# Similarity-based machine learning

+ application: override youtube/amazon recommedatation (do it better)

+ theory: compare two documents?

### Can we compare two documents? How?

+ d1 = “I love you”

+ d2 = “I love her”

+ d3 = “I love love you”

+ d4 = “they hate her”

+ s12 = simf(d1, d2)

+ s13 = simf(d1, d3)

+ s14 = simf(d1, d4) = 0 ???

Answer can be found in <http://nlp.stanford.edu/IR-book/>

### Yes, we can, with the set model

+ in one sentence: convert document into set; comparing two sets (jaccard index)

+ d1 ----- s1 = toSet(d1) --------

+ d2 ------- s2 = toSet(d2) --------- s12 =jaccard(s1, s2)

|  |
| --- |
| **public** **class** task1\_compare\_two\_documents  {  **public** **static** **void** main(String[] args)  {  *testString*("I love you", "I love you");  *testString*("I love you", "I love love you");    *testString*("I love you", "I love her");  *testString*("I love you", "I love her more");  *testString*("I love you", "they hate her");  }  **private** **static** **void** testString(String d1, String d2)  {  JaccardEngine eng = **new** JaccardEngine();  **double** s12 = eng.jaccardIndex(d1, d2);  System.***out***.println(d1 + " ---- " + d2 + " ---- " + s12);  }  } |
| I love you ---- I love you ---- 1.0  I love you ---- I love love you ---- 1.0  I love you ---- I love her ---- 0.5  I love you ---- I love her more ---- 0.4  I love you ---- they hate her ---- 0.0 |

### Can we reflect counts/frequences in comparision/similarity?

+ In math/concept, we expect to see  
Simf("I love you”, “I love you”) > Simf("I love you”, “I love love you”) > Simf("I love you”, “I love you I love you”)

+ In Manning’s book, we can assign each word with its count (I->1, you->2, love->5). This is call “histogram/bag of words”

+ See <https://en.wikipedia.org/wiki/Bag-of-words_model>

|  |
| --- |
| **package** tasks.task1\_comparing;  **public** **class** task2\_compare\_two\_documents\_with\_BOW  {  **public** **static** **void** main(String[] args)  {  *testString*("I love you", "I love you");  *testString*("I love you", "I love love you");  *testString*("I love you", "I love you I love you");  *testString*("I love you", "I love you I love you I love you");  }  **private** **static** **void** testString(String d1, String d2)  {  JaccardEngine eng = **new** JaccardEngineBow();  **double** s12 = eng.jaccardIndex(d1, d2);  System.***out***.println(d1 + " ---- " + d2 + " ---- " + s12);  }    } |
| I love you ---- I love you ---- 1.0  I love you ---- I love love you ---- 0.75  I love you ---- I love you I love you ---- 0.5  I love you ---- I love you I love you I love you ---- 0.3333333333333333 |

### Can we compare two movies based on what we have learned from document?

+ this is the data to work on

|  |  |  |
| --- | --- | --- |
|  | M1 | M2 |
| title | x-men first class | x-men the last stand |
| cast | lorrence | lorrence, huge jackman |
| characters | mythique, professor X, magnito | mythique, professor X, magnito |
| plot | cuba missil crisis and the men try to prevent | the return of pheunix girl and the cure for mutant |

+ s12 = similar(M1, M2) = ?

+ xmen ~ xmen -> high score  
+ xmen ~ finding dory -> low score

+ bin-to-bin matching:

|  |
| --- |
| s12 = similar(M1, M2)  = similar(M1.title, M2.title) \* w\_title  …  + similar(M1.plot, M2.plot) \* w\_plot |
| if you want to ignore title, set w\_title = 0 |

+ cross-bin matching: match title – plot, match year – title; works when we don’t have enough data   
(title: tom and jerry ~ cast:tom hank)

### Can we compare two songs based on what we have learned from document?

+ the data we work on

|  |  |  |
| --- | --- | --- |
|  | S1 | S2 |
| writer |  |  |
| lyric |  |  |
| title |  |  |
| year |  |  |

+ can we make use of cross-bin or bin-to-bin matching?

+ can we make use of repetition / patterns in song?

+ can we compare two writers? Base on their words?

# Next topics and experiments

### Topics

+ similarity based on similarity

+ similarity-based machine learning

+ word/language processing

### Collect movie data

+ 40K movies from wiki

### Write similarity function

+ write bin-to-bin similiarity function

+ write cross-bin similiarity function