CSE 5334 Final Preparation

* Information Retrival Systems

- Word : a unit separated by whitespace or punctuation
- Term : a meaningful unit
- Token : a unit after tokenization
- Type : a disjonset of tokens from documents

Term ≒ Type ⊂ Word
Token

* Normalization

* Tokenization

- numbers
- stopwords
- Type construction : ① Lemmatization ② Stemming

ex. Porter, Lovins, Paice

A few rules of Porter Stemmer

Rule			Example		
sses	→	ss	caresses	→	caress
ies	→	i	ponies	→	poni
ss	→	ss	caress	→	caress
s	→		cats	→	cat

* Ranked Retrieval

① Jaccard Coefficient : from non-empty sets A, B

$$Jaccard(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

due to the Set feature, 1) X considers the freq. of terms
2) X weights the important words.
3) X implies the length of documents

② tf-matching-score

- Term Frequency tf = # of terms in documents
  - log weighted : $\begin{cases} 1 + \log_{10}(tf) & : tf > 0 \\ 0 & : \text{otherwise} \end{cases}$

⇒ $\sum 1 + \log_{10} tf$

③ TF-IDF Matrix & Cosine Similarity

- Inversed Document Frequency $idf = \log_{10} \frac{N}{df_t}$
  - N : # of documents
  - df_t : # of documents which contains term

- TF-IDF Matrix

term \ doc	d_1	d_2 ⋯
t_1	v_{11}	v_{21}
t_2	v_{12}	v_{22}

⇒ $d_1 = [v_{11}, v_{12}]$
$d_2 = [v_{21}, v_{22}]$

- Cosine Similarity : $\frac{\vec{q} \cdot \vec{d}}{\|\vec{q}\| \times \|\vec{d}\|}$

ex. lnc.ltn → lnc for doc. ltn for query

Table 5. Components of tf-idf

Term Frequency tf		Document Frequency df		Normalization	
n (natural)	$tf_{t,d}$	n (no)	1	n (none)	1
l (logarithmic)	$1 + \log(tf_{t,d})$	t (idf)	$\log \frac{N}{df_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2 + w_2^2 + \cdots + w_M^2}}$
a (augmented)	$0.5 + 0.5 \times \frac{tf_{t,d}}{\max_{t' \in d} tf_{t',d}}$	p (prob idf)	$max(0, \log \frac{N - df_t}{df_t})$	u (pivoted unique)	$\frac{1}{u}$
b (boolean)	1, if $tf_{t,d} > 0$, otherwise 0			b (byte size)	$\frac{1}{\text{CharLength}^\alpha}, \alpha < 1$
L (log average)	$\frac{1 + \log tf_{t,d}}{1 + \log(ave_{t' \in d} tf_{t',d})}$				

* Data Mining

- 4Vs : Volume, Velocity, Variety, Veracity
  - Veracity = quality
- Structed / Semi-Structed / Unstructed data
- Nominal — eye color, student ID
  Ordinal — student grades, satisfaction level, height
  Interval — Celsius & Fahrenheit Temperature, dates on calendar
  Ratio — Kelvin Temperature

* Nom. ⊂ Ord. ⊂ Int. ⊂ Rat.

* Data Preprocessing

- aggregation, sampling, dimensionality reduction, feature subset selection, feature creation, discretization and binarization, attribute transformation
  - random sampling
  - w/o replacement sampling
  - w/ replacement sampling
  - Stratified sampling

* Similarity and Distance

Attribute	Similarity	Dissimilarity
Nominal	$s = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q \end{cases}$	$d = \begin{cases} 0 & \text{if } p = q \\ 1 & \text{if } p \neq q \end{cases}$
Ordinal	$s = 1 - \frac{\lvert p-q \rvert}{n-1}$ $(s = 1 - d)$ (where n is the number of values, mapping the values to integers 0 to n-1)	$d = \frac{\lvert p-q \rvert}{n-1}$
Interval or Ratio	$s = -d, s = \frac{1}{1+d}$	$d = \lvert p - q \rvert$

- Distance

$$\text{dist} = \left(\sum_{i=1}^{n} |p_i - q_i|^r \right)^{1/r} \begin{cases} r = 1, & \text{Manhattan distance, L1 norm} \\ r = 2, & \text{Euclidean distance, L2 norm} \\ r = \infty, & \text{Supremum (Chebyshev) distance, } L_{max} \text{ norm, } L_\infty \text{ norm} \end{cases}$$

- Similarity

$$M_{pq} = \begin{cases} M_{01} & p = 0, q = 1 \\ M_{10} & p = 1, q = 0 \\ M_{11} & p = 1, q = 1 \\ M_{00} & p = 0, q = 0 \end{cases}$$

(The number of attributes for each case p and q)

	SMC	Jaccard Coefficient
	$\frac{M_{11}+M_{00}}{M_{11}+M_{00}+M_{10}+M_{01}}$	$\frac{M_{11}}{M_{11}+M_{10}+M_{01}}$
ex. $p = 1000000000, q = 0000001001$	$\frac{0+7}{0+7+1+2} = \frac{7}{10}$	$\frac{0}{0+1+2} = 0$

- if M_{00} has an important meaning → SMC :)
- if M_{00} is meaningless → Jaccard :)

$$\text{Pearson Correlation Coefficient} = \frac{\text{covariance}_{x,y}}{\sigma_x \sigma_y} = \frac{\sum_{i=1}^{n}(p_i - \bar{p})(q_i - \bar{q})}{\sqrt{\sum_{i=1}^{n}(p_i - \bar{p})^2}\sqrt{\sum_{i=1}^{n}(q_i - \bar{q})^2}}$$

$\bar{p} = \frac{1}{n}\sum p_i$ $\bar{q} = \frac{1}{n}\sum q_i$

- Overall Similarity (w/ many types of attributes)

1. Define an indicator variable σ_k for the k-th attribute as follows:

$$\sigma_k = \begin{cases} 0 & \text{(if the } k^{\text{th}} \text{ attribute is a binary asymmetric attribute AND both objects have a 0 value)} \\ & \text{or (if one of the object has a missing value for the } k^{\text{th}} \text{ attribute)} \\ 1 & \text{otherwise} \end{cases}$$

2. With the similarity of the k^{th} attribute s_k, the overall similarity between two objects is calculated as:

$$s(p,q) = \frac{\sum_{k=1}^{n} s_k \cdot \sigma_k}{\sum_{k=1}^{n} \sigma_k}$$

* Decision Tree

- Hunts

```
Start from a tree with a single root node containing all the training data.

Recursive:
    1. Check if the node is homogeneous (pure).
        - If true, make the node a leaf node and label it with the class. END the branch.
        - If not, continue to the next step.
    2. Check if the node is empty.
        - If true, make the node a leaf node. END the branch.
        - If not, continue to the next step.
    3. Check whether the node has conflicting data, a same label with different values.
        - If true, mark the node as a leaf node. END the branch.
        - If not, continue to the next step.
    4. Split the node into child nodes based on the attribute.
        - During the split, the algorithm calculates the impurity of the child nodes
            using the 1) Gini index, 2) entropy, or 3) misclassification error.
        - After splitting, the Gain is recalculated to renew the tree state.

Terminate:
    1. If all nodes become leaf nodes during the recursive process.
    2. If the split does not show certain improvement than beforehanded-set threshold, regarding the impurity
```

- Impurity Metric

① Misclassification Error : $1 - \max(p_i)$
  - $\max(p_i)$ = the most correct value

ex. 3:3:3 → $\frac{3}{9}$ ⇒ M.E. = $\frac{6}{9}$
2:3:5 → $\frac{5}{10}$ ⇒ M.E. = $\frac{5}{10}$

② $\text{Gini} = 1 - \sum_{i=1}^{c} p_i^2$ $\begin{cases} c = \text{the number of classes} \\ p_i = \text{the probability of selecting an item of class } i \text{ in the node} \end{cases}$

cf. minimum = 0 maximum = $1 - \frac{1}{c}$

$$\text{Gain} = \text{Gini(parent)} - \sum_{\text{child} \in \text{children}} \frac{\text{the number of data in the child}}{\text{the number of data in the parent}} \times \text{Gini(child)}$$

(2)

$$\text{Entropy}(t) = -\sum_j p(j|t)\log_2 p(j|t)$$

- Minimum : 0
- Maximum : $\log_2 C$

Information Gain

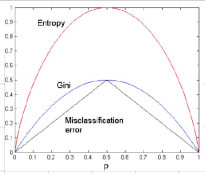
$$\text{Gain}_{\text{split}} = \text{Entropy}(\text{parent}) - \sum_{\text{child}\in\text{children}} \frac{\text{the number of data in the child}}{\text{the number of data in the parent}} \times \text{Entropy}(\text{child})$$

Split Info

$$\text{Split Info} = -\sum_{\text{child}\in\text{children}} \frac{\text{the number of data in the child}}{\text{the number of data in the parent}} \times \log_2 \frac{\text{the number of data in the child}}{\text{the number of data in the parent}}$$

Gain Ratio

$$\text{Gain Ratio} = \frac{\text{Gain}_{\text{split}}}{\text{Split Info}}$$



- Entropy is able to represent the maximum impurity and its derivatives along with prob. among metrics

- Pre-pruning : ① Stop when the tree reaches a certain depth
② Stop when the # of instances in a node is less than a certain threshold
③ Stop if all instances in a node belong to the same class.
④ Stop if the split does not improve impurity like Gini, Information Gain

- Post-pruning : 1) Pessimistic Error
2) Optimistic Error
3) Reduced Error
└ with validation data, prune or not.

optimistic: Training Error (before splitting) $= \frac{10}{30}$ (# of Nodes)

Pessimistic Error (before splitting) $= \frac{10+(0.5\times 1)}{30}$

optimistic: Training Error (after splitting) $= \frac{4+3+1+1}{30} = \frac{9}{30}$

Pessimistic Error (after splitting) $= \frac{9+(0.5\times 4)}{30}$

* Naive Bayes Classifier

Naive Bayes Classifier : $P(C \mid A_1, A_2, \ldots, A_n)$

$$\begin{cases} P(C \mid A_1, A_2, \ldots, A_n) = \frac{P(A_1, A_2, \ldots, A_n \mid C)P(C)}{P(A_1, A_2, \ldots, A_n)} \\ P(A_1, A_2, \ldots, A_n \mid C) = P(A_1 \mid C)P(A_2 \mid C)\ldots P(A_n \mid C) \end{cases}$$

$$P(C_j) = \frac{\text{Number of data in class } C_j}{\text{Total number of data}}$$

e.g. $P(\text{No}) = \frac{7}{10} = 0.7, P(\text{Yes}) = \frac{3}{10} = 0.3$

Discrete Attributes: $P(A_i \mid C_k) = \frac{\text{Number of instance having attribute } A_i}{\text{Number of Class } C_k}$

e.g. $P(\text{Status=Married} \mid \text{No}) = \frac{4}{7}$

$P(\text{Refund=Yes} \mid \text{Yes}) = 0$

Continuous Attributes: $P(A_i \mid C_k) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

e.g. $P(\text{Income=120} \mid \text{No}) = \frac{1}{\sqrt{2\pi(54.54)}} e^{-\frac{(120-110)^2}{2(2975)}} = 0.0072$

$\sigma = \frac{1}{n-1}\sum_{i=1}^{n}(x_i-\mu)^2$

$X = (\text{Refund=No, Status=Married, Income=120K})$

$P(X \mid \text{Class=No}) = P(\text{Refund=No} \mid \text{Class=No}) \times P(\text{Status=Married} \mid \text{Class=No}) \times P(\text{Income=120K} \mid \text{Class=No}) = \frac{4}{7}\cdot\frac{4}{7}\cdot 0.0072 = 0.0024$

$P(X \mid \text{Class=Yes}) = P(\text{Refund=No} \mid \text{Class=Yes}) \times P(\text{Status=Married} \mid \text{Class=Yes}) \times P(\text{Income=120K} \mid \text{Class=Yes}) = 1\cdot 0\cdot(1.2\times 10^{-9}) = 0$

$P(X \mid \text{Class=No})\cdot P(\text{Class=No}) = 0.0024\cdot 0.7 = 0.0017$

$P(X \mid \text{Class=Yes})\cdot P(\text{Class=Yes}) = 0\cdot 0.3 = 0$

- Prevent P=0 (only for Discrete Attributes)

Laplace	$P(A_i \mid C) = \frac{\text{Number of instance having attribute } A_i + 1}{\text{Number of Class } C_k + N}$
m-estimate	$P(A_i \mid C) = \frac{\text{Number of instance having attribute } A_i + m\cdot P(A_i)}{\text{Number of Class } C_k + m}$

N : # of classes
m : Smoothing Parameters

* KNN

- to avoid ties — 1) add K value
2) weight factor $w = \frac{1}{\text{distance}^2}$

- when K↓ → sensitive to noise
K↑ → acc. decrease.

* PEBLS

- Distance $(V_1, V_2) = \sum_{i\in C}\left|\frac{n_{1i}}{n_1} - \frac{n_{2i}}{n_2}\right|$ ⇒

class	Yes	No

Distance(Single, Married) $= \left|\frac{2}{4}-\frac{0}{4}\right| + \left|\frac{2}{4}-\frac{4}{4}\right| = 1$

Distance(Single, Divorced) $= \left|\frac{2}{4}-\frac{1}{2}\right| + \left|\frac{2}{4}-\frac{1}{2}\right| = 0$

Distance(Married, Divorced) $= \left|\frac{0}{4}-\frac{1}{2}\right| + \left|\frac{4}{4}-\frac{1}{2}\right| = 1$

Distance(Refund=Yes, Refund=No) $= \left|\frac{0}{3}-\frac{3}{7}\right| + \left|\frac{3}{3}-\frac{4}{7}\right| = \frac{6}{7}$

- Weighting Method

$$\delta(\mathbf{X}, \mathbf{Y}) = W_X W_Y \sum_{i=1}^{d} \text{Distance}(X_i, Y_i)^2$$

Where $W_X = \frac{\text{The number of times that X is used for prediction}}{\text{The number of times that X is used to predict correctly}}$

* SVM

- when X_i, Y_i are given, Find $Y_i(w^T X_i + b) \geq 1$
- non-linearly separable data : slack variable
non-linear decision boundary : Kernel trick

* Evaluation

- Underfitting : train & test Err.↑ ⊕ a model is too simple
Overfitting : train ↓ test ↑ (Err.) ⊕ a model is too complex
- Occam's Razor → MDL $L(M, D) = L(M) + L(D|M)$ $\begin{cases} L(M) & \text{is the length of the model} \\ L(D|M) & \text{is the length of the data given the model} \end{cases}$

- Missing Values

Train
1) when computing Impurity
- exclude in counting child nodes
- include in counting parent node
2) when distributing instances
- (3 T, 1 F)
(0 T, 3 F) (2 T, 4 F) ⇒ (3/9 T, 3 F) (6/9 T, 4 F)

Test - 3) From trained tree, follow the most # of value in missing attribute

* Metrics

- Confusion Matrix

Actual \ Pred.	P	N
P	TP	FN
N	FP	TN

↓ precision → recall ⇒ f: $\frac{2rp}{r+p}$

- Roc Curve

$TPR = \frac{TP}{TP+FN}$ $FPR = \frac{FP}{FP+TN}$

Class	+	-	+	-	-	+	-	+	+	
>=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	1	0	
FP	5	5	4	4	3	2	1	1	0	0	
TN	0	0	1	1	2	3	4	4	5	5	
FN	0	1	1	2	2	2	2	3	4	5	
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.2	0	
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	

draw the ROC curve by moving $\frac{1}{\#\text{ of positive labeled samples}}$ or $\frac{1}{\#\text{ of negative labeled samples}}$ to the bottom or left for each threshold respectively on the each point of the ROC curve from the $(1,1)$ data point with the lowest threshold

- Confidence Interval

Model 1 in dataset A (size n_1, error e_1) ⇒ $d = e_1 - e_2$, $\sigma^2 = \sum \frac{e_i(1-e_i)}{n_i}$

" 2 " B (" n_2, " e_2) $d_t = d \pm z_{\alpha/2}\times\sigma$

* Sampling & Validation Techniques

① Holdout ② Random Subsampling (K Holdout) ③ Cross Valid. (k=N : Leave-one-out)
④ Stratified Sampling ⑤ Bootstrap

* Clustering

- Goal : Minimize Intra-Cluster distance
Maximize Inter-Cluster distance

- Partitional Clustering
$P = K!/K^K$
- Bisecting k-means : Divide the ↑SSE cluster into 2
- $SSE = \sum_{i=1}^{K}\sum_{x\in C_i} \text{dist}^2(m_i, x)$

- Hierarchical Clustering
· Agglomerative
· Divisive — Build MST → ↑Dist. 부터 갈라 clusters들!

* Association Rule Mining

Rule: {Milk, Diaper} → {Beer}

$$\begin{cases} \text{Support } s = \frac{\sigma(\{\text{Milk, Diaper, Beer}\})}{\text{Total number of transaction}} \\ \text{Confidence } c = \frac{\sigma(\{\text{Milk, Diaper, Beer}\})}{\sigma(\{\text{Milk, Diaper}\})} \end{cases}$$

- # of Possible Association Rule : $3^n - 2^{n+1} + 1$
- # of " Itemsets : 2^n
- Apriori Algorithm
1) Frequent Itemset Generation ← (Apriori Principle)

① 1-itemsets

item	support
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

2-itemsets

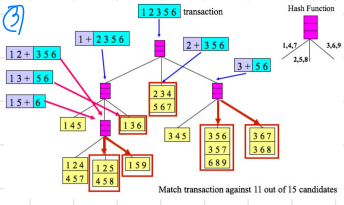
item	support
Bread, Milk	3
Bread, Beer	2
Bread, Diaper	3
Milk, Beer	2
Milk, Diaper	3
Beer, Diaper	3

3-itemsets

item	support
Bread, Milk, Diaper	3
Bread, Diaper, Beer	3
Milk, Diaper, Beer	3

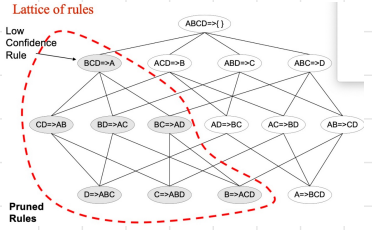
② Build Hash Tree from ①

③



2) Rule Generation

Confidence$(A, B, C \to D) \geq$ Confidence$(A, B \to C, D)$
$\geq$ Confidence$(A \to B, C, D)$



To solve the problem, **1)** multiple runs of the algorithm might be considered, but it is not efficient and does not guarantee the best solution with a very small probability as well. **2)** Hierarchical clustering to find initial centroids and **3)** selecting more than K initial centroids and then selecting the most widely separated among these initial centroids are another solutions. **4)** Postprocessing such as reassigning outliers or splitting and merging clusters, and **5)** bisecting K-means are also the solutions.

Handling empty clusters is another issue in the basic K-means algorithm. This could be solved by **1)** choosing the point that has the maximum distance from the centroid of the cluster with the maximum SSE and reassigning the point to the empty cluster. **2)** Repeating the algorithm is also the solution for this issue too. **3)** When it comes to running the K-means algorithm, rather than calculating the distance between all data points and all centroids after they all belong to clusters, updating the centroids after each data point is assigned to a cluster converges faster and never gets an empty cluster, but it is more expensive and introduces an order dependency. The order dependency means that the result could be different depending on which point is calculated first.

K-means algorithm itself also has limitations when the clusters of the data have different sizes, densities, or shapes. Firstly, the algorithm assumes that all clusters have the same size. However, the radius of the original clusters might be different, causing the non-optimal clusters. Secondly, the algorithm presumes that all clusters have the same density. The problem is derived from the same radius among all clusters likewise the first limitation. This leads clusters to have two dense clusters or splitted one cluster comparing from the original clusters. Lastly, the algorithm is not able to find non-globular shapes such as star-shaped clusters.