TEXT SIMILARITY

David Kauchak
CS159 Spring 2011

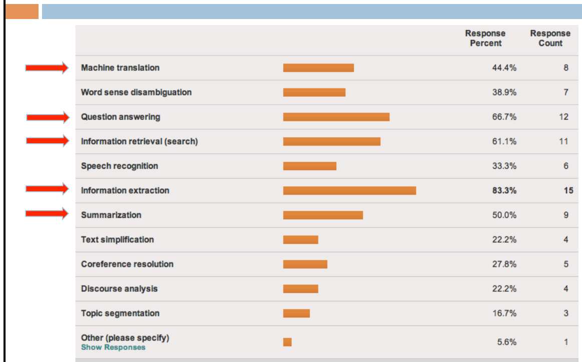
Quiz #2

- Out of 30 points
- High: 28.75
- Ave: 23
- Will drop lowest quiz
- I do not grade based on absolutes

Class feedback

- Thanks!
- Specific comments:
  - "Less/no Java :)"
    - http://www.langpop.com/
    - http://www.devtopics.com/most-popular-programming-languages/
  - "tell us to get up more often and stretch and high-five"
  - "Drop lowest quiz grade"
  - "more labs"

Class presentations



Class presentations

- Presentations done in pairs (and one triplet)
- 25 minutes for presentation 10 min. for Q+A
- In the week following your presentation, come by and see me for 5-10 min. for feedback
- 5% of your grade is based on your presentation
  - I will also be looking for improvement from this presentation to your final project presentation
- If you are not presenting, you should spend at least 30 min. on each paper reading it before class

Class presentations

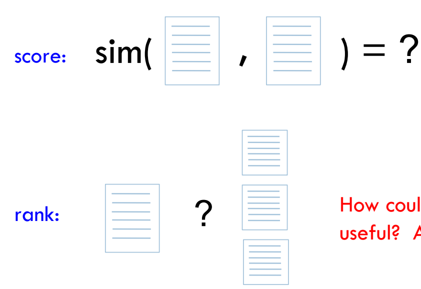
- 7 of you still haven't e-mailed me preferences!
- If you e-mail me by 5pm today, I'll take those into account
- I will post the assignments later today
  - I'll try and give everyone their first choice

Other Admin

- Assignment 5 (last assignment!) will be posted soon and due next Friday (4/1)
- I will post final project deadlines, specifications, etc. soon
  - Groups 2-3 (possibly 4)
  - ~4 weeks of actual coding/writing
  - Start thinking about final projects
  - Project proposals will be due ~ April 4
- How many of you are seniors?
  - I will have to shift some things in the schedule since you're grades are due early ☺

Text Similarity

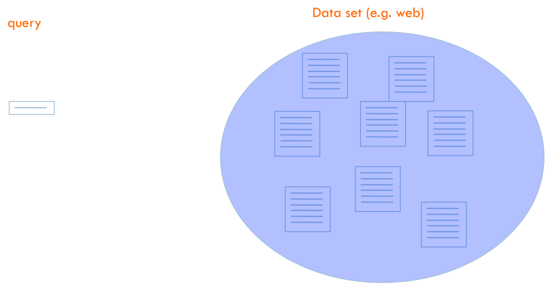
- A common question in NLP is how similar are texts



How could these be useful? Applications?

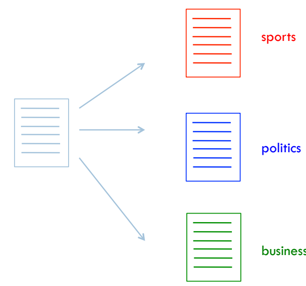
Text similarity: applications

- Information retrieval (search)



Text similarity: applications

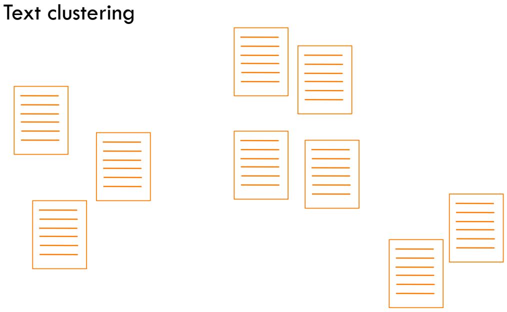
- Text classification



These “documents” could be actual documents, for example using k-means or pseudo-documents, like a class centroid/ average

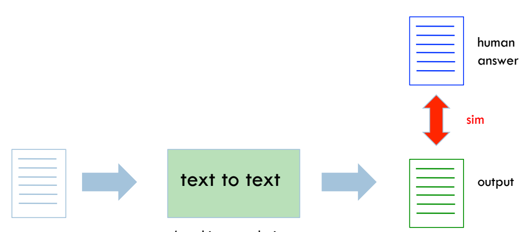
Text similarity: applications

- Text clustering



Text similarity: application

- Automatic evaluation



(machine translation, summarization, simplification)

Text similarity: applications

- Word similarity

  sim(*banana*, apple) = ?

- Word-sense disambiguation

  I went to the *bank* to get some money.

  financial bank   river bank

Text similarity: application

- Automatic grader

Question: what is a variable?
Answer: a location in memory that can store a value

How good are:

- a variable is a location in memory where a value can be stored
- a named object that can hold a numerical or letter value
- it is a location in the computer 's memory where it can be stored for use by a program
- a variable is the memory address for a specific type of stored data or from a mathematical perspective a symbol representing a fixed definition with changing values
- a location in memory where data can be stored and retrieved

Text similarity

- There are many different notions of similarity depending on the domain and the application
- Today, we'll look at some different tools
- There is no one single tool that works in all domains

Text similarity approaches

sim( , ) = ?

A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him.

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him.

How can we do this?

The basics: text overlap

- Texts that have overlapping words are more similar

A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him.

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him.

Word overlap: a numerical score

- Idea 1: number of overlapping words

A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him.

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him.

sim(T1, T2) = 11

problems?

Word overlap problems

- Doesn't take into word order
- Related: doesn't reward longer overlapping sequences

A: defendant his the When lawyer into walked backs him the court, of supporters and some the victim turned their backs him to.

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him.

sim(T1, T2) = 11

Word overlap problems

Doesn't take into account length

A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him.

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him. I ate a large banana at work today and thought it was great!

sim(T1, T2) = 11

Word overlap problems

Doesn't take into account synonyms

A: When the defendant and his *lawyer* walked into the *court*, some of the victim supporters turned their backs to him.

B: When the defendant walked into the *courthouse* with his *attorney*, the crowd truned their backs on him.

sim(T1, T2) = 11

Word overlap problems

Doesn't take into account spelling mistakes

A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him.

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him. I ate a large banana at work today and thought it was great!

sim(T1, T2) = 11

Word overlap problems

Treats all words the same

A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him.

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him.

Word overlap problems

May not handle frequency properly

A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him. I ate a ***banana*** and then another ***banana*** and it was good!

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him. I ate a large ***banana*** at work today and thought it was great!

Word overlap: sets

A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him.

and
backs
court
defendant
him
…

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him.

and
backs
courthouse
defendant
him
…

Word overlap: sets

- What is the overlap, using sets?
  - |A∧B| the size of the intersection
- How can we incorporate length/size into this measure?

Word overlap: sets

- What is the overlap, using sets?
  - |A∧B| the size of the intersection
- How can we incorporate length/size into this measure?
- Jaccard index (Jaccard similarity coefficient)

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

- Dice's coefficient

$$Dice(A,B) = \frac{2\,|A \cap B|}{|A| + |B|}$$

Word overlap: sets

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \qquad Dice(A,B) = \frac{2\,|A \cap B|}{|A| + |B|}$$

How are these related?

Hint: break them down in terms of

$|A - B|$ words in A but not B

$|B - A|$ words in B but not A

$|A \cap B|$ words in both A and B

Word overlap: sets

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$$= \frac{|A \cap B|}{|A-B| + |B-A| + |A \cap B|}$$

in A but not B

in B but not A

$$Dice(A,B) = \frac{2\ |A \cap B|}{|A| + |B|}$$

$$= \frac{2\ |A \cap B|}{|A-B| + |B-A| + 2\ |A \cap B|}$$

Dice's coefficient gives twice the weight to overlapping words

Set overlap

- Our problems:
  - word order
  - length
  - synonym
  - spelling mistakes
  - word importance
  - word frequency

Set overlap measures can be good in some situations, but often we need more general tools

Bag of words representation

For now, let's ignore word order:

Clinton said banana repeatedly last week on tv, "banana, banana, banana"

(4, 1, 1, 0, 0, 1, 0, 0, …)

banana clinton said california across tv wrong capital

Frequency of word occurrence

Vector based word

A

a_1: When	1
a_2: the	2
a_3: defendant	1
a_4: and	1
a_5: courthouse	0
…	

B

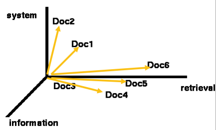
b_1: When	1
b_2: the	2
b_3: defendant	1
b_4: and	0
b_5: courthouse	1
…	

Think of these as feature vectors

How do we calculate the similarity based on these feature vectors?

Vector based similarity

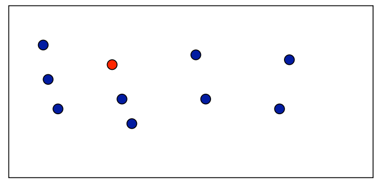
- We have a |V|-dimensional vector space
- Terms are axes of the space
- Documents are points or vectors in this space
- Very high-dimensional
- This is a very sparse vector - most entries are zero



What question are we asking in this space for similarity?

Vector based similarity

- Similarity relates to distance
- We'd like to measure the similarity of documents in the |V| dimensional space
- What are some distance measures?



Distance measures

- Euclidean (L2)

$$sim(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

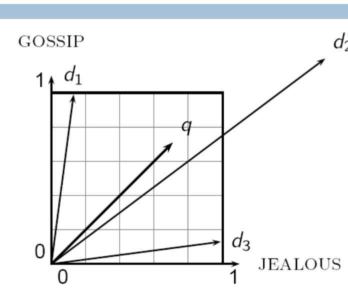
- Manhattan (L1)

$$sim(A,B) = \sum_{i=1}^{n}|a_i - b_i|$$

Distance can be problematic

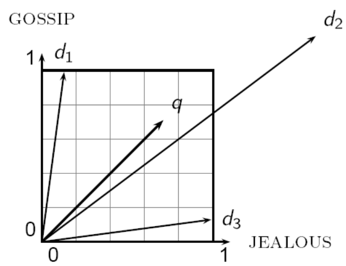
Which d is closest to q using one of the previous distance measures?

Which do you think should be closer?



Distance can be problematic

The Euclidean (or L1) distance between q and d_2 is large even though the distribution of words is similar

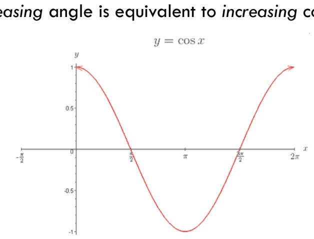


Use angle instead of distance

- Thought experiment:
  - take a document d
  - make a new document d' by concatenating two copies of d
  - "Semantically" d and d' have the same content
- What is the Euclidean distance between d and d'? What is the angle between them?
  - The Euclidean distance can be large
  - The angle between the two documents is 0

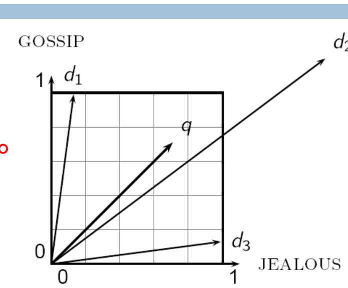
From angles to cosines

- Cosine is a monotonically decreasing function for the interval [0°, 180°]
- *decreasing* angle is equivalent to *increasing* cosine



cosine

How do we calculate the cosine between two vectors?



cosine

Dot product

$$sim_{\cos}(A,B) = A \cdot B = \sum_{i=1}^{n} a_i b_i$$

Just another distance measure, like the others:

$$sim_{L2}(A,B) = \sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$$

$$sim_{L1}(A,B) = \sum_{i=1}^{n}|a_i - b_i|$$

Dealing with length

- Thought experiment, revisited:
  - take a document d
  - make a new document d' by concatenating two copies of d
- How does sim_{cos}(d,d) relate to sim_{cos}(d, d')?
- Does this make sense?

Cosine of two vectors

$$A \cdot B = \|A\|\|B\|\cos\theta$$

$$\cos\theta = \frac{A \cdot B}{\|A\|\|B\|}$$

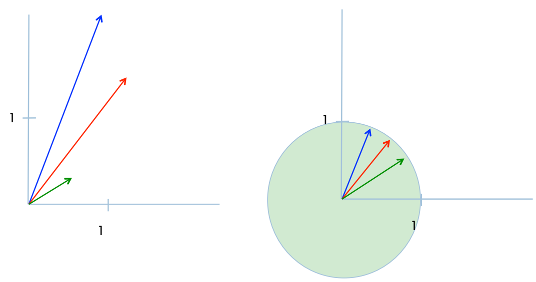
Length normalization

- A vector can be length-normalized by dividing each of its components by its length
- Often, we'll use L_2 norm (could also normalize by other norms):

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$

- Dividing a vector by its L_2 norm makes it a unit (length) vector

Unit length vectors



In many situations, normalization improves similarity, but not in all situations

Normalized distance measures

- Cosine

$$sim_{\cos}(A,B) = A \cdot B = \sum_{i=1}^{n} a_i' b_i' = \frac{\sum_{i=1}^{n} a_i b_i}{\boxed{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}}}$$

- L2

$$sim_{L2}(A,B) = \sqrt{\sum_{i=1}^{n} (a_i' - b_i')^2}$$

- L1

$$sim_{L1}(A,B) = \sum_{i=1}^{n} |a_i' - b_i'|$$

a' and b' are length normalized versions of the vectors

Cosine similarity with 3 documents

How similar are the novels:

SaS: *Sense and Sensibility*

PaP: *Pride and Prejudice*, and

WH: *Wuthering Heights*?

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

Term frequencies (counts)

Length normalized

term	SaS	PaP	WH
affection	115	58	20
jealous	10	7	11
gossip	2	0	6

term	SaS	PaP	WH
affection	0.99	0.99	0.84
jealous	0.08	0.1	0.46
gossip	0.02	0	0.25

Often becomes much clearer after length normalization

Our problems

- Which of these have we addressed?
  - word order
  - length
  - synonym
  - spelling mistakes
  - word importance
  - word frequency

Our problems

- Which of these have we addressed?
  - word order
  - length
  - synonym
  - spelling mistakes
  - word importance
  - word frequency

Word overlap problems

Treats all words the same

A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him.

B: When the defendant walked into the courthouse with his attorney, the crowd truned their backs on him.

Word importance

- Include a weight for each word/feature

A

a_1: When	1	w_1
a_2: the	2	w_2
a_3: defendant	1	w_3
a_4: and	1	w_4
a_5: courthouse	0	w_5
…		…

B

b_1: When	1	w_1
b_2: the	2	w_2
b_3: defendant	1	w_3
b_4: and	0	w_4
b_5: courthouse	1	w_5
…		…

Distance + weights

- We can incorporate the weights into the distances
- Think of it as either (*both work out the same*):
  - preprocessing the vectors by multiplying each dimension by the weight
  - incorporating it directly into the similarity measure

$$sim_{\cos}(A,B) = A \cdot B = \frac{\sum_{i=1}^{n} w_i a_i w_i b_i}{\sqrt{\sum_{i=1}^{n} (w_i a_i)^2} \sqrt{\sum_{i=1}^{n} (w_i b_i)^2}}$$

Idea: use corpus statistics

the

defendant

What would be a quantitative measure of word importance?

Document frequency

- document frequency (df) is one measure of word importance
- Terms that occur in many documents are weighted less, since overlapping with these terms is very likely
  - In the extreme case, take a word like the that occurs in EVERY document
- Terms that occur in only a few documents are weighted more

Document vs. overall frequency

- The overall frequency of a word is the number of occurrences in a dataset, counting multiple occurrences
- Example:

Word	Overall frequency	Document frequency
insurance	10440	3997
try	10422	8760

- Which word is a better search term (and should get a higher weight)?

Document frequency

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

Document frequency is often related to word importance, but we want an actual weight. Problems?

$$sim_{\cos}(A,B) = A \cdot B = \frac{\sum_{i=1}^{n} w_i a_i w_i b_i}{\sqrt{\sum_{i=1}^{n} (w_i a_i)^2} \sqrt{\sum_{i=1}^{n} (w_i b_i)^2}}$$

From document frequency to weight

Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

- weight and document frequency are **inversely** related
  - higher document frequency should have lower weight and vice versa
- document frequency is unbounded
- document frequency will change depending on the size of the data set (i.e. the number of documents)

Inverse document frequency

$$\text{idf}_w = \log \frac{N}{\text{df}_w}$$

\# of documents in dataset

document frequency of w

- idf is inversely correlated with df
  - higher df results in lower idf
- N incorporates a dataset dependent normalizer
- log dampens the overall weight

idf example, suppose *N*= 1 million

term	df_t	idf_t
calpurnia	1	
animal	100	
sunday	1,000	
fly	10,000	
under	100,000	
the	1,000,000	

What are the idfs assuming log base 10?

idf example, suppose N= 1 million

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1,000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

There is one idf value/weight for each word

idf example, suppose N= 1 million

term	df_t	idf_t
calpurnia	1	
animal	100	
sunday	1,000	
fly	10,000	
under	100,000	
the	1,000,000	

What if we didn't use the log to dampen the weighting?

idf example, suppose N= 1 million

term	df_t	idf_t
calpurnia	1	1,000,000
animal	100	10,000
sunday	1,000	1,000
fly	10,000	100
under	100,000	10
the	1,000,000	1

What if we didn't use the log to dampen the weighting?

TF-IDF

- One of the most common weighting schemes
- TF = term frequency
- IDF = inverse document frequency

$$a'_i = a_i \times \underbrace{\log N / df_i}_{\text{word importance weight}}$$

We can then use this with any of our similarity measures!