Proximity between binary attributes 1/2

- A binary attribute has only two states: 0 (absence), 1 (presence)
- A contingency table for binary data

Instance i \ *Instance j*	1	0	sum
1	q	r	$q+r$
0	s	t	$s+t$
sum	$q+s$	$r+t$	p

q = the number of attributes where i was 1 and j was 1
t = the number of attributes where i was 0 and j was 0

s = the number of attributes where i was 0 and j was 1
r = the number of attributes where i was 1 and j was 0

- Simple matching coefficient

  (for symmetric binary variables)

$$d(i,j) = \frac{r+s}{q+r+s+t}$$

- for asymmetric binary variables:

$$d(i,j) = \frac{r+s}{q+r+s}$$

- Jaccard coefficient

  (for *asymmetric* binary variables)

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s}$$

Proximity between binary attributes 2/2

- Example:

Name	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	1	0	1	0	0	0
Mary	1	0	1	0	1	0
Jim	1	1	0	0	0	0

$$d(jack, mary) = \frac{0+1}{2+0+1} = 0.33$$

$$d(jack, jim) = \frac{1+1}{1+1+1} = 0.67$$

$$d(jim, mary) = \frac{1+2}{1+1+2} = 0.75$$

(from previous slide)

q = the number of attributes where i was 1 and j was 1
t = the number of attributes where i was 0 and j was 0

s = the number of attributes where i was 0 and j was 1
r = the number of attributes where i was 1 and j was 0

$$d(i, j) = \frac{r+s}{q+r+s}$$

Proximity between categorical attributes

- A nominal attribute has >2 states (generalization of a binary attribute)
  - e.g. color={red, blue, green}
- Method 1: Simple matching
  - m: # of matches, p: total # of variables

$$d(i,j) = \frac{p - m}{p}$$

Name	Hair color	Occupation
Jack	Brown	Student
Mary	Blond	Student
Jim	Brown	Architect

- Method 2: Map it to binary variables
  - create a new binary attribute for each of the *M* nominal states of the attribute

Name	Brown hair	Blond hair	IsStudent	IsArchitect
Jack	1	0	1	0
Mary	0	1	1	0
Jim	1	0	0	1

Selecting the right proximity measure

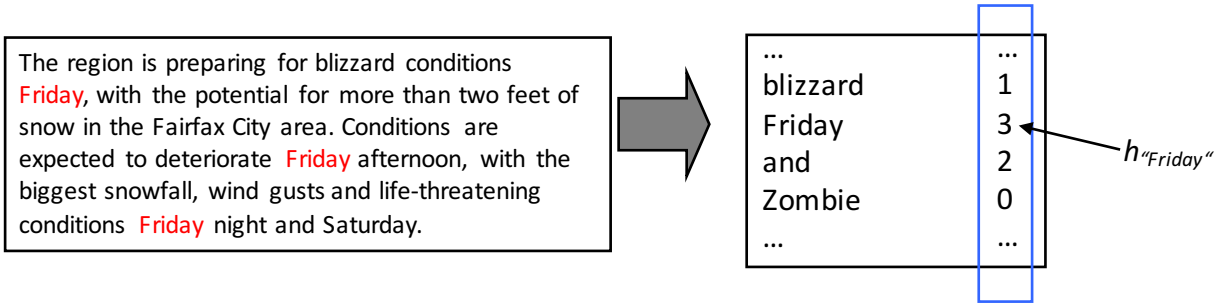
- The proximity function should fit the type of data
  - For dense continuous data, metric distance functions like Euclidean are often used.
  - For sparse data, typically measures that ignore 0-0 matches are employed
    - We care about characteristics that objects share, not about those that both lack
- Domain expertise is important, maybe there is already a state-of-the-art proximity function in a specific domain and we don't need to answer that question again.
- In general, choosing the right proximity measure can be a very time consuming task
- Other important aspects: How to combine proximities for heterogenous attributes (binary and numeric and nominal etc.)
  - e.g., using attribute weights

Outline

- Data preprocessing
- Decomposing a dataset: instances and features
- Basic data descriptors
- Feature spaces and proximity (similarity, distance) measures
- Feature transformation for text data
- Homework/ Tutorial
- Things you should know from this lecture

Feature transformations for text data 1/6

- Text represented as a set of terms ("Bag-Of-Words" model)
  - Terms:
    - Single words ("cluster", "analysis"..)
      or
    - bigrams, trigrams, …n-grams ("cluster analysis"..)
  - Transformation of a document d in a vector $r(d) = (h_1, ..., h_d)$, $h_i \geq 0$: the frequency of term t_i in d



Feature transformations for text data 2/6

- Challenges/Problems in Text Mining:
  1. Common words ("e.g.", "the", "and", "for", "me")
  2. Words with the same root ("fish", "fisher", "fishing",…)
  3. Very high-dimensional space (dimensionality $d > 10.000$)
  4. Not all terms are equally important
  5. Most term frequencies $h_i = 0$ ("sparse feature space")
- More challenges due to language:
  - Different words have same meaning (synonyms)
    - "freedom" – "liberty"
  - Words have more than one meanings
    - e.g. "java", "mouse"

Feature transformations for text data 3/6

- Problem 1: Common words (“e.g.”, “the”, “and”, “for”, “me”)
  - Solution: ignore these terms (Stopwords)

    There are stopwords list for all languages in WWW.

- Problem 2: Words with the same root (“fish”, “fisher”, “fishing”,…)
  - Solution: Stemming

    Map the words to their root

    - "fishing", "fished", "fish", and "fisher" to the root word, "fish".

    For English, the Porter stemmer is widely used.
    (Porters Stemming Algorithms: http://tartarus.org/~martin/PorterStemmer/index.html)

    Stemming solutions exist for other languages also.

    The root of the words is the output of stemming.

Feature transformations for text data 4/6

- Problem 3: Too many features/ terms
  - Solution: Select the most important features ("Feature Selection")
  - Example: average document frequency for a term
    - Very frequent items appear in almost all documents
    - Very rare terms appear in only a few documents

Ranking procedure:

1. Compute document frequency for all terms t_i :
2. Sort terms w.r.t. $DF(t_i)$ and get $rank(t_i)$
3. Sort terms by score(t_i)= $DF(t_i) \cdot$ rank(t_i)
   e.g. score(t_{23}) = $0.82 \cdot 1 = 0.82$
   score(t_{17}) = $0.65 \cdot 2 = 1.3$
4. Select the k terms with the largest score(t_i)

$$DF(t_i) = \frac{\#Docs\ containing\ t_i}{\#All\ documents}$$

Rank	Term	DF
1.	t_{23}	0.82
2.	t_{17}	0.65
3.	t_{14}	0.52
4.	…	…

Feature transformations for text data 5/6

- Problem 4: Not all terms are equally important
  - Idea: Very frequent terms are less informative than less frequent words. Define such a term weighting schema.
  - Solution: TF-IDF (Term Frequency · Inverse Document Frequency)

    Consider both the importance of the term in the document and in the whole collection of documents.

$$TF(t,d) = \frac{n(t,d)}{\sum_{w \ d} n(w,d)}$$

The relative frequency of term t in d [n(t,d) = # t in d]

$$IDF(t) = \log\left(\frac{|DB|}{|\{d \mid d \in DB \wedge t \in d\}|}\right)$$

Inverse frequency of term t in all DB

$$TF \cdot IDF = TF(t,d)\,IDF(t)$$

Feature vector with TF IDF : $r(d) = (TF(t_1,d) \cdot IDF(t_1), \ldots, TF(t_n,d) \cdot IDF(t_n))$

Feature transformations for text data 6/6

- Problem 5: for most of the terms $h_i = 0$
  - Euclidean distance is not a good idea: it is influenced by vectors lengths
  - Idea: use more appropriate distance measures

Jaccard Coefficient: Ignore terms absent in both documents

$$d_{Jaccard}(d_1, d_2) = 1 - \frac{|d_1 \cap d_2|}{|d_1 \cup d_2|} = \frac{|\{t | t \in d_1 \wedge t \in d_2\}|}{|\{t | t \in d_1 \vee t \in d_2\}|}$$

Cosine Coefficient: Consider term values (e.g. TFIDF values)

$$d_{cosinus}(d_1, d_2) = 1 \quad \frac{\langle d_1, d_2 \rangle}{\|d_1\| \ \|d_2\|} = 1 \quad \frac{\sum_{i=0}^{n} \left(d_{1,i} \ d_{2,i}\right)}{\sqrt{\sum_{i=0}^{n} d_{1,i}^{2}} \ \sqrt{\sum_{i=0}^{n} d_{2,i}^{2}}}$$