduce fatty acids of different chain lengths from C6 to

Tackling spelling errors or OCR errors

- How to build a spell checker?
  http://norvig.com/spell-correct.html
- Using a dictionary, computing editing distance, and ranking candidates.
- Peter's solution assumes at most 2 edits from original string.
- String alignment: Smith-waterman algorithm, etc.
- What else?
- Do not think at token level.
- Language model to be discussed. How to train the model?
- What about symbols used in technical documents? E.g., "10g/mL".

Tackling spelling errors or OCR errors

- How to build a spell checker?
  http://norvig.com/spell-correct.html
- Using a dictionary, computing editing distance, and ranking candidates.
- Peter's solution assumes at most 2 edits from original string.
- String alignment: Smith-waterman algorithm, etc.
- What else?
- Do not think at token level.
- Language model to be discussed. How to train the model?
- What about symbols used in technical documents? E.g., "10g/mL".

Tackling spelling errors or OCR errors

- How to build a spell checker?
  http://norvig.com/spell-correct.html
- Using a dictionary, computing editing distance, and ranking candidates.
- Peter's solution assumes at most 2 edits from original string.
- String alignment: Smith-waterman algorithm, etc.
- What else?
- Do not think at token level.
- Language model to be discussed. How to train the model?
- What about symbols used in technical documents? E.g., "10g/mL".

Tackling spelling errors or OCR errors

- How to build a spell checker?
  http://norvig.com/spell-correct.html
- Using a dictionary, computing editing distance, and ranking candidates.
- Peter's solution assumes at most 2 edits from original string.
- String alignment: Smith-waterman algorithm, etc.
- What else?
- Do not think at token level.
- Language model to be discussed. How to train the model?
- What about symbols used in technical documents? E.g., "10g/mL".

Tackling spelling errors or OCR errors

- How to build a spell checker?
  http://norvig.com/spell-correct.html
- Using a dictionary, computing editing distance, and ranking candidates.
- Peter's solution assumes at most 2 edits from original string.
- String alignment: Smith-waterman algorithm, etc.
- What else?
- Do not think at token level.
- Language model to be discussed. How to train the model?
- What about symbols used in technical documents? E.g., "10g/mL".

Tackling spelling errors or OCR errors

- How to build a spell checker?
  http://norvig.com/spell-correct.html
- Using a dictionary, computing editing distance, and ranking candidates.
- Peter's solution assumes at most 2 edits from original string.
- String alignment: Smith-waterman algorithm, etc.
- What else?
- Do not think at token level.
- Language model to be discussed. How to train the model?
- What about symbols used in technical documents? E.g., "10g/mL".

Tackling spelling errors or OCR errors

- How to build a spell checker?
  `http://norvig.com/spell-correct.html`
- Using a dictionary, computing editing distance, and ranking candidates.
- Peter's solution assumes at most 2 edits from original string.
- String alignment: Smith-waterman algorithm, etc.
- What else?
- Do not think at token level.
- Language model to be discussed. How to train the model?
- What about symbols used in technical documents? E.g., "10g/mL".

Tackling spelling errors or OCR errors

- How to build a spell checker?
  http://norvig.com/spell-correct.html
- Using a dictionary, computing editing distance, and ranking candidates.
- Peter's solution assumes at most 2 edits from original string.
- String alignment: Smith-waterman algorithm, etc.
- What else?
- Do not think at token level.
- Language model to be discussed. How to train the model?
- What about symbols used in technical documents? E.g., "10g/mL".

Outline

Building an inverse index

- An inverse index allows us to quickly locate a string (all occurrences or just one) in the corpus.
- Suppose I am building a file scanner for all files on my computer in `whoosh`.
- Begin with creating a schema, the structure of the index.

```
import whoosh, whoosh.fields, whoosh.analysis
schema = whoosh.fields.Schema(\
  path=whoosh.fields.ID(stored=True),\
  body=whoosh.fields.TEXT(\
       analyzer=whoosh.analysis.StemmingAnalyzer(),
       stored=True),\
  page=whoosh.fields.NUMERIC(int, 64, signed=False, stored=
      True))
```

- You may imagine the schema as the headers of a table.
- This index contains 3 fields, path, body and page, which are ID ("the entire value of the field as a single unit (that is, it doesnt break it up into individual words)."), text and numeric, respectively.
- "A field can be indexed (meaning it can be searched) and/or stored (meaning the value that gets indexed is returned with the results"

Building an inverse index

- An inverse index allows us to quickly locate a string (all occurrences or just one) in the corpus.
- Suppose I am building a file scanner for all files on my computer in `whoosh`.
- Begin with creating a schema, the structure of the index.

```
import whoosh, whoosh.fields, whoosh.analysis
schema = whoosh.fields.Schema(\
  path=whoosh.fields.ID(stored=True),\
  body=whoosh.fields.TEXT(\
       analyzer=whoosh.analysis.StemmingAnalyzer(),
       stored=True),\
  page=whoosh.fields.NUMERIC(int, 64, signed=False, stored=
      True))
```

- You may imagine the schema as the headers of a table.
- This index contains 3 fields, path, body and page, which are ID ("the entire value of the field as a single unit (that is, it doesnt break it up into individual words)."), text and numeric, respectively.
- "A field can be indexed (meaning it can be searched) and/or stored (meaning the value that gets indexed is returned with the results"

Building an inverse index

- An inverse index allows us to quickly locate a string (all occurrences or just one) in the corpus.
- Suppose I am building a file scanner for all files on my computer in `whoosh`.
- Begin with creating a schema, the structure of the index.

```
import whoosh, whoosh.fields, whoosh.analysis
schema = whoosh.fields.Schema(\
  path=whoosh.fields.ID(stored=True),\
  body=whoosh.fields.TEXT(\
       analyzer=whoosh.analysis.StemmingAnalyzer(),
       stored=True),\
  page=whoosh.fields.NUMERIC(int, 64, signed=False, stored=
      True))
```

- You may imagine the schema as the headers of a table.
- This index contains 3 fields, path, body and page, which are ID ("the entire value of the field as a single unit (that is, it doesnt break it up into individual words)."), text and numeric, respectively.
- "A field can be indexed (meaning it can be searched) and/or stored (meaning the value that gets indexed is returned with the results"

Building an inverse index

- An inverse index allows us to quickly locate a string (all occurrences or just one) in the corpus.
- Suppose I am building a file scanner for all files on my computer in `whoosh`.
- Begin with creating a schema, the structure of the index.

```
import whoosh, whoosh.fields, whoosh.analysis
schema = whoosh.fields.Schema(\
  path=whoosh.fields.ID(stored=True),\
  body=whoosh.fields.TEXT(\
       analyzer=whoosh.analysis.StemmingAnalyzer(),
       stored=True),\
  page=whoosh.fields.NUMERIC(int, 64, signed=False, stored=
      True))
```

- You may imagine the schema as the headers of a table.
- This index contains 3 fields, path, body and page, which are ID (“the entire value of the field as a single unit (that is, it doesnt break it up into individual words).”), text and numeric, respectively.
- “A field can be indexed (meaning it can be searched) and/or stored (meaning the value that gets indexed is returned with the results”

Building an inverse index

- An inverse index allows us to quickly locate a string (all occurrences or just one) in the corpus.
- Suppose I am building a file scanner for all files on my computer in `whoosh`.
- Begin with creating a schema, the structure of the index.

```
import whoosh, whoosh.fields, whoosh.analysis
schema = whoosh.fields.Schema(\
  path=whoosh.fields.ID(stored=True),\
  body=whoosh.fields.TEXT(\
        analyzer=whoosh.analysis.StemmingAnalyzer(),
        stored=True),\
  page=whoosh.fields.NUMERIC(int, 64, signed=False, stored=
       True))
```

- You may imagine the schema as the headers of a table.
- This index contains 3 fields, path, body and page, which are ID ("the entire value of the field as a single unit (that is, it doesnt break it up into individual words)."), text and numeric, respectively.
- "A field can be indexed (meaning it can be searched) and/or stored (meaning the value that gets indexed is returned with the results"

Building an inverse index

- An inverse index allows us to quickly locate a string (all occurrences or just one) in the corpus.
- Suppose I am building a file scanner for all files on my computer in `whoosh`.
- Begin with creating a schema, the structure of the index.

```
import whoosh, whoosh.fields, whoosh.analysis
schema = whoosh.fields.Schema(\
  path=whoosh.fields.ID(stored=True),\
  body=whoosh.fields.TEXT(\
        analyzer=whoosh.analysis.StemmingAnalyzer(),
        stored=True),\
  page=whoosh.fields.NUMERIC(int, 64, signed=False, stored=
       True))
```

- You may imagine the schema as the headers of a table.
- This index contains 3 fields, path, body and page, which are ID ("the entire value of the field as a single unit (that is, it doesnt break it up into individual words)."), text and numeric, respectively.
- "A field can be indexed (meaning it can be searched) and/or stored (meaning the value that gets indexed is returned with the results"

Building an inverse index

- Then you create an index object.

```
ix = whoosh.index.create_in("/home/forrest/index_dir",
     schema)
writer = ix.writer()
```

- The index will be saved in the directory.
- Then you add documents (not necessarily a real file but a piece of text) into it. Just like filling rows in a table.

```
writer.add_document(path=unicode("/mylife/preschool.log"),
     body=unicode("I was born"), page=1)
writer.add_document(path=unicode("/mylife/preschool.log"),
     body=unicode("I went to school"), page=2)
```

- Always remember to commit after you are done with adding documents each time.

```
writer.commit()
```

Building an inverse index

- Then you create an index object.

```
ix = whoosh.index.create_in("/home/forrest/index_dir",
     schema)
writer = ix.writer()
```

- The index will be saved in the directory.
- Then you add documents (not necessarily a real file but a piece of text) into it. Just like filling rows in a table.

```
writer.add_document(path=unicode("/mylife/preschool.log"),
     body=unicode("I was born"), page=1)
writer.add_document(path=unicode("/mylife/preschool.log"),
     body=unicode("I went to school"), page=2)
```

- Always remember to commit after you are done with adding documents each time.

```
writer.commit()
```

Building an inverse index

- Then you create an index object.

```
ix = whoosh.index.create_in("/home/forrest/index_dir",
     schema)
writer = ix.writer()
```

- The index will be saved in the directory.
- Then you add documents (not necessarily a real file but a piece of text) into it. Just like filling rows in a table.

```
writer.add_document(path=unicode("/mylife/preschool.log"),
     body=unicode("I was born"), page=1)
writer.add_document(path=unicode("/mylife/preschool.log"),
     body=unicode("I went to school"), page=2)
```

- Always remember to commit after you are done with adding documents each time.

```
writer.commit()
```

Building an inverse index

- Then you create an index object.

```
ix = whoosh.index.create_in("/home/forrest/index_dir",
     schema)
writer = ix.writer()
```

- The index will be saved in the directory.
- Then you add documents (not necessarily a real file but a piece of text) into it. Just like filling rows in a table.

```
writer.add_document(path=unicode("/mylife/preschool.log"),
     body=unicode("I was born"), page=1)
writer.add_document(path=unicode("/mylife/preschool.log"),
     body=unicode("I went to school"), page=2)
```

- Always remember to commit after you are done with adding documents each time.

```
writer.commit()
```

Building an inverse index

- Now, let's search by creating a query.

```
import whoosh.qparser
qp = whoosh.qparser.QueryParser("body", schema = ix.schema)
     # the query is on body field
q = qp.parse(u"born")      # the query value is "born"
```

- then pass the query to a searcher

```
searcher = ix.searcher()
results = searcher.search(q)
```

- Here is the results:

```
  for result in results: # each result is for one document
      which could contain multiple matches
        print "found match(es) in file %s".format(result["
            path"])
```

Building an inverse index

- Now, let's search by creating a query.

```
import whoosh.qparser
qp = whoosh.qparser.QueryParser("body", schema = ix.schema)
     # the query is on body field
q = qp.parse(u"born")      # the query value is "born"
```

- then pass the query to a searcher

```
searcher = ix.searcher()
results = searcher.search(q)
```

- Here is the results:

```
  for result in results: # each result is for one document
      which could contain multiple matches
        print "found match(es) in file %s".format(result["
            path"])
```

Building an inverse index

- Now, let's search by creating a query.

```
import whoosh.qparser
qp = whoosh.qparser.QueryParser("body", schema = ix.schema)
     # the query is on body field
q = qp.parse(u"born")          # the query value is "born"
```

- then pass the query to a searcher

```
searcher = ix.searcher()
results = searcher.search(q)
```

- Here is the results:

```
  for result in results: # each result is for one document
      which could contain multiple matches
        print "found match(es) in file %s".format(result["
            path"])
```

Cloud APIs for syntactic analysis

- Google: https://cloud.google.com/natural-language/
- Microsoft: https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/

Other libraries

- TextBlob: built on top of NLTK
- SpaCy: much faster than NLTK

Homework 3

- Write a Python program that passes a query (which may contain non-alphabetnumerics) to Google using GET method and extract the first sentence of each sample result on the first page. In the example below, you should extract “academia is broken”, “academia is isolating and fundamentally sick”, and others.