TEXT SIMILARITY

David Kauchak
CS159 Spring 2019

Admin

Assignment 4a

- Solutions posted
- If you're still unsure about questions 3 and 4, come talk to me.

Assignment 4b

Grading

Quiz #2 next Thursday covering material through 3/6

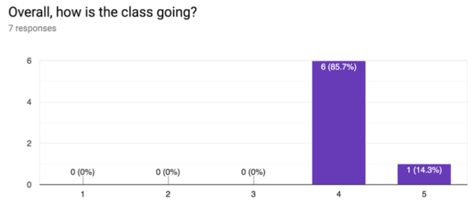
Admin

Office hours between:

- M: 3 - 3:50pm
- **T: 11am - 12**
- **Th: 11am - 12:30**
- **F: 10 - 11am**

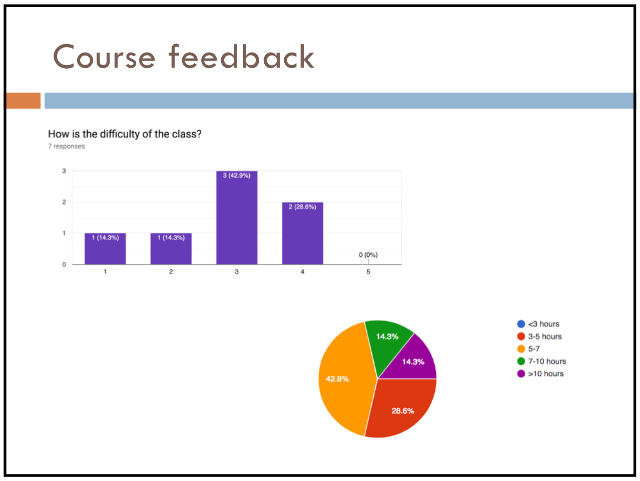
Course feedback

Thanks!


Overall, how is the class going?

7 responses

Course feedback



Course feedback

Assignments can be tough but overall are doable. Toughness is due to debugging at times.

The pace is too slow. For example, we took too much time to cover n-grams.

If the homeworks had more regular due-date (eg. every friday), it'd be easier to plan an NLP schedule.

Course feedback

It's difficult to tell exactly what parts of the material we cover we'll be expected to remember.

I know it would be a lot of work, but if possible I'd appreciate Python starter code.

Grade homeworks more quickly

Due to there being only one instructor, sometimes it is hard to get advice/answers to questions.

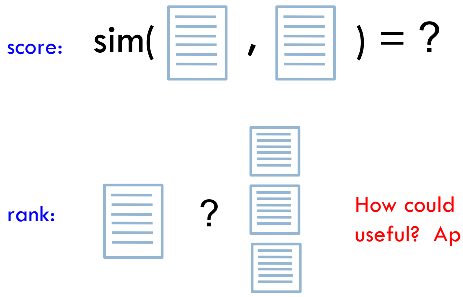
Course feedback

This is the only stem class I've taken where I haven't collaborated in person with any other students. I'm not suggesting mandatory group assignments, but I more so miss the environment mentor sessions provide. Has not been an issue though, just a consideration.

Organize mentor-less mentor sessions (some teachers in the math department do this when there are no TAs available). Students work together and answer each other's questions.

Text Similarity

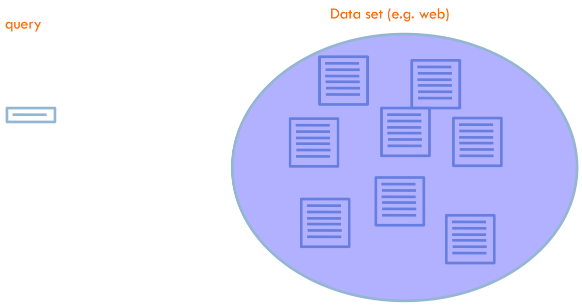
A common question in NLP is how similar are texts



How could these be useful? Applications?

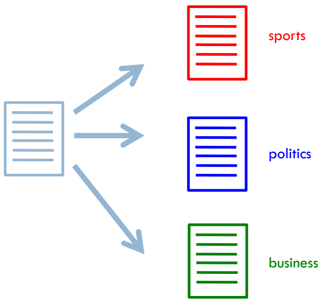
Text similarity: applications

Information retrieval (search)



Text similarity: applications

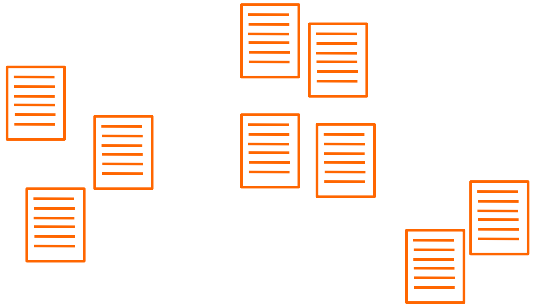
Text classification



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

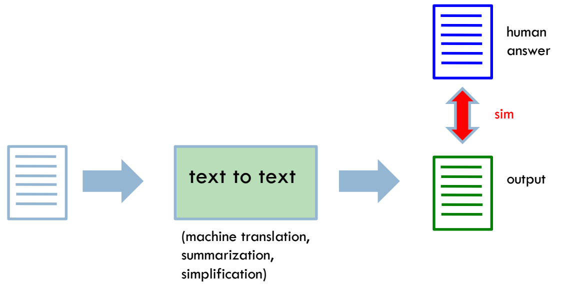
Text similarity: applications

Text clustering



Text similarity: applications

Automatic evaluation



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 account 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.

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

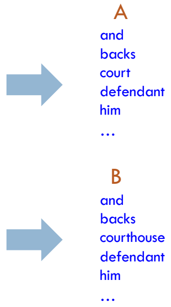
A: When the defendant and his lawyer walked into the court, some of the victim supporters turned their backs to him. I ate a *banana* and then another *banana* and it was good!

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.

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



Word overlap: sets

What is the overlap, using set notation?

- |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

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

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

What information do we lose?

Bag of words representation

For now, let's ignore word order:

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

(4, 1, 1, 0, 0, 1, 0, 0, …)

banana obama said california across tv wrong capital

"Bag of words representation": multi-dimensional vector, one dimension per word in our vocabulary

Frequency of word occurrence

Bag of words representation

http://membercentral.aaas.org/blogs/member-spotlight/tom-mitchell-studies-human-language-both-man-and-machine

Vector based word

A

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

Multi-dimensional vectors, one dimension per word in our vocabulary

B

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

How do we calculate the similarity based on these vectors?

Vector based similarity

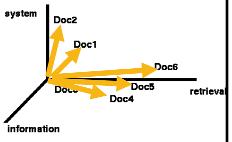
We have a |V|-dimensional vector space

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



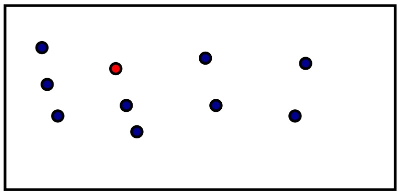
What question are we asking in this space for similarity?

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)

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

Manhattan (L1)

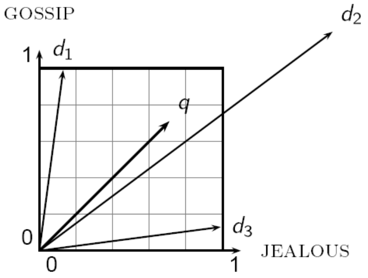
$$dist(A,B) = \sum_{i=1}^{n}|a_i - b_i|$$

What do these mean for our bag of word vectors?

Distance can be problematic

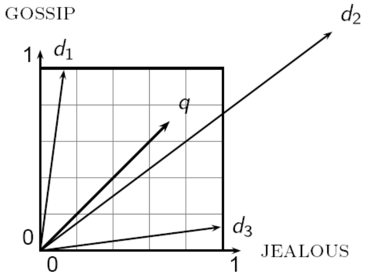
Which d is closest to q using one of the previous distance measures?

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

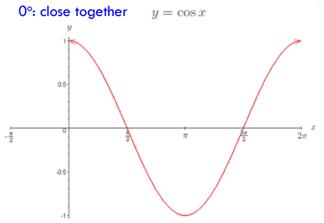
From angles to cosines

Cosine is a monotonically decreasing function for the interval [0°, 180°]

decreasing angle is equivalent to *increasing* cosine of that angle

(larger cosine means more similar)

0°: close together



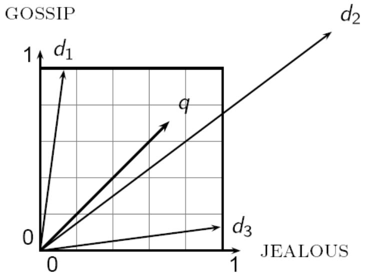
180°: far apart

Near and far

https://www.youtube.com/watch?v=iZhEcRrMA-M

cosine

How do we calculate the cosine between two vectors?



Cosine of two vectors

Dot product

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

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

Dot product between unit length vectors

Cosine as a similarity

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

ignoring length normalization

Just another distance measure, like the others:

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

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

Cosine as a similarity

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

ignoring length normalization

For bag of word vectors, what does this do?

Cosine as a similarity

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

ignoring length normalization

Only words that occur in *both* documents count towards similarity

Words that occur more frequently in both receive more weight

Length normalization

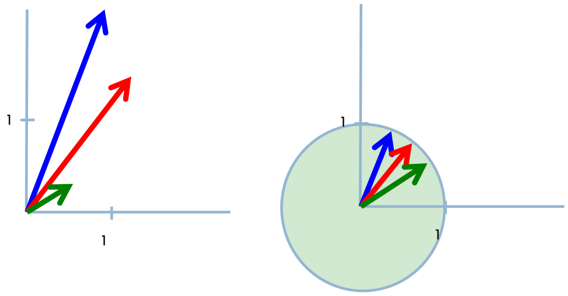
A vector can be length-normalized by dividing each of its components by its length

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

$$dist_{L2}(A,B) = \sqrt{\sum_{i=1}^{n} (a'_i - b'_i)^2}$$

L1

$$dist_{L1}(A,B) = \sum_{i=1}^{n} |a'_i - b'_i|$$

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

Distance measures

Cosine

$$sim_{\cos}(A,B) = A \cdot B = \sum_{i=1}^{n} a'_i b'_i$$

L2

$$dist_{L2}(A,B) = \sqrt{\sum_{i=1}^{n} (a'_i - b'_i)^2}$$

L1

$$dist_{L1}(A,B) = \sum_{i=1}^{n} |a'_i - b'_i|$$

Cosine is the most common measure. Why do you think?

Distance measures

Cosine

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

L2

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

L1

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

- L1 and L2 penalize sentences for not having words, i.e. if a has it but b doesn’t
- Cosine can be significantly faster since it only calculates over the intersection

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.

Ideas?

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?

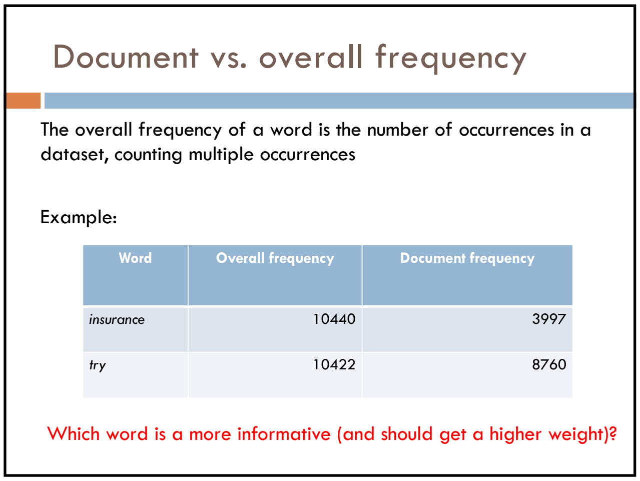
Document frequency

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 almost EVERY document

Terms that occur in only a few documents are weighted more



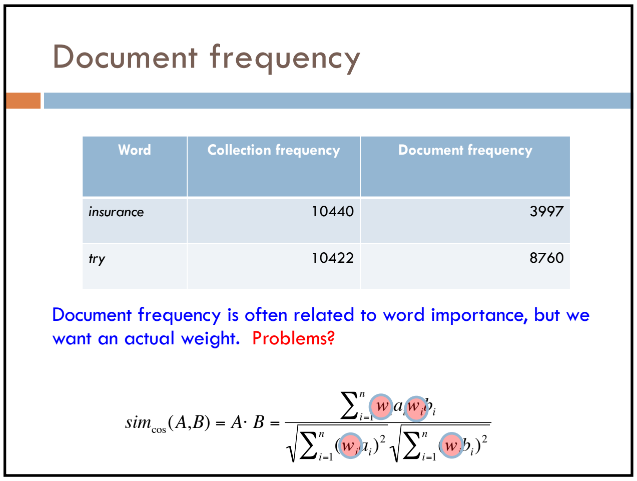
Document vs. overall frequency

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 more informative (and should get a higher weight)?

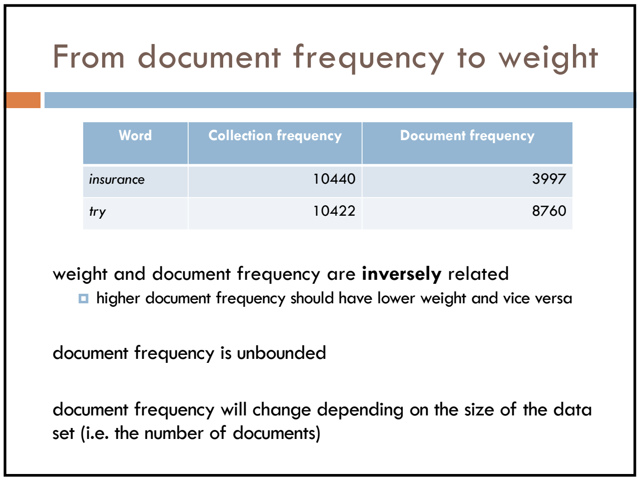


Document frequency

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

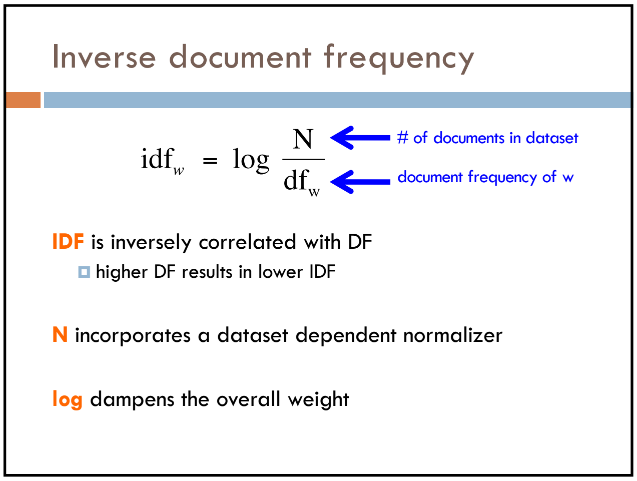
Word	Collection frequency	Document frequency
insurance	10440	3997
try	10422	8760

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

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

# of documents in dataset

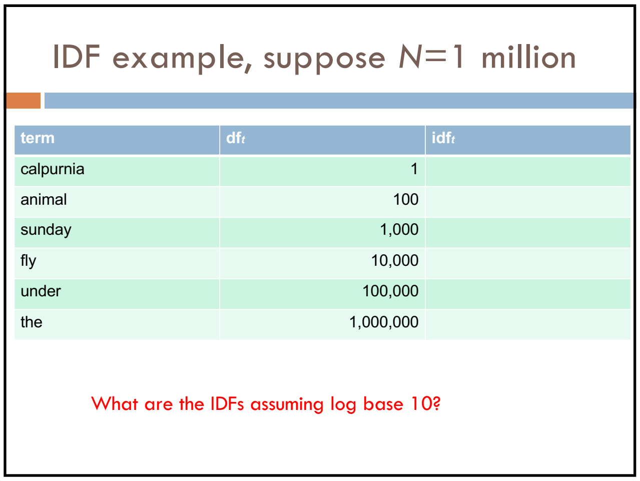
document frequency of w

IDF is inversely correlated with DF

- higher DF results in lower IDF

N incorporates a dataset dependent normalizer

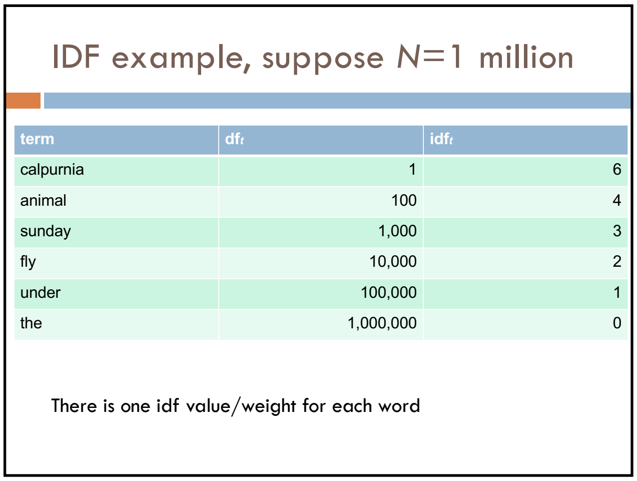
log dampens the overall weight



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	

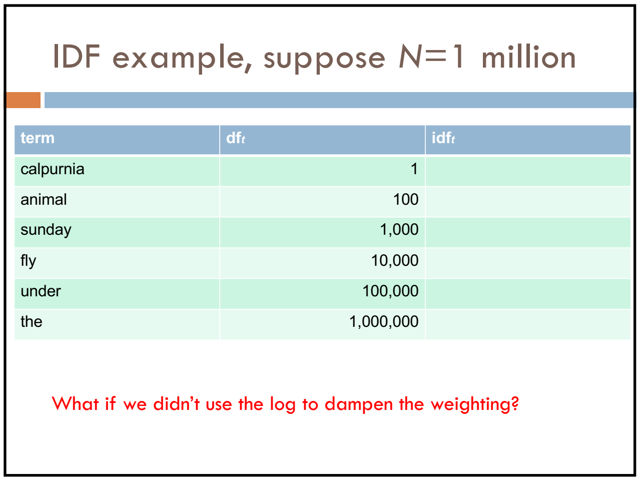
What are the IDFs assuming log base 10?



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

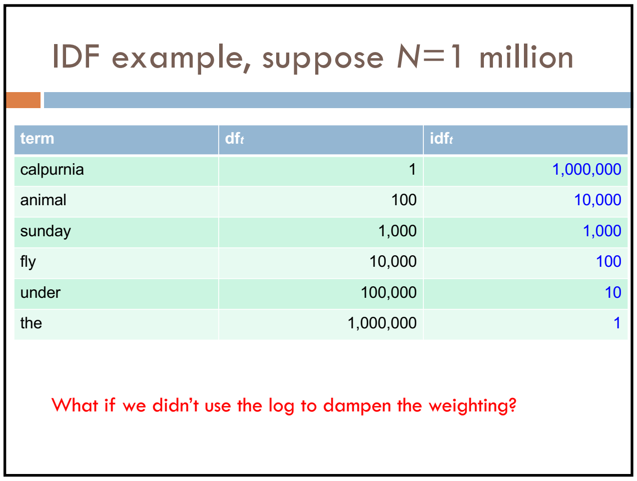
There is one idf value/weight for each word



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

$$a'_i = \underbrace{a_i}_{\text{TF}} \times \underbrace{\log N / \text{df}_i}_{\text{IDF (word importance weight)}}$$

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

Stoplists: extreme weighting

Some words like ‘a’ and ‘the’ will occur in almost every document

- IDF will be 0 for any word that occurs in all documents
- For words that occur in almost all of the documents, they will be nearly 0

A ***stoplist*** is a list of words that should **not** be considered (in this case, similarity calculations)

- Sometimes this is the *n* most frequent words
- Often, it’s a list of a few hundred words manually created

Stoplist

I
a
aboard
about
above
across
after
afterwards
against
agin
ago
agreed-upon
ah
alas
albeit
all
all-over
almost
along
alongside
altho
although
amid
amidst
among
amongst
an
and
another
any
anyone
anything
around
as
aside
astride
at
atop
avec
away
back
be
because
before
beforehand
behind
behynde
below
beneath
beside
besides
between
bewteen
beyond
bi
both
but
by
ca.
de
des
despite
do
down
due
durin
during
each
eh
either
en
every
ever
everyone
everything
except
far
fer
for
from
go
goddamn
goody
gosh
half
have
he
hell
her
herself
hey
him
himself
his
ho
how

If most of these end up with low weights anyway, why use a stoplist?

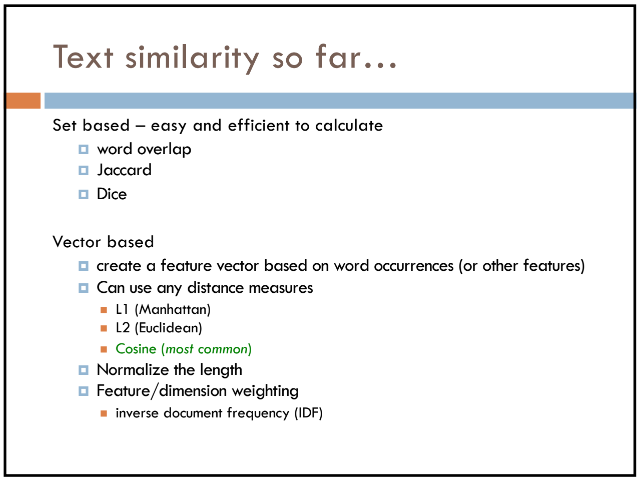
Stoplists

Two main benefits

- More fine grained control: some words may not be frequent, but may not have any content value (alas, teh, gosh)
- Often does contain many frequent words, which can drastically reduce our storage and computation

Any downsides to using a stoplist?

- For some applications, some stop words may be important



Text similarity so far…

Set based – easy and efficient to calculate

- word overlap
- Jaccard
- Dice

Vector based

- create a feature vector based on word occurrences (or other features)
- Can use any distance measures
  - L1 (Manhattan)
  - L2 (Euclidean)
  - Cosine (*most common*)
- Normalize the length
- Feature/dimension weighting
  - inverse document frequency (IDF)