一、概念题（共55分）
1. Precision, recall (4pt) (chap 1)
2. Maximum likelihood estimate (2pt) (chap 11, chap 13)
3. Maximum a posteriori estimate (2pt) (chap 11, chap 13)
4. Second price auction (2pt) (chap 19)
   The advertiser pays the minimum amount necessary to maintain their position in the auction (plus 1 cent).
5. Jaccard coefficient (2pt) (chap 3,19)
   \[
   \text{JACCARD}(A, B) = \frac{|A \cap B|}{|A \cup B|}
   \]
   \[
   \text{JACCARD}(A, A) = 1
   \]
   \[
   \text{JACCARD}(A, B) = 0 \text{ if } A \cap B = 0
   \]
6. K nearest neighbor classification (2pt) (chap 14)
   Assign each test document to the majority class of its \( k \) nearest neighbors in the training set.
7. Linear classifier (3pt) (chap 14)
   a two-class classifier that decides class membership by comparing a linear combination of
   the features to a threshold.
   \[
   \text{A linear classifier computes a linear combination or weighted sum } \sum_i w_i x_i \text{ of the feature values.}
   \]
   \[
   \text{Classification decision: } \sum_i w_i x_i > \theta?
   \]
   \[
   \ldots \text{where } \theta \text{ (the threshold) is a parameter.}
   \]
8. Standing query (2pt) (chap 13)
   A standing query is like any other query except that it is periodically executed on a collection
   to which new documents are incrementally added over time.
9. Add-one / Laplace smoothing (2pt) (chap 13)
   To eliminate zeros, we use add-one or Laplace smoothing, which simply adds one to each count
\[ \hat{P}(t|c) = \frac{T_{ct} + 1}{\sum_{t' \in V} (T_{ct'} + 1)} = \frac{T_{ct} + 1}{(\sum_{t' \in V} T_{ct'}) + B'} \]

where \( B = |V| \) is the number of terms in the vocabulary. Add-one smoothing can be interpreted as a uniform prior (each term occurs once for each class) that is then updated as evidence from the training data comes in.

10. Microaveraging (2pt) (chap 13)

Microaveraging

a) Compute TP, FP, FN for each of the \( C \) classes
b) Sum these \( C \) numbers (e.g., all TP to get aggregate TP)
c) Compute \( F_1 \) for aggregate TP, FP, FN

11. Query likelihood model, multinomial model (4pt) (chap 12)

The query likelihood model is a language model used in Information Retrieval. A language model is constructed for each document in the collection. It is then possible to rank each document by the probability of specific documents given a query. This is interpreted as being the likelihood of a document being relevant given a query.

The multinomial unigram language model is commonly used to achieve this. We have:

\[ P(q|M_d) = K_q \prod_{t \in V} P(t|M_d)^{tf_{t,d}} \]

where the multinomial coefficient is

\[ K_q = \frac{L_d^{|q|}}{t.f_{t1,d}!t.f_{t2,d}!...t.f_{tM,d}!} \]

for query \( q \). Here, \( L_d \) is the length of document \( d \), \( M \) is the size of the term vocabulary.


If the retrieved documents (w.r.t a query) are ranked decreasingly on their probability of relevance, then the effectiveness of the system will be the best that is obtainable.


- ‘Binary’ (equivalent to Boolean): documents and queries represented as binary term incidence vectors
  - E.g., document \( d \) represented by vector \( x = (x_1, \ldots, x_M) \), where \( x_t = 1 \) if term \( t \) occurs in \( d \) and \( x_t = 0 \) otherwise
  - Different documents may have the same vector representation
- ‘Independence’: no association between terms (not true, but practically works - ‘naive’ assumption of Naive Bayes models)

14. Structural term (2pt) (chap 10)

Index all paths that end in a single vocabulary term, in other words all XML-context term pairs. We call such an XML-context term pair a structural term and denote it by \(<c, t>\): a
pair of XML-context \( c \) and vocabulary term \( t \).

15. Marginal relevance (2pt) (chap 8)
   whether a document still has distinctive usefulness after the user has looked at certain other documents.

16. R-precision(2pt) (chap 8)
   It requires having a set of known relevant documents \( \text{Rel} \), from which we calculate the precision of the top \( \text{Rel} \) documents returned.
   If there are \( |\text{Rel}| \) relevant documents for a query, we examine the top \( |\text{Rel}| \) results of a system, and find that \( r \) are relevant, then by definition, not only is the precision (and hence R-precision) \( r/|\text{Rel}| \), but the recall of this result is also \( r/|\text{Rel}| \).

17. Document-at-a-time (2pt) (chap 7)
   We order the documents consistently by some common ordering: typically by document ID or by static quality scores. Such a common ordering supports the concurrent traversal of all of the query terms’ postings lists, computing the score for each document as we encounter it. Computing scores in this manner is sometimes referred to as document-at-a-time scoring.

18. Tiered indexes(2pt) (chap 7)
   19. Pivot normalization (2pt) (chap 6)
      Cosine normalization produces weights that are too large for short documents and too small for long documents (on average).
      Adjust cosine normalization by linear adjustment: “turning” the average normalization on the pivot
      Effect: Similarities of short documents with query decrease; similarities of long documents with query increase.
      This removes the unfair advantage that short documents have.

20. Heap's law (2pt) (chap 5)
   Heaps’ law is linear in log-log space.
   a) It is the simplest possible relationship between collection size and vocabulary size in log-log space.
   b) Empirical law
   Heaps’ law: \( M = kT^b \)
   \( M \) is the size of the vocabulary, \( T \) is the number of tokens in the collection.
   Typical values for the parameters \( k \) and \( b \) are: \( 30 \leq k \leq 100 \) and \( b \approx 0.5 \).

   We also want to know how many frequent vs. infrequent terms we should expect in a collection.
Zipf’s law: The $i^{th}$ most frequent term has frequency $c_i$ proportional to $1/i$.

   SPIMI uses terms instead of termIDs, writes each block's dictionary to disk, and then starts a new dictionary for the next block.

23. Permuterm index(2pt) (chap 3)
   Basic idea: Rotate every wildcard query, so that the * occurs at the end.
   Store each of these rotations in the dictionary, say, in a B-tree

24. Stemming, lemmatization(4pt) (chap 2)

二、简答题（共15分）
1. Q: Mutual information measures how much information the presence / absence of a term contributes to making the correct classification decision on $c$. Formally:

$$I(U; C) = \sum_{e_t \in \{1, 0\}} \sum_{e_c \in \{1, 0\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)}$$

The “export”/POULTRY contingency table is as follows:

<table>
<thead>
<tr>
<th></th>
<th>$e_c = e_{poultry} = 1$</th>
<th>$e_c = e_{poultry} = 0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_t = e_{export} = 1$</td>
<td>$N_{11} = 49$</td>
<td>$N_{10} = 27,652$</td>
</tr>
<tr>
<td>$e_t = e_{export} = 0$</td>
<td>$N_{01} = 141$</td>
<td>$N_{00} = 774,106$</td>
</tr>
</tbody>
</table>

Based on maximum likelihood estimates, please get the value of $I(\text{export}; \text{POULTRY})$. (5pt) (chap 13)

a:

$$I(U; C) = \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{N_1N_1} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{N_0N_1} + \frac{N_{10}}{N} \log_2 \frac{NN_{10}}{N_1N_0} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{N_0N_0}$$

$$I(U; C) = \frac{49}{801,948} \log_2 \frac{801,948 \cdot 49}{(49 + 27,652)(49 + 141)} + \frac{141}{801,948} \log_2 \frac{801,948 \cdot 141}{(141 + 774,106)(49 + 141)} + \frac{27,652}{801,948} \log_2 \frac{801,948 \cdot 27,652}{(49 + 27,652)(27,652 + 774,106)} + \frac{774,106}{801,948} \log_2 \frac{801,948 \cdot 774,106}{(141 + 774,106)(27,652 + 774,106)}$$

$$\approx 0.0001105$$

2. Q: Suppose we have a collection that consists of the 4 documents given in the below
table.

<table>
<thead>
<tr>
<th>docID</th>
<th>Document text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>click go the shears boys click click click</td>
</tr>
<tr>
<td>2</td>
<td>click click</td>
</tr>
<tr>
<td>3</td>
<td>metal here</td>
</tr>
<tr>
<td>4</td>
<td>metal shears click here</td>
</tr>
</tbody>
</table>

Build a query likelihood language model for this document collection. Assume a mixture model between the documents and the collection, with both weighted at 0.5. Maximum likelihood estimation (MLE) is used to estimate both as unigram models. Work out the model probabilities of the query **click shears** for each document, and use those probabilities to rank the documents. Fill in these probabilities in the below table:

<table>
<thead>
<tr>
<th>Query</th>
<th>Doc 1</th>
<th>Doc 2</th>
<th>Doc 3</th>
<th>Doc 4</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>click shears</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

What is the final ranking of the documents for the query click shears? (5pt) (chap 12)

3. Q: Give three reasons why relevance feedback has been little used in web search. (5pt) (chap 9)
   a:  i. RF slows down returning results as you need to run two sequential queries, the second of which is slower to compute than the first. Web users hate to be kept waiting.
        ii. RF is mainly used to increase recall, but web users are mainly concerned about the precision of the top few results.
        iii. RF is one way of dealing with alternate ways to express an idea (synonymy), but indexing anchor text is commonly already a good way to solve this problem.
        iv. RF is an HCI failure: it’s too complicated for the great unwashed to understand.