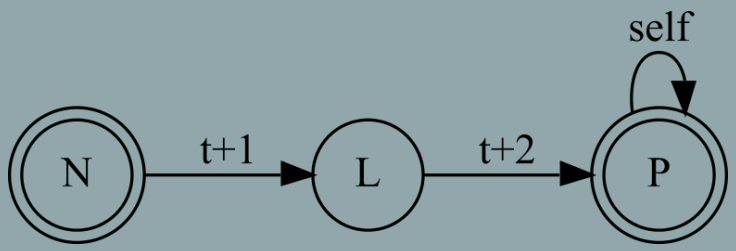
LING-362

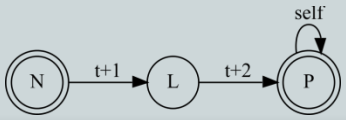
Introduction to **N**atural **L**anguage **P**rocessing

Topic Modelling I



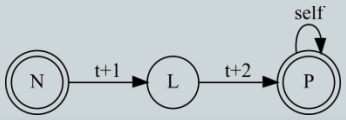
Parsing – review

- CFG parsers work by constructing a tree of phrases – grammars are bi-directional:
  - Top down: LHS > RHS
  - Bottom up: LHS < RHS
- Algorithm for possible (binary) trees: CKY
  - Try to identify places to split the input
  - Do two things show up as a RHS somewhere?
  - Use dynamic programming: keep a table of all possible pairs, look up only once
  - Requires grammars in **CNF**



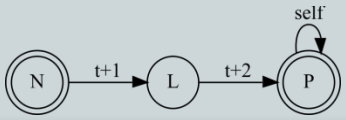
Parsing – review

- Some big problems in rule-based parsing:
  - Data sparseness:
    - Some rules in Treebank very rare (17,000+ rules from PTB!)
    - Presumably many others are unattested
  - Ambiguity:
    - Many sentences have enormous number of possible parses
- PCFGs:
  - Use probabilities to disambiguate possible parses
  - Probabilities for decompositions of each LHS



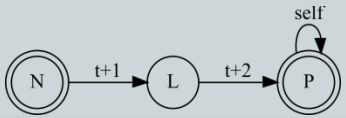
Parsing – review

- Context sensitivity and lexicalized parsing:
  - Parsers always choose most frequent decomposition – rare readings can become impossible
  - Sometimes probability differs in syntactic context (object/subject NP, certain words)
    - Use **parent annotation** (NP^S = subject NP)
    - Use **lexicalized head percolation** ($NP_{[PRP,me]}$)
    - Optimize **label splitting / merging**



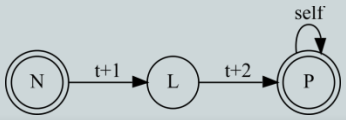
Information retrieval and NLP

- So far we've been looking at **forms**
  - Morphological affixes
  - Parts of speech
  - Sentence structure – *colorless green ideas…*
- Today we take a look at broader **meanings**
  - What kind of **information** are you looking for?
  - What is a document **about**?
  - Do these pieces of text mean something **similar**?



From words to documents

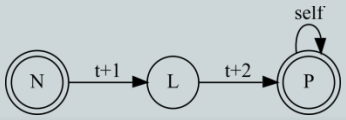
- Words and sentences are not the only data points for processing in NLP
- Some processing methods target **documents**
  - Each document is a data point
  - Try to extract certain properties from each data point
  - Common applications:
    - **Information retrieval**
    - **Document similarity**
    - **Topic modelling**



Topic Modelling

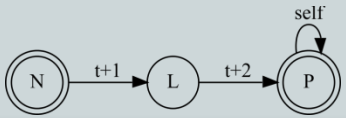
- Trying to figure out **topics** in documents
  - Approaches from automatic text summarization:
    - Extract **snippets** – *this sentence is the most pertinent because…*
      (may require **coreference resolution**, **templates**, other types of **pre-processing**…)
    - Give more 'semantic' answers – return single phrase answer via key word extraction (TF/IDF, semantic parsing…)

extractive ⟷ abstractive



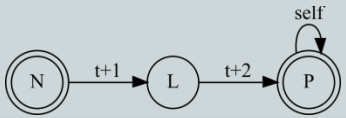
Topic Modelling

- Trying to figure out **topics** in documents
  - **Similarity based** approaches:
    - *This document is about* Middle East politics *because it's* ***similar to*** other stories in this rubric*… (→* ***supervised****)*
    - Or: we don't know what these documents are about **but –** they're similar **to each other** *(→* ***unsupervised****)*
- If we obtain 10 good clusters for 10,000 documents, naming those clusters may be a small problem ☺



What documents are about

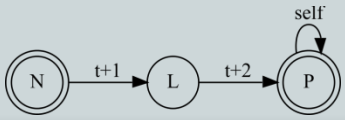
- Documents can be seen as sequences of tokens and other features:
  - We know how to tokenize plain text
  - Tokens can carry hidden features (POS tags and more)
  - Groups of tokens can form higher structures (trees)
- Which of these are most indicative of document meaning?



How do we know what this is about?

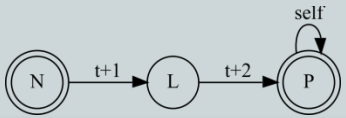
The U.S. Navy and Coast Guard rescued three mariners from a remote, uninhabited Pacific island Thursday after a Navy plane spotted palm fronds spelling the word "help" on the sand.

The castaways who constructed the makeshift S.O.S. had been stranded on Fanadik Island for three days, according to the Coast Guard. This island lies about 2,600 miles southwest of Honolulu.



Some words are “not interesting”

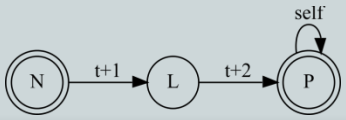
- A simple approach would be to only look at ‘content words’ (nouns and verbs?)
  - But POS tags like NN don’t guarantee content-y-ness
    - *A/DT **lot/NN** of/IN …*
    - ***Take/VB** for/IN **example/NN** …*
- More often, **stop lists** are used:
  - Manual curation and/or top frequencies
  - *if, in, take, do, lot, make, while, to* …



What would you consider a ‘stop word’?

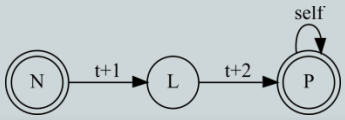
- Try crossing out words that you think are not useful for establishing what the text on the sheet is about
  - Can we agree on the stop words?
  - Can you still understand the text with the words crossed out?

~~The~~ U.S. Navy and Coast Guard rescued three mariners from a remote, uninhabited Pacific island Thursday after a Navy plane spotted palm fronds spelling the word "help" on the sand.



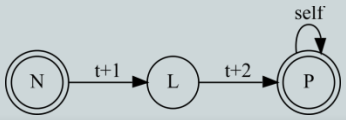
Just ‘content words’

U.S. Navy Coast Guard rescued mariners remote, uninhabited Pacific island Thursday Navy plane spotted palm fronds spelling word "help" sand. castaways constructed makeshift S.O.S. stranded Fanadik Island Coast Guard. island lies southwest Honolulu.



But what is it about?

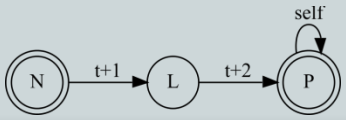
- So the article is not about the word ‘the’
  - But is it about the coast guard?
  - About Honolulu?
  - About palm fronds?



But what is it about?

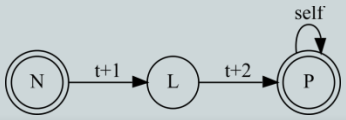
- Answer 1: use…
- Frequencies!
  - Island 3
  - Navy 2
  - Coast 2
  - Guard 2
  - palm 1
  - fronds 1
  - castaways 1
  - Honolulu 1
  - Fanadik 1
  - stranded 1

We'd like some things to go together



MWEs, lemmatization & stemming

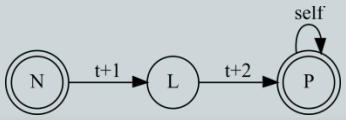
- Some words go together and form a unit:
  - **Multiword Expressions:** *Coast Guard, as well*
- And some words are different forms of the same lexical item:
  - *rescue, rescued* → **lemma**: rescue
- And some words share the same **stem**:
  - *rescue, rescued, rescuer* → **stem**: rescue



How to lemmatize and stem

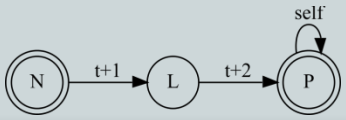
- Very many stemmers out there
- Porter and Snowball are freely available classics:

```
from nltk import word_tokenize
from nltk.stem import snowball

text = "Instructive instructions are less useful than exemplifying examples"

tokens = word_tokenize(text)
my_stemmer = snowball.SnowballStemmer("english")

for token in tokens:
  print(my_stemmer.stem(token))
```



How to lemmatize and stem

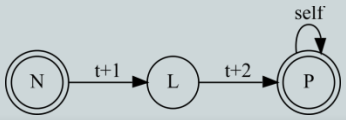
instruct
instruct
are
less
use
than
exemplifi
exampl



How to lemmatize and stem

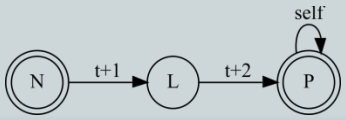
- Lemmatization is often more reliable, but collapses fewer forms
- We can use the nltk WordNetLemmatizer (not great but works mostly)

```
from nltk.stem import WordNetLemmatizer
from nltk import pos_tag, word_tokenize
from nltk.corpus import wordnet
```



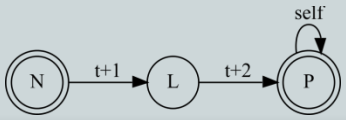
How to lemmatize and stem

```
text = "Instructive instructions are less useful than
exemplifying examples"
tokens = word_tokenize(text)
tagged = pos_tag(tokens)

my_lemmatizer = WordNetLemmatizer()

for token, tag in tagged:
    wordnet_tag = get_wordnet_pos(tag)
    if wordnet_tag is not None:
        print(token + "\t" +tag + "\t" +
                    my_lemmatizer.lemmatize(token,wordnet_tag))
    else:
        print(token + "\t" +tag + "\t" + token)
```



How to lemmatize and stem

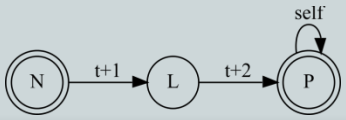
```
def get_wordnet_pos(PTB_tag):
    if PTB_tag.startswith('J'):
        return wordnet.ADJ
    elif PTB_tag.startswith('V'):
        return wordnet.VERB
    elif PTB_tag.startswith('N'):
        return wordnet.NOUN
    elif PTB_tag.startswith('R'):
        return wordnet.ADV
    else: # indeclinable word
        return None
```



Collapsed frequencies

- Answer 2: use…
- Collapsed frequencies!
  - Island 3
  - Coast Guard 2
  - Navy 1
  - U.S. Navy 1
  - palm frond 1
  - castaway 1
  - Honolulu 1
  - Fanadik 1
  - strand 1

We have better grouping now… But what is this document like?

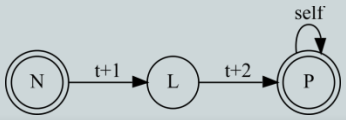


Is the following document similar?

One man went from camper to castaway in a turn of the tide.

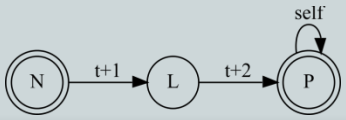
An Illinois man who drifted in a dinghy from the Florida Keys to a desolate island in the Bahamas is lucky to be alive after six days spent on the open ocean and then marooned, according the U.S. Coast Guard who rescued him Monday night.

Larry Sutterfield, 39, planned to go camping when he took his dinghy under the Seven Mile Bridge in Marathon, Fla. But the current took him toward the Florida Straits and then to a small spit of land along the Cay Sal Bank in the Bahamas, The Sun-Sentinel reported.



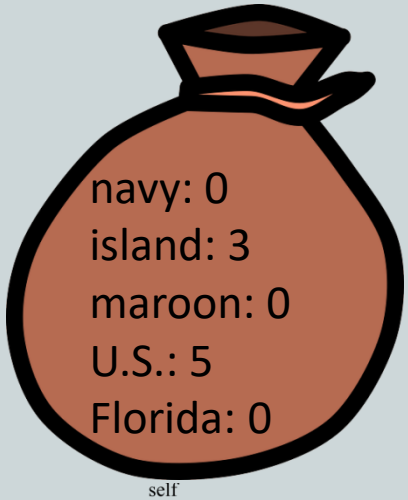
Is the following document similar?

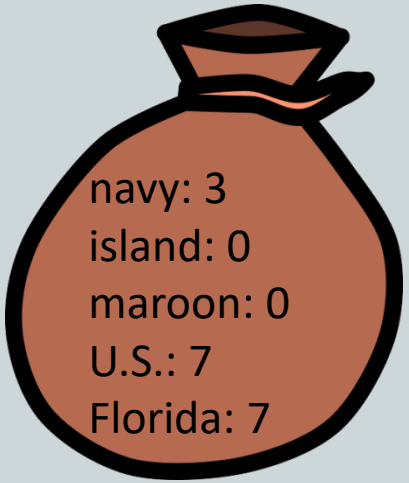
• Island	3	• dinghy	2
• Coast Guard	2	• Florida	2
• Navy	1	• Bahama	2
• U.S. Navy	1	• man	2
• palm frond	1	• U.S.	1
• castaway	1	• Coast Guard	1
• Honolulu	1	• rescue	1
• Fanadik	1	• island	1
• strand	1	• maroon	1

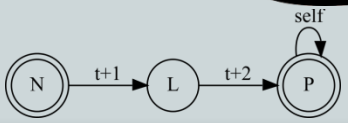


Bag of words model

- We're basically treating documents as frequency lists
  - No information about **word order**
  - Not all documents have all words – so lots of 0's





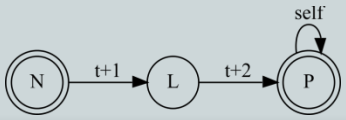


Can we build our own BoW model?

- Think about the tagging assignment's common noun frequency counts
- What kind of data structure would you get if you collected frequencies for the **same** words for multiple input documents?

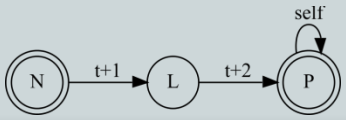
JJ? → +=1

words["..."]



Bags of words as vectors in a matrix

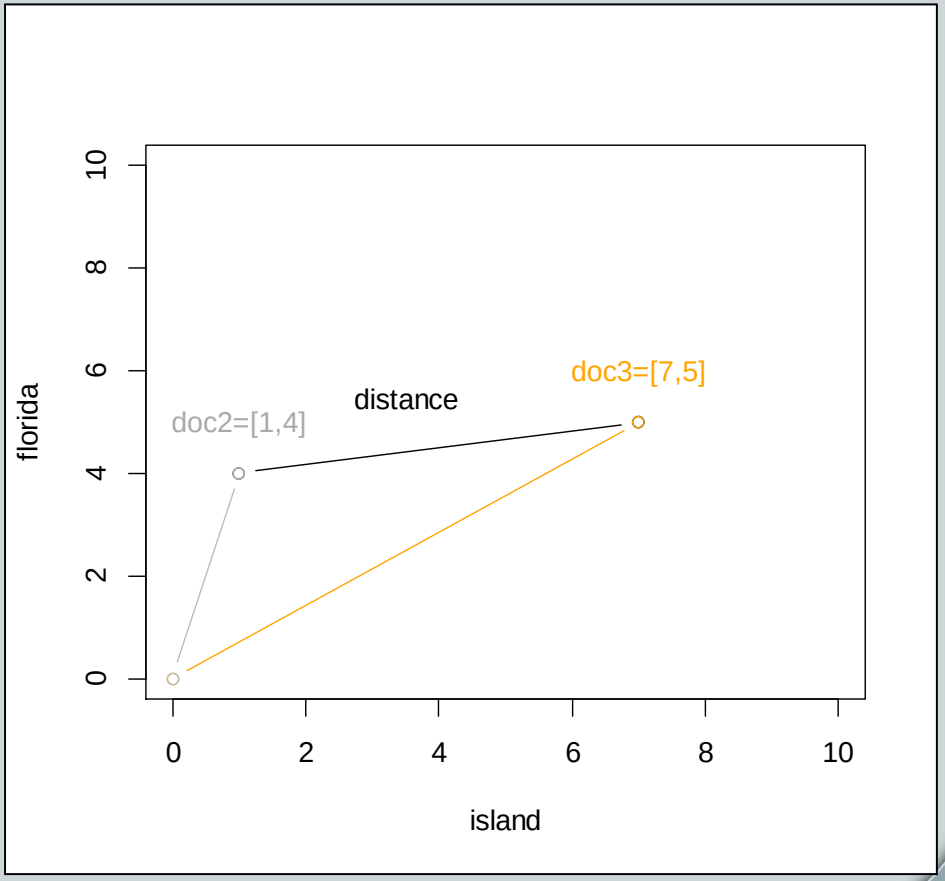
	doc1	doc2	doc3
navy	0	3	3
Island	3	2	0
maroon	0	4	0
U.S.	5	1	7
Florida	0	1	7

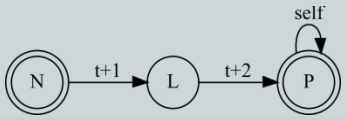


Words in **Vector Space**

- If all we care about is frequency similarity for ***island*** and ***florida***:
  - Measure similarity as Euclidean distance
  - Each feature is a coordinate

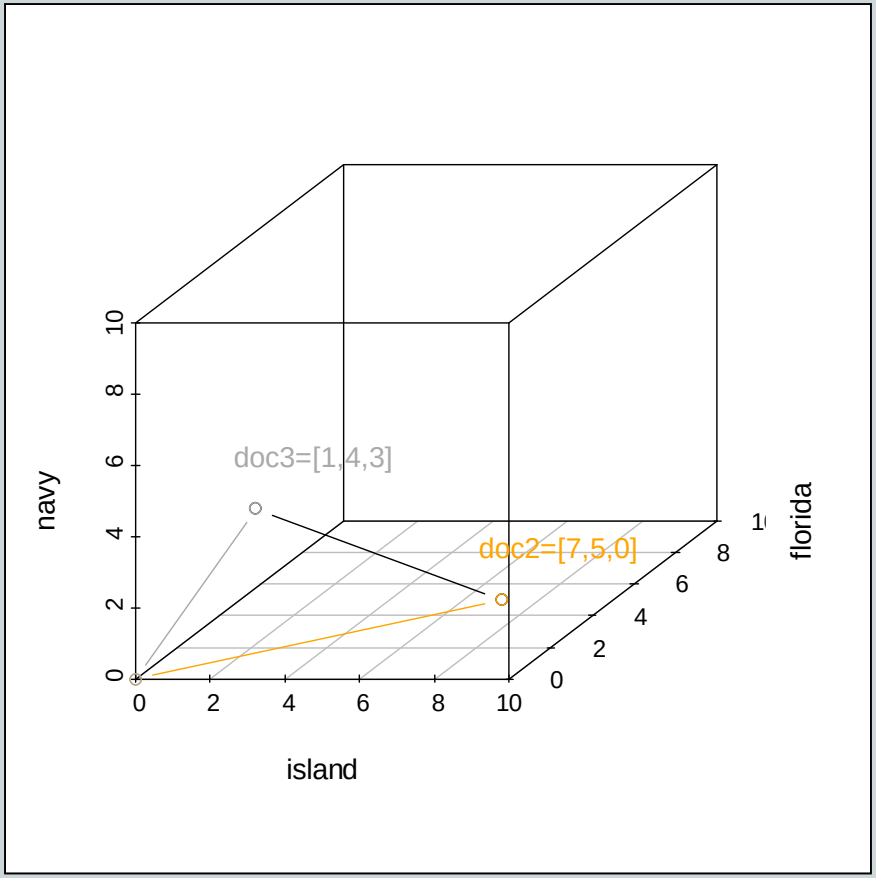
$$\sqrt{\sum_{i=1}^{N}(a_i - b_i)^2}$$

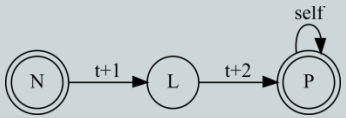




How to measure distance?

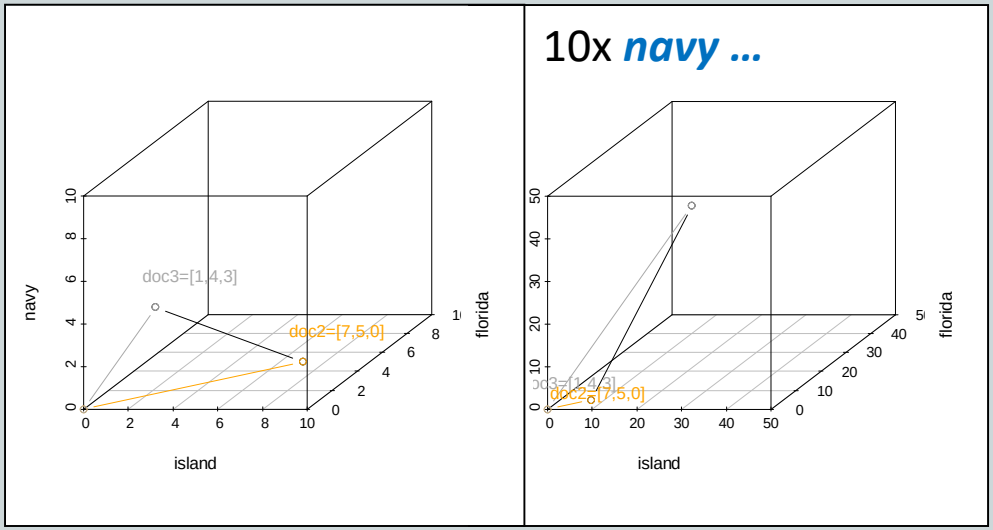
- We can add a third dimension (recall embeddings)
- Adding more?
  - Hard to visualize
  - Distance still calculable
  - But should each word be a dimension?
- Problems:
  - **Document length**
  - **Scaling**
  - **Term specificity**
  - **Collinearity**

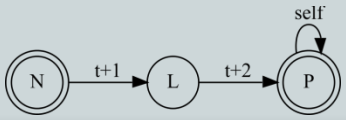




Document length

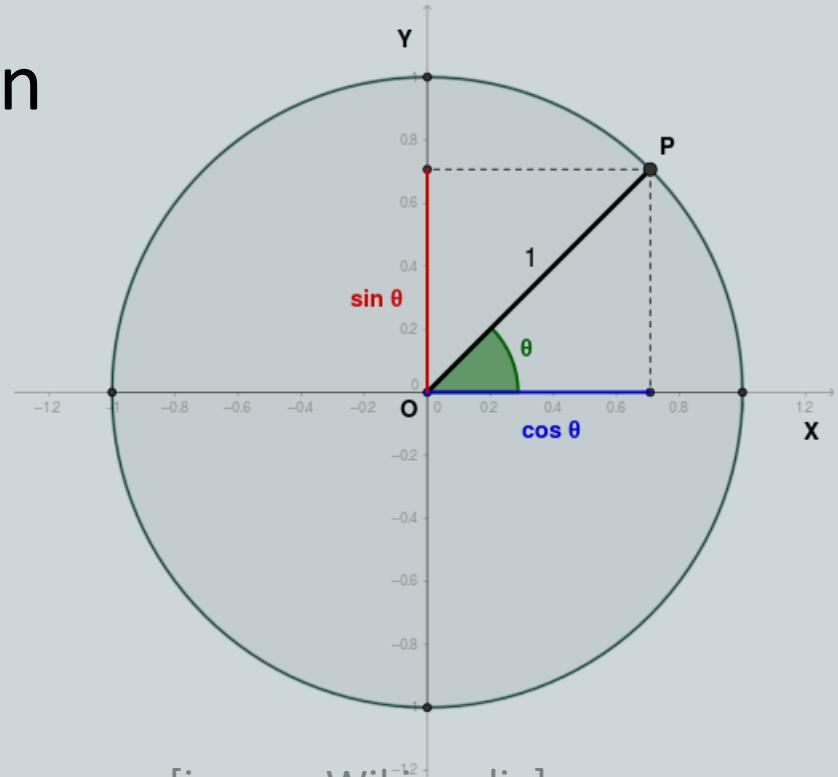
- Distance might seem like a good idea but…
  - Longer documents have more words
    - A short document might not mention the U.S. Coast Guard often
    - But the fact that it does so in just 100 words seems significant
  - Still, **the angle** remains the same
  - ➢ Normalize length
  - ➢ **Cosine similarity**



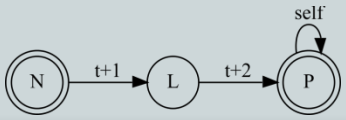


Cosine similarity

- Works much like cosine in trigonometry:
  - **1 -> zero angle (same)**
  - **0 -> orthogonal (90°)**
  - **-1 -> opposite (180°)**



[image: Wikimedia]

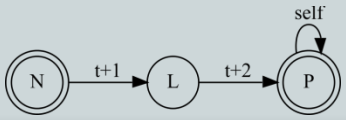


Cosine similarity

- But still works for n-dimensional vectors!

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

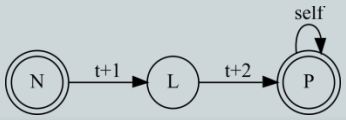
  - Dot product of two vectors divided by the product of their magnitudes
    - Dot product: multiply vectors cell-wise and sum
    - Magnitude: $\|X\| = \sqrt{x_1^2 + x_2^2 .. + x_n^2}$



Scaling

- Even with cosine similarity, the proportion of word frequencies gives the direction
  - If word 1 appears **once** and word 2 appear **twice:**
    - proportion 1:2
  - Now consider words appearing 10 vs. 20 times:
    - proportion 1:2

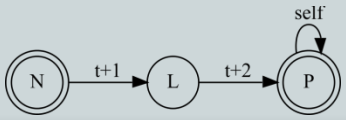
- Is a word appearing twice really twice as important as one appearing once?
- Is it the same for 10 vs. 20?

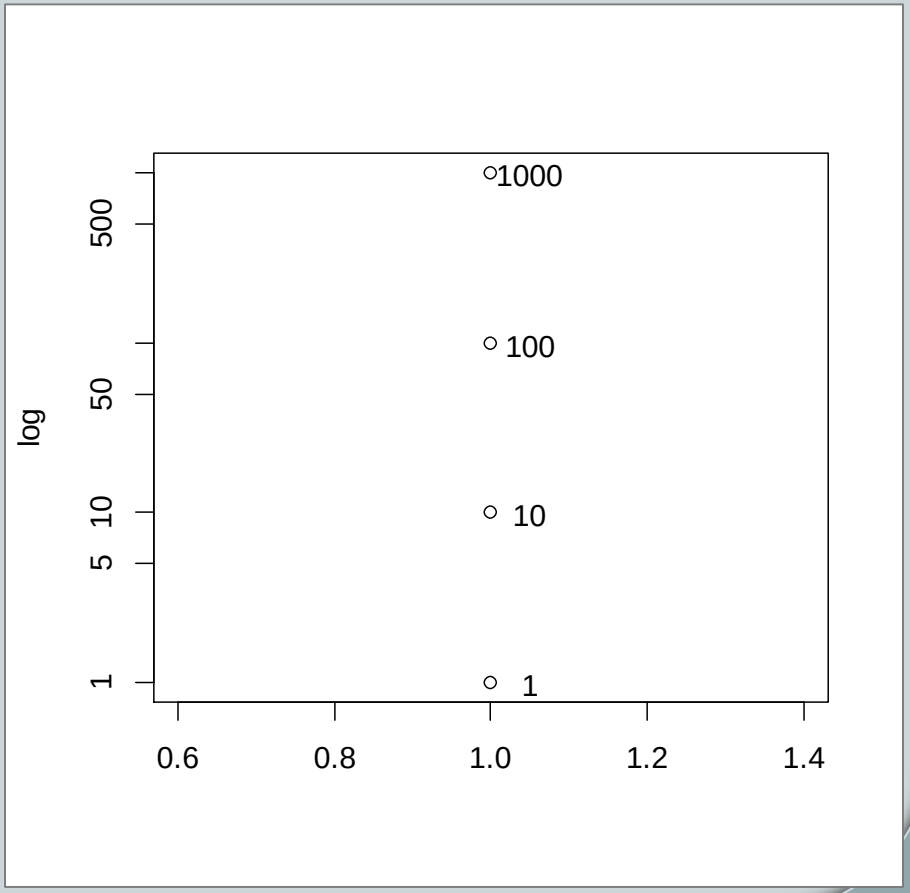


Scaling

- A common solution is to take **log** frequencies
  - E.g. log base 10
  - Difference between 1 and 10 same as difference between 10 and 100, 100 and 1000, …

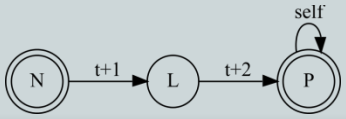
```
from math import log10
print(log10(5))
```





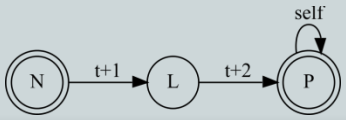
Term specificity

- A more fundamental problem with VSMs is that we have different ideas about what's important
  - Very frequent (non-stop) word is important?
  - Suppose our document contains:
    - ***insurance*** 10
    - ***try*** 10
    - ***story*** 10
  - What is it about?
  - How can we tell which is more important?



Collection frequency

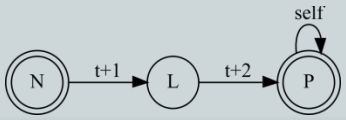
- Some terms might generally be very frequent
  - Appearance less surprising – assign less importance
  - Use **Collection Frequencies** (sum over all documents, adapted NYT example from Manning & Schütze 1999)
    - ***insurance*** 10440
    - ***try*** 10422
    - ***story*** 23591 (less surprising)
  - How are ***insurance*** and ***try*** still different?



Document frequency

- Even with equal collection frequency, terms appear in different amounts of **documents**

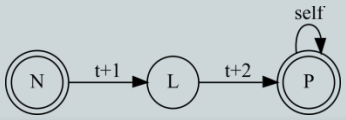
	term	*collection*	*document*
• ***insurance***	10	10440	3997
• ***try***	10	10422	8760
• ***story***	10	23591	10897



So if we know all this

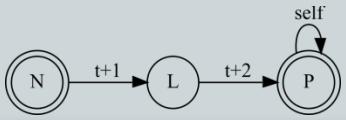
- Answer 3: use…
- Collapsed collection and document frequencies!
  - Island 3
  - Coast Guard 2
  - Navy 1
  - U.S. Navy 1
  - palm frond 1
  - castaway 1
  - Honolulu 1
  - Fanadik 1
  - Strand 1

Is this as much about Honolulu as it is about Fanadik?



Just one number

- To get just one number representing a term's relevance in a document:
  - Use log term frequency (TF): log(TF)
  - Weight it by proportion of documents with this term (DF) in an N document collection
- But we want **inverse** weighting – high document count is bad, so:
  - Inverse document frequency (IDF): log(N/DF)



TF-IDF

- Most common weight function in Information Retrieval:
  - Weight for term *i* in document *j* (or 0 if unattested):

$$\text{weight}(i,j) = \left(1 + \log\left(TF_{i,j}\right)\right) \cdot \log\left(\frac{N}{DF_i}\right)$$

  - The IDF weighting for a unique term is maximal: log(N)
  - For a term appearing in all documents: log(1) = 0