Logistics

- Lab 5 done expect grades by this weekend
- Midterm Wednesday
  - 50min, close book, in class
- Topics, readings, review questions
- No class this Friday – home coming
- Lab 6 on text processing/NLP
  - Material will be posted Friday
  - In-class Lab and quiz coming Monday

Outline

- (postponed) Sequence models and algorithms
- (postponed) Relation Extracting, word2vec..

- Ngram Language Models
- TFIDF scores
- t-test hypothesis testing review

Natural Language Processing

Some basic terms:

- Syntax: the allowable structures in the language: sentences, phrases, affixes (-ing, -ed, -ment, etc.).
- Semantics: the meaning(s) of texts in the language.
- Part-of-Speech (POS): the category of a word (noun, verb, preposition etc.).
- Bag-of-words (BoW): a featurization that uses a vector of word counts (or binary) ignoring order.
- N-gram: for a fixed, small N (2-5 is common), an n-gram is a consecutive sequence of words in a text. (vs. q-gram)

Bag of words Featurization

Assuming we have a dictionary mapping words to a unique integer id, a bag-of-words featurization of a sentence could look like this:

Sentence:	The	cat	sat	on	the	mat
word id's:	1	12	5	3	1	14

The BoW featurization would be the vector:

Vector	2,	0,	1,	0,	1,	0,	0,	0,	0,	0,	0,	1,	0,	1
position	1		3		5							12		14

In practice this would be stored as a sparse vector of (id, count)s:

(1,2),(3,1),(5,1),(12,1),(14,1)

Note that the original word order is lost, replaced by the order of id's.

BoW vs. N-grams

Because word order is lost, the sentence meaning is weakened. This sentence has quite a different meaning but the same BoW vector:

Sentence:	The	mat	sat	on	the	cat
word id s:	1	14	5	3	1	12

BoW featurization:

Vector 2, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

But word order is important, especially the order of nearby words.

N-grams capture this, by modeling tuples of consecutive words.

N-grams

Sentence: The cat sat on the mat

2-grams: the-cat, cat-sat, sat-on, on-the, the-mat

Notice how even these short n-grams "make sense" as linguistic units. For the other sentence we would have different features:

Sentence: The mat sat on the cat

2-grams: the-mat, mat-sat, sat-on, on-the, the-cat

We can go still further and construct 3-grams:

Sentence: The cat sat on the mat

3-grams: the-cat-sat, cat-sat-on, sat-on-the, on-the-mat

Which capture still more of the meaning:

Sentence: The mat sat on the cat

3-grams: the-mat-sat, mat-sat-on, sat-on-the, on-the-cat

N-grams Features

Typically, its advantages to use multiple n-gram features in machine learning models with text, e.g.

unigrams + bigrams (2-grams) + trigrams (3-grams).

The unigrams have higher counts and are able to detect influences (i.e., features) that are weak, while bigrams and trigrams capture strong influences that are more specific. Examples?

e.g. “the white house” will generally have very different influences from the sum of influences of “the”, “white”, “house”.

N-grams size

N-grams pose some challenges in feature set size.

If the original vocabulary size is |V|, the number of 2-grams is $|V|^2$

While for 3-grams it is $|V|^3$

Luckily natural language n-grams (including single words) have a **power law** frequency structure. This means that most of the n-grams you see are common. A dictionary that contains the most common n-grams will cover most of the n-grams you see.

N-grams size

Because of this you may see values like this:

- Unigram dictionary size: 40,000
- Bigram dictionary size: 100,000
- Trigram dictionary size: 300,000

With coverage of > 80% of the features occurring in the text.

N-gram Language Models

N-grams can be used to build statistical models of texts.

When this is done, they are called n-gram language models.

An n-gram language model associates a probability with each n-gram, such that the sum over all n-grams (for fixed n) is 1.

You can then determine the overall likelihood of a particular sentence:

The cat sat on the mat

Is much more likely than

The mat sat on the cat

Q: how can we compare the similarity of two documents?

The TF/IDF Measure: Motivation

- uses the TF/IDF notion commonly used in IR
  - two strings are similar if they share distinguishing terms
  - e.g., x = Apple Corporation, CA
    y = IBM Corporation, CA
    z = Apple Corp
  - s(x,y) > s(x,z) using edit distance or Jaccard measure, so x is matched with y ➔ incorrect
  - TF/IDF measure can recognize that Apple is a distinguishing term, whereas Corporation and CA are far more common ➔ correctly match x with z

Term Frequencies and Inverse Document Frequencies

- Assume x and y are taken from a collection of strings
- Each string is converted into a bag of terms called a document
- Define term frequency tf(t,d) = number of times term t appears in document d
- Define inverse document frequency idf(t) = N / $\mathbf{N}_d$, number of documents in collection devided by number of documents that contain t
  - note: in practice, idf(t) is often defined as log(N / $\mathbf{N}_d$), here we will use the above simple formula to define idf(t)

Example

$x = aab \Rightarrow B_x = \{a, a, b\}$

$y = ac \Rightarrow B_y = \{a, c\}$

$z = a \Rightarrow B_z = \{a\}$

$tf(a, x) = 2$

$tf(b, x) = 1$

...

$tf(c, z) = 0$

$idf(a) = 3/3 = 1$

$idf(b) = 3/1 = 3$

$idf(c) = 3/1 = 3$

Feature Vectors

- Each document d is converted into a feature vector $\mathbf{v}_d$
- $\mathbf{v}_d$ has a feature $\mathbf{v}_d(\mathbf{t})$ for each term t
  - value of $\mathbf{v}_d(\mathbf{t})$ is a function of TF and IDF scores
  - here we assume $\mathbf{v}_d(\mathbf{t})$ = tf(t,d) * idf(t)

$x = aab \Rightarrow B_x = \{a, a, b\}$

$y = ac \Rightarrow B_y = \{a, c\}$

$z = a \Rightarrow B_z = \{a\}$

tf(a, x) = 2
tf(b, x) = 1
...
tf(c, z) = 0

idf(a) = 3/3 = 1
idf(b) = 3/1 = 3
idf(c) = 3/1 = 3

	a	**b**	**c**
$\mathbf{v}_x$	2	3	0
$\mathbf{v}_y$	3	0	3
$\mathbf{v}_z$	3	0	0

TF/IDF Similarity Score

- Let p and q be two strings, and T be the set of all terms in the collection
- Feature vectors $\mathbf{v}_p$ and $\mathbf{v}_q$ are vectors in the |T|-dimensional space wher each dimension corresponds to a term
- TF/IDF score of p and q is the cosine of the angle between $\mathbf{v}_p$ and $\mathbf{v}_q$
  - $s(p,q) = \sum_{t \in T} v_p(t) * v_q(t) \; / [\sqrt{\sum_{t \in T} v_p(t)^2} * \sqrt{\sum_{t \in T} vq(t)^2}]$

TF/IDF Similarity Score

- Score is high if strings share many frequent terms
  - terms with high TF scores
- Unless these terms are common in other strings
  - i.e., they have low IDF scores
- Dampening TF and IDF as commonly done in practice
  - use v_d(t) = log(tf(t,d) + 1) * log(idf(t)) instead of
    v_d(t) = tf(t,d) * idf(t)
- Normalizing feature vectors
  - v_d(t) = v_d(t) / $\sqrt{\sum_{\{t \in T\}} v_d(t)^2}$

Hypothesis testing

- One sample t-test
  - Sample standard deviation – s
  - Degrees of freedom – df
  - Test statistic
  - T-distribution
  - T-test vs. z-test