file/after/tracklang.stypackage/after/bidipackage/after/luabidi

Evaluation

There are several ways of evaluating the performance of our model. **Recall** is the fraction of relevant documents that have been retrieved. **Precision** is the fraction of the *retrieved* documents that are relevant.

The precision is the percentage of relevant datasets that our model has retrieved from the "correct" answer set. The **precision versus recall curve** is the measure of the precision of our model when the relevant documents are greater.

We then test over the different recall levels. The **Mean Average Precision** measures the precision obtained from the top k documents.

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} P(R_{jk}) \tag{1}$$

Sometimes, some numbers can summarize a set of queries. The **Mean Reciprocal Rank** is the multiplicative inverse of the rank of the first correct answer in our retrieved items.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{-}$$

which is essentially a measure of how long it takes for the first correct answer takes to appear in the retrieved documents.

Some other alternative measures include **harmonic measures**, which is the inverse of the arithmetic mean. Why would this be useful?

$$F(j) = \frac{2}{\frac{1}{r(j)} + \frac{1}{P(j)}} \tag{3}$$

We can use a measure such as β to give a weight to the precision and recall. Another measure is that of the **confusion matrix**, where we have a measure of the True positives (the elements we correctly include), true negatives (the element's we're not suposed to include), False positives (the items we thought relevant but not), and false negatives (the relevant items we missed in our query). We can place all the items in a 2x2 matrix.

There are several interesting results we can get, e.g. the **accuracy** of a test is the measure of how many of the True items we classified correctly, so $(TP + TN)/N$. Measuring the accuracy has its problems,

for example if the classes we are trying to predict are not balanced, then by trivially classifying all the items as one class we can still achieve a really high accuracy (this would probs only work on a binary classfication problem).

There are other measures we can also use which take into account the **cost** of making a wrong or right decision, it could be like a weight of some sort. Precision, recall, and F-measures all have their biases,

- Precision is based towards true positives, $C(yes|yes)$
- Recall is based towards true positives, $C(yes|yes)$
- F-measure is biased towards all except the True negatives.

The **Receiver operating characteristic** is a measure of the sensitivity and the specificity of the test. We can think of the true positives on one axis and the false positives on the other. This means that we are measuring the rate at which we choose false positives vs. true positives. We obvs. want to choose more true positives, so our plot should look like a curve lying above the line $y = x$.

ROC curves have a couple benefits: they are invariant to class distribution (doesn't matter how many items per class and their proportion); good measure of our model's ability to produce good positive results.

> What is the relationship between F1 and break-even point? When the recall and precision are the same, the F1 is 1. ## Methods of Estimation How do we actually measure the performance of our dataset?

- Holdout: Keep a fraction of the dataset for testing (2/3, 1/3 split)
- Random subsampling: Repeated holdout
- Cross-validation: Take the dataset and split it into k disjoint subsets. Then train on $k-1$ subsets and use the remaining one to check accuracy.

These work for any type of data, but suppose we have a ranked list, how can we measure if a model is better? 80% accuracy on 30 items vs 75% on 3000 items is better?

For ranked items, we can use **Discounted Cumulative Gain**, where we measure the usefulness of a document based on its position in the list. We have some measure of the relevance rel_i. The discounted cumulative gain is the sum of the relevance of the items at rank i. However, to the have the position influence the value of the relevance, we can use

$$DCG_p = rel_1 + \frac{rel_i}{\log_2 i} \tag{4}$$

Cohen's Kapp correlation coefficient measures two classifiers who classify the data. We then measure the probability of their agreement and the probability that they agreed on their rating by accident.

Where are the following most useful? - NDCG (ranked list, where position in the ranked query is of relevance) - Recall: How useful the model is in capturing mo - Top-1 precision: precision for the top documents, how useful the first documents are. - F1 : Useful if we want to weight the precision and recall. - Novelty: When we have relevant documents to each user.