Admin

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

Assignment 4b

Quiz #2 next Wednesday covering material through 3/1
Completing the coding assignments and playing with the algorithms.

I do like how we really get into the trenches about implementing the algorithms.

It's very engaging and the homeworks are good for my coding skills.

I am ambivalent towards the assignments. On one hand, they are extremely time intensive, but on the other, I feel proud after having completed them, and I feel I strengthened myself as a programmer.

Maybe including some content related to more recent NLP stuff would be interesting.

Assignments need to be less vague, more detail on how we actually implement it. They are very interesting, but also very difficult.

Assignments could be more detailed.
Course feedback
Not gonna lie, this class feels a lot harder than most of my other cs classes. The structure of the class feels less intuitive, and though related to the class material, the homeworks seem much harder than what we go over in class
way too much workload
the homeworks are veeeery hard and time consuming

Course feedback
Maybe if we had a constant due date for assignments (like same day every week), we’d be able to organize time better. There have been times I was less free before deadlines which made completing the assignment and attending mentor sessions harder. If it was the same time every week, I feel like I would eventually find a routine for getting things done in time.

Text Similarity
A common question in NLP is how similar are texts
score: \( \text{sim}(\text{query}, \text{data set}) = ? \)
rank: ?
How could these be useful? Applications?

Text similarity: applications
Information retrieval (search)
query
Data set (e.g. web)
These “documents” could be actual documents, for example using k-means or pseudo-documents, like a class centroid/average.

Automatic evaluation

\[
sim(\text{banana}, \text{apple}) = ?
\]

Word-sense disambiguation

I went to the \text{bank} to get some money.
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
- 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

$\text{sim}(\text{A}, \text{B}) = ?$

**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 turned 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 turned 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 turned their backs on him.

\[ \text{sim}(T_1, T_2) = 11 \]

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 to.

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

\[ \text{sim}(T_1, T_2) = 11 \]

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 turned their backs on him.

\[ \text{sim}(T_1, T_2) = 11 \]

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 turned their backs on him.

\[ \text{sim}(T_1, T_2) = 11 \]
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.

\[ \text{sim}(T1, T2) = 11 \]

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 turned their backs on him.

Word overlap: sets

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!

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 turned their backs on him.
Word overlap: sets

What is the overlap, using set notation?
- \( |A \cap B| \) 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**

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

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

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.

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, 0, 1, 0, 0, ...)

```
banana    obama    across    tv
wrong     capital    banana
```

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

Frequency of word occurrence

Bag of words representation

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?

Distance measures:

Euclidean (L2)

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

Manhattan (L1)

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

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?

---

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 \( d' \)?
- 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 \([0^\circ, 180^\circ]\)

decreasing angle is equivalent to increasing cosine of that angle
(larger cosine means more similar)

---
Near and far

https://youtu.be/E9luXEwpU7U

How do we calculate the cosine between two vectors?

Cosine of two vectors

Dot product

\[ A \cdot B = \|A\| \|B\| \cos \theta \]

\[ \cos \theta = \frac{A \cdot B}{\|A\| \|B\|} = \frac{A \cdot B}{\|A\| \cdot \|B\|} \]

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 \quad \text{(ignoring length normalization)}

Just another distance measure, like the others:

\[ dist_{d_2}(A, B) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \]

\[ dist_{d_1}(A, B) = \sum_{i=1}^{n} |a_i - b_i| \]
Cosine as a similarity

\[ \text{sim}_{\cos}(A, B) = A \cdot B = \sum_{i=1}^{n} a_i b_i \]

Ignoring length normalization

For bag of word vectors, what does this do?

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):

\[ \|\mathbf{x}\|_2 = \sqrt{\sum x_i^2} \]

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

<table>
<thead>
<tr>
<th>Cosine</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{sim}<em>{\text{cos}}(A,B) = A \cdot B = \sum</em>{i=1}^{n} a_i b_i' )</td>
<td></td>
</tr>
<tr>
<td>( \text{dist}<em>{\text{L2}}(A,B) = \sqrt{\sum</em>{i=1}^{n} (a'_i - b'_i)^2} )</td>
<td></td>
</tr>
<tr>
<td>( \text{dist}<em>{\text{L1}}(A,B) = \sum</em>{i=1}^{n}</td>
<td>a'_i - b'_i</td>
</tr>
</tbody>
</table>

L2 and L1 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.

Distance measures

<table>
<thead>
<tr>
<th>Cosine</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{sim}<em>{\text{cos}}(A,B) = A \cdot B = \sum</em>{i=1}^{n} a_i b_i' )</td>
<td></td>
</tr>
<tr>
<td>( \text{dist}<em>{\text{L2}}(A,B) = \sqrt{\sum</em>{i=1}^{n} (a'_i - b'_i)^2} )</td>
<td></td>
</tr>
<tr>
<td>( \text{dist}<em>{\text{L1}}(A,B) = \sum</em>{i=1}^{n}</td>
<td>a'_i - b'_i</td>
</tr>
</tbody>
</table>

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

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 turned their backs on him.

Ideas?

Word importance
Include a weight for each word/feature

A
- When 1 w1
- the 2 w2
- defendant 1 w3
- and 1 w4
- courthouse 0 w5
- ...

B
- When 1 w1
- the 2 w2
- defendant 1 w3
- and 0 w4
- courthouse 1 w5
- ...

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

\[
sim_{\text{cos}}(A,B) = \frac{\sum_{i=1}^{n} w_i a_i b_i}{\sqrt{\sum_{i=1}^{n} (w_i a_i)^2} \sqrt{\sum_{i=1}^{n} (w_i b_i)^2}}
\]
Idea: use corpus statistics

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:

<table>
<thead>
<tr>
<th>Word</th>
<th>Overall frequency</th>
<th>Document frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>insurance</td>
<td>10440</td>
<td>3997</td>
</tr>
<tr>
<td>try</td>
<td>10422</td>
<td>8760</td>
</tr>
</tbody>
</table>

Which word is a more informative (and should get a higher weight)?

---

**Document frequency**

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

$$\text{sim}_{\text{cos}}(A, B) = \frac{A \cdot B}{\sqrt{\sum_i (A_i)^2} \sqrt{\sum_i (B_i)^2}}$$
From document frequency to weight

<table>
<thead>
<tr>
<th>Word</th>
<th>Collection frequency</th>
<th>Document frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>insurance</td>
<td>10440</td>
<td>3997</td>
</tr>
<tr>
<td>try</td>
<td>10422</td>
<td>8760</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>term</th>
<th>df</th>
<th>idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>calpurnia</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>animal</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>sunday</td>
<td>1,000</td>
<td>3</td>
</tr>
<tr>
<td>fly</td>
<td>10,000</td>
<td>2</td>
</tr>
<tr>
<td>under</td>
<td>100,000</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>1,000,000</td>
<td>0</td>
</tr>
</tbody>
</table>

What are the IDFs assuming log base 10?

IDF example, suppose $N=1$ million

<table>
<thead>
<tr>
<th>term</th>
<th>df</th>
<th>idf</th>
</tr>
</thead>
<tbody>
<tr>
<td>calpurnia</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>animal</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>sunday</td>
<td>1,000</td>
<td>3</td>
</tr>
<tr>
<td>fly</td>
<td>10,000</td>
<td>2</td>
</tr>
<tr>
<td>under</td>
<td>100,000</td>
<td>1</td>
</tr>
<tr>
<td>the</td>
<td>1,000,000</td>
<td>0</td>
</tr>
</tbody>
</table>

There is one idf value/weight for each word
**IDF example, suppose N=1 million**

<table>
<thead>
<tr>
<th>term</th>
<th>dfi</th>
<th>idfi</th>
</tr>
</thead>
<tbody>
<tr>
<td>calpurnia</td>
<td>1</td>
<td>1,000,000</td>
</tr>
<tr>
<td>animal</td>
<td>100</td>
<td>10,000</td>
</tr>
<tr>
<td>sunday</td>
<td>1,000</td>
<td>1,000</td>
</tr>
<tr>
<td>fly</td>
<td>10,000</td>
<td>10</td>
</tr>
<tr>
<td>under</td>
<td>100,000</td>
<td>10</td>
</tr>
<tr>
<td>the</td>
<td>1,000,000</td>
<td>1</td>
</tr>
</tbody>
</table>

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

$$a'_i = a_i \times \log \frac{N}{df_i}$$

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

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

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)