cs276a PS2 Review

1. Relevance Feedback
    a. In addition to the query, the user tells which of the documents returned are relevant. How can we use this new information?
    b. Example
        i. Query "fishing equipment"
        ii. Every relevant document contains the word "boat"
        iii. Then "fishing equipment boat" would be a better query
    c. In the example above, formally, how do we get "boat" into query?
        i. Original query vector q_0: q_0["fishing"]=1, q_0["equipment"]=1, all other q_0 components =0
        ii. Let d be one of the relevant documents, then d is a vector, and d["boat"]=1 (ignore other components of d for now)
        iii. $q_m=q_0+d$ and we get q_m["boat"]=1
            1. But d has lots of words, now they are all in q_m, we've completely drawn out the original query. That is why we need to scale down the contribution from d with β, eg $q_m=q_0+\beta d$
            2. But there are several relevant documents $d_1,\ldots,d_n$, not just one. So average them first:

$$q_m = q_0 + \beta d_{avg} = q_0 + \beta \frac{d_1 + d_2 + \cdots + d_n}{\text{\# of relevant documents}}$$

    d. What about non-relevant documents?
        i. Suppose every non-relevant document contains "equipment"
        ii. Now we want to downplay "equipment" in the query
        iii. Same as before, but now we subtract, if d non-relevant, then d["equipment"]=1, then for $q_m=q_0-d$, q_m["equipment"]=0. Success.
        iv. Then average and scale just like for relevant documents.
    e. The Formula (Rocchio Algorithm)
        i. Now we are ready for the full formula, call the set of relevant documents C_r, non-relevant C_{nr}; add a scale factor α for q_0 to be

able to control tradeoff between the original query and the relevant/non-relevant docs, and we get

$$q_m = \alpha q_0 + \beta\left(\frac{1}{|C_r|}\sum_{d_r \in C_r} d_r\right) - \gamma\left(\frac{1}{|C_{nr}|}\sum_{d_{nr} \in C_{nr}} d_{nr}\right)$$

2. Probabilities, Language Models and Naïve Bayes
    a. Example
        i. Have document with 1,000 words, D=$w_1,\ldots,w_{1,000}$
        ii. P(D)=P($w_1,\ldots w_{1,000}$)=P(w_1)*P($w_2|w_1$)*P($w_3|w_1,w_2$)*…*P($w_{1,000,}$ $|w_1,w_2,\ldots w_{999}$)
            1. The **<u>Chain Rule of Probability</u>**
            2. **Always** true, like 2=2, based on the axioms of probability
        iii. Consider P($w_{1,000,}$ $|w_1,w_2,\ldots w_{999}$), lets say $w_{1,000}$="toys"
            1. P("toys"|w_1="A", $w_2,\ldots,w_{999}$) vs P("toys"|w_1="Today",$w_2,\ldots,w_{999}$)
            2. Would not expect them to be very different
            3. Thus P($w_{1,000,}$ $|w_1,w_2,\ldots w_{999}$)≈P($w_{1,000,}$ $|$ $w_2,\ldots w_{999}$)
        iv. How about P($w_{1,000,}$ $|w_{\mathbf{1}},\ldots w_{999}$)≈P($w_{1,000,}$ $|$ $w_{\mathbf{3}},\ldots w_{999}$)? P($w_{1,000,}$ $|w_{\mathbf{1}},\ldots w_{999}$)≈P($w_{1,000,}$ $|$ $w_{\mathbf{100}},\ldots w_{999}$)?
        v. Why not P($w_{1,000,}$ $|w_1,\ldots w_{999}$)≈P($w_{1,000}$)?
            1. Why not indeed, this is precisely the Naïve Bayes assumption, and it can work pretty well.
            2. Note that almost certainly P($w_{1,000,}$ $|w_1,\ldots w_{999}$)≠P($w_{1,000}$), but if it is close, we'll be ok.
    b. Language Model
        i. Unigram: P($w_{n,}$ $|w_1,\ldots w_{n-1}$)≈P(w_n)=$\dfrac{\#\text{ of times } w_n \text{ occurs}}{\text{total }\#\text{ of words}}$
            1. makes the Naïve Bayes assumption
        ii. Bigram: P($w_{n,}$ $|w_1,\ldots w_{n-1}$)≈P($w_n|w_{n-1}$)= $\dfrac{\#\text{ of times } w_n \text{ follows } w_{n-1}}{\#\text{ of times } w_{n-1} \text{ occurs}}$
            1. According to a bigram model, and the chain rule of probability, P(w_1,w_2,w_3)=…?

c. Naïve Bayes classification

   i. Before wanted to know $P(D)=P(w_1,\ldots w_{1,000})$

   ii. Now want to know $P(D|c)=P(w_1,\ldots w_{1,000}|c)$, where c is some class of documents

      1. eg c="su.class.cs276a", if D is in this newsgroup how likely of a document is it (for this newsgroup)? A somewhat weird quantity, but hang on.

      2. Or could think about it generationally: if I was randomly writing a document based on other documents I've seen in c, with what probability would I write D?

   iii. Structurally identical to what we did above for language models, still comes apart via the chain rule

      1. $P(D|c)=P(w_1,\ldots w_{1,000}|c)=P(w_1|c)*P(w_2|w_1,c)*P(w_3|w_1,w_2,c)*\ldots*P(w_{1,000,}|w_1,w_2,\ldots w_{999},c)$

      2. Apply the Naïve Bayes assumption, then for any $w_{k,}$ $P(w_k|w_1,w_2,\ldots w_{k-1},c)=P(w_k|c)$, and we have $P(w_1,\ldots w_{1,000}|c)=P(w_1|c)*P(w_2|c)*\ldots*P(w_{1,000}|c)$

      3. What precisely is $P(w_k|c)$?

         a. How likely you are to see the word w_k in a document from class c

         b. If w_k="homework" and c="su.class.cs276a" newsgroup, then

$$P(\text{"homework"}|\text{ newsgroup "su.class.cs276a"}) = \frac{\text{\# of times "homework" appears in "su.class.cs.276a"}}{\text{\# of words in "su.class.cs276a"}}$$

         c. This is the **training**, $P(w_k|c)$ is a **parameter** of our model

   iv. Why did we want to compute $P(D|c)$? Recall, it is a rather weird quantity. We did it to get to $P(c|D)$.

      1. $P(c|D)$ tells us for class c, how likely it is that D is in it

a. Not clear how to compute directly, so we flip it with Bayes Rule, $P(c \mid D) = \frac{P(D \mid c)P(c)}{P(D)}$.

2. If we can compute P(c|D) then we can find the most likely class $c_{MAP} = \underset{c}{\operatorname{argmax}} P(c \mid D) = \underset{c}{\operatorname{argmax}} \frac{P(D \mid c)P(c)}{P(D)}$.

   a. Called **<u>maximum a posteriori</u>** class, eg **after** we see the document as opposed to **<u>prior</u>** P(c), eg **before** we see the document. If I pick a newsgroup, and ask you to guess which one I picked, you best bet is to go with the biggest one, one that maximizes P(c), but if I then give you a document D from the newsgroup I picked and D is about fishing, then you best bet is pick some fishing related newsgroup, the one that will maximize P(c|D).

3. How do we compute $\frac{P(D \mid c)P(c)}{P(D)}$?

   a. P(c) is easy

      i. eg for newsgroups it is $\frac{\text{size of newsgroup c}}{\text{combined total size of all newsgroups}}$

   b. P(D) drops out since it is the same for every class, and won't effect selection of c_{MAP}, so $c_{MAP} = \underset{c}{\operatorname{argmax}} P(c)P(D \mid c)$

   c. P(D|c) we've seen how to compute, so $c_{MAP} = \underset{c}{\operatorname{argmax}} P(c)\left(P(w_1 \mid c)P(w_2 \mid c) * \cdots * P(w_{1,000} \mid c)\right)$

4. Now given D, we can figure its most likely class c_{MAP} (eg what newsgroup it belongs to, is it spam or not, etc. whatever classes we've trained on)

3. Probability vs Similarity
    a. Given a query q, P(user is looking for document D)=P(document D is relevant for query q)=P(document D is relevant | q)=P(relevant | D, q)=P(R|D,q)
        i. You hope it is proportional to similarity(q,D)
            1. if D is very similar to q, P(relevant | D,q) should be high
            2. if not, you hope P(relevant | D,q) is low
        ii. If it is like similarity, then we can rank by it, if $P(R|D_1,q) > P(R|D_2,q)$, then D_1 is more relevant, return it higher on the list – Probability Ranking Principle
    b. Why are we replacing similarity with probability?
        i. It has advantages – firmer theoretical foundation, easier to integrate in other sources of information, maybe it will work much better (this last one hasn't happened)
    c. How to compute P(R|D,q)? Remember R and NR (not relevant) are just classes, like newsgroups, or spam and not spam, apply the same machinery as before
        i. First figure out how to represent D and q with some numbers
            1. Binary representation – D represented by $x=(x_1,\ldots,x_n)$; $x_i=1$ if word i (eg "boat") occurs in D, otherwise $x_i=0$
        ii. Now we want to compute P(R|x,q). Apply Bayes Rule whenever you can, so $P(R \mid x,q) = \dfrac{P(R \mid q)P(x \mid R,q)}{P(x \mid q)}$
        iii. Would be nice not to compute P(x|q). How do we avoid it?
            1. We are not really interested in P(R|x,q) as a number, only want to know given another document y, which should be higher in the ranking, eg is it P(R|x,q) > P(R|y,q) or P(R|x,q) <P(R|y,q)?
                a. If P(R|x,q) > P(R|y,q)

i. Then $\frac{P(R \mid x,q)}{1-P(R \mid x,q)} > \frac{P(R \mid y,q)}{1-P(R \mid y,q)}$

since $\begin{array}{c} P(R|x,q)(1-P(R|y,q) > P(R|y,q)(1-P(R|x,q) \\ P(R|x,q)-P(R|x,q)P(R|y,q) > P(R|y,q)-P(R|y,q)P(R|x,q) \\ P(R|x,q) > P(R|y,q) \end{array}$

b. That last one, while true, is a bit weird, why would we want to do that? Because 1-P(R|x,q)=P(NR|x,q) (NR=not relevant)

c. So $\frac{P(R \mid x,q)}{P(NR \mid x,q)} > \frac{P(R \mid y,q)}{P(NR \mid y,q)}$ iff P(R|x,q) > P(R|y,q)

i. Aha, we can use $\frac{P(R \mid x,q)}{P(NR \mid x,q)}$ instead of P(R|x,q) to do ranking

ii. Flip with Bayes Rule and get $\frac{P(R \mid x,q)}{P(NR \mid x,q)}$

$$= \frac{\frac{P(R \mid q)P(x \mid R,q)}{\mathrm{P(x \mid q)}}}{\frac{P(NR \mid q)P(NR \mid R,q)}{\mathrm{P(x \mid q)}}} = \frac{P(R \mid q)P(x \mid R,q)}{P(NR \mid q)P(x \mid NR,q)}$$

iii. Voila, we don't need P(x|q), this is called the **odds ratio** (odds like in Vegas, 2 to 1, etc) $O(R \mid x,q) = \frac{P(R \mid q)P(x \mid R,q)}{P(NR \mid q)P(x \mid NR,q)}$

iv. Again, not interested in O(R|x,q) as a number, **only** for ranking versus some other document y, eg is it O(R|x,q)>O(R|y,q) or O(R|x,q)<O(R|y,q)? Therefore the $\frac{P(R \mid q)}{P(NR \mid q)}$ term drops out, and we are left computing $\frac{P(x \mid R,q)}{P(x \mid NR,q)}$

d. Recap: want P(user is looking for document D)= P(document D is relevant | q)=P(R|D,q) → *change to binary representation* → P(R|x,q) → *get rid of the need to compute P(x|q)* → O(R|x,q) → *drop P(R|q)* → $\frac{P(x \mid R,q)}{P(x \mid NR,q)}$.

e. How do we compute $P(x|R,q)=P(x_1,\ldots,x_n|R,q)$?

   i. Same as before, remember R is just a class, like a newsgroup, make the Naïve Bayes Assumption, then $P(x_1,\ldots,x_n|R,q)= P(x_1|R,q)*\ldots* P(x_n|R,q)$

   ii. $P(x|NR,q)$ splits apart in the same way

   iii. All that is left is to train, and estimate each individual parameter (eg $P(x_k|R,q)$) by counting frequencies. See Lecture 10 for specifics

f. The model above (binary data representation + Naïve Bayes assumption) is called **<u>Binary Independence Model</u>**

4. Feature Representation

   a. A feature is a piece of information that we represent numerically

      i. eg whether a word k is in document D – if it is, then we set $w_k=1$, otherwise $w_k=0$, w_k is a feature

      ii. We could have had w_k={# of times k appears in D}, or w_k={# number of pigeons in California}. This last one is probably less informative for text retrieval then the first two. In the same way, the second one – the frequency of k in D – may be more informative than just knowing whether k occurred in D or not

   b. Once the feature set is selected, we can churn it through all of our formulas above, and make a classifier (or a probabilistic ranker given some query)

   c. Suppose we have a document D=”the sunny sunny sky”, and a lexicon={“bright”, “sunny”, “sky”, “the”}. How to represent D?

      i. Lets do it like in Binary Independence Model

         1. $D = (\overset{\text{bright}}{0}, \overset{\text{sunny}}{1}, \overset{\text{sky}}{1}, \overset{\text{the}}{1})$

         2. eg $D=(w_1,\ldots,w_n)$, $w_k=1$ iff word k occurs in D

         3. We call this representation **<u>Multivariate Binomial</u>**

      ii. Lets do another model, assign a number to every word:

      $\{\overset{1}{\text{"bright"}},\overset{2}{\text{"sunny"}},\overset{3}{\text{"sky"}},\overset{4}{\text{"the"}}\}$

1. $D = (\overset{\text{the}}{4}, \overset{\text{sunny}}{2}, \overset{\text{sunny}}{2}, \overset{\text{sky}}{3})$
2. eg $D=(w_1,\ldots,w_n)$, $w_i=k$ iff word k occurs at position i
3. Call this one **Multinomial**

d. Under any of the representations, we can still churn through all of our Bayes Rule/Naïve Bayes math, so what is different?

   i. The parameters are very different
      1. Multivariate Binomial: $P(w_k|c)$={how likely we are to observe word k in a document from class c}
      2. Multinomial: $P(w_i=k|c)$={how likely we are to observe word k in position i in a document from class c}
      3. All parameters still estimated with frequencies during training

   ii. Multinomial keeps position and frequency information