CS6370 - PROJECT REPORT (TEAM-46)[⋆]

Rahul Vippala(ME16B171) and Nagaraj Chandru(CS20M041)

Indian Institute of Technology Madras
`me16b171@smail.iitm.ac.in`
`cs20m041@smail.iitm.ac.in`

Abstract. The main goal of this course project is to build an improved Information Retrieval(IR) system compared to the Vector Space Model(VSM) built as part of the course assignment. This report starts out with a description the fore-mentioned VSM model which is used as a baseline, and then we address few of the limitations of baseline model using Latent Semantic Analysis(LSA) and ngram(bigram, trigram) indexing of documents instead of unigrams. Finally, we build ngram Language model for query auto-completion and spell checking queries.

Keywords: IR · VSM · LSA · ngram Language Model.

1 Introduction

Vector Space Model : A Vector Space Model represents documents and queries as vectors in the space of terms. In a simple model, weights of these vectors would be product of term frequencies(TF), inverse document frequencies(IDF). And at the time of retrieval, we compute the cosine similarity between the query vector and document vectors and retrieve the documents in decreasing order of similarity.

ngram indexing of documents We use upto ngrams(unigrams, bigrams,..,ngrams) of terms to index documents and queries, instead of indexing with unigrams to retain the some information of the order of occurrence of terms.

Latent Semantic Analysis(LSA) LSA is a method for automatic indexing and retrieval. This approach takes advantage of implicit higher-order structure in the association of terms with documents ("semantic structure") in order to improve the detection of relevant documents on the basis of terms found in queries using a technique called Singular Value Decomposition.

Query Auto-Completion By using the text entered by the user, query auto-completion algorithm predicts the next m-words with the help of a ngram Language model.

Spell Corrector The text entered by the user would be checked for spelling errors and corrected using a noisy channel model.

2 Vector Space Model

In this section, we explain the methodology of Vector Space model that we followed to build VSM-base1 (same as assignment). Another Vector space model is built with more preprocessing that **VSM-base1** called **VSM-base2**. We then present the results and discuss the limitations of Vector space models.

[⋆] Guided by Prof. Sutanu Chakraborty

2.1 Methodology

For VSM-base1 model

- Data Preprocessing:
  - Sentence segmentation: segmenting each document into sentences using punkt sentence tokenizer.
  - Tokenisation: Splitting each sentences into words using PennTreeBank tokenizer.
  - Lemmatization: Reducing the word tokens into their lemma using WordNet Lemmatizer.
  - Stopword Removal: Removing words that occur very frequently in all documents like 'a', 'the', 'is' etc.
- Indexing: Create a lookup table for terms and their frequencies in each document.
- Ranking and Evaluation:
  - Compute the TF-IDF matrix by using the lookup table,
  - Similarly do the same for queries as well.
  - Now, compute the cosine similarity between all pairs and rank the documents for each query in descending order of similarity.
  - Once the ranking is done, we can compute different evaluation metrics like Precision@k, Recall@k, F-Score@k, MAP@k and nDCG@k.

For VSM-base2 model We do everything similar to VSM-base1 model and also add one additional preprocessing step of *text cleaning*, in this, after tokenisation of sentences, we convert all letters to lower case and delete all characters other than alphabets.

2.2 Results

	BASE-1					BASE-2				
k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k
1	0.636	0.109	0.179	0.636	0.511	0.667	0.114	0.187	0.667	0.526
2	0.544	0.18	0.254	0.684	0.475	0.544	0.182	0.256	0.704	0.479
3	0.477	0.225	0.285	0.691	0.451	0.481	0.229	0.288	0.714	0.457
4	0.421	0.258	0.296	0.69	0.435	0.428	0.265	0.303	0.703	0.445
5	0.38	0.285	0.301	0.681	0.428	0.393	0.297	0.313	0.69	0.441
6	0.353	0.315	0.308	0.672	0.431	0.361	0.322	0.315	0.685	0.441
7	0.331	0.34	0.31	0.665	0.435	0.342	0.35	0.32	0.675	0.446
8	0.316	0.364	0.312	0.65	0.443	0.319	0.369	0.317	0.667	0.449
9	0.297	0.381	0.309	0.646	0.446	0.298	0.385	0.311	0.655	0.452
10	0.277	0.393	0.301	0.635	0.448	0.281	0.401	0.306	0.642	0.456

Table 1. Evaluation metric values for VSM base-1 and base-2 models

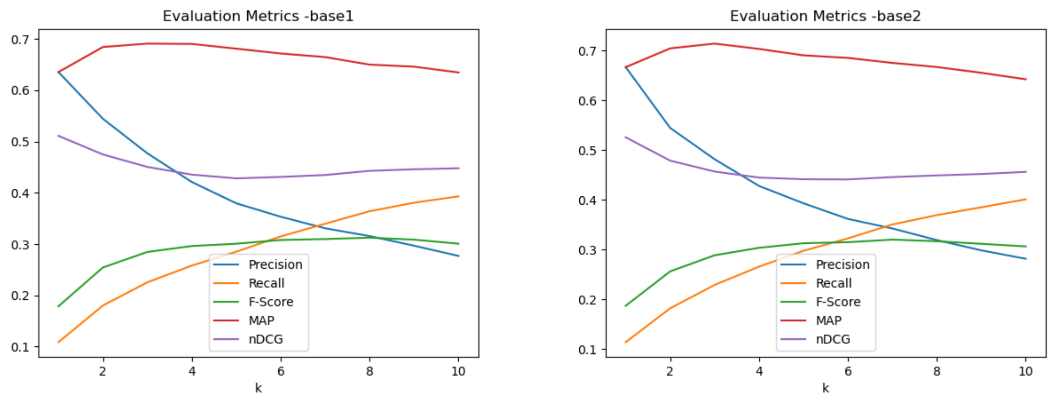

Fig. 1. Evaluation metric plots for VSM base-1 and base-2 models

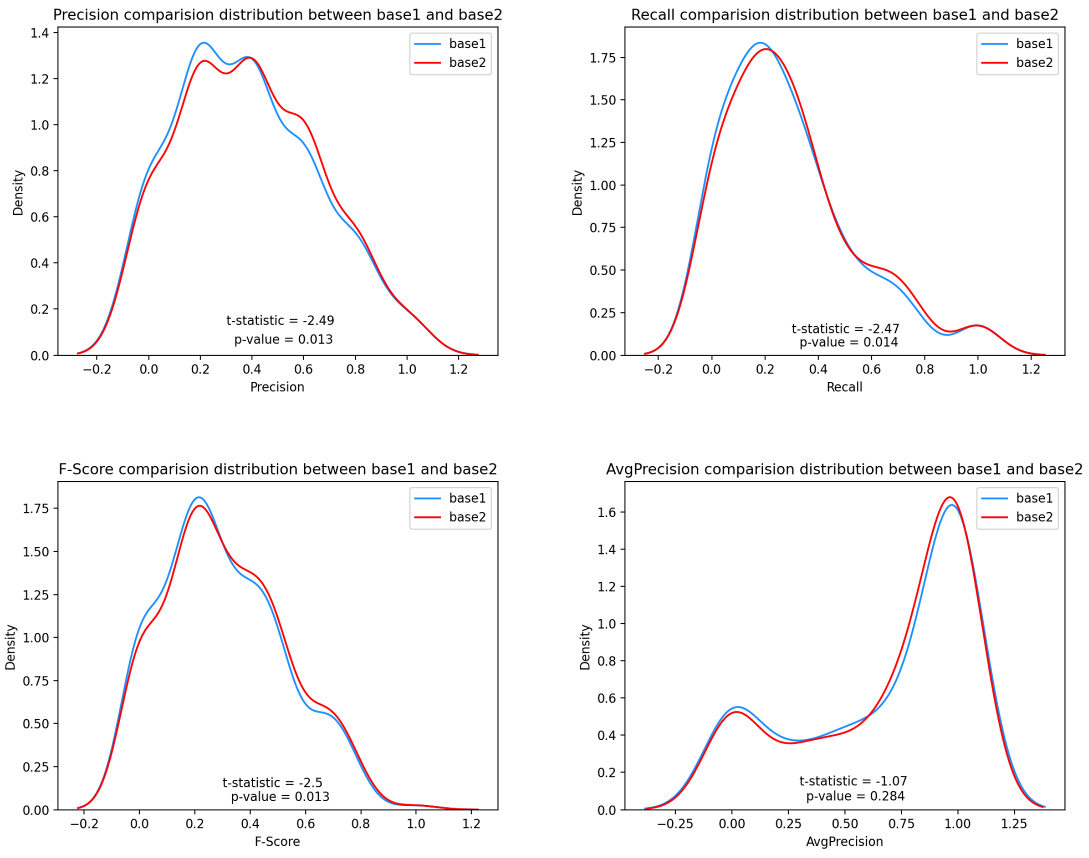

Fig. 2. Query score distribution plots for different metrics for VSM base-1 and base-2 models

2.3 Hypothesis Testing

Null Hypothesis VSM-base1 and VSM-base2 perform similarly in terms of nDCG metric.
Alternate Hypothesis VSM-base1 and VSM-base2 doesn't perform similarly in terms of nDCG metric.
Result We apply the paired two tail t-test on the nDCG@5 scores for all queries from both models to evaluate our hypothesis. We got t-statistic = -2.64 and a p-value of 0.009. Hence we reject the Null Hypothesis and accept the Alternate Hypothesis as the p-value is less than 0.05. We can say that VSM-base2 is performing not similar to VSM-base1 and from all the plots we can see that VSM-base2 scores are better than VSM-base1. Hence, performance VSM-base2 is better than VSM-base1.

2.4 Limitations

- The order in which the terms appear in the document is lost in the vector space representation.
- Vector Space Model assumes orthogonality of terms which is usually not true.
- Documents with similar content but different vocabularies may result in a poor similarity scores
- Documents are sparsely represented and hence poor similarity scores

3 ngram indexing of documents

By using ngram indexing for documents and queries, we can now preserve some sense of order of terms occurrence in the document and hence addressing one of the limitation of the baseline model. However, using just an ngram index will make the document representation even sparser and we might miss out on important discriminative term. Hence we will represent the document using upto ngrams(i.e. unigrams + bigrams+... ngrams). In this project, we are only going try till trigrams as this is not memory efficient.

3.1 Methodolgy

After the data preprocessing of the documents as in VSM, instead of just indexing unigrams, we will now index upto ngrams depending on the model and then follow the ranking and evaluation process similar to VSM.
Here we will build 2 models namely **bigram** and **trigram** and present its results.

3.2 Results

	bigram model					BASE-2				
k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k
1	0.658	0.114	0.187	0.658	0.551	0.667	0.114	0.187	0.667	0.526
2	0.571	0.189	0.268	0.727	0.513	0.544	0.182	0.256	0.704	0.479
3	0.499	0.237	0.299	0.728	0.476	0.481	0.229	0.288	0.714	0.457
4	0.441	0.273	0.313	0.725	0.464	0.428	0.265	0.303	0.703	0.445
5	0.405	0.307	0.323	0.707	0.461	0.393	0.297	0.313	0.69	0.441
6	0.367	0.33	0.321	0.7	0.457	0.361	0.322	0.315	0.685	0.441
7	0.345	0.355	0.324	0.686	0.461	0.342	0.35	0.32	0.675	0.446
8	0.32	0.376	0.319	0.673	0.465	0.319	0.369	0.317	0.667	0.449
9	0.294	0.388	0.31	0.668	0.466	0.298	0.385	0.311	0.655	0.452
10	0.277	0.401	0.304	0.664	0.468	0.281	0.401	0.306	0.642	0.456

Table 2. Evaluation metric values for LSA410 and VSM base-2 models

	trigram model					BASE-2				
k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k
1	0.64	0.11	0.18	0.64	0.534	0.667	0.114	0.187	0.667	0.526
2	0.584	0.192	0.273	0.722	0.515	0.544	0.182	0.256	0.704	0.479
3	0.498	0.237	0.3	0.722	0.475	0.481	0.229	0.288	0.714	0.457
4	0.439	0.274	0.313	0.714	0.463	0.428	0.265	0.303	0.703	0.445
5	0.389	0.3	0.314	0.704	0.452	0.393	0.297	0.313	0.69	0.441
6	0.358	0.323	0.314	0.697	0.452	0.361	0.322	0.315	0.685	0.441
7	0.335	0.349	0.316	0.683	0.456	0.342	0.35	0.32	0.675	0.446
8	0.311	0.365	0.31	0.674	0.456	0.319	0.369	0.317	0.667	0.449
9	0.294	0.387	0.309	0.659	0.463	0.298	0.385	0.311	0.655	0.452
10	0.276	0.398	0.302	0.653	0.464	0.281	0.401	0.306	0.642	0.456

Table 3. Evaluation metric values for trigram and VSM base-2 models

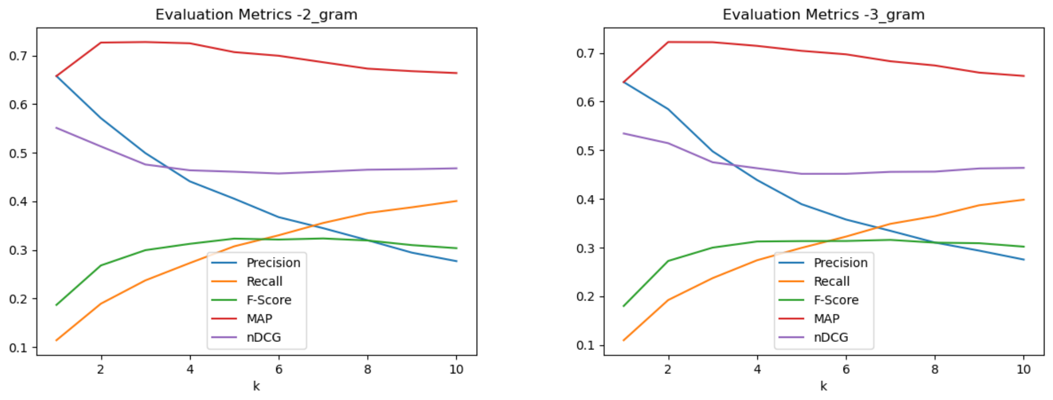

Fig. 3. Evaluation metric plots for bigram and trigram models

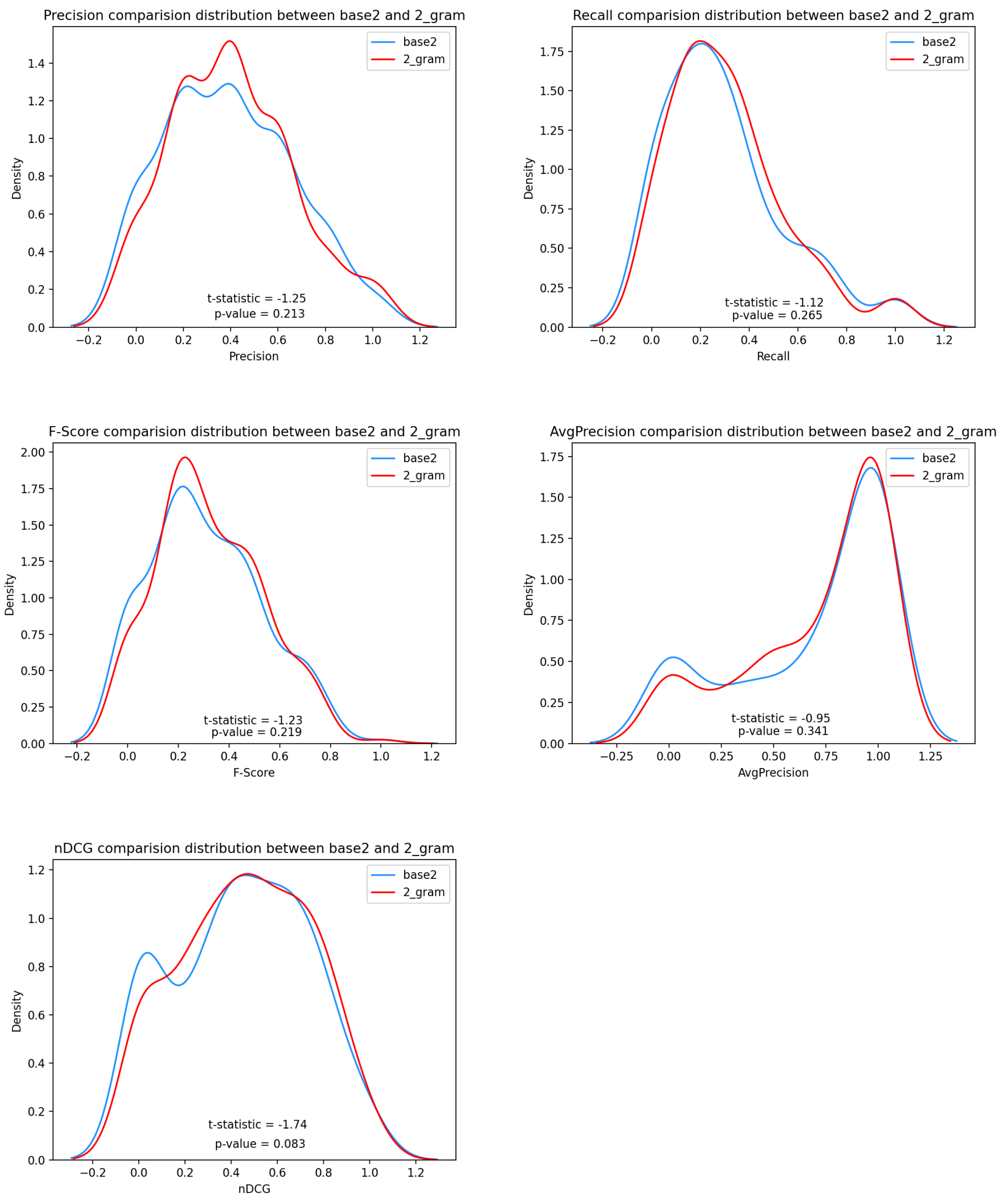

Fig. 4. Query score distribution plots for different metrics for VSM base-2 and bigram models

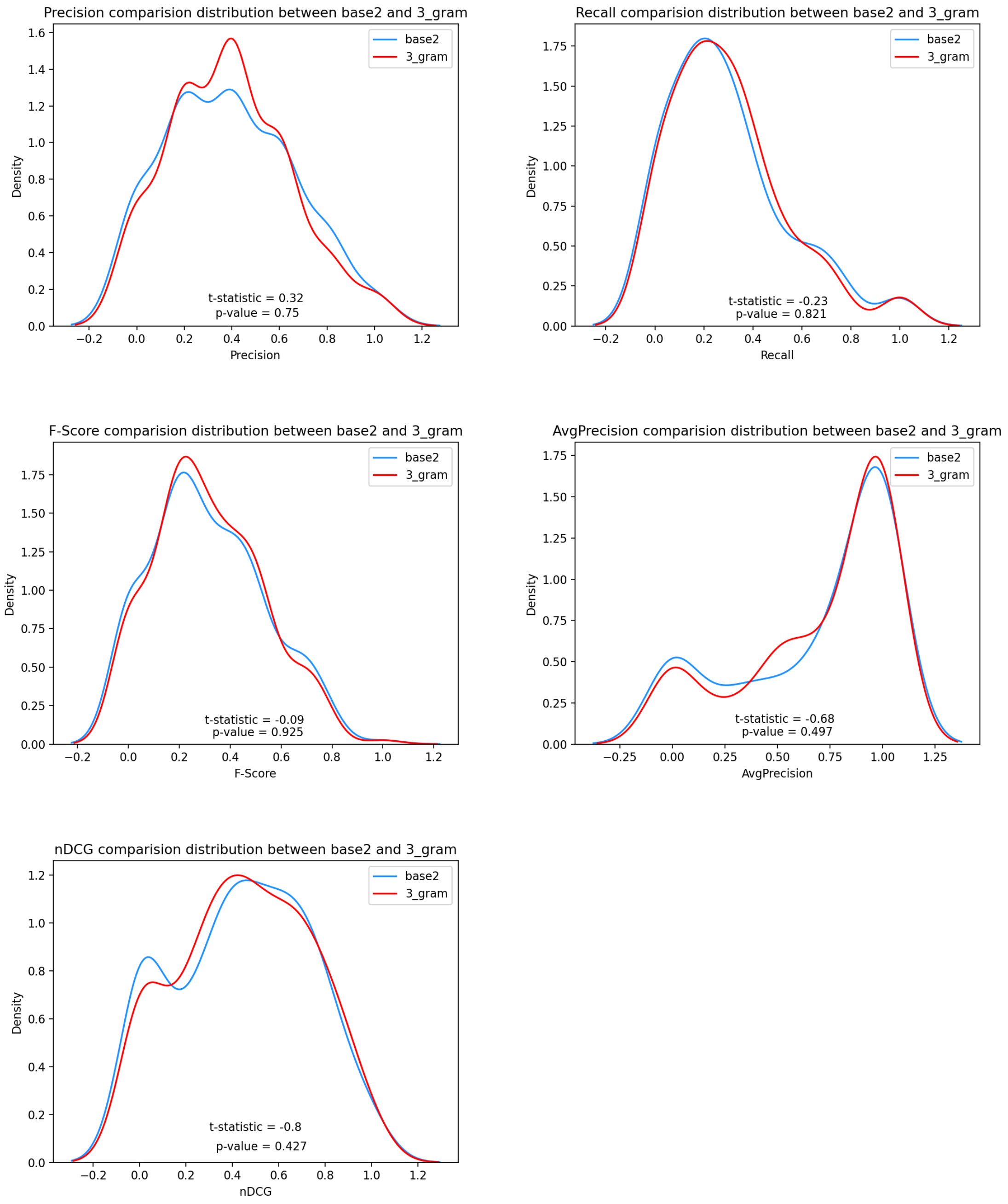

Fig. 5. Query score distribution plots for different metrics for VSM base-2 and trigram models

3.3 Hypothesis Testing

Null Hypothesis bigram model and VSM-base2 perform similarly in terms of nDCG metric.
Alternate Hypothesis bigram and VSM-base2 doesn't perform similarly in terms of nDCG metric.
Result We apply the paired two tail t-test on the nDCG@5 scores for all queries from both models to evaluate our hypothesis. We got t-statistic = -1.74 and a p-value of 0.083. Hence we can't reject the Null Hypothesis and accept the Alternate Hypothesis as the p-value is more than 0.05. Therefore bigram model and VSM-base2 model perform similarly in terms of nDCG@5.

Null Hypothesis trigram model and VSM-base2 perform similarly in terms of nDCG metric.

Alternate Hypothesis trigram and VSM-base2 doesn't perform similarly in terms of nDCG metric.
Result We apply the paired two tail t-test on the nDCG@5 scores for all queries from both models to evaluate our hypothesis. We got t-statistic = -0.8 and a p-value of 0.427. Hence we can't reject the Null Hypothesis and accept the Alternate Hypothesis as the p-value is more than 0.05. Therefore trigram model and VSM-base2 model perform similarly in terms of nDCG@5.

4 Latent Semantic Analysis

LSA assumes that there is some underlying latent semantic structure in the data that is partially obscured by the randomness of word choice with respect to retrieval. Singular Value Decomposition is used to estimate this latent structure, and get rid of the obscuring "noise.". Singular-value decomposition allows the arrangement of the space to reflect the major associative patterns in the data, and ignore the smaller, less important influences. As a result, terms that did not actually appear in a document may still end up close to the document, if that is consistent with the major patterns of association in the data[3]. Hence this addresses limitations 2,3,4 mentioned above.

4.1 Methodology

After computing the TF-IDF matrix for the documents as in VSM or n-gram indexing model, we apply SVD to obtain a k rank approximation of the TF-IDF matrix, where k is a hyper parameter and represents the number of latent features. Now, to compute similarity between documents and queries, we transform both og them into the fitted k-rank latent space and compute cosine similarity and rank based on it just like VSM model.
We will build 2 LSA models, **unigram-LSA** and **bigram-LSA**, based on VSM-base2 and bigram models.

4.2 Hyper parameter tuning: Latent features

To find the best value for k, we use nDCG@5 metric and search for the best k.

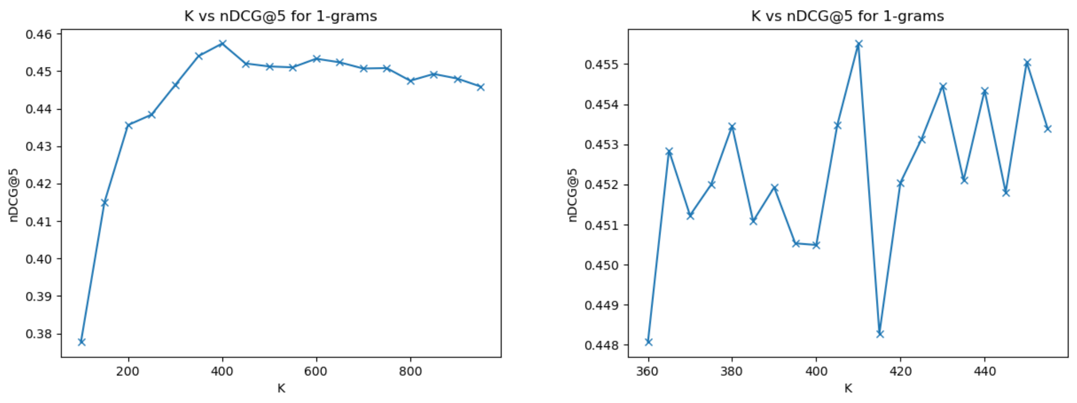

Fig. 6. Latent features 'K' VS nDCG@5 for unigram LSA model

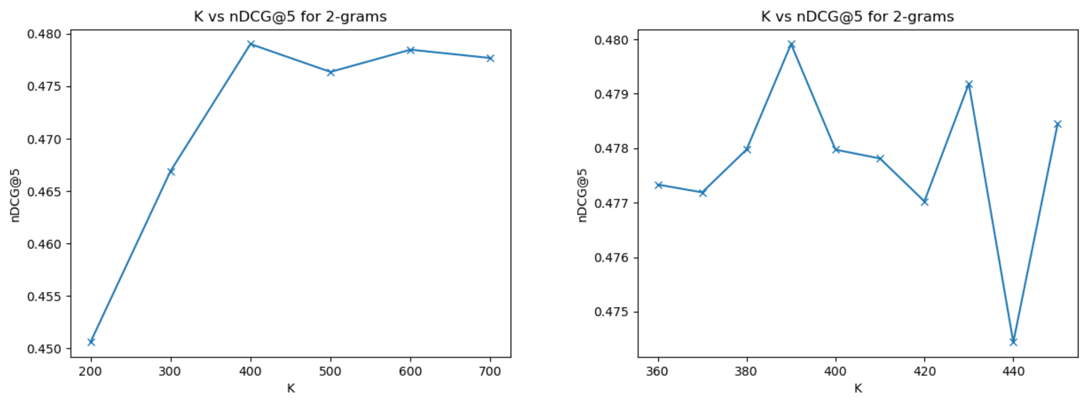

Fig. 7. Latent features 'K' VS nDCG@5 for bigram LSA model

– From the plots, we chose **k=410** unigram-LSA model and **k=390** for bigram-LSA model

4.3 Results

	unigram-LSA-410					BASE-2				
k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k
1	0.644	0.111	0.182	0.644	0.502	0.667	0.114	0.187	0.667	0.526
2	0.576	0.188	0.267	0.709	0.487	0.544	0.182	0.256	0.704	0.479
3	0.511	0.246	0.308	0.713	0.471	0.481	0.229	0.288	0.714	0.457
4	0.448	0.281	0.32	0.711	0.456	0.428	0.265	0.303	0.703	0.445
5	0.411	0.312	0.328	0.698	0.452	0.393	0.297	0.313	0.69	0.441
6	0.387	0.346	0.337	0.687	0.458	0.361	0.322	0.315	0.685	0.441
7	0.363	0.371	0.338	0.678	0.462	0.342	0.35	0.32	0.675	0.446
8	0.342	0.396	0.339	0.67	0.468	0.319	0.369	0.317	0.667	0.449
9	0.318	0.407	0.33	0.664	0.469	0.298	0.385	0.311	0.655	0.452
10	0.296	0.42	0.321	0.658	0.47	0.281	0.401	0.306	0.642	0.456

Table 4. Evaluation metric values for unigram-LSA410 and VSM base-2 models

	bigram-LSA-390					BASE-2				
k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k	Precision@k	Recall@k	F-Score@k	MAP@k	nDCG@k
1	0.689	0.114	0.188	0.689	0.547	0.667	0.114	0.187	0.667	0.526
2	0.589	0.191	0.271	0.724	0.512	0.544	0.182	0.256	0.704	0.479
3	0.516	0.243	0.308	0.731	0.488	0.481	0.229	0.288	0.714	0.457
4	0.477	0.296	0.339	0.721	0.484	0.428	0.265	0.303	0.703	0.445
5	0.435	0.329	0.347	0.713	0.479	0.393	0.297	0.313	0.69	0.441
6	0.401	0.359	0.35	0.706	0.481	0.361	0.322	0.315	0.685	0.441
7	0.374	0.382	0.35	0.698	0.483	0.342	0.35	0.32	0.675	0.446
8	0.349	0.403	0.346	0.688	0.487	0.319	0.369	0.317	0.667	0.449
9	0.328	0.423	0.342	0.679	0.491	0.298	0.385	0.311	0.655	0.452
10	0.307	0.437	0.334	0.673	0.493	0.281	0.401	0.306	0.642	0.456

Table 5. Evaluation metric values for bigram-LSA-390 and VSM base-2 models

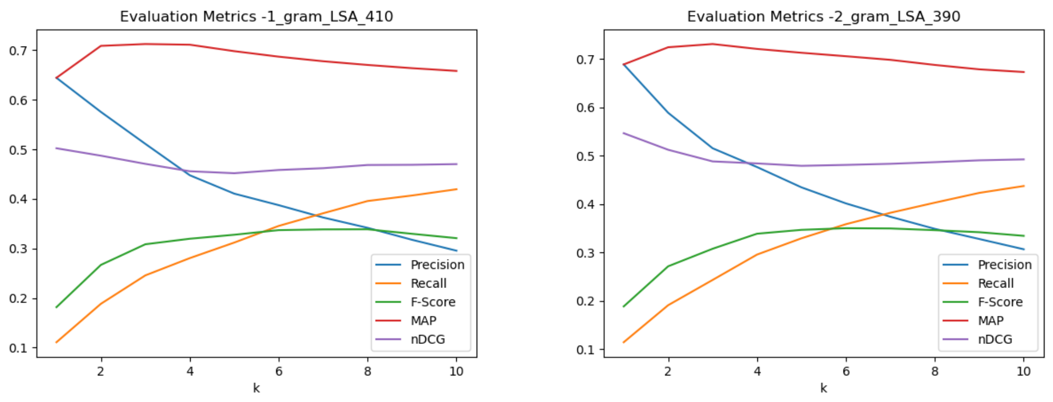

Fig. 8. Evaluation metric plots for unigram-LSA410 and bigram-LSA390 models

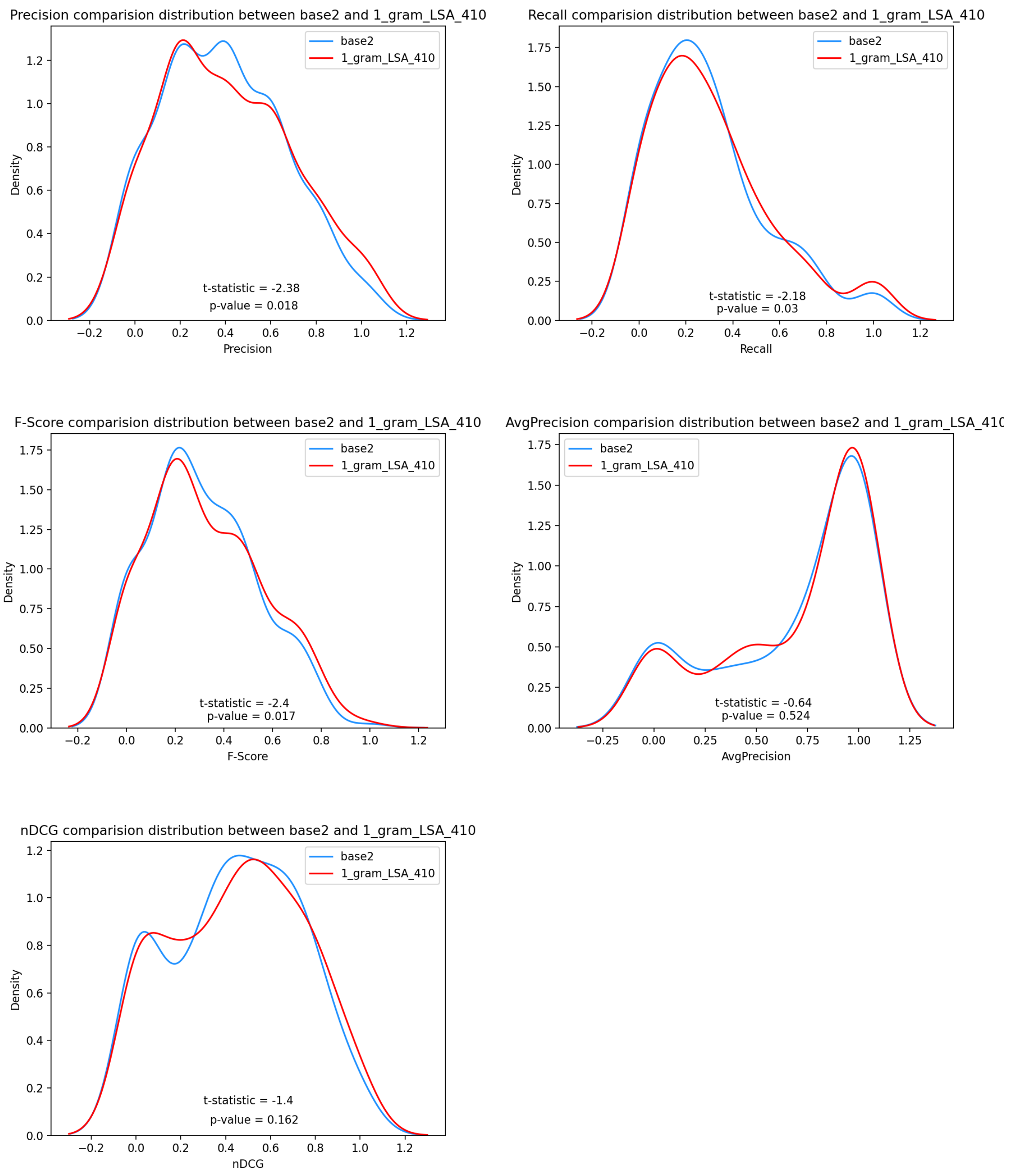

Fig. 9. Query score distribution plots for different metrics for VSM base-2 and unigram-LSA-410 models

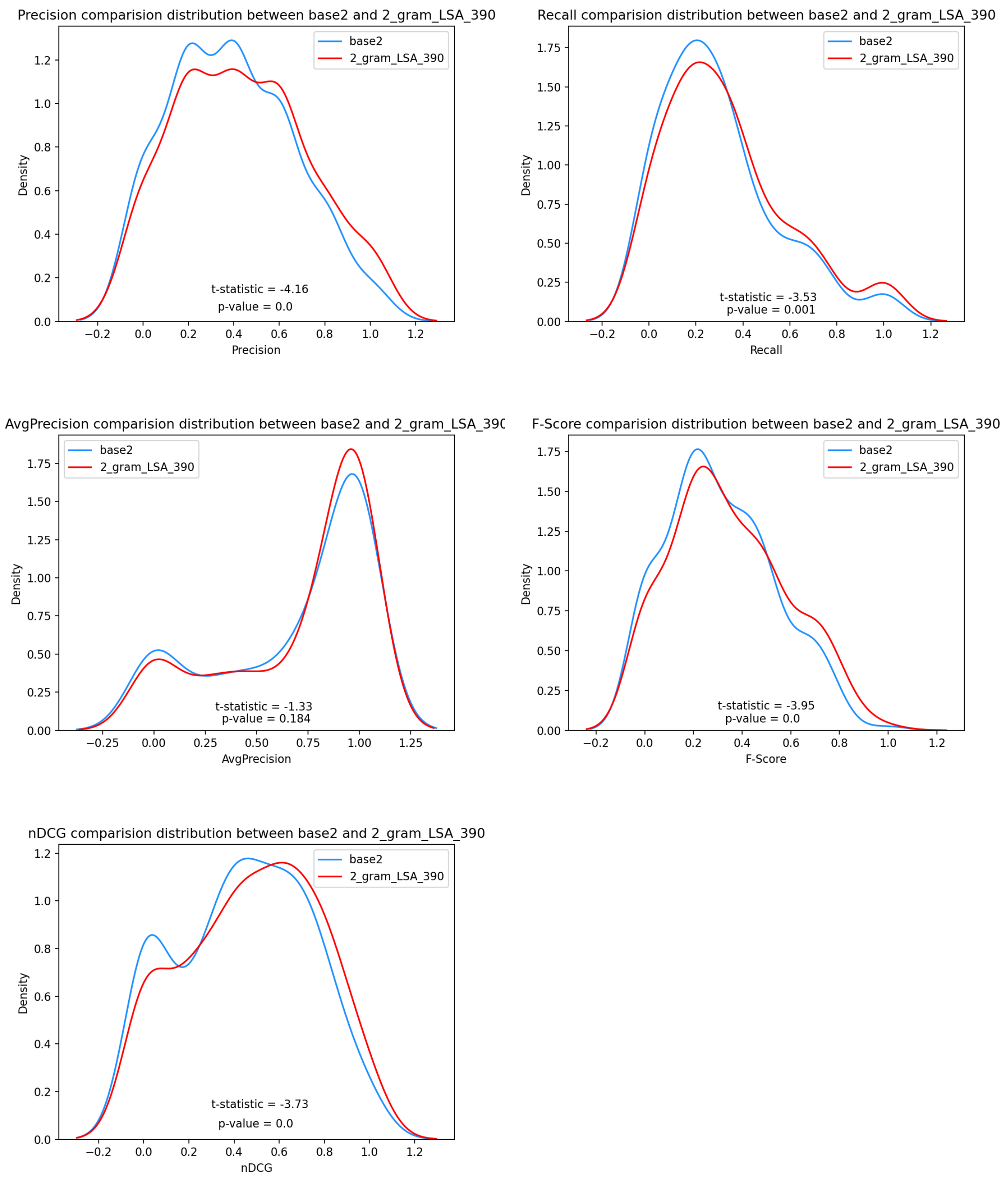

Fig. 10. Query score distribution plots for different metrics for VSM base-2 and bigram-LSA-390 models

4.4 Hypothesis Testing

Null Hypothesis unigram-LSA and VSM-base2 perform similarly in terms of nDCG metric.
Alternate Hypothesis unigram-LSA and VSM-base2 doesn't perform similarly in terms of nDCG metric.
Result We apply the paired two tail t-test on the nDCG@5 scores for all queries from both models to evaluate our hypothesis. We got t-statistic = -1.4 and a p-value of 0.162. Hence we cannot reject the Null Hypothesis and accept the Alternate Hypothesis as the p-value is less than 0.05. Hence, we conclude that unigram-LSA and VSM-base2 models perform similarly in terms of nDCG@5

Null Hypothesis bigram-LSA and VSM-base2 perform similarly in terms of nDCG metric.

Alternate Hypothesis bigram-LSA and VSM-base2 doesn't perform similarly in terms of nDCG metric.
Result We apply the paired two tail t-test on the nDCG@5 scores for all queries from both models to evaluate our hypothesis. We got t-statistic = -3.73 and a p-value of 0.000... Hence we reject the Null Hypothesis and accept the Alternate Hypothesis as the p-value is less than 0.05. Hence, we conclude that bigram-LSA and VSM-base2 models doesn't perform similarly in terms of nDCG@5. So, based on the plots we can infer that bigram-LSA model performs better than VSM-base2 model.

5 ngram Language Model

An ngram model predicts x_i based on $x_{i-(n-1)}, \dots, x_{i-1}$. In probability terms, this is $P(x_i \mid x_{i-(n-1)}, \dots, x_{i-1})$. When used for language modeling, independence assumptions are made so that each word depends only on the last $n-1$ words. This Markov model is used as an approximation of the true underlying language. This assumption is important because it massively simplifies the problem of estimating the language model from data.[2]

5.1 Methodology

- Datapreprocessing: sentence segmentation, tokenization and token cleaning just like VSM-base2 model.
- Indexing of ngram tokens is done and frequency counts is stored. If smoothing method Simple Good Turing is chosen, then a log curve between ngram counts(C) and Number of types with counts-c (N_c) is fitted using Linear regression.
- Probability of an ngram can now be computed based on the index and type of smoothing method. Only add-k Laplace smoothing and Simple Good Turing smoothing methods are implemented.

This Language model can now be used for downstream tasks like auto complete and spell correction.

5.2 Results

Perplexity on cranfield dataset is computed for different configurations of n-gram Language model and presented in table below.

n	Smoothing	Perplexity
1	Laplace-1	209.24
2	Laplace-1	231.86
3	Laplace-1	277.92
1	SGT	209.7
2	SGT	353.134
3	SGT	445.67

Table 6. Model vs Perplexity

5.3 Query Auto Completion

Query Auto completion is useful for positive user experience while using a search engine. And almost all search engines on web have this and are very effective. A good auto completion model requires a good language model.

Qualitative Results for Auto Complete : Here are couple of examples for query auto complete; We used trigram language model with Simple Good turing smoothing for this,

```
python main.py -LM_n 3 -custom -autocomplete -LM_smooth SGT
>Enter query below:
paper on shear buckling
>proposed completed query:- paper on shear buckling of thin cylindrical shells
```

```
python main.py -LM_n 3 -custom -autocomplete -LM_smooth SGT
>Enter query below
what analytical solutions
>proposed completed query:- what analytical solutions are obtained for incompressible
```

5.4 Spell Corrector

We have implemented a noisy channel spell corrector as described in [1]. Details are listed below

1. Made a simplifying assumption that every sentence has only one error.
2. For each word, used current word and words with edit distance 1 that occur in the vocabulary as candidates.
3. Assumed that 1 in 10 words that people write are spelled wrong and used a channel probability distribution with $\alpha = 0.9$:

$$p(x|w) = \begin{cases} \alpha & x = w \\ \frac{1-\alpha}{|Candidates(x)|} & w \in Candidates(x) \\ 0 & else \end{cases}$$

4. Used the Ngram Model built above to compute the probabilities of sentence ($P(W)$)
5. Finally, returned the candidate sentence with highest probability ($P(X|W).P(W)$).

Qualitative Results for Spell corrector : Here are couple of examples for query spell correction; We used trigram language model with Simple Good turing smoothing for this,

```
python main.py -LM_n 3 -custom -spell_check -LM_smooth SGT
>Enter query below:
peper on shear buckling
>proposed corrected query:-  paper on shear buckling
```

```
python main.py -LM_n 3 -custom -spell_check -LM_smooth SGT
>Enter query below
hat analytical solutions
>proposed corrected query:-  what analytical solutions
```

6 Conclusion

– We have built a total of 6 models VSM-base1, VSM-base2, bigram, trigram, unigram-LSA and bigram-LSA for the IR system and by hypothesis testing we concluded that bigram-LSA performed the best in terms of nDCG@5 which supports our proposal.
– We built a Language model using the same Cranfield dataset and used it for query auto complete and spell correction. Although the qualitative results shown seem decent, the perplexity of the Language model on the same dataset surprising increased with increase in n. This could be because of the small training set and the blunt smoothing methods.

References

1. Chapter B.2, Speech and Language Processing. Daniel Jurafsky & James H. Martin.
2. n-gram, Wikipedia
3. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K. and Harshman, R. (1990), Indexing by latent semantic analysis. J. Am. Soc. Inf. Sci., 41: 391-407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6¡391::AID-ASI1¿3.0.CO;2-9