**Information Retrieval HW4**

Done by: A0168082B, A0170229J

**Query Refinement Documentations**

In this assignment we tried to implement 2 query refinement techniques, Pseudo Relevance Feedback and Thesaurus-Based Query Expansion with Princeton’s wordnet.

During the implementation, we decided to make some of our own minor modifications to both query refinement techniques. The main reasons for such minor modifications is to ensure that the query time is able to meet the 1 minute time limit. Such modifications will be explained within the code.

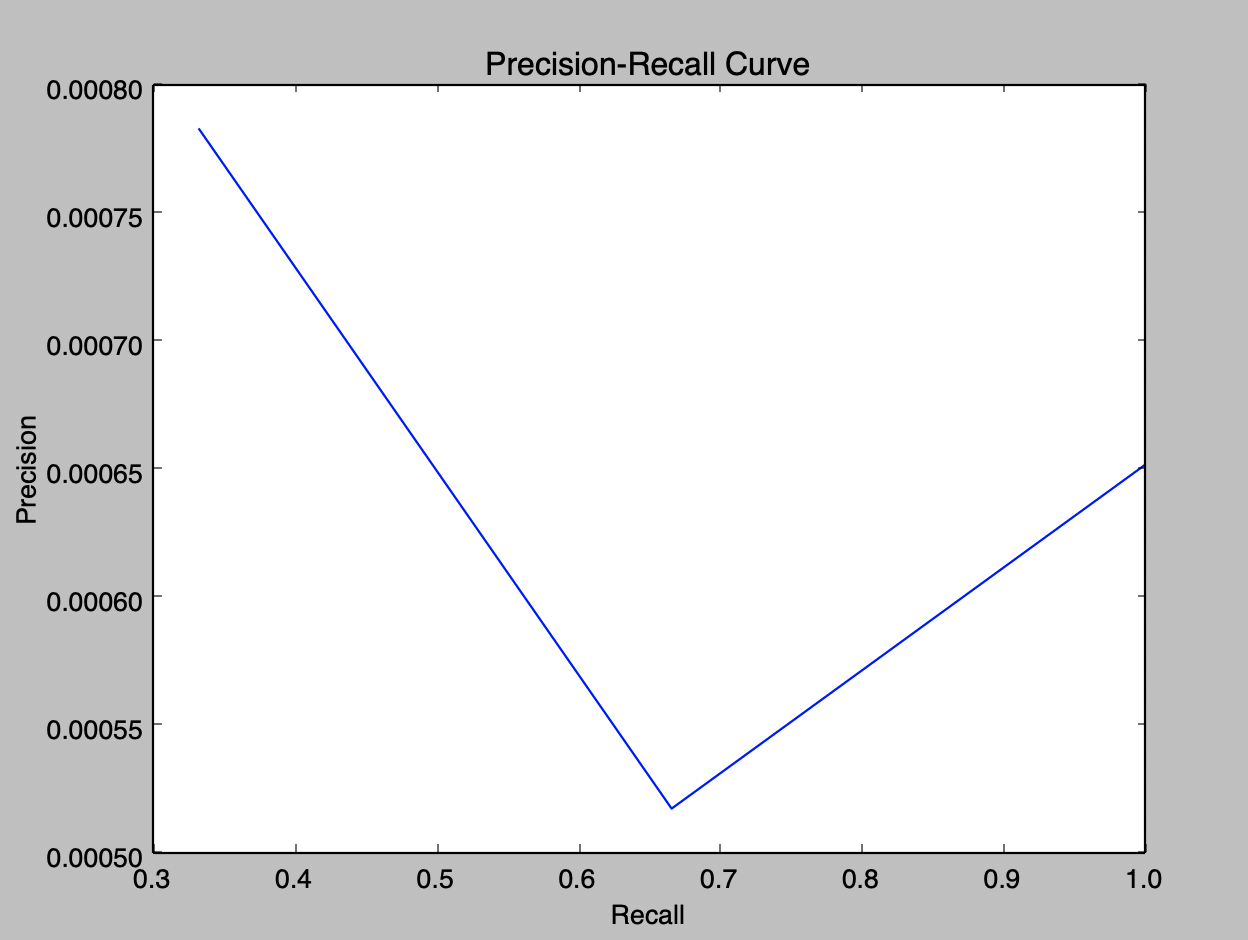
Before, we go on to discuss our implementation and results, we will first be discussing the performance of our baseline system without any form of refinement techniques.

We will also be treating all our Boolean queries as Freetext queries due to the seemingly better performance during our testing.

**Baseline**

For the baseline, without any sort of query refinement techniques, the following are some of our results.

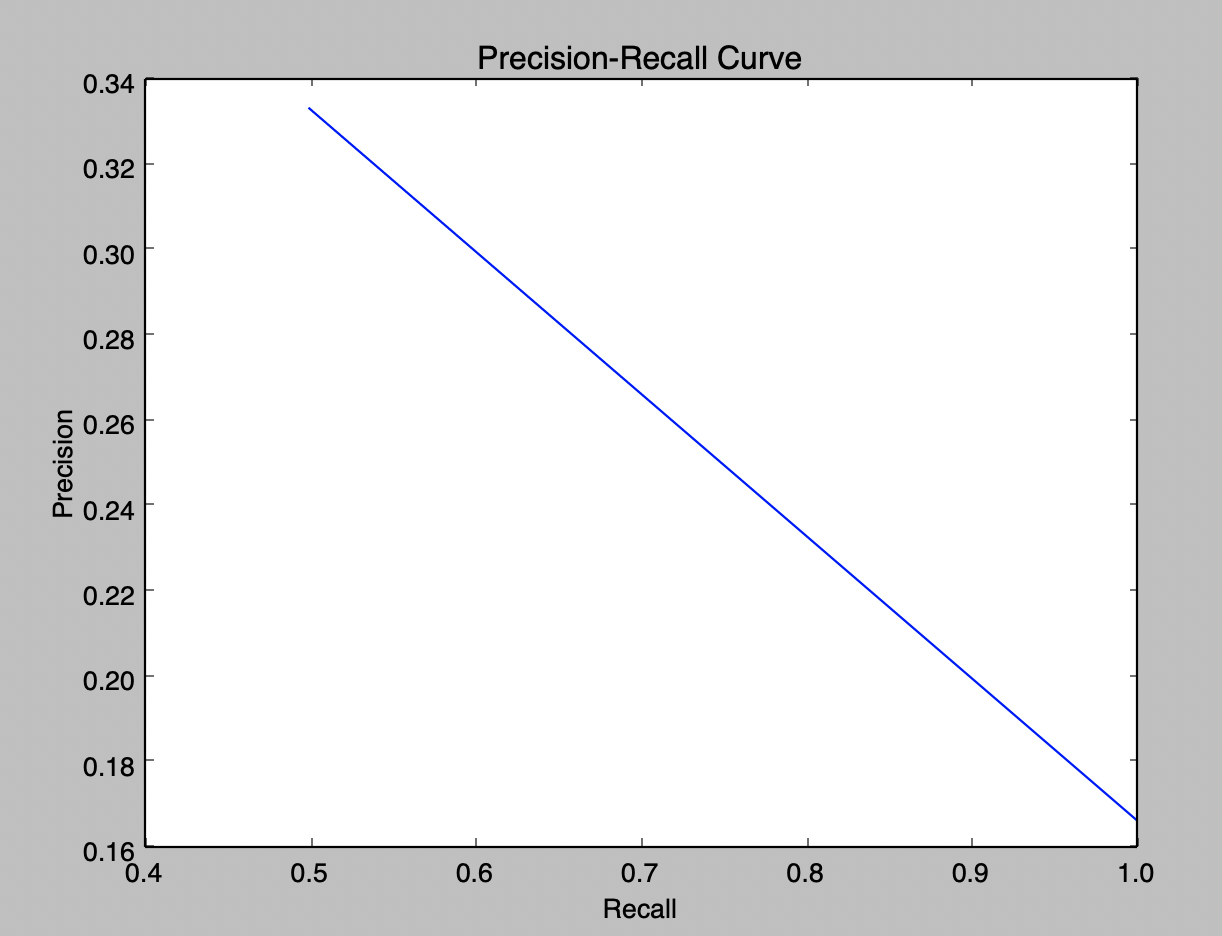
Query: “quiet phone call”



Average F2: 0.00323225906821

Average Precision: 0.0006947832389

Query: “good grades exchange scandal”

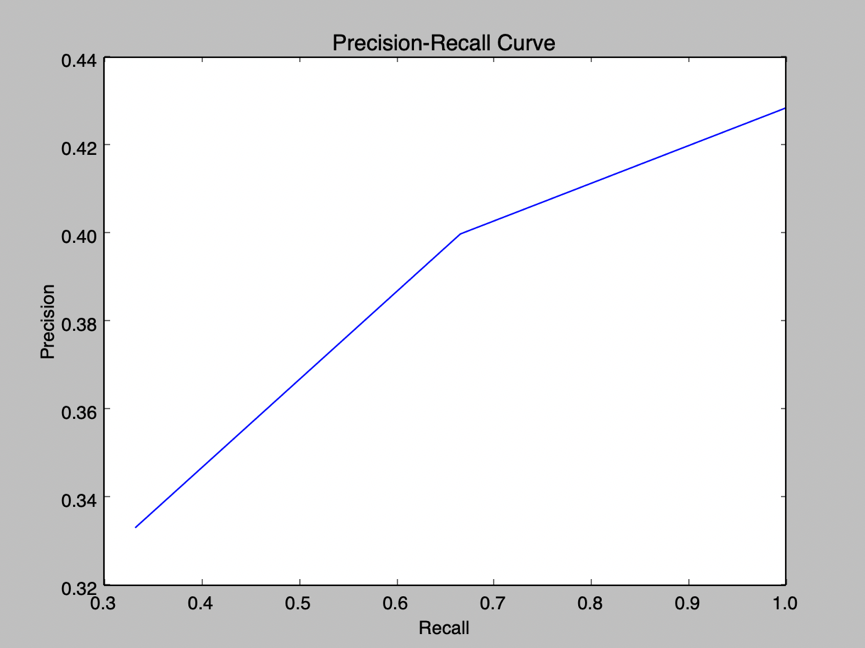


Average F2: 0.41958041958

Average Precision:

0.291666665

Query: "fertility treatment" AND damages



Average F2: 0.486531986532

Average Precision: 0.3624338524

**Overall: Mean Average F2: 0.3031148884**

**Mean Average Precision: 0.2182651002**

**Query Expansion with Princeton’s wordnet**

When experimenting with Query Expansion with the Princeton’s wordnet. We found out that when trying to find synonymous words for a given term, most of the synonymous terms that is generated is normally nonsensical given a certain context.

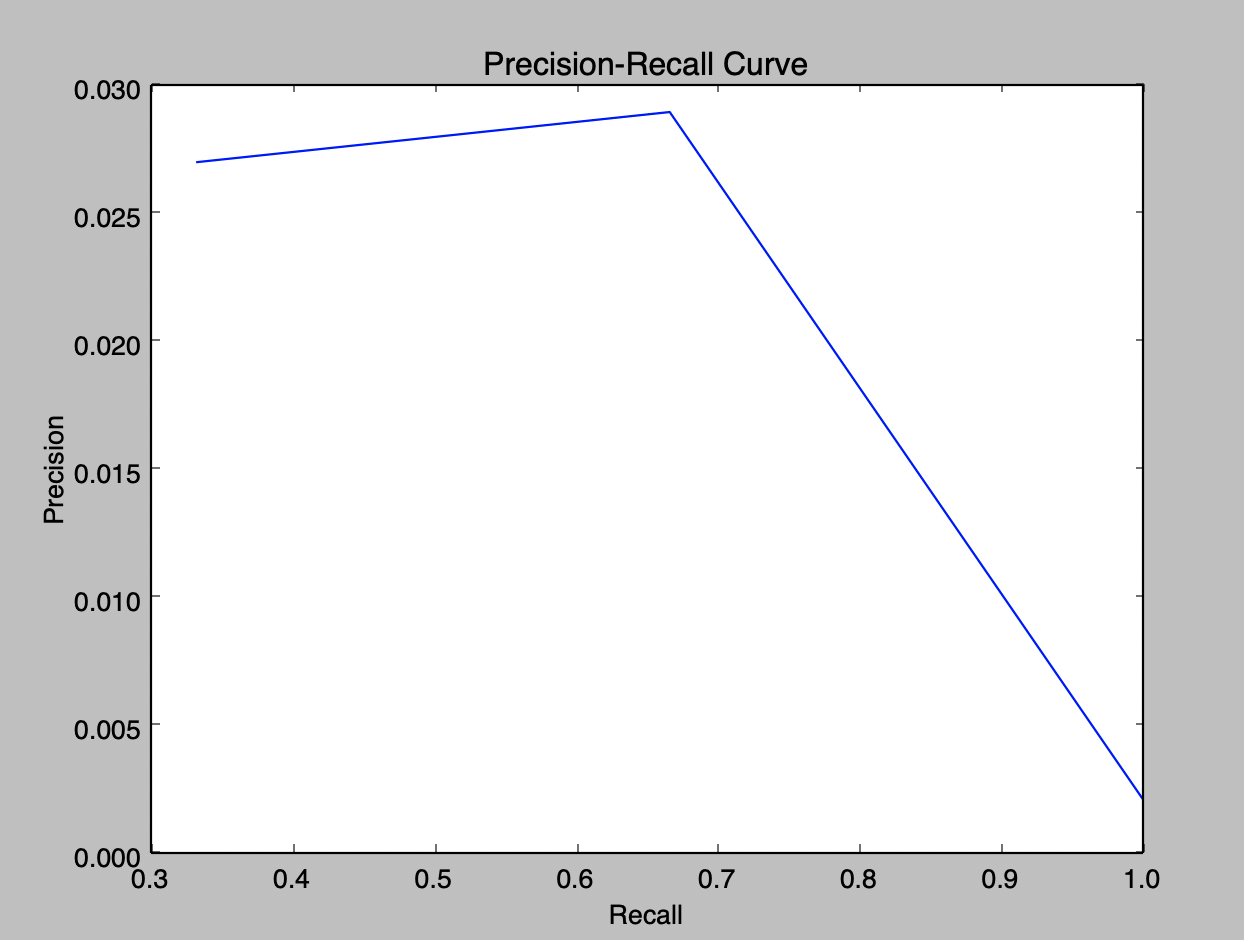
For example, for the query “quiet phone call”, the wordnet will regard the word “call” as one of the words synonymous with “phone”. If we replace the word “phone” with “call”, we will get the query “quiet call call” which is nonsensical in this scenario.

To try and filter out such terms, we will be utilizing the Lesk algorithm provided by NLTK, that helps us with word sense disambiguation. Lesk algorithm helps us prevent some of the above-mentioned situation as synonymous words returned by the Lesk algorithm will also consider the context of the rest of the query.

We will first be presenting our findings of query expansion without the Lesk Algorithm, and then with Lesk algorithm next.

**Without Lesk Algorithm**

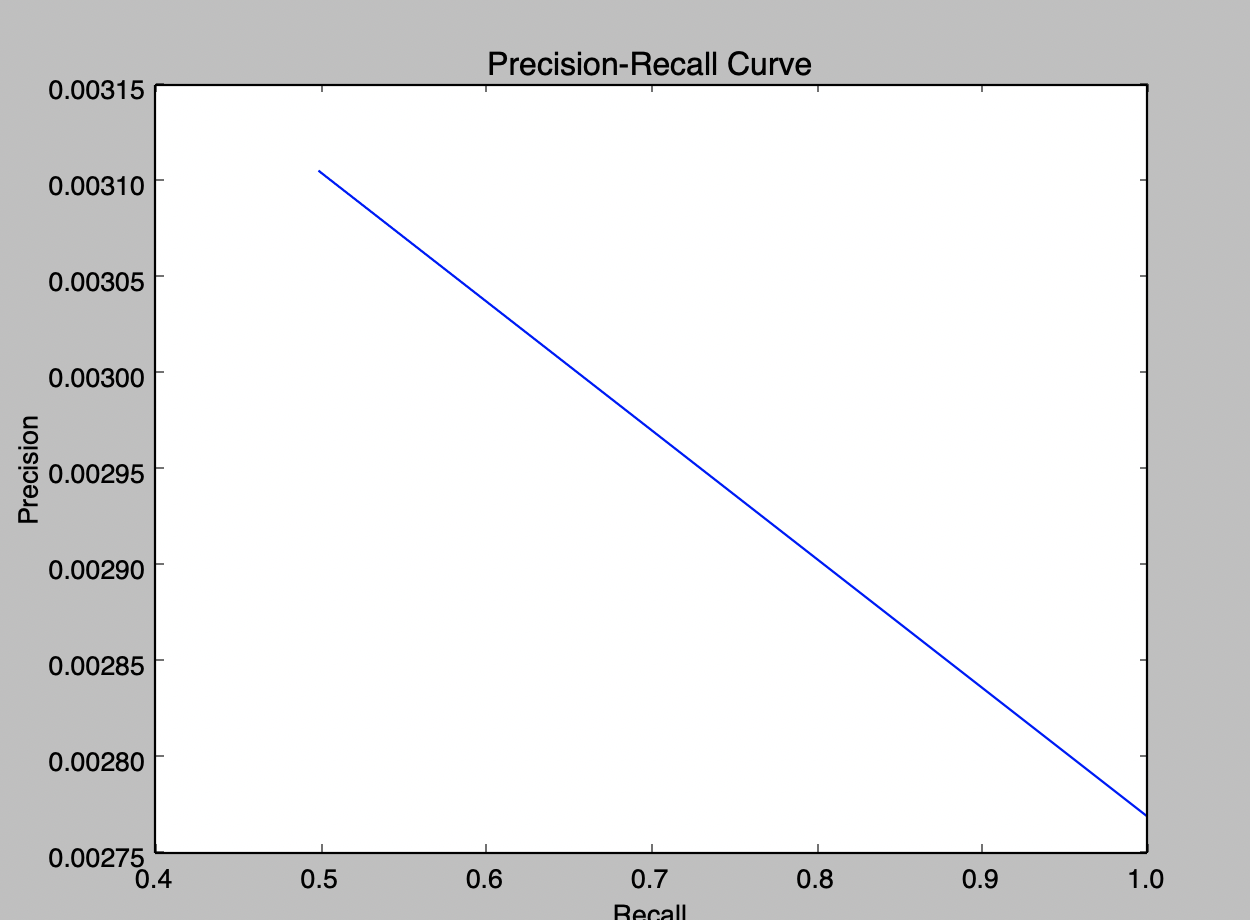
Query: “quiet phone call”



Average F2: 0.0753439300371

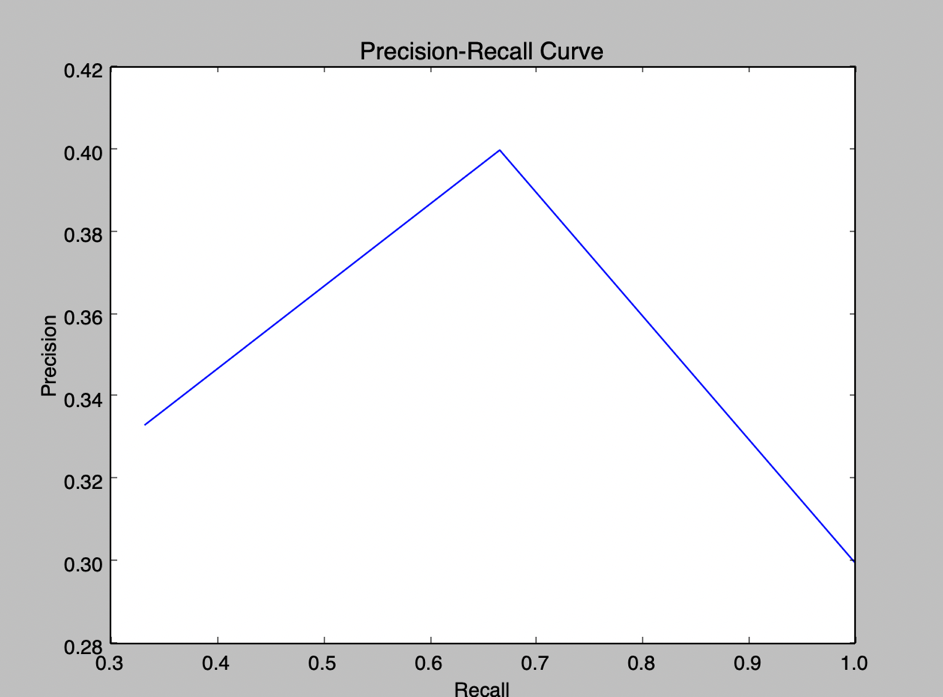
Average Precision: 0.02481135531

Query: “good grades exchange scandal”



Average F2: 0.0143607215748

Average Precision: 0.003021713322

Query: "fertility treatment" AND damages

Average F2: 0.459259259259

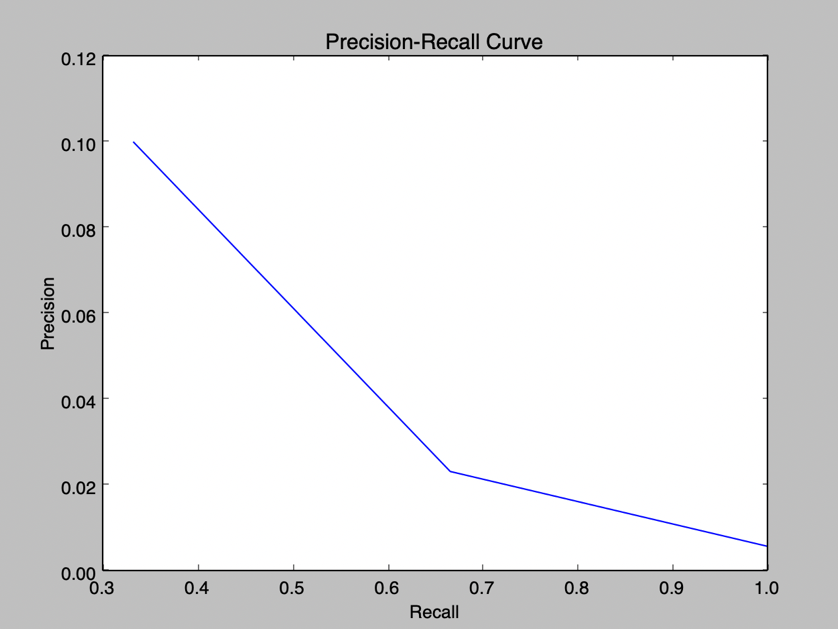
Average Precision: 0.3481481471

**Overall: Mean Average F2: 0.1829879703**

**Mean Average Precision: 0.1253270719**

**With Lesk Algorithm**

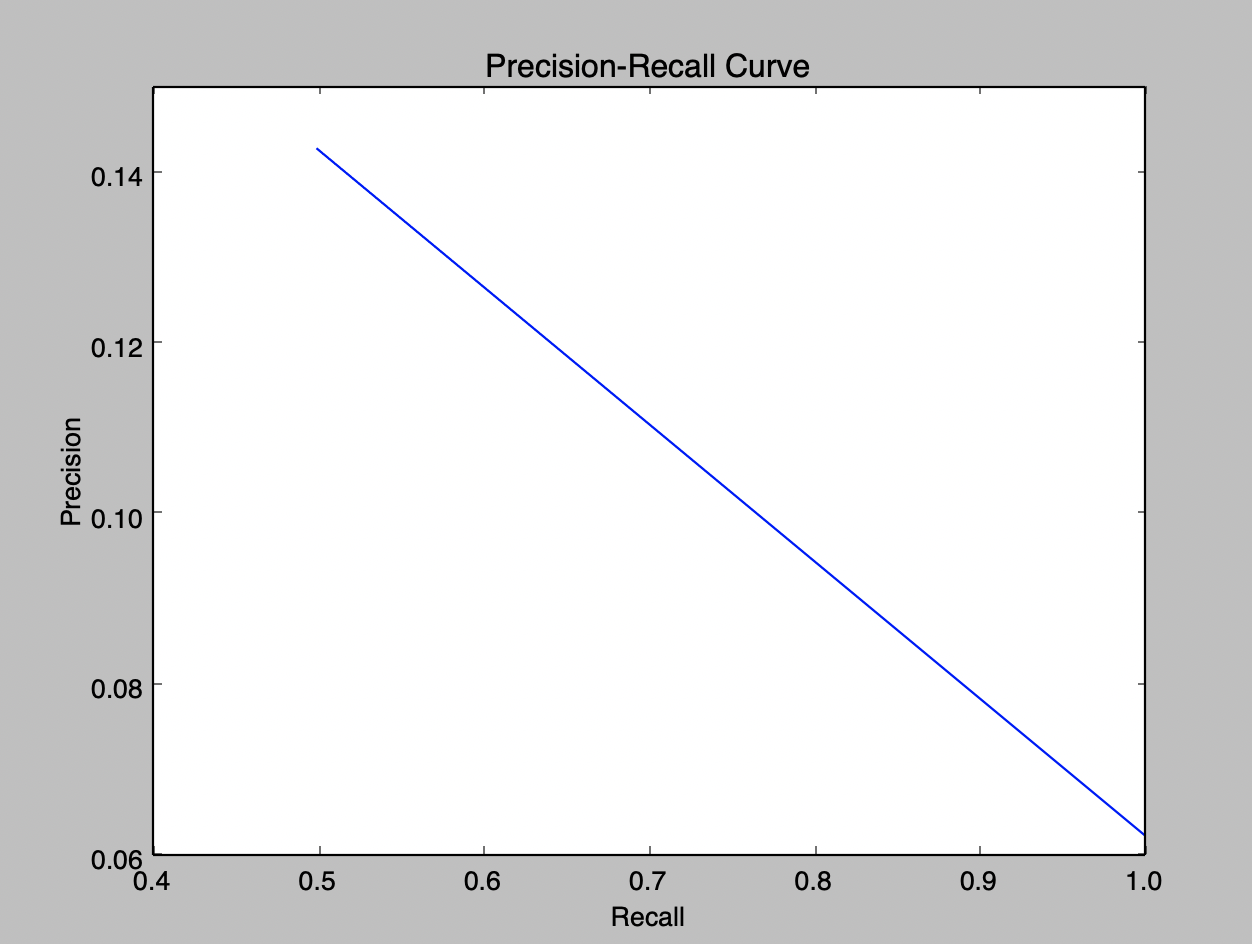
Query: “quiet phone call”



Average F2: 0.109121762838

Average Precision: 0.06821956786

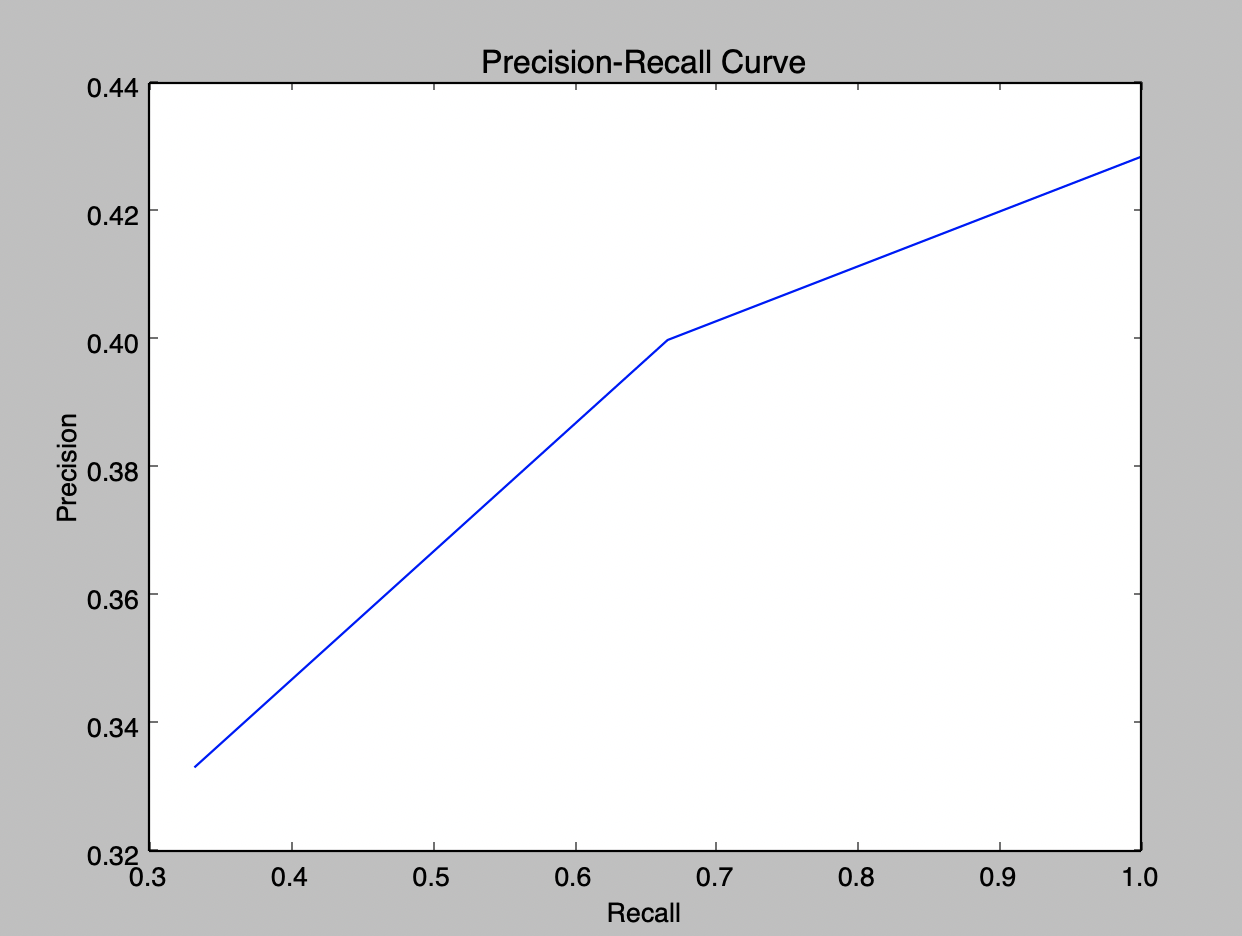
Query: “good grades exchange scandal”



Average F2: 0.266106442577

Average Precision: 0.1227678571

Query: "fertility treatment" AND damages



Average F2: 0.486531986532

Average Precision: 0.3624338524

**Overall: Mean Average F2: 0.2872533973**

**Mean Average Precision: 0.1844737291**

From the above results, the performance of Query Expansion with Lesk Algorithm performs better than Query Expansion without Lesk Algorithm.

However, both MAF2 and MAP of Query Expansion with Lesk Algorithm is still lower than that of the Baseline. But it is hard to conclude whether if the performance of query expansion is indeed worse than our baseline system without any query refinement technique due to the lack of actual relevance judgements that we can compare with.

But we will still keep the implementation of Query Expansion within our search engine since it still performs pretty well within the Competition framework. (Better than the provided TF-IDF baseline)

**Pseudo Relevance Feedback (PRF)**

Pseudo Relevance Feedback is something that we did not incorporate in our search engine.

There are 2 main reasons on why we chose not in incorporate PRF into our search engine. The first reason is that our approach to PRF requires us to load a large file of document vectors, and unfortunately, the time it takes for us to unpickle the file takes more than 1 minute on our machine.

The second reason is that we are unsure whether if we can achieve a good performance with PRF. There are several parameters to tune, and after trying out several of them, we are unable to find the optimal set of parameters to improve the performance of our search engine.

For example, one of the parameters that we are unable to accurately determine is how many documents should we retrieve first and assume it as relevant. From our experience while designing the search engine, we would sometime need to retrieve a large number of documents before we retrieve our first relevant document. PRF is heavily dependent on the search engine being able to retrieve a relevant document early, and in some cases we are unable to do so.

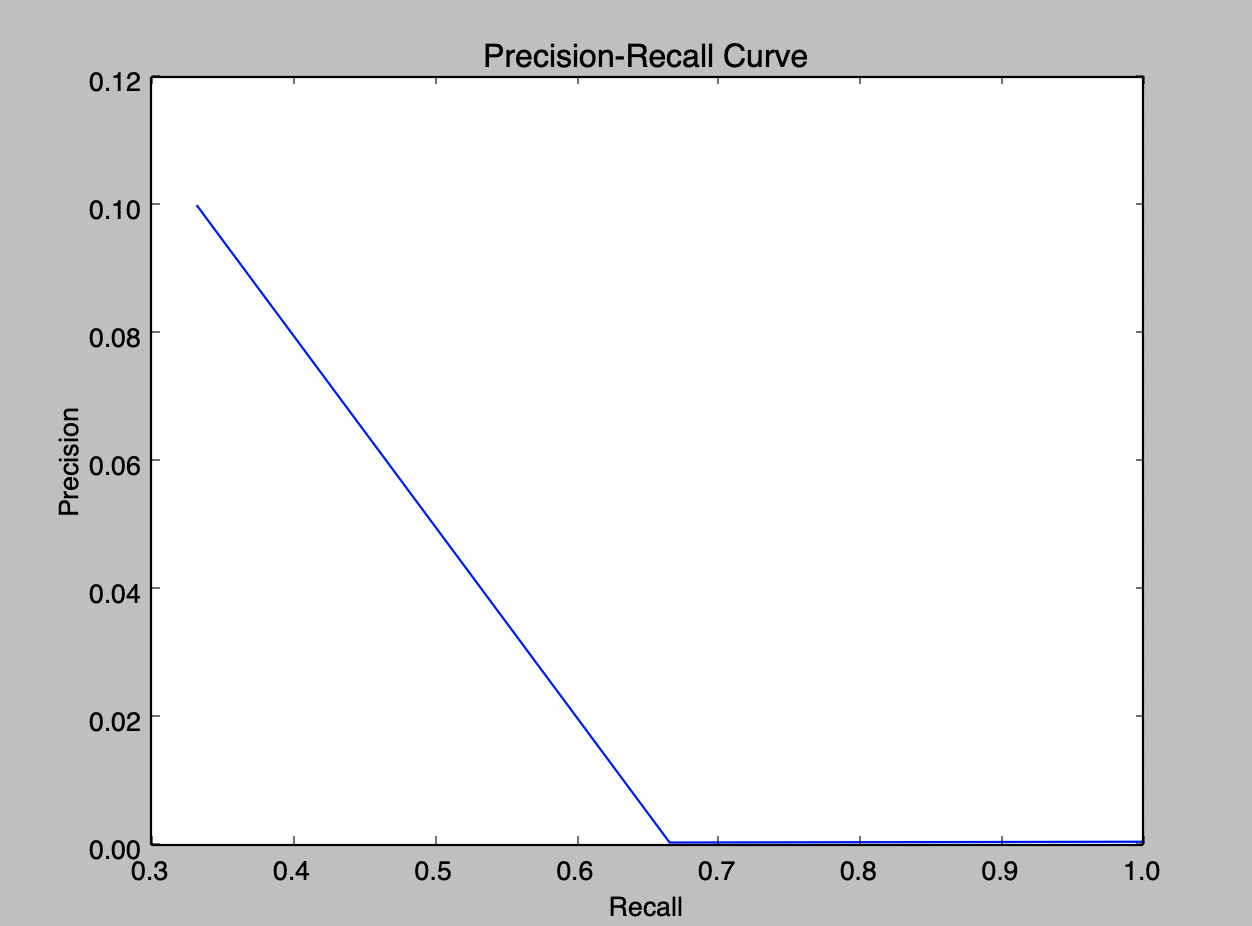
However, we will still be discussing some of the results of PRF below.

One thing to take note is that our implementation of PRF is built upon our search engine with Query Expansion implemented. So we will be making our comparisons with that.

To perform PRF, we utilize the Rocchio Formula and we set our alpha to 0.75 and beta to 0.25 and the following are some of our results. We will not be considering negative feedback without our Rocchio Formula

We will be performing our expansion by assuming that the top 10 retrieved documents are relevant.

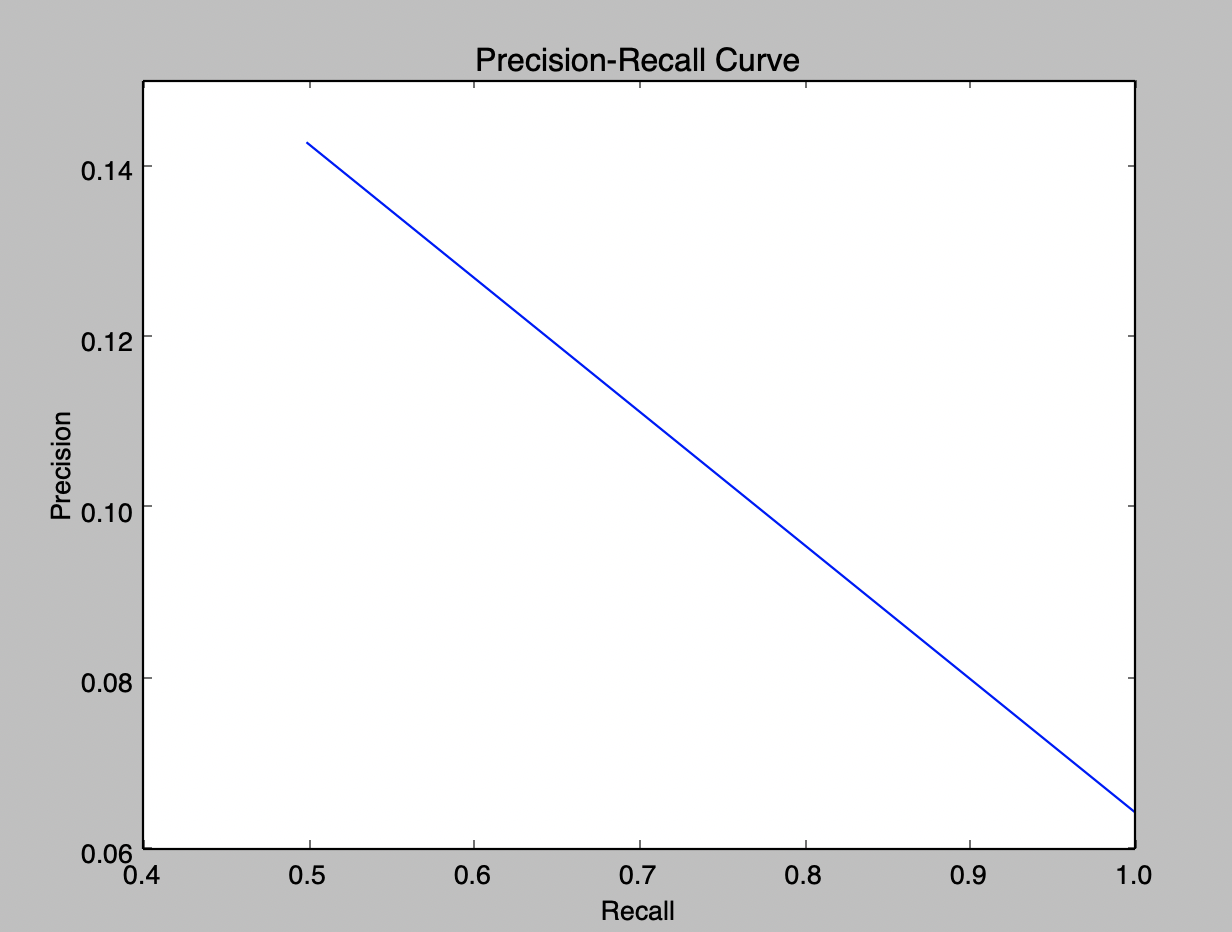
Query: “quiet phone call”



Average F2: 0.0686737867134

Average Precision: 0.0613356*2091*

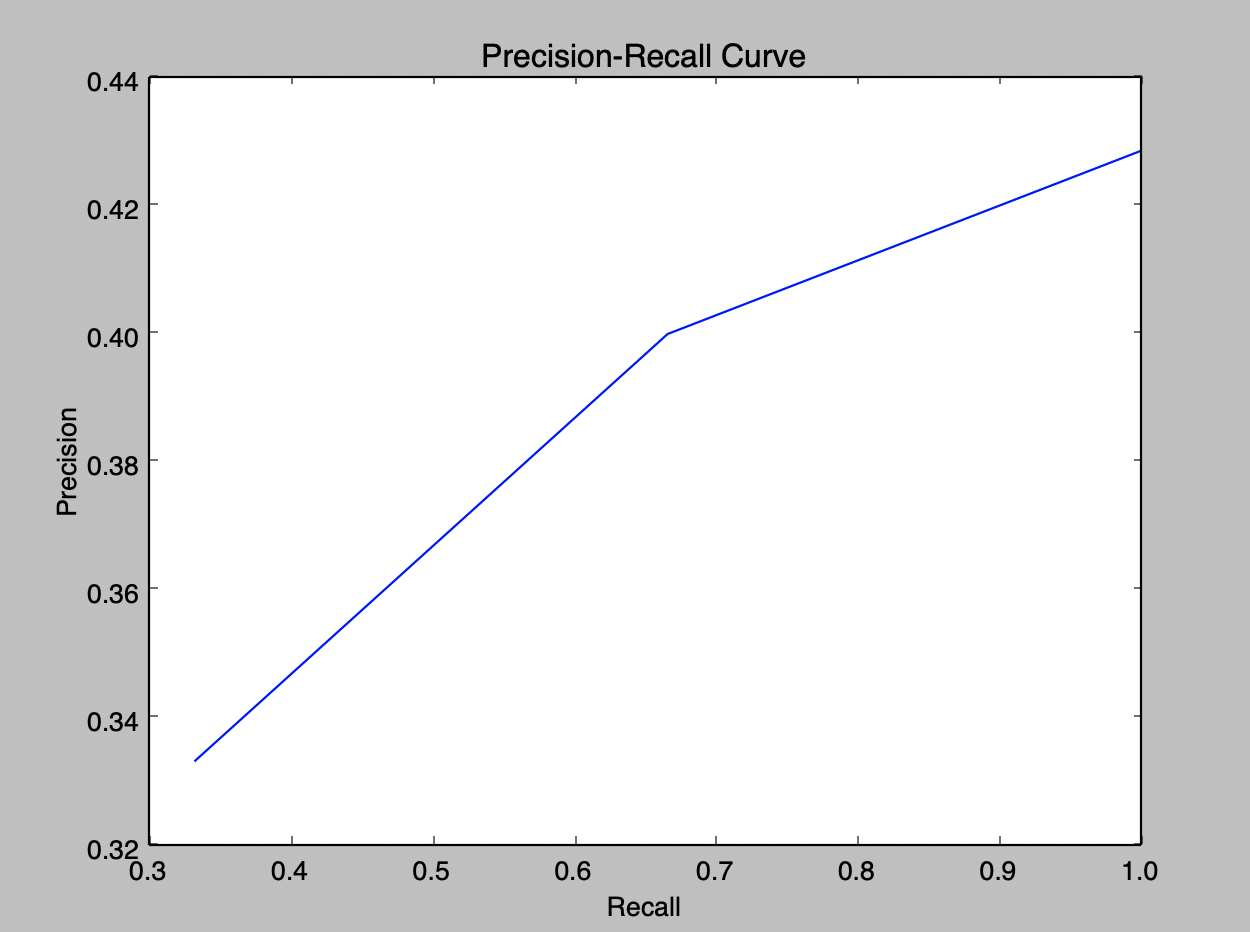
Query: “good grades exchange scandal”



Average F2: 0.269010043042

Average Precision: 0.1232718894

Query: "fertility treatment" AND damages



Average F2: 0.486531986532

Average Precision: 0.3624338623

**Overall: Mean Average F2: 0.2747386052**

**Mean Average Precision: 0.1823471242**

From the results, the implementation of PRF over a search engine with Query Expansion performs slightly worser. However, as mentioned previously, the problem might be due to how we chose to tune the parameters. As such we decided not to implement PRF into our search engine