Automatic Spoiler Tagging

- Contributors:

- Rushikesh Tammewar CS17BTECH11041
- Atharva Sarage CS17BTECH11005
- Niraj Kamble CS17BTECH11024
- Ajinkya Bokade CS17BTECH11001
- Shivashish Suman CS17BTECH11037

ABSTRACT:

- We study automatic methods to detect comments and reviews that contain spoilers and apply them to reviews from the IMDB website.
- A spoiler is a comment that,when disclosed, would ruin a surprise or reveal an important plot detail.
- Experimental results demonstrate the effectiveness of our technique over four movie-comment datasets of different scales.

- We have used Four Models
  - 1 . Plain LDA
  - 2. LDA with synonyms
  - 3. Simple LPU
  - 4. LPU with LDA and synonyms

Data Set :

- We have chosen four movies, and collected each movies comments from Kaggle IMDB spoiler DataSet as well as synopsis from IMDB.
- For LDA purpose we do not need labeled comment but for LPU model we have to choose positive comments as per IMDB tag.
- For testing purpose we have used spoiler or non spoiler tag from dataset.
- For LDA model we split data in 8:2 for training and testing.

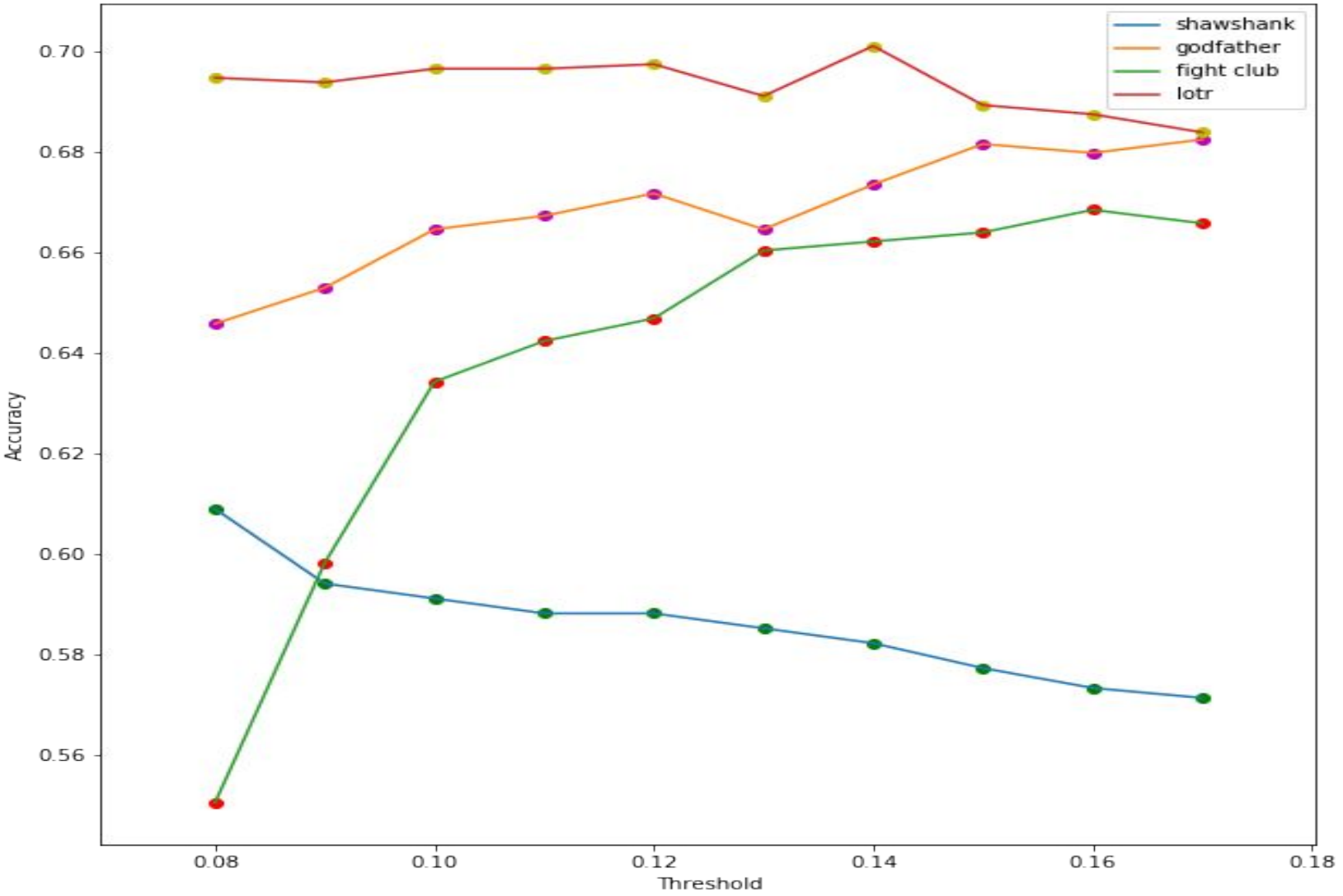
Plain LDA :

- First we have formed unlabeled training corpus to estimate LDA model using Gensim library.
- We have done simple preprocessing on data such as removing punctuation mark, make text lowercase, remove word containing digits.
- Then we have created count matrix of data and formed dictionary and applied LdaModel on corpus.
- LDA gives specified number of topics with word probability distribution of each topic.

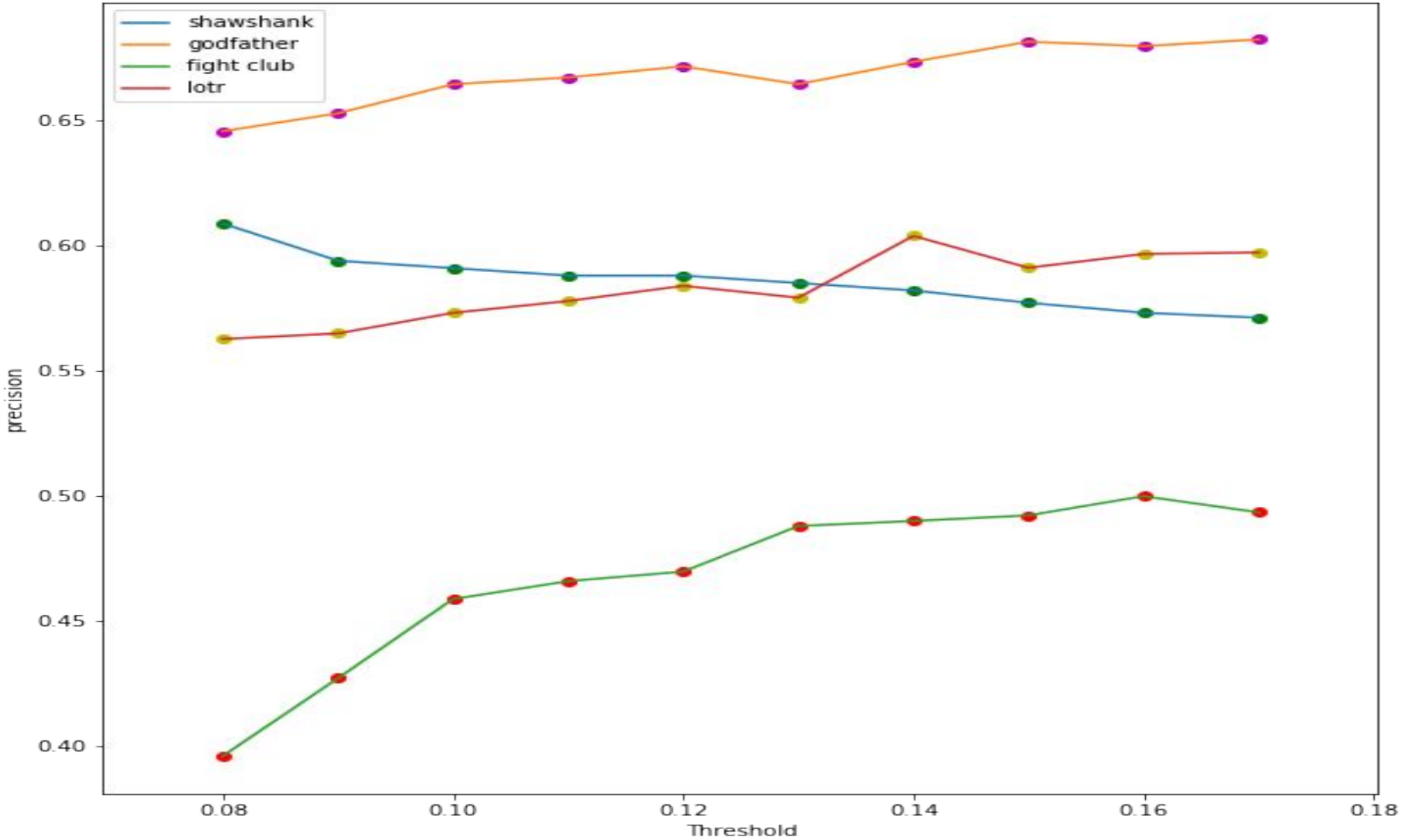
- In the same way we have applied LdaModel to synopsis and we get probabilistic distribution of synopsis.
- For each word of new comment we have assigned, maximum probability that word have in different topics in lda distribution.
- Thus we get a probabilistic distribution of new comment, this probabilistic distribution denotes how much similar added comment is with rest of comments.
- As IMDB synopsis contain detail description of movie and plot summary, thus we assume that more the similarity with synpsis greater is the chance that comment is spoiler.

- Now we have two probabilistic distribution and we have to find out similarity of these two distributions.
- Here we can not use KL divergence as sum of probabilities of word in new comments distribution is not equal to one. Best way to find similarity here is cosine similarity.
- We have to classify comment as spoiler or non spoiler on the basis of resultant similarity.
- We have to choose appropriate threshold to classify comment as spoiler or non spoiler.
- Threshold is hyperparameter of this model.

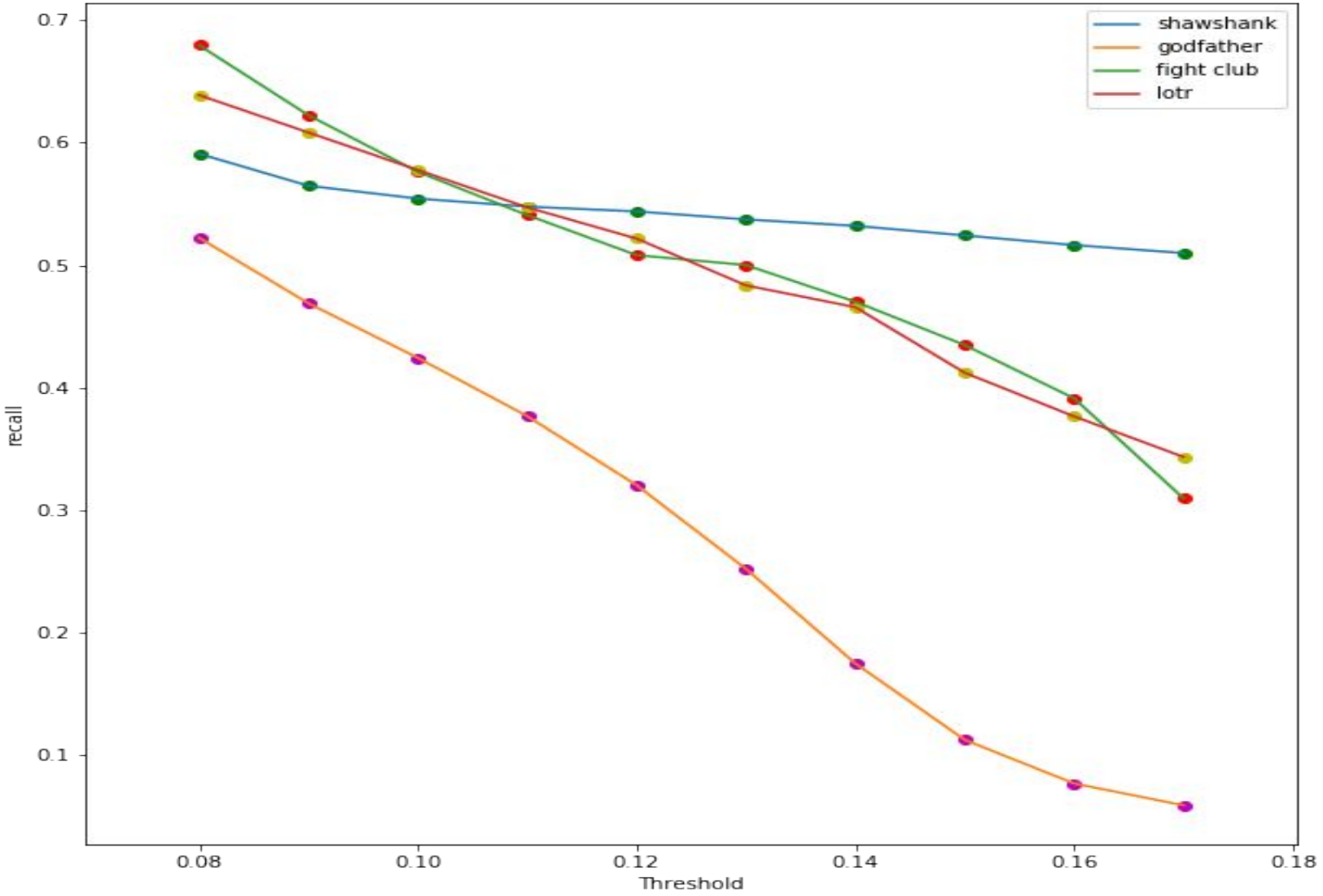
Accuracy vs Threshold



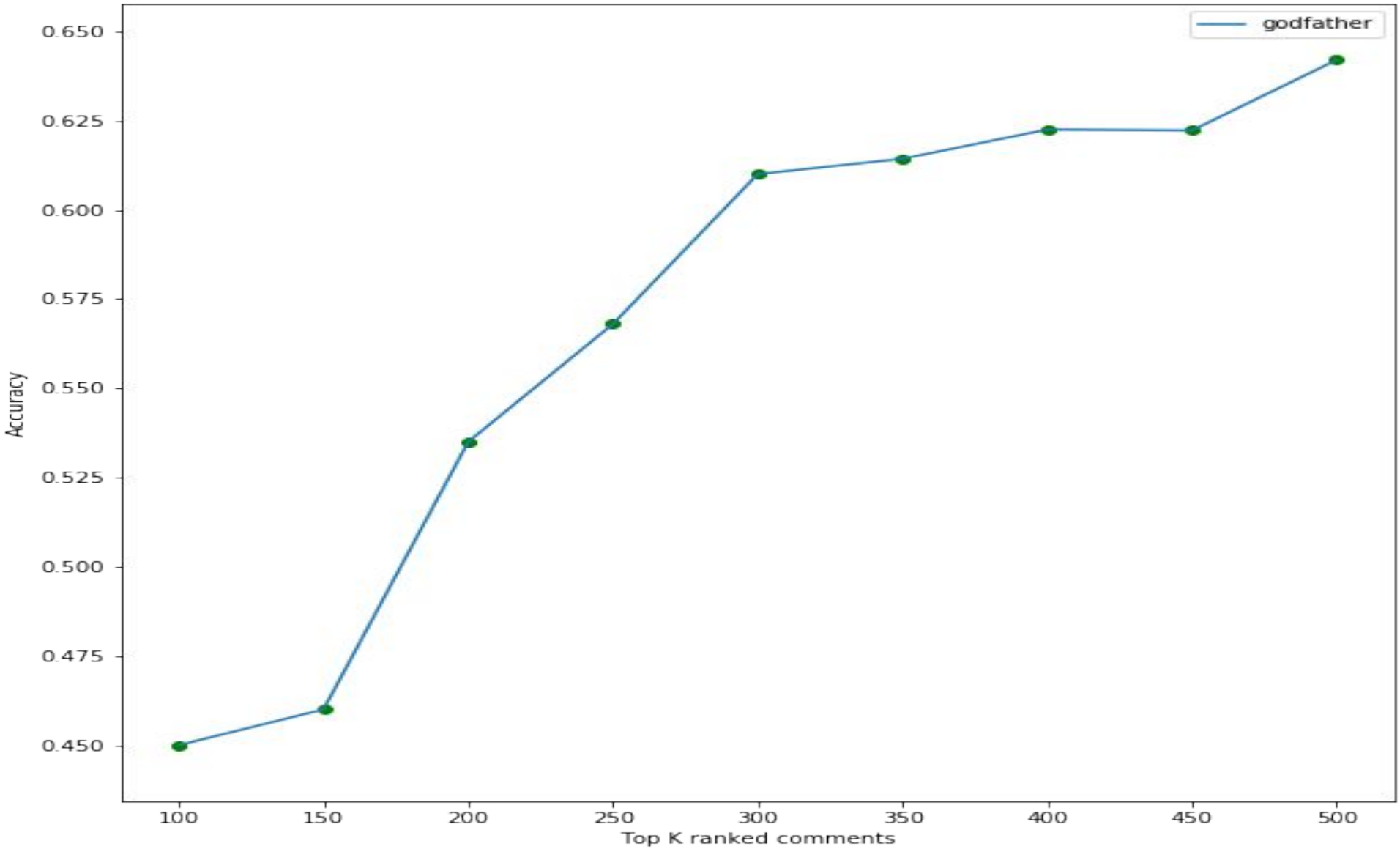
Precision vs Threshold



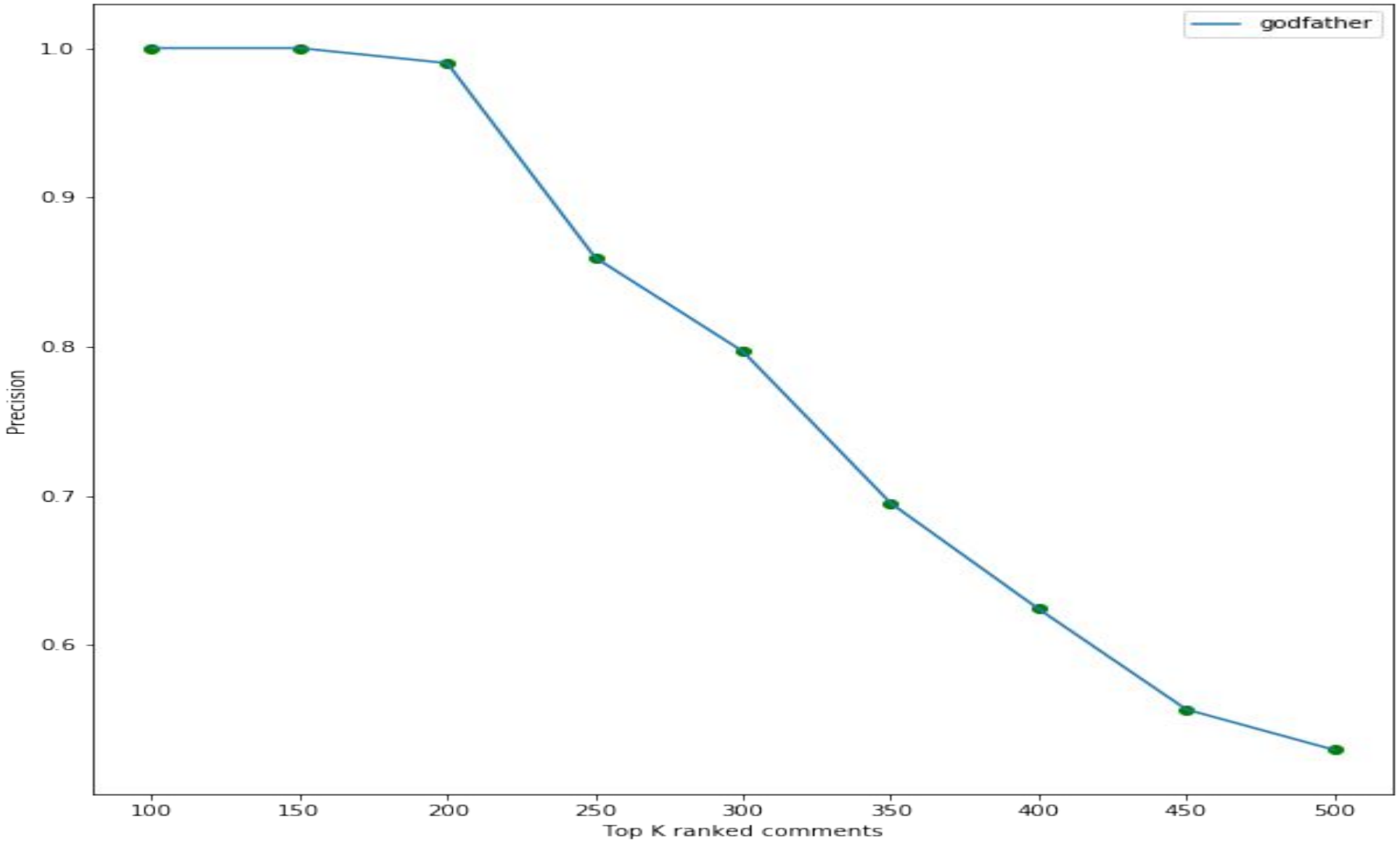
Recall vs Threshold



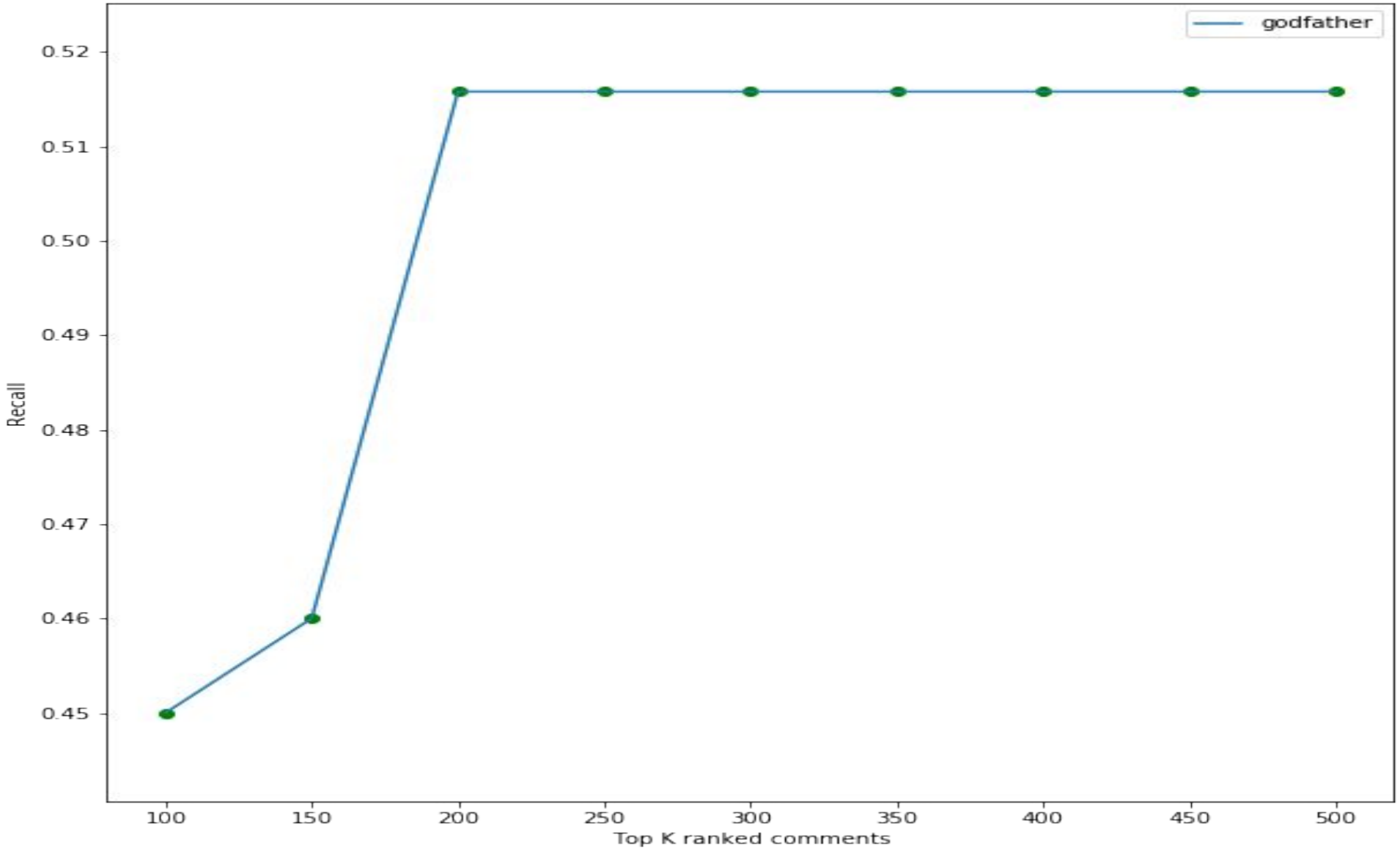
Accuracy vs Top k comments:



Precision vs Top K comments:



Recall vs Top K comments:



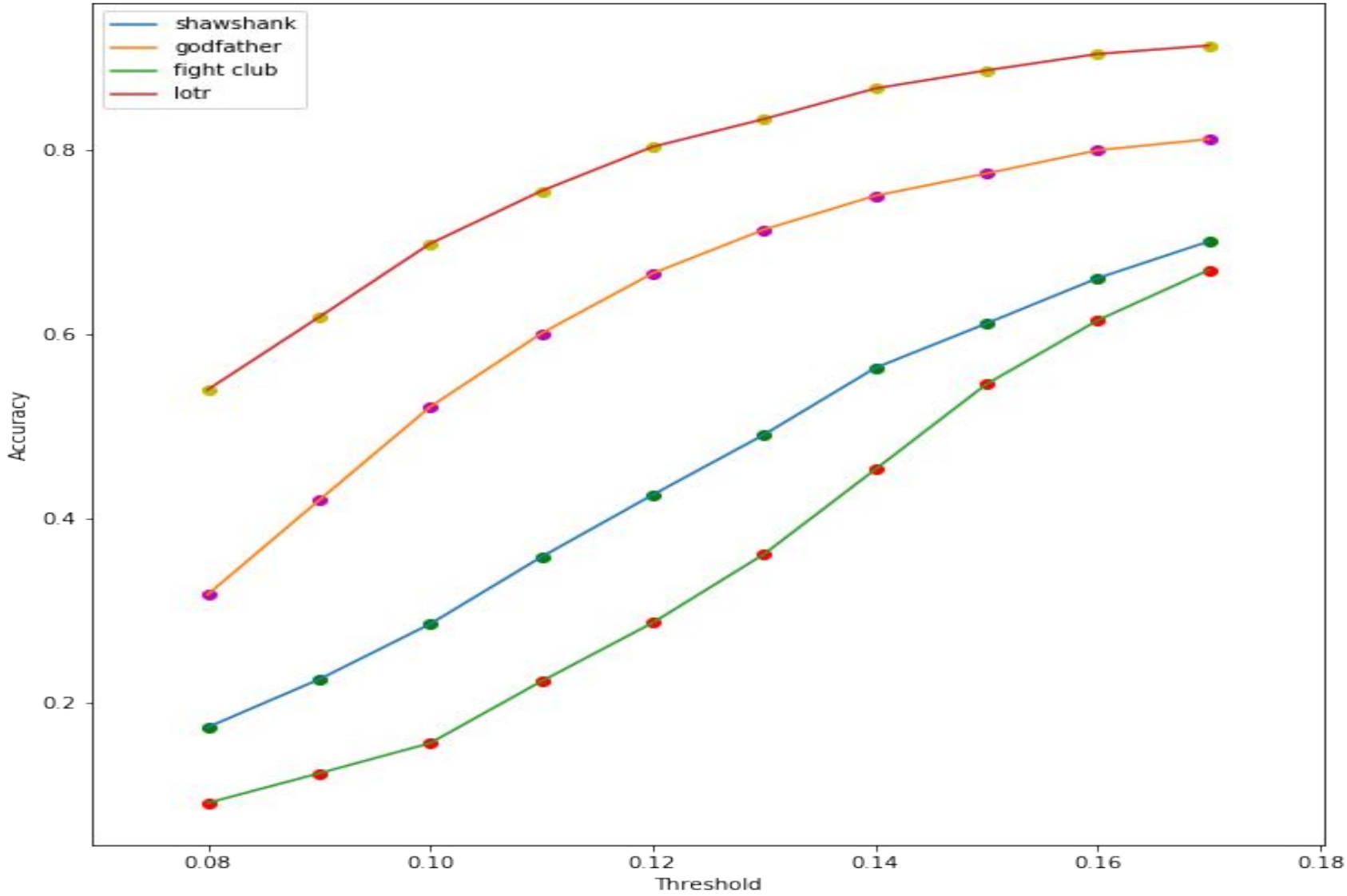
LDA with Synonyms

- This model is built upon the simple LDA with added support of matching synonyms of words present in a new comment.
- Preprocessing on dataset has been done similar to previous models.
- “Wordnet” in NLTK library is used to find the set of synonyms for every word (synsets) in a new comment.
- We check if the word or its synomym is present in the Topic distribution given by the LDA on corpus of training comments.

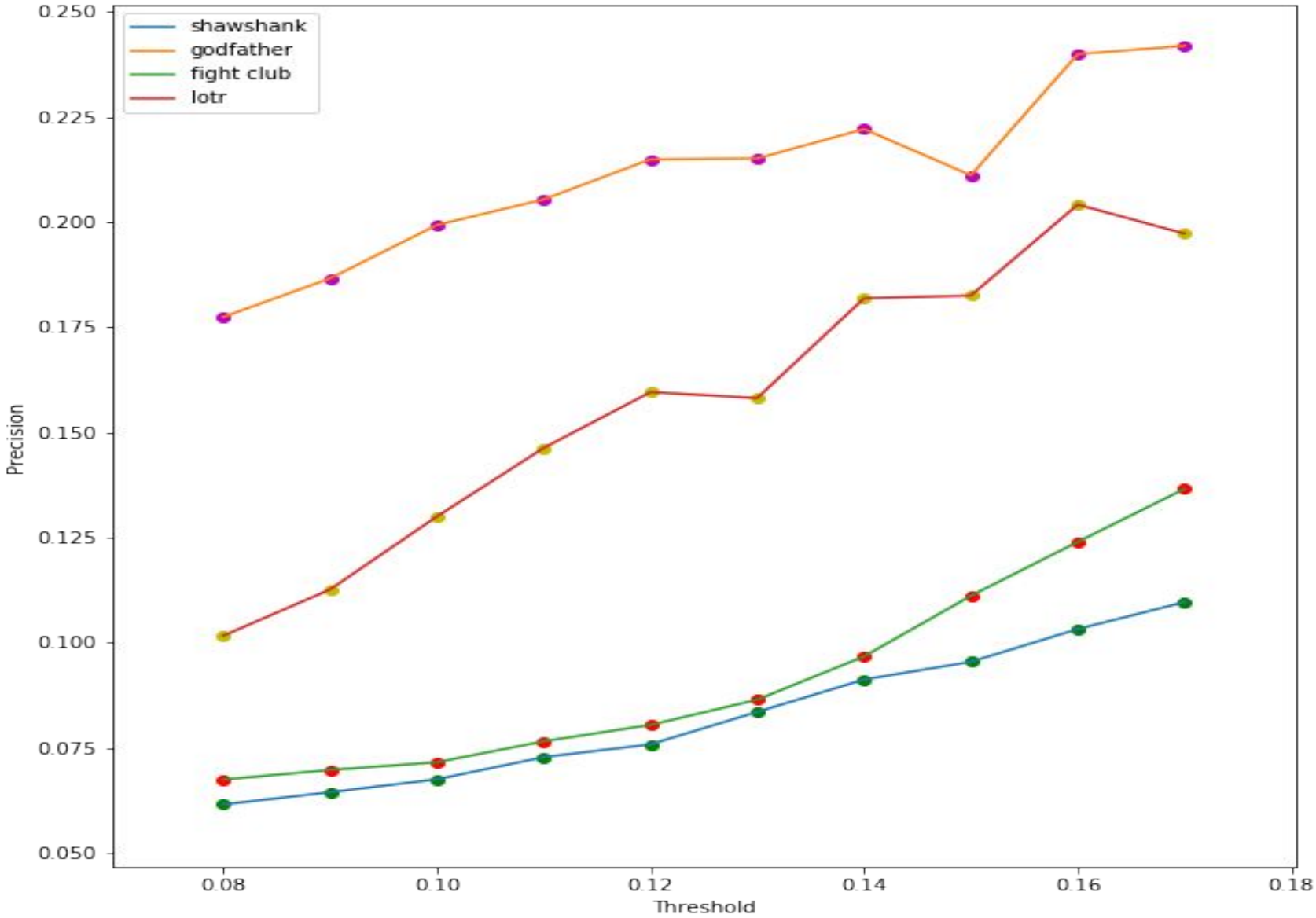
- In a new comment for synonym of each word from synsets of that word, we store the maximum probability of that word from comments' LDA model and from synopsis LDA model.
- We then check if the cosine similarity on word weights is greater than a threshold and then classify accordingly .

- We have set the total number of topics as “100” to increase the no of words in distribution hence we can match more words while obtaining similarity increasing the chances for a better prediction.
- By varying the threshold we have compared the accuracy, precision, recall for the comments of 4 different movies.

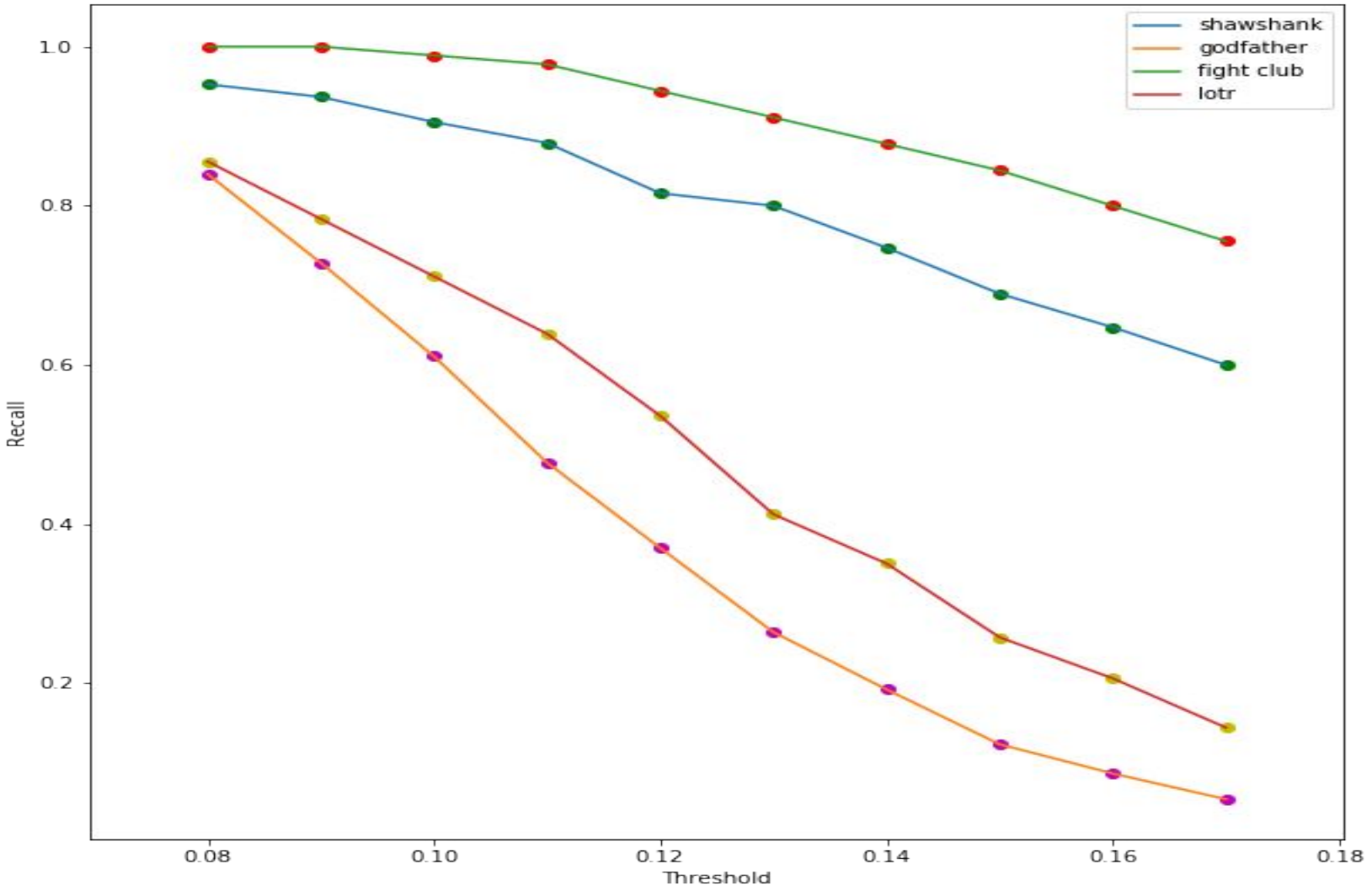
Accuracy vs Threshold



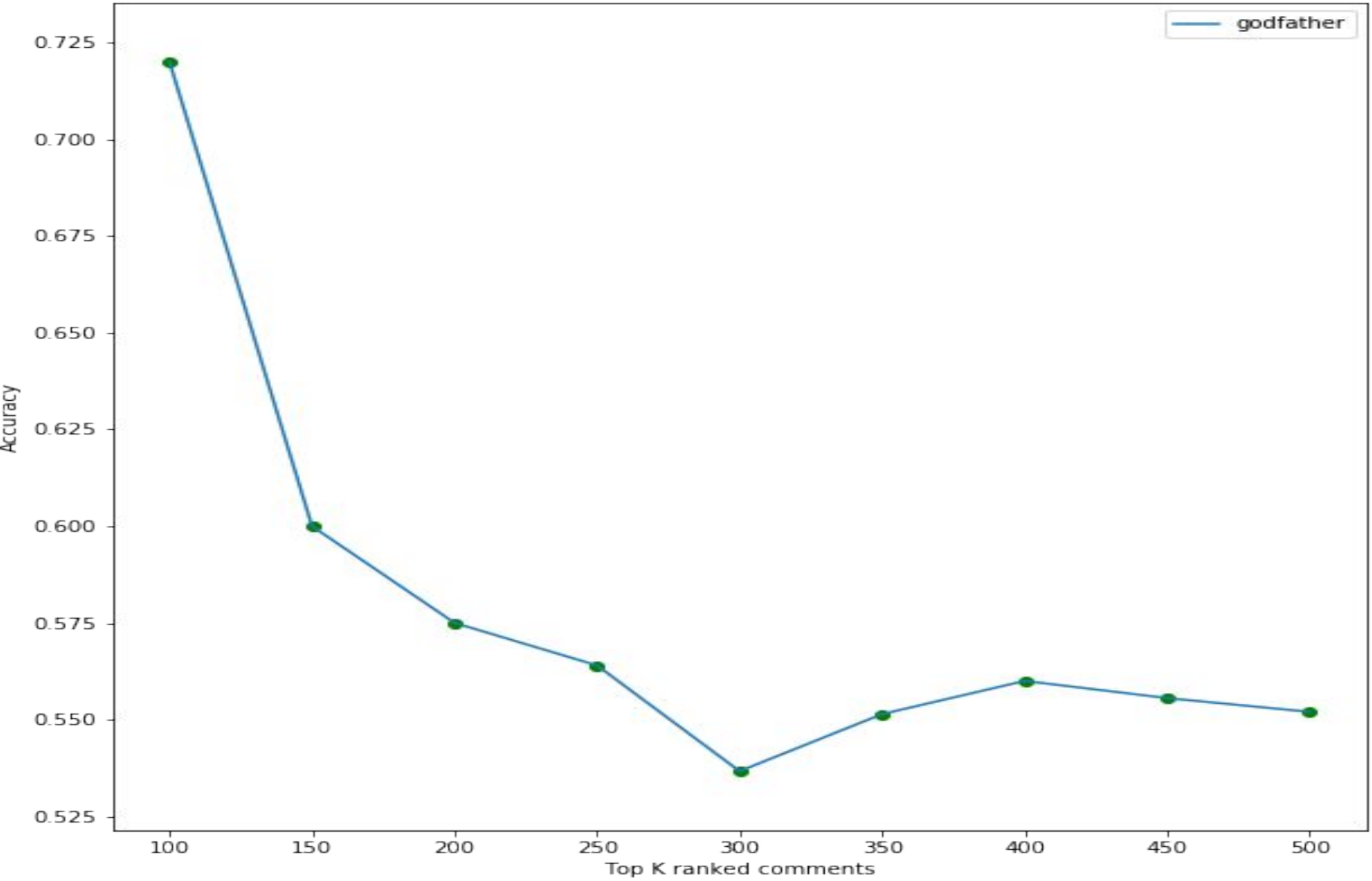
Precision vs Threshold



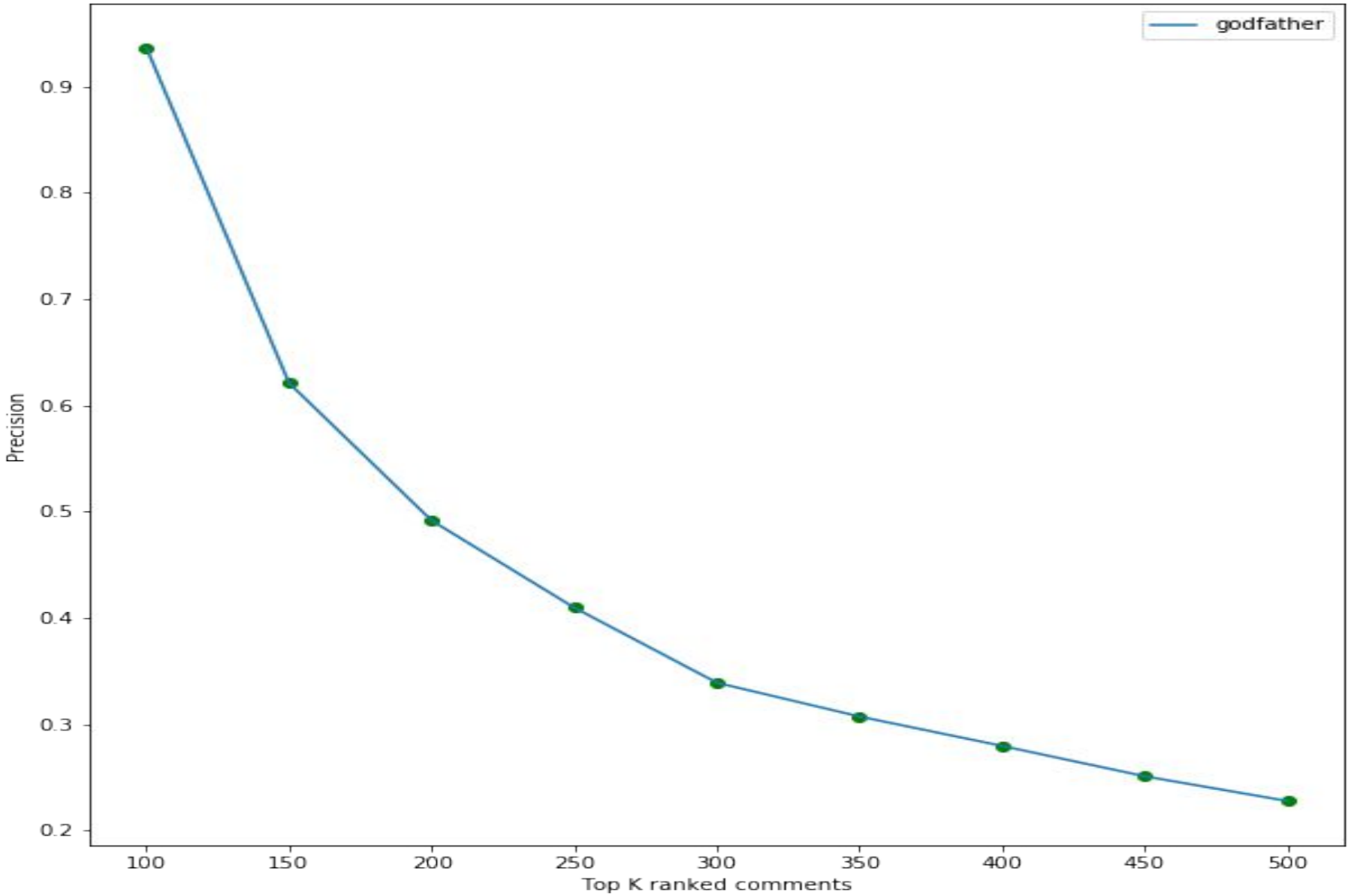
Recall vs Threshold



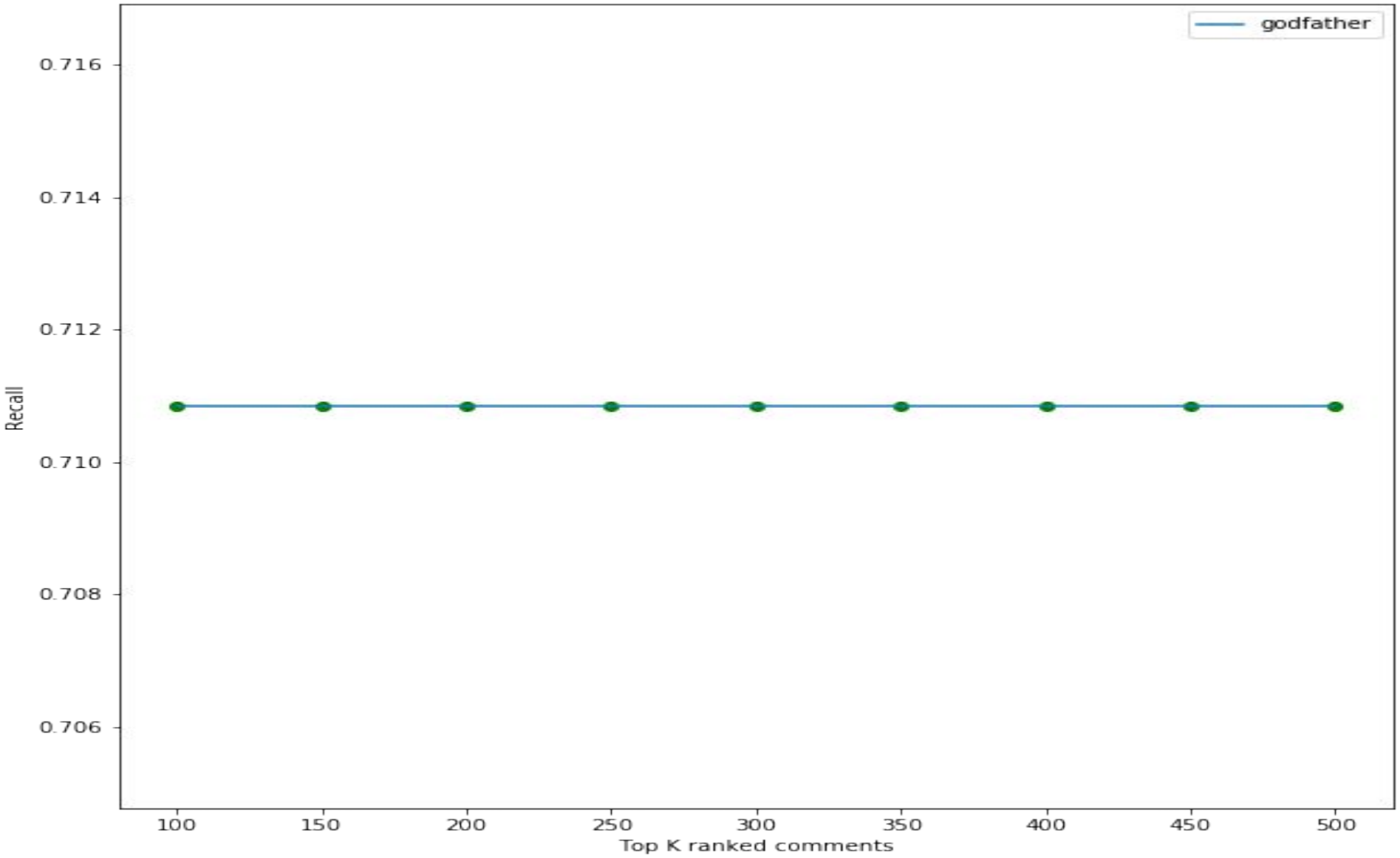
Accuracy vs Top k comments:



Precision vs Top k comments:



Recall vs Top K comments:



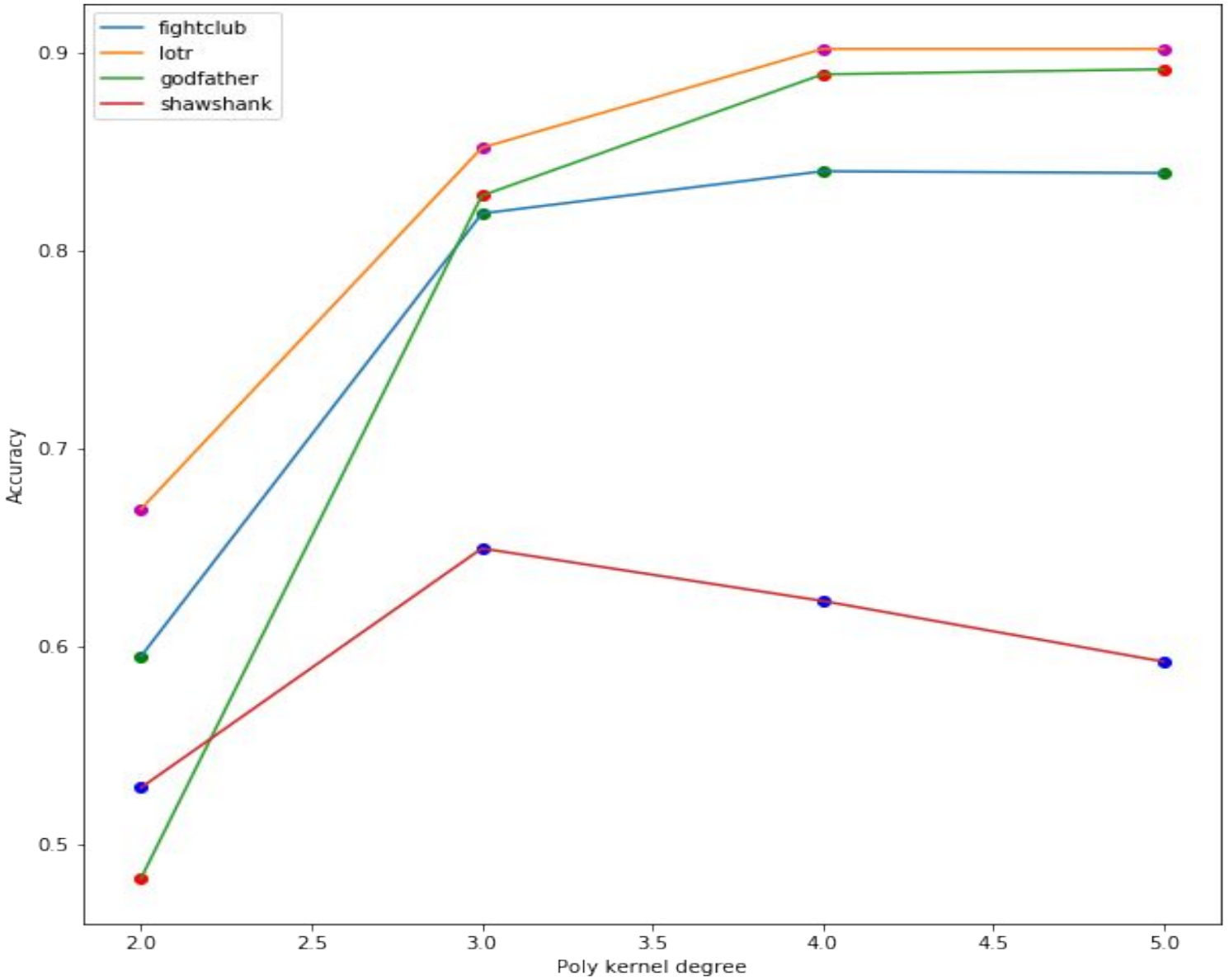
LPU(Learning with Positive and Unlabeled Data)

- LPU assumes we have access to only Positive and unlabeled data examples.
- Unlabeled data may contain positive or negative samples.
- Synopsis and some comments identified as spoilers are treated as positive training data for training the model .
- **Preprocessing**:The Data is cleaned of punctuation marks and the words present in reviews are converted to lower case.

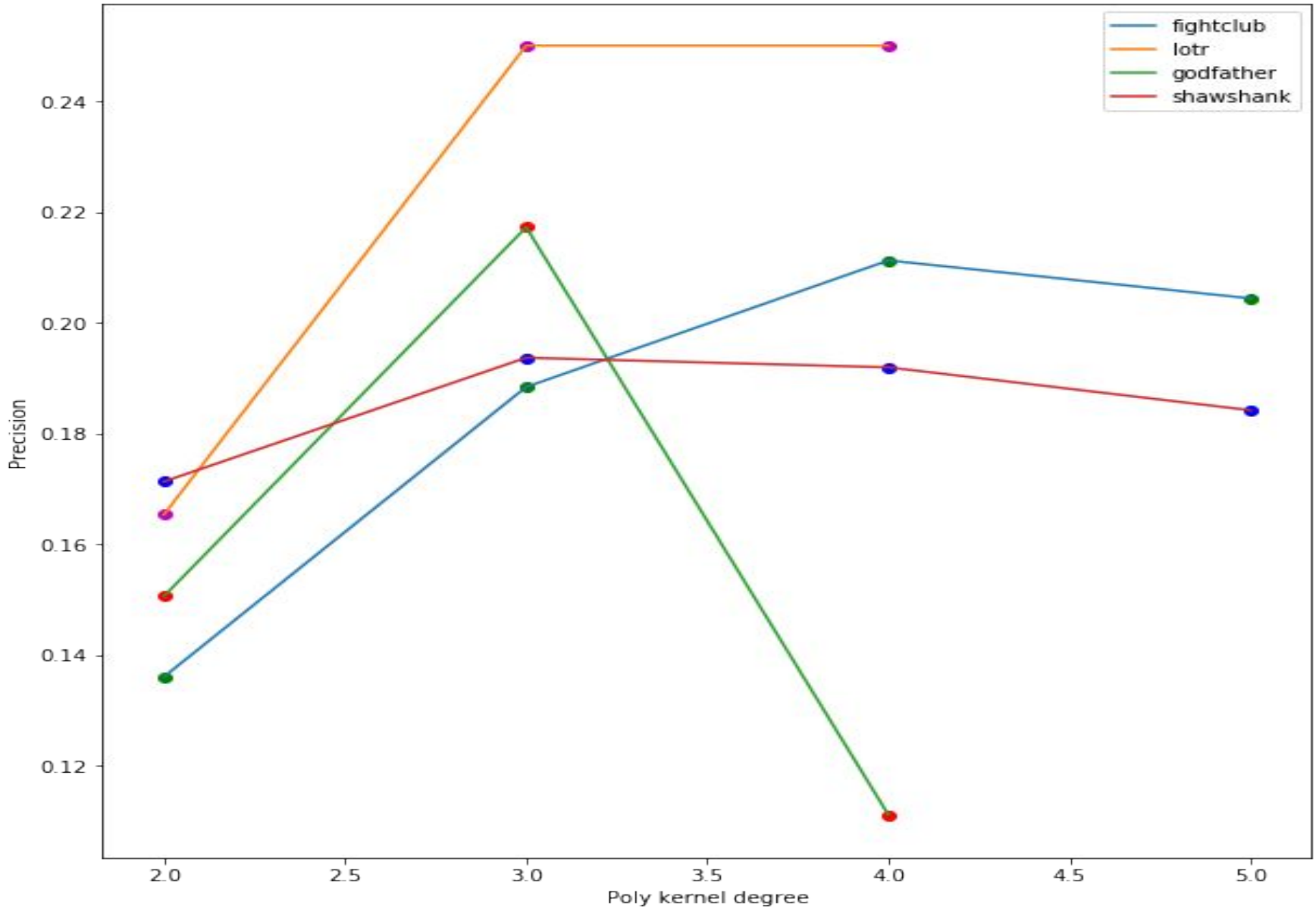
- Half of the Positive examples present in dataset have been considered for training classifier while remaining are appended to test data(unlabeled data).
- From the set of positive examples a training set is created with features as the count of each distinct word present in our data.
- CountVectorizer utility from gensim extracts the features(words)and word counts from the comments to form the training set.
- Common “Stop words” in english vocabulary have been ommited as they carry less weight in Prediction.

- OneClassSVM from SCIKIT library is used to classify.
- By varying the threshold we have compared the accuracy, precision, recall for the comments of 4 different movies.
- Accuracy of our model is computed for different kernels in One Class SVM
- Linear and polynomial kernels are used where degree of polynomial kernels is varied.
- Better performance of model is observed for polynomial kernels with higher degree (~5).

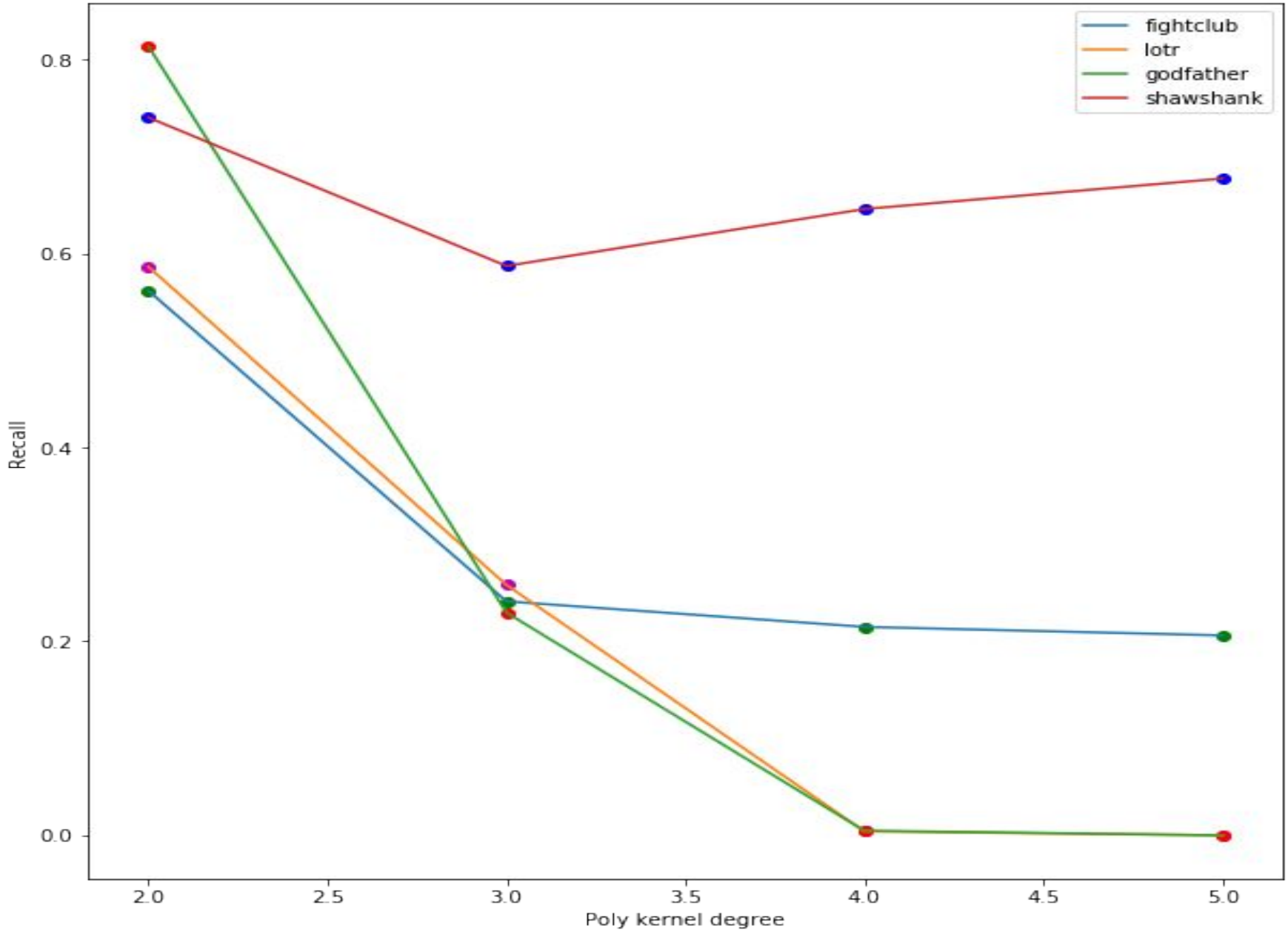
Accuracy vs Kernel degree



Precision vs Kernel degree



Recall vs Kernel degree



LPU with LDA :

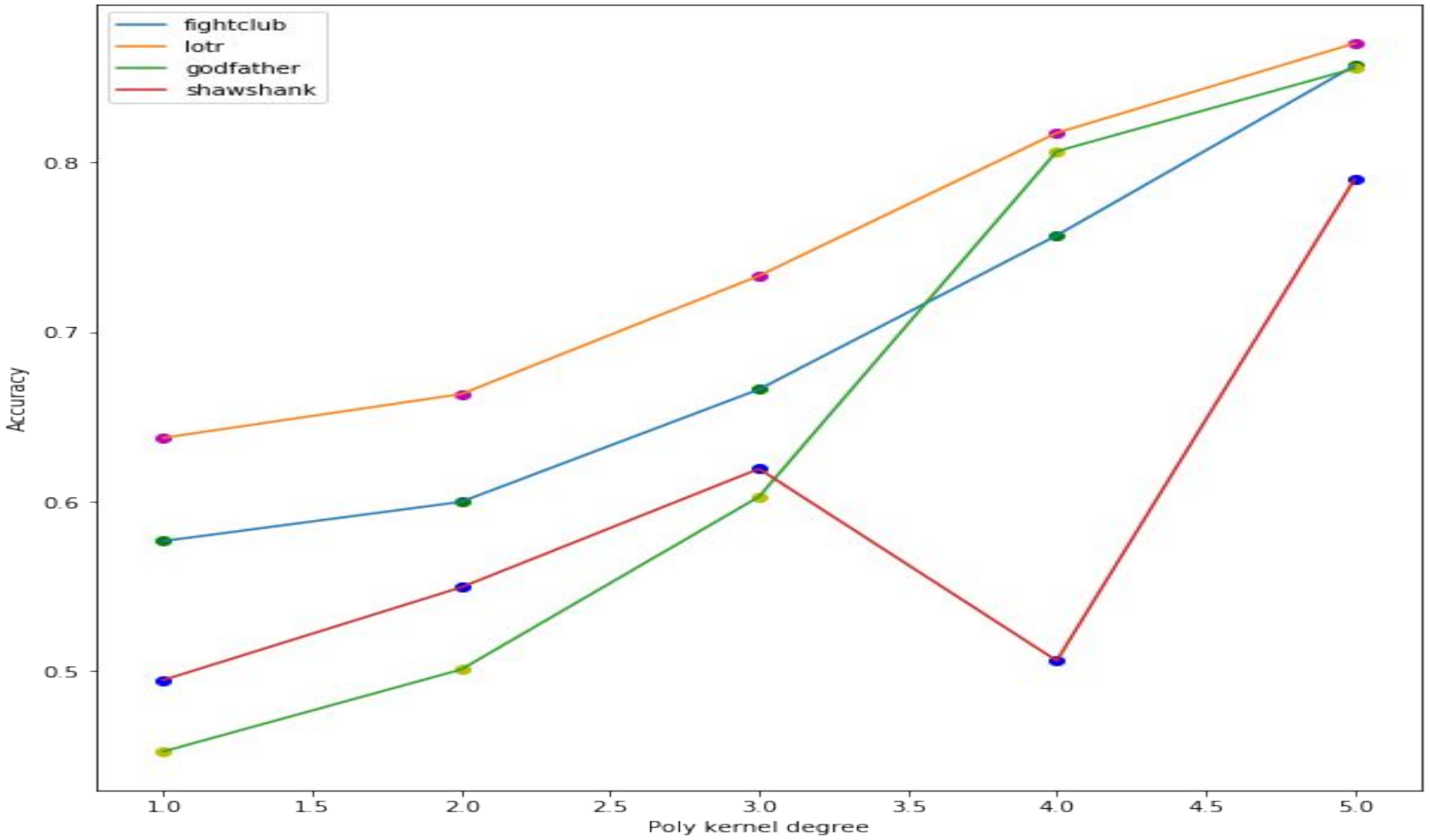
- As explained in Simple LPU, we are using one class svm, each word in dataset is consider as feature .
- In simple LPU we have used all words from training as a feature of SVM .
- In this model we have extracted important words from dataset using LDA .
- LDA gives us distribution of words in topic. Here we are selecting all those words as feature of one class svm.
- One advantage of this model is that it reduces number of feature which improves the speed of execution as compared to simple LPU.

- While forming count vector for each comment we might not get exact same word , as given by LDA model.
- Using ‘wordnet’ from NLTK library we are finding out set of similar words for each word of comment, and if its similar word is part of LDA output then we are incrementing count of LDA word with count of comments word.
- One-class SVM is an unsupervised algorithm that learns a decision function for novelty detection. classifying new data as similar or different to the training set.
- Here we used polynomial kernels with different degrees as part of comaparison.

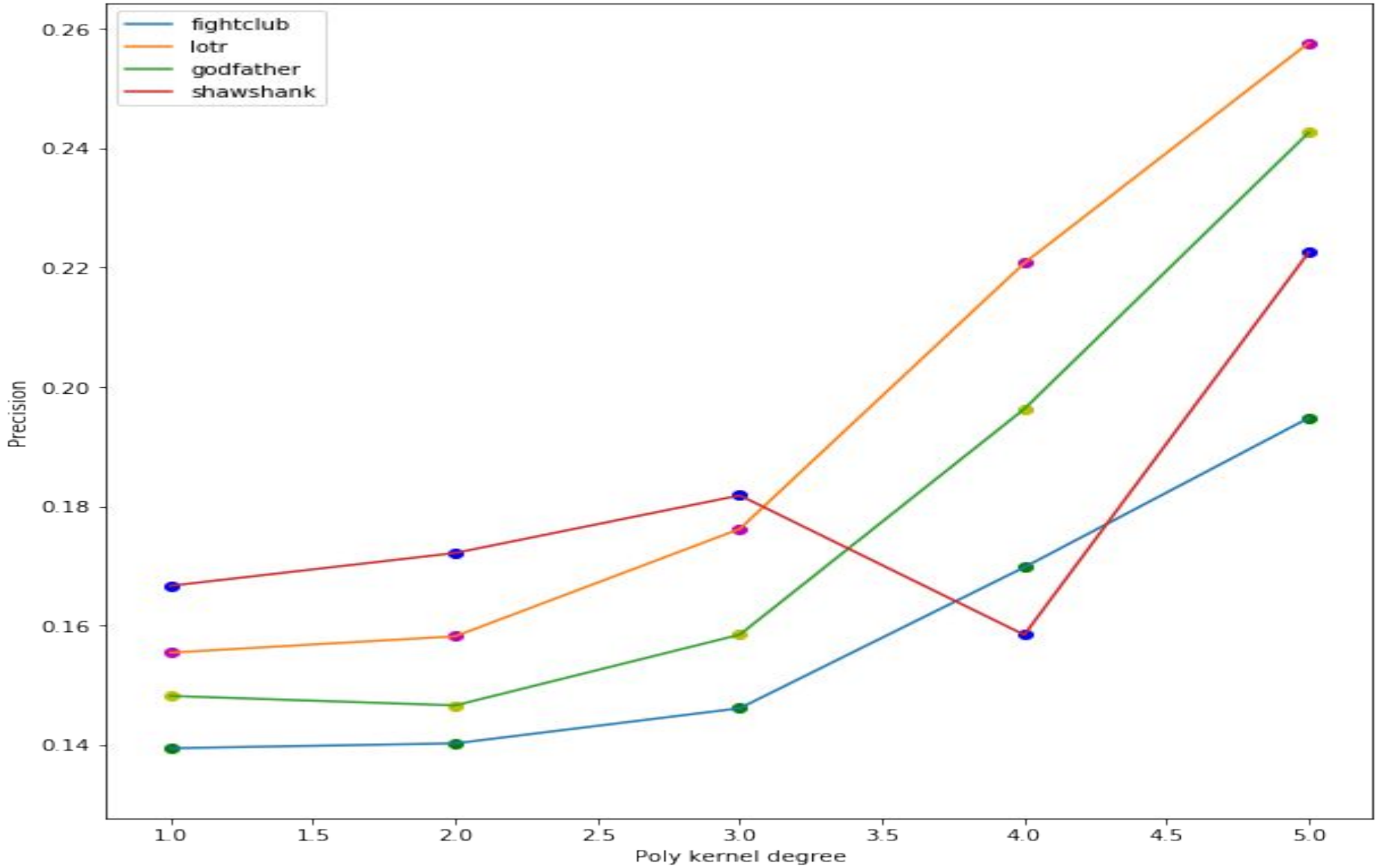
- We can change number of topics in LDA ,we have observed that as we increase the number of topics accuracy was increasing and time of execution also increasing.

- We observed that as we increase degree of polynomial kernel accuracy and precision was increasing because higher-degree polynomial kernels allow a more flexible decision boundary .

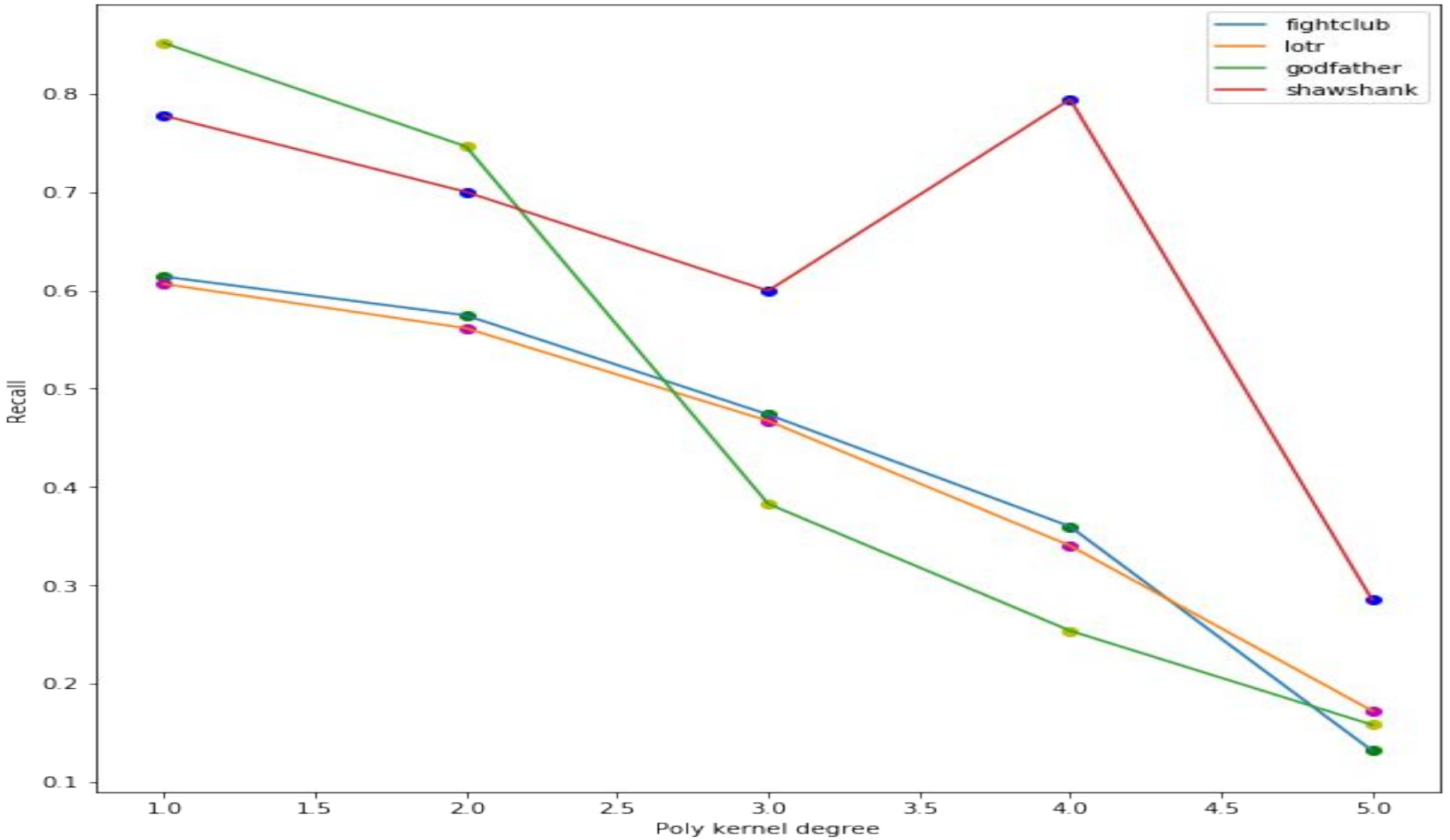
Accuracy vs Kernel Degree



Precision vs Kernel Degree



Recall vs Kernel Degree



Important Libraries Used :

- SVM Classifiers have been imported from “scikit-learn library”.
- For Formatting Dataset and “NUMPY” and “PANDAS” libraries are used
- LDA has been used from “GENSIM” library.
- “WordNet” from “NLTK” library for synsets of a word.
- Graphs have been plotted using “MatPlotLib” Python Library.