Natural Language Processing

Dr. Karen Mazidi

Information Extraction

- IE involves identifying specific, important information in unstructured text
  - Key terms
  - Proper names of people, organizations
  - Dates and times
  - Events
- A form of text mining
- Related topic: IR information retrieval – finding documents in a corpus based on keyword search

IE approaches

Rules-based approaches

- using regex to find patterns like phone numbers
- using pre-defined lists of persons, places, …

Statistical approaches

- Use count-based metrics to find important terms
  - tf
  - tf-idf

ML approaches

- Use a corpus of annotated data
- Use text features such as POS, capitalization as features

Finding important terms

$$tf = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

- Are the most frequent words the most important words?
  - Stop words?
  - Common words?
- Term frequency is the count of a word in a document, often divided by the number of tokens in the document
- Sometimes the log is taken

idf – inverse document frequency

$$idf = log\left(\frac{N}{|d \in D \& t \in d|}\right)$$

- idf down-votes common words that appear in most documents
- these common words don't tell you much about a document
- idf divides the number of documents (N) by the number of documents in which the term appears, then the log is taken

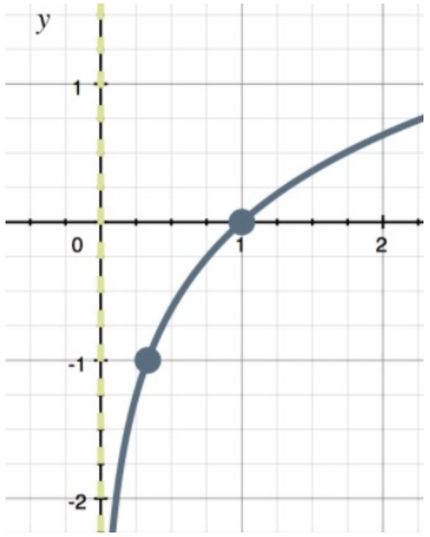
- Where
  - N is the number of documents
  - D is the set of documents in a corpus

idf

$$idf = log\left(\frac{N}{|d \in D \& t \in d|}\right)$$

- As the value inside log(*) approaches 1 (100% of documents), the log approaches 0

- smoothing will be needed to prevent divide by zero in tf-idf and deal with negative idf



tf-idf

Multiply tf * idf

The more a term appears in a document, the higher the tf

The more a term appears in many documents, the lower the idf which makes tf*idf lower

Creating tf-idf scores from scratch

- 4 documents: lower cased and tokenized
- Delete tokens that are stop words or not alpha
- Number of unique words in this corpus is around 4K
- First, create a tf dictionary
- The tf value is normalized by number of tokens in the document

Creating tf-idf scores from scratch

Code 13.1.1 — Create a tf dictionary. Function Definition

```
def create_tf_dict(doc):
# returns a normalized term frequency dict
# doc is a set of processed tokens

    tf_dict = {}
    tokens = word_tokenize(doc)
    tokens = [w for w in tokens if w.isalpha()
              and w not in stopwords]

    # get term frequencies
    for t in tokens:
        if t in tf_dict:
            tf_dict[t] += 1
        else:
            tf_dict[t] = 1

    # normalize tf by number of tokens
    for t in tf_dict.keys():
        tf_dict[t] = tf_dict[t] / len(tokens)

    return tf_dict
```

Creating tf-idf scores from scratch

- Notice that the word ‘work’ appears in all 4 documents

```
tf for "work" in anat = 0.00046040515653775324
tf for "work" in buslaw = 0.0027739251040221915
tf for "work" in econ = 0.0006854009595613434
tf for "work" in geog = 0.0009285051067780873
```

Creating tf-idf scores from scratch

Code 13.1.2 — Create an idf dictionary. One dict for the corpus

```
import math

idf_dict = {}

vocab_by_topic = [tf_anat.keys(), tf_buslaw.keys(),
                  tf_econ.keys(), tf_geog.keys()]

for term in vocab:
    temp = ['x' for voc in vocab_by_topic if term in voc]
    idf_dict[term] = math.log((1+num_docs) / (1+len(temp)))
```

- Now create an idf dictionary for every word in the corpus vocabulary
- Divide number of docs, 4, by the number of docs that a given term appears in
- Do some fiddling around to avoid dividing by zero and negative idf values
- Take the log

Creating tf-idf scores from scratch

Notice:

- The idf for 'work' is 0 because it appears in all docs
- The idf for 'inflation' was 0.9; it appears in one document

Code 13.1.2 — Create an idf dictionary. One dict for the corpus

```
import math

idf_dict = {}

vocab_by_topic = [tf_anat.keys(), tf_buslaw.keys(),
                  tf_econ.keys(), tf_geog.keys()]

for term in vocab:
    temp = ['x' for voc in vocab_by_topic if term in voc]
    idf_dict[term] = math.log((1+num_docs) / (1+len(temp)))
```

Creating tf-idf scores from scratch

- Create a tf-idf dictionary for each document by multiplying the tf times the idf

Code 13.1.3 — tf-idf. Function Definition

```
def create_tfidf(tf, idf):
    tf_idf = {}
    for t in tf.keys():
        tf_idf[t] = tf[t] * idf[t]

    return tf_idf
```

Creating tf-idf scores from scratch

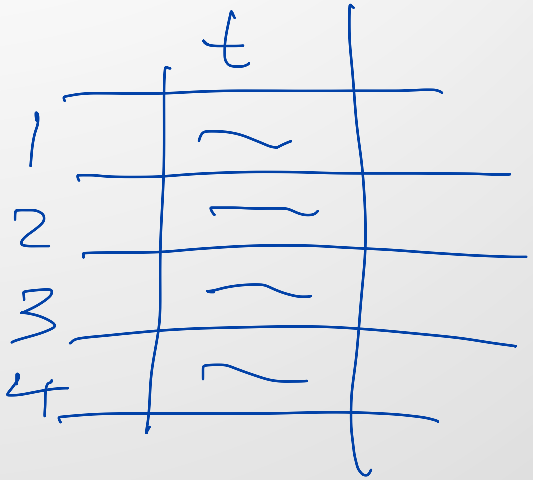
- Sort the dictionaries to find the most important terms

```
anatomy: sympathetic, system, autonomic, parasympathetic, receptors

business law: damages, party, breach, nonbreaching, remedies

economics: inflation, prices, index, price, basket

geography: islands, island, antartica, ozone, pacific
```

- Notice 'island' and 'islands' in geography
  - Lemmatization would have been helpful

tf-idf in sklearn

- Creates a document-term matrix
- we will revisit this in Chapter 19



NER

Named Entity Recognition

NER

- Identify and label sequences of words that represent persons, countries, organizations, etc.
- Important for question answering systems, information retrieval, etc.

> The European Union's top court gave judges in the bloc broader power to order the removal of Facebook (FB 1.13%) posts, dealing a fresh blow to the U.S. tech giant as it faces growing regulatory headwinds on both sides of the Atlantic.

- Results from AllenNLP:

```
ORG:   European Union
ORG:   Facebook
ORG:   FB
PERCENT:   1.13 %
GPE:   U.S.
LOC:   Atlantic
```

Stanford NER

- 20+ classes

- Names: PERSON, LOCATION, ORGANIZATION, MISC
- Numeric: MONEY, NUMBER, ORDINAL, PERCENT
- Temporal: DATE, TIME, DURATION, SET
- Additional classes: EMAIL, URL, CITY, STATE_OR_PROVINCE, COUNTRY, NATIONALITY, RELIGION, (job) TITLE, IDEOLOGY, CRIMINAL_CHARGE, CAUSE_OF_DEATH, Twitter, etc. HANDLE

Stanford NER

Code 13.2.1 — NER. Stanford CoreNLP

```
from stanfordnlp.server import CoreNLPClient

import os
os.environ['CORENLP_HOME'] =
        r'/your path here/stanford-corenlp-full-2018-10-05'

# set up the client
with CoreNLPClient(annotators=['tokenize','ssplit','pos','lemma','ner'],
        timeout=60000, memory='16G') as client:
    # submit the request to the server
    ann = client.annotate(text)

    print('\nTokens \t POS \t NER')
    sentence_count = 1
    for sentence in ann.sentence:
        print('\nSentence', sentence_count)
        for token in sentence.token:
            if token.ner != 'O':
                print (token.word, '\t', token.pos, '\t', token.ner)

        sentence_count += 1
```

Stanford NER

- Input:

The Hawaiian Islands became the fiftieth US state in 1959. Since the passage of the Social Security Indexing Act of 1972, the level of Social Security benefits increases each year along with the Consumer Price Index. The leading case, perhaps the most studied case, in all the common law is Hadley v. Baxendale, decided in England in 1854. Lyndon Baines Johnson (August 27, 1908 – January 22, 1973), often referred to as LBJ, was an American politician who served as the 36th president of the United States from 1963 to 1969.

- NER:

```
Sentence 1
Hawaiian    JJ    LOCATION
Islands    NNPS    LOCATION
fiftieth    NN    ORDINAL
US    NNP    COUNTRY
1959    CD    DATE
. . .
```

SpaCy NER

- Uses models trained on OntoNotes5, a hand-annotated corpus covering multiple genres
- 18 entity types: PERSON, NORP (nationalities or religious or polical groups), FAC (facilities), ORG, GPE, LOC, PRODUCT, EVENT, WORK_OF_ART, LAW, LANGUAGE, DATE, TIME, PERCENT, MONEY, QUANTITY, ORDINAL, CARDINAL

SpaCy NER

- Using medium model
- Recognized LBJ as a person

Code 13.2.2 — NER. SpaCy

```
import spacy
nlp = spacy.load('en_core_web_md')

doc = nlp(text)

for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
Hawaiian Islands GPE
fiftieth ORDINAL
US GPE
1959 DATE
the Social Security Indexing Act LAW
1972 DATE
Social Security ORG
each year DATE
Hadley v. Baxendale PERSON
England GPE
1854 DATE
Lyndon Baines Johnson PERSON
August 27, 1908 - January 22, 1973 DATE
LBJ PERSON
American NORP
36th ORDINAL
the United States GPE
1963 to 1969 DATE
```

Open IE

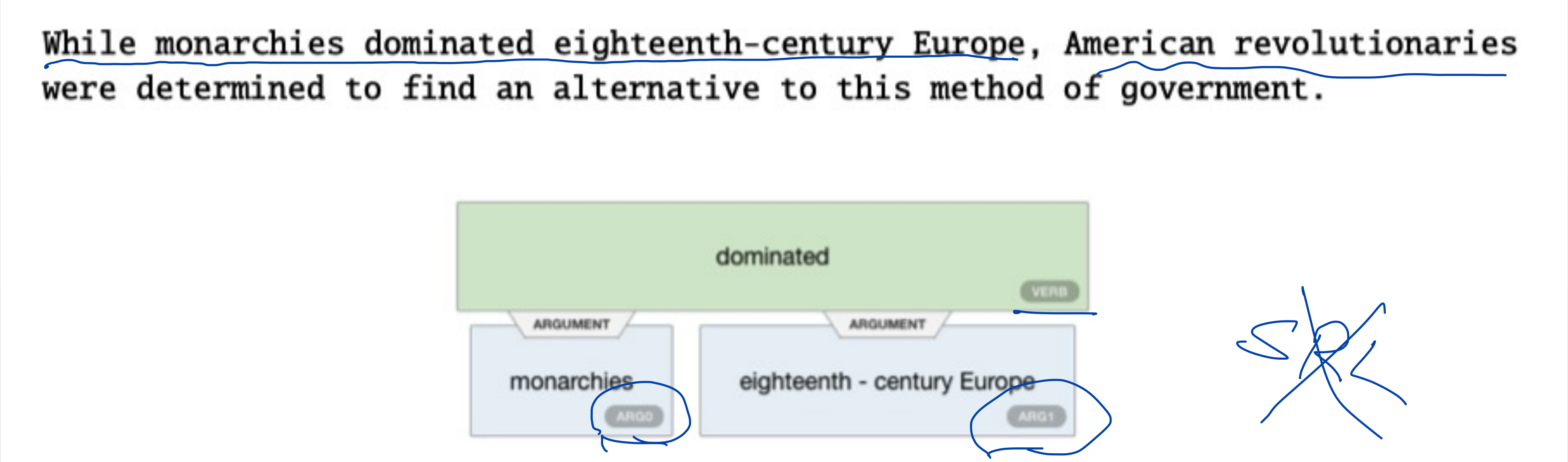
extracts tuples of information

Open IE

- Information Extraction looks for specific information: person, org, date, etc.
- Open IE extracts tuples:
  - `type(arg1, arg2)`
- Example event extraction:
  - `born-in(Barack-Obama, Hawaii)`
- These tuples could be part of:
  - a logic reasoning system
  - a knowledge base
  - any downstream NLP application

AllenNLP example

- Similar to SRL parse



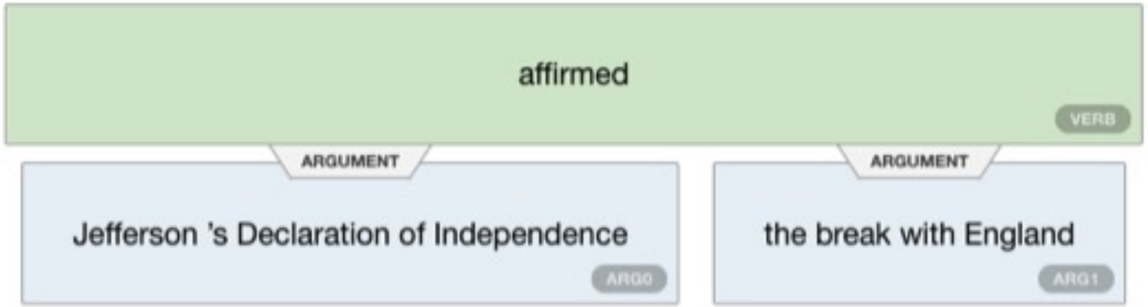
- Notice that nothing was extracted from the main clause

AllenNLP example

affirmed(Dec —, the br...)

- Extracted from main but not subordinate clause

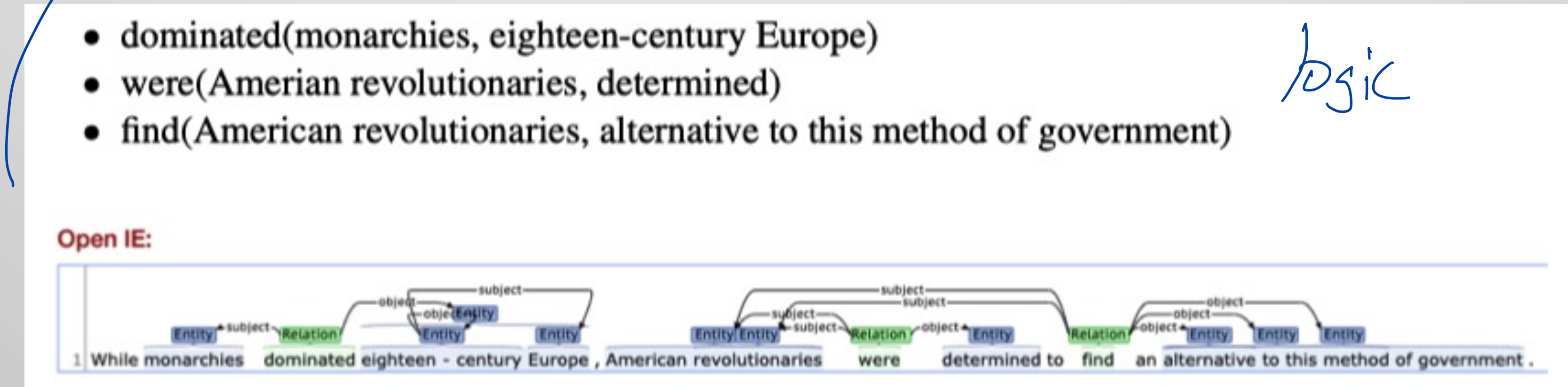
Jefferson's Declaration of Independence affirmed the break with England but did not suggest what form of government should replace monarchy, the only system most English colonists had ever known.



Stanford CoreNLP Open IE

1. Extracts clauses from sentences
2. Shortens
3. Reduced to triples

- https://nlp.stanford.edu/software/openie.html



logic

IE-related conferences and companies

- Conferences: https://www.aclweb.org/portal/category/topics/information-extraction
- Information retrieval: https://sigir.org/sigir2021/
- Many companies in this space:
- https://www.ontotext.com/
- https://ai.googleblog.com/2020/06/extracting-structured-data-from.html
- Knowledge page from IBM:
- https://www.ibm.com/docs/en/db2/9.7?topic=studio-information-extraction

Essential points to note

The field of IE includes:

- extracting important terms
- extracting named entities
- extracting tuples
- open IE