Lecture 3 (unfinished)

Evaluation of IR systems.

Suppose we have a search engine. A natural question is how well does it work, i.e. do the documents that it returns satisfy the user's request.
Suppose we are to choose between two search engines developed by different companies. Then we may want to know which is the better one.
Suppose our search engine has a parameter which we can tune. This parameter affect the quality of the result list the search engine returns. What value of the parameter should we choose?
To answer those questions we want to put a number to each search engine that measures the quality of its results. To this end we need a mathematical definition (a formula) that from a set of results produced by the search engine computes a measure that corresponds to the quality of the results, and therefore to the quality of the search engine. The above questions reflect the point of view of the system designer.

If you are a user of a search engine you do not care what algorithm and what tuning parameters it uses internally. Your satisfaction with the search engine is based on whether it "does what you want". If a researcher cares to account for the user point of view, then she conducts user studies where real people interact with the search engine. The researcher then gathers and analyses data from the experiments. Interviews with the participants may be conducted; sometimes the participants are even videotaped.

The evaluation measures that reflect the systems point of view are called **system-oriented measures**. The measures in the second group which reflect the user point of view are called **user-oriented**.

System-oriented measures are cheap to use because once an evaluation setup is created (the setup is a collection, a set of queries and relevance judgments, see later about them) then only computers are required.

To carry out an evaluation of a search engine we need

- a collection of documents (corpus)
- a set of queries
- for each query we need an extensive set of relevance judgments.

A relevance judgment is a decision whether a particular document in the corpus is relevant with respect to a given query. In other words a person (often called judge) has to decide whether the document satisfies the information need that the query tries to express. The information need is specified to the judge in detail, while the query may contain only a short description. For example the following is a query from TREC. The query can be hunger but the relevance is decided upon the narrative and the description.

<num> Number: 453
<title> hunger

<desc> Description:
Find documents that discuss organizations/groups that are aiding in the eradication of the worldwide hunger problem.

<narr> Narrative:

Relevant documents contain the name of any organization or group that is attempting to relieve the hunger problem in the world. Documents that address the problem only without providing names of organizations/groups that are working on hunger are irrelevant.

It is common for the collection to contain millions of documents. In the best case the collection should contain real world data. Some of collections the collections used by academia are from newswire, for example Reuters and Associated Press. Trying to get closer to the real world, the latest TREC tasks use data that is collected from the web. TREC stands for Text Retrieval Conference. It is organized by National Institute of standards and Technology. Its goal is to promote and evaluate research in Information Retrieval.

We would also like to have many queries and those queries to be close to what people in the real world are inputting. Imaging deciding that system A is better than system B on the basis of one query alone. Would you trust the conclusion? In past years, TREC has used 50 queries to evaluate search engines.

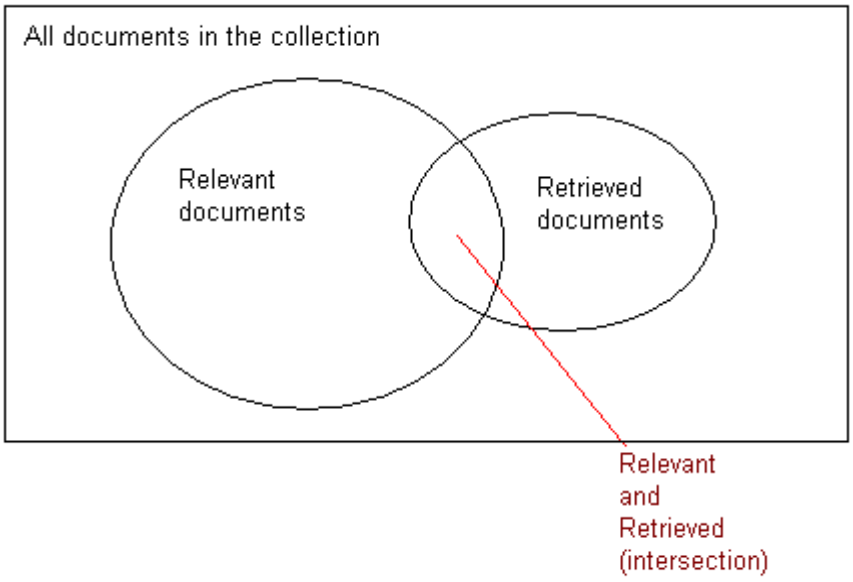
We would also like to have extensive judgments. When the collection contains over a million documents it is impossible even for a single query to have information whether every document is relevant. What we can hope is that we have information at least for those documents that are very likely to be relevant. A major question here is how to decide which documents to judge. Common methods are pooling and sampling.

We would also like the judgments to be accurate. The judgments (binary decisions which documents are relevant) are made by people. Those judgments are not the absolute truth. It is common that two judges disagree about the relevance of a document; it has also been found out that the same judge will judge a document differently at different times. One may also disagree that relevance judgments should be binary since clearly some documents are more relevant than others.

Since there is so much possibility for errors (judges disagree, we do not have relevance for all documents, etc.) one can wonder how it is possible to evaluate search engines. The answer is that if many judges judge the same document (usually that does not happen in practice because of budget constraints) and we use many queries then the errors made will cancel out. That is even if an individual judgment may be wrong, as a whole the judgments are correct.

The first problem we deal with is evaluation of **set retrieval** (documents are a set, not a ranked list) for a **single query** assuming the **relevance judgments are accurate**.
We are given a set of documents, each of which is classified as relevant or not relevant by a judge. We care only about the decisions which we represent as R for relevant and N for non-relevant and not about the documents themselves. We also need to know how many relevant documents are altogether in the collection for the given query. Their number we represent by R.

Example: the set of retrieved documents is {R, N, N, R, R, N, N, R, R, N} . Assume there are 20 relevant documents in the corpus.



Define **Precision = |Relevant And Retrieved| / |Retrieved|**. The precision is what fraction of the retrieved documents are relevant. In our example

Precision = 5 / 10 because out of the 10 retrieved documents only 5 are relevant.

Precision can also be interpret as the conditional probability that a document is relevant given the information that it as retrieved. If I tell you that a document is in the retrieved set in this example you can conclude that it has 50% chance of being relevant because 5 out of 10 documents are relevant. We denote this conditional probability as Precision = P(Relevant | Retrieved).

Define **Recall = | Relevant and Retrieved| / |Relevant|**. The recall is what fraction of the relevant documents are retrieved. In the example

Recall = 5 / 20 because there 5 relevant documents out of 20 were retrieved.

Recall can also be written as the conditional probability

Recall = P(Retrieved | Relevant). In our example this would mean that if one knows that some document is relevant, then one can conclude that the chance to be retrieved is 5/20.

One can think of precision as focusing on the garbage (the nonrelevant) documents that the system returned, while recall emphasizes the missed opportunities.

So, low precision = lot's of garbage in the retrieved set

low recall = a lot of missed opportunities, that is many relevant documents were not found.

The best situation is to have high precision and high recall. Later we will argue that this is hard, or even impossible to attain. To put it simply: high precision implies low recall. This means there are only a few documents for which the system can be very certain that they are correct.

High recall implies low precision. This means if you require to retrieve most of the relevant documents you have to tolerate lots of noise or garbage.

There is a trade-off between precision and recall. The trade-off is usually given by one's tolerance level for garbage (or irrelevant documents). Now we give the F1-measure that measures the trade-off:

F1 = 2 / (1/Precision + 1 / Recall).

What does the F1-measure measure? To find out we replace Precision and Recall by their formulas
Precision = |Relevant and Retrieved| / |Retrieved|,
Recall = | Relevant and Retrieved| / |Recall|

F1 = 2 |Relevant and Retrieved| / (|Relevant| + |Retrieved|).
We need to look at the Venn diagram of the intersection of Relevant and Retrieved documents for intuition. The F1 measure corresponds to the intersection of relevant and Retrieved normalized by the sum of the sizes of the relevant and retrieved set. We need to compare it with the formula for the Dice coefficient, which is defined as

Dice(A, B) = 2 |A intersect B| / (|A| + |B|)

to find out that F1-measure is the same thing. The Dice coefficient measures the agreement or amount of commonness between two sets A and B.
We conclude that the F1-measure measures the commonness between the set of Relevant and Retrieved documents. The F1-measure can also be read as the harmonic mean between the variables Precision and Recall. The following inequalities between the Harmonic, Geometric and Arithmetic means are famous

$$2/(1 / x + 1 / y) <= (x * y)^{1/2} <= ½ (x + y)$$

Ranked retrieval: Here we evaluate the quality of lists are opposed to sets.
Consider a system A which retrieves a the ranked list R, R, N, N and system B which retrieves the list N, N, R, R. Obviously system A is better than B. However any set retrieval measure will report that system A and system B have the same quality. This is provides motivation for introducing retrieval measures based on ranked lists instead on sets. Another way to say the same thing is that relevant documents at higher ranks are more important than relevant documents at lower ranks and we want our measures to account for that.

Example: Consider the following ranked list returned by a system
RNNRR NNRNR. Assume that there are 20 relevant documents in the collection for this query. We can think of this list as sequence of increasing sets A1, A2, ..., A10 where
set A1 = {R} contains only the first relevant document, the set A2 = {R, R} contains the first and the second document, A3 = {R, R, N} and so on. A natural way to think about this sets is that every person who would browse the list would start at the top. A person whose attention span is only one document will read only the first documents. One who's attention span is 10 documents will read the top 10 documents. For each set A(i) we can calculate its Precision and Recall. Let us do so in the table below.

Rank	Rele-vance	The set of documents at rank higher or equal (that is the number of the rank is lower or equal)	Precision	Recall
1	R	{R}	1	0.05
2	N	{R, N}	0.5	0.05
3	N	{R, N, N}	0.33	0.05
4	R	{R, N, N, R}	0.5	0.1
5	R	{R, N, N, R, R}	0.6	0.15
6	N	{R, N, N, R, R, N}	0.5	0.15
7	N	{R, N, N, R, R, N, N}	0.43	0.15
8	R	{R, N, N, R, R, N, N, R}	0.5	0.2
9	N	{R, N, N, R, R, N, N, R, N}	0.44	0.2
10	R	{R, N, N, R, R, N, N, R, N, R}	0.5	0.25

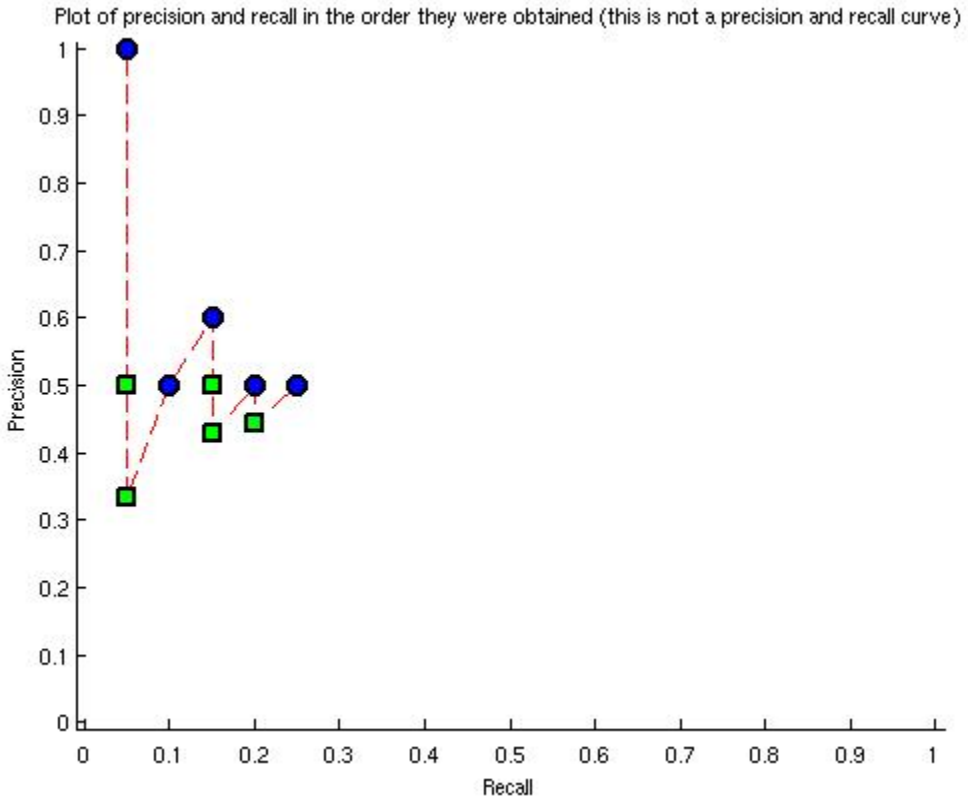
Let's make some observations. First, as we go deeper in the list Recall increases. That is because recall increases every time we see a relevant document. If we see a non-relevant document, then recall stays the same.
We can also notice that precision tends to decrease as we go lower in the list although it can jump up. Specifically if we see a non-relevant document Precision drops.
If we see a relevant document the precision stays the same or goes higher. It can also happen that for a long time precision is zero and then increases till the end of the list (See example:). When we say that precision tends to decrease we mean that 1) this is the natural thing to happen (the pathological example happens if a system flips its list) and 2) this is what usually happens in practice.

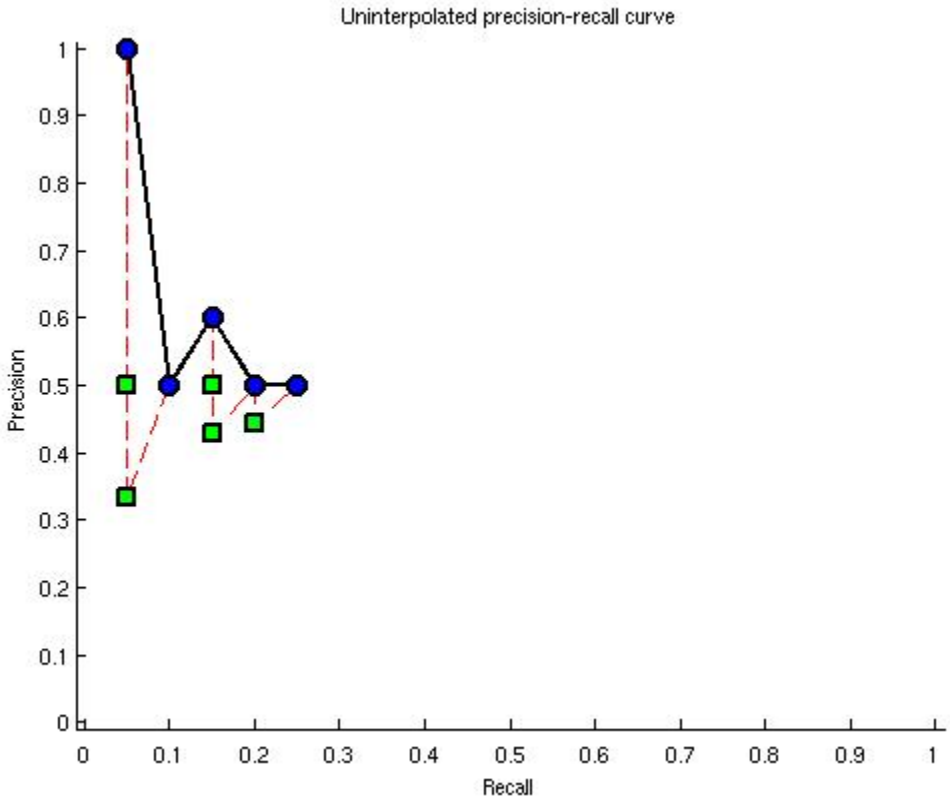
We can plot the precision and recall for easy inspection. Here one can see the trend that precision decreases with recall. Notice that that are exactly 10 markers (circles and squares) on the plot. The blue circles correspond to relevant documents while the green squares correspond to non-relevant documents. This plot will be a starting point for two types of curves we will use: the uninterpolated and interpolated precision-recall curve. One can see the saw-toothed shape that results if we are to just connect the points. The curves however are smoother.

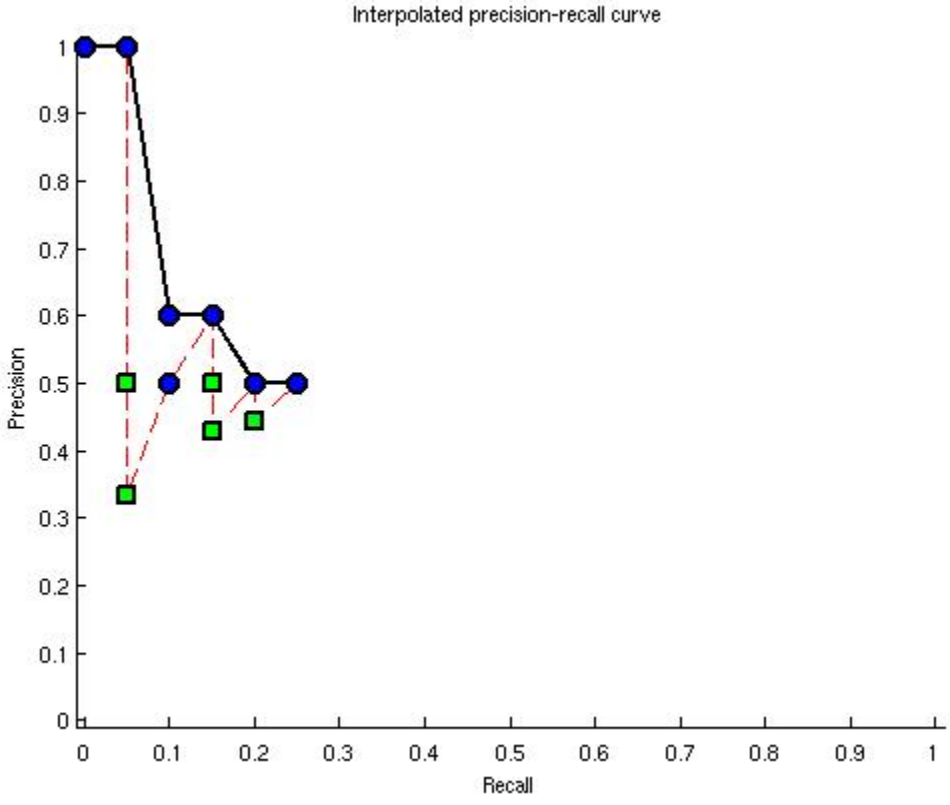
The uninterpreted curve: one can see that at the same recall level it is possible to have different precisions. For example at recall level 0.5 we have precisions equal to 1.0, 0.5, 0.33. Now to each recall level that appears in the list set the precision to be maximum precision at that recall level. For example for recall level 0.5 we set precision to be max(1.0, 0.5, 0.33) = 1.0. For the next recall level of 0.1 we have precision is max(0.5) = 0.5, and so on. See table. This operation correspond to raising the green points vertically up to the corresponding blue point. Then we simply connect the points. See the resulting plot.

Interpolated precision recall curve: at each achieved level of recall we set the precision to be equal to the maximum precision at this or higher recall levels. For example for recall level 0.1 we set the precision to be equal to max(0.5, 0.6, 0.5, 0.43, 0.5, 0.44, 0.5) = 0.6. This corresponds to raising all points with x coordinate equal to 0.1 (in our case only one point, but there can be many) to the maximum of its level and the level of the highest point to the right of it (towards 1 on the x-axis). See figure .

Rank	Relevance	Precision	Recall		Max precision at that recall level (use for uninterpolated prec-recall curve)	Max precision at that recall level or higher (use for interpolated prec-recall curve)
1	R	1	0.05		1	1
2	N	0.5	0.05		1	1
3	N	0.33	0.05		1	1
4	R	0.5	0.1		0.5	0.6
5	R	0.6	0.15		0.6	0.6
6	N	0.5	0.15		0.6	0.6
7	N	0.43	0.15		0.43	0.6
8	R	0.5	0.2		0.5	0.5
9	N	0.44	0.2		0.5	0.5
10	R	0.5	0.25		0.5	0.5




Uninterpolated precision-recall curve
Precision
Recall
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


Interpolated precision-recall curve
Precision
Recall
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

Precision at cut-off k. This is the precision calculated for the set of the top-k documents returned. We denote it as PC(k).
Using example precision at cut-off 1 is PC(1) = 1 and precision at cut-off 7 is PC(7) = 0.43

Precision at k% recall (also known as **interpolated precision** at k% recall, although there is no interpolation here). This is the maximum of the precisions at any achieved level of recall m that is higher or equal to k (m >= k). Using the example precision at 0 % recall is 1; Prec. at 17.5 % recall is 0.5.

Average precision and R-precision.

Average precision is defined in the following way. Let R be the number of documents relevant to the query. Let r_1, r_2, ..., r_m be the ranks where relevant documents that appear in the retrieved list (the list contains exactly m relevant documents). We can also suppose $r_1 < r_2, <... <r_m$. PC(k) is the precision at cut-off k as defined below.

$AP = (1 / R) * (PC(r_1) + PC(r_2) + ... + PC(r_m))$.

We emphasize:

- divide by R, which is the number of relevant documents in the collection for this query. It is not how many documents the system retrieved.
- Use only precisions at ranks where relevant documents were retrieved.

Example:
RNNRR NNRNR is the list. It contains 10 documents, 5 of which are relevant. They were retrieved at ranks r1 = 1, r2 = 4, r3 = 5, r4 = 8, r5 = 10.
Let R = 20.
Then AP = (1/20) * (1/1 + 2/4 + 3/5 + 4/8 + 5 / 10)
We can conclude that AP can also be calculated according to this formula

$AP = (1/r_1 + 2/r_2 + 3/r_3 + ... + m / r_m) / R$

R-precision: R-precision is defined as precision at cut-off R, where R is the number of relevant documents for the query. If this list contains less than R documents, we can always pad with list non-relevant documents and evaluate R-precision of this list.

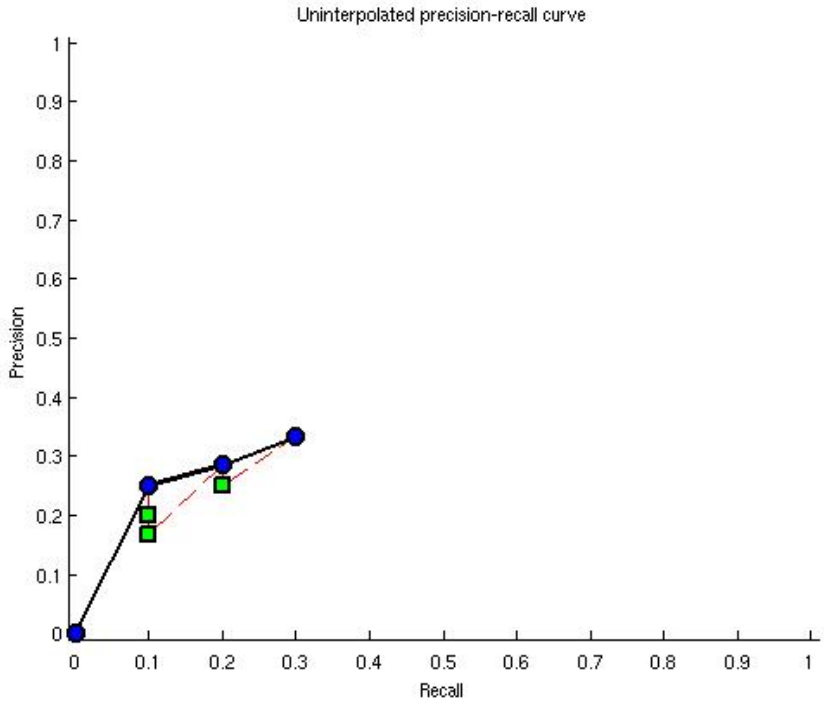
Example: List is RNNRR NNRNR NNNRN RNNNN. Assume that are 11 relevant documents for the query. The R-precision is equal to precision at cut-off 11, or the precision for the set of the first 11 documents retrieved (precision for RNNRR NNRNR N).
R-precision = 5/11
What is the recall at that point: Recall = |Relevant and Retrieved| / |Relevant| = 5/11.
We conclude that R-precision is the point where precision is equal to recall.
To prove that formally:
R-precision = PC(R) = |Relevant and Retrieved| / |Retrieved| = |Relevant and Retrieved| / R
Recall(R) = |Relevant and Retrieved| / |Relevant| = |Relevant and Retrieved| / R

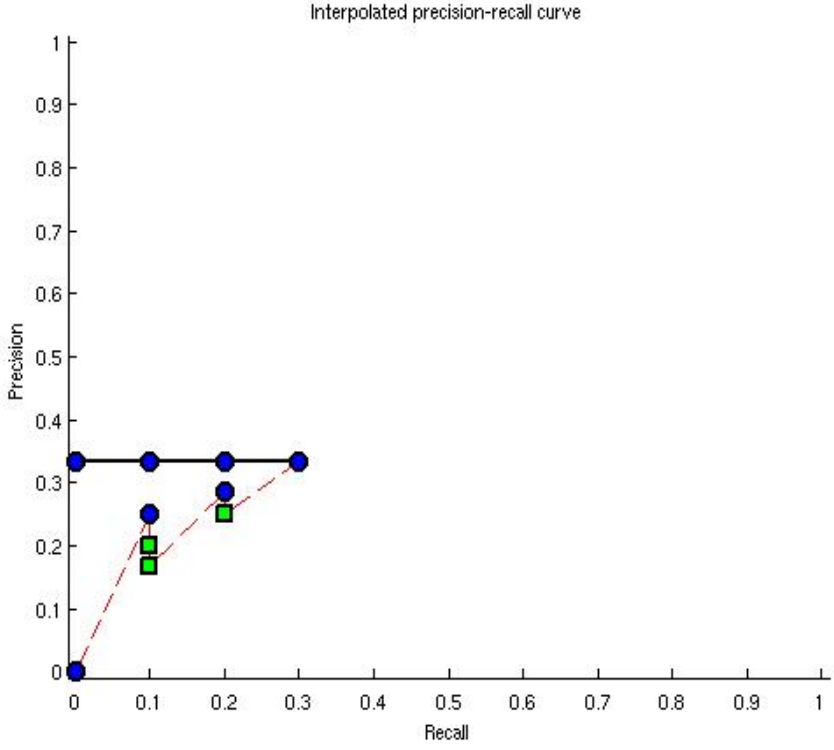
Relationship of Average precision to Precision-Recall curve. **Average precision approximates the area under precision recall curve.**
Proof:

Relationship between R-precision and AP. In homework 1 you have to prove that R-precision and AP are approximately the same, because both approximate the area under the precision recall curve. Here we show a case where R-precision is 0 but AP is 0.5. Consider the following list NR and R (number of relevant documents for the query is 1). Then R-prec = 0/1 = 0
and AP = 1/1(½) = 0.5

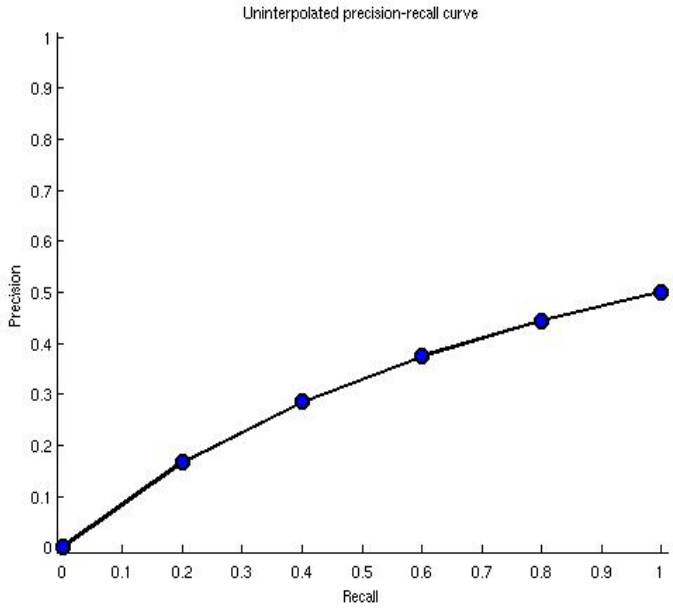
This example demonstrates that AP is more robust than R-precision because the list NR is of good quality but R-precision could not detect it.

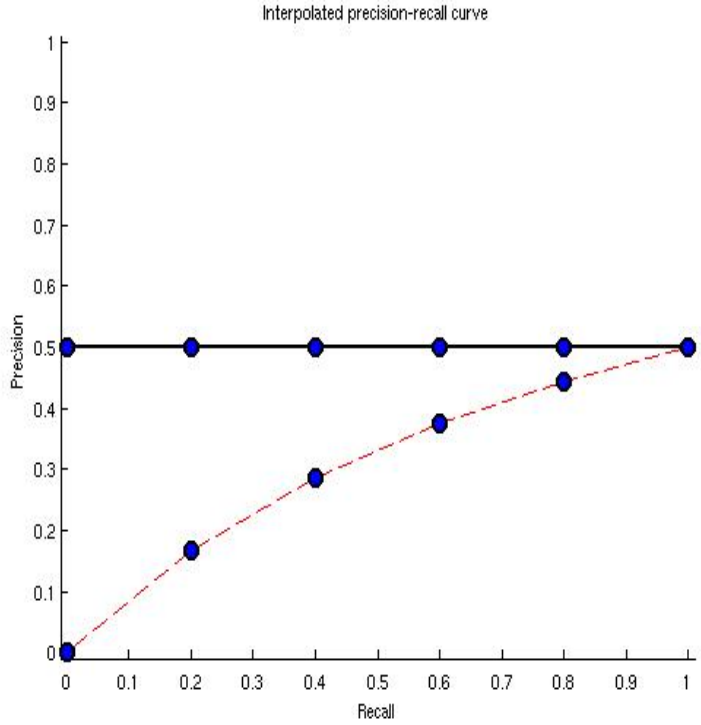
Examples for uninterpolated and interpolated precision-recall curves.
Example:
list = [N, N, N, R, N, N, R, N, R]
R = number of relevant documents for the query = 10



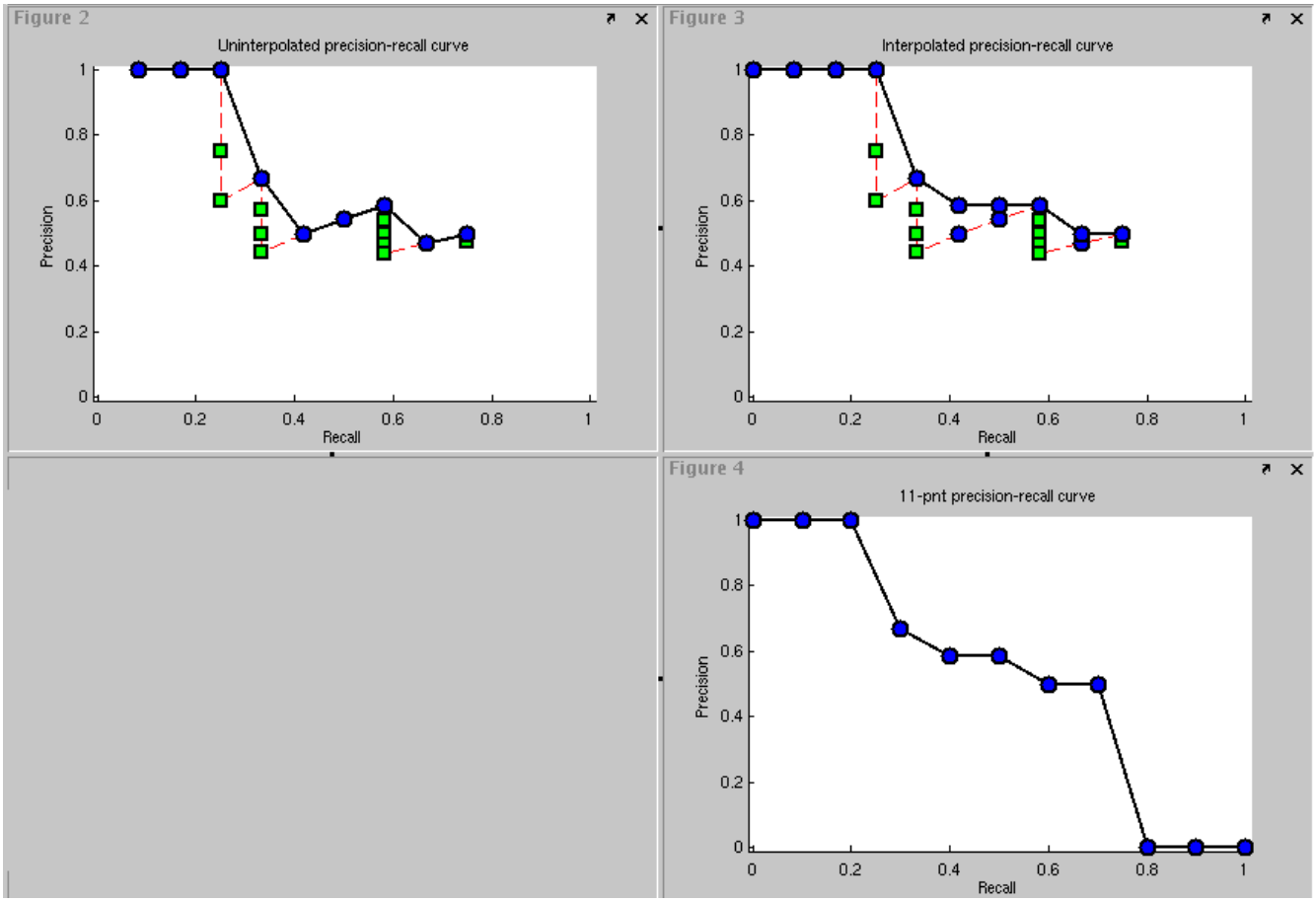


Example:
list = [N, N, N, N, N, R, R, R, R, R]
R = number of relevant = 5





List = L = [1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0] (here 1 means relevant , 0 means non-relevant)
R = 12



Examples of evaluation of IR systems by TREC (for the project it will be good if you produce such plots)

http://trec.nist.gov/pubs/trec8/appendices/A/adhoc_results/READWARE.table.pdf

http://trec.nist.gov/pubs/trec8/appendices/A/adhoc_results/Sab8A2.table.pdf

http://trec.nist.gov/pubs/trec8/appendices/A/adhoc_results/UT810.table.pdf

TO DO: 11-pnt Precision-Recall Curve, ROC curve