Plot Based Semantic Search Engine for Movies

Sai Venkatesh Gatiganti sg629@mail.umkc.edu
Karthik Vundela kvxc5@mail.umkc.edu
Sri Chaitanya Patluri sp6f6@mail.umkc.edu
Chaitanya Sai Manne cm7cd@mail.umkc.edu

***Abstract*— A semantic search engine can take the context into the account and make the search more accurate and sensible.Movies are an area where this can be done in a very elegant way.T There are existing Semantic search engines to find the details about the movies. We can find the details by giving a key word about the movie like movie name, cast etc. However we came up with something different. This paper describes about search engine which find the details based on the plot given by the user.**

I. INTRODUCTION

The Semantic search engine proposed should search for the movies based on given text by the user. This cannot be done in traditional key word based search as given text contains so many words and they will be random words. We have to do analysis on the given text using Natural Language Processing techniques for getting base forms of words, whether they are names of companies, people,their parts of speech etc.Another problem in the text is it contains so many unwanted words like is,the. So we use stop word remover from the Apache Spark machine learning pipeline and we used tokenizer and tf-idf from it in the pipeline.In order to get the movie out of plot we have to train the search engine with the plots.We took the training data from dbpedia data of movies including plots.We take this data as database and the text given by the user is taken as input to search.The whole data in database is processed using Natural language Processing techniques and will be give to pipeline. Detailed explanation will be given in the next sections of the paper regarding particular NLP techniques used. we used NaiveBase for classification and LDA for topics recognition these techniques are also explained in detail in next sections.

Next section of the paper are as follows

II. RELATED WORK

Several works on movie recommendations and summarization already in exist. Apart from that a few on semantic searching are present as well. For instance the movie ontology(MO) at www.movieontology.org aims to provide a controlled vocabulary to semantically describe movie related concepts like movie Genre, Director Actor and the corresponding individual (Age Drama, Steven Spielberg or Johnny Depp). The Web Ontology Language (OWL) is used to specify the MO ontology. Several other ontologies that are provided in the Linked Data cloud are considered and integrated to highly couple the MO ontology.

*
1
2

The movie ontology focuses on concepts and the semantic relations among those concepts. In addition, existing movie ontologies define individuals or instances of a concept superficially. A semantic movie ontology should provide on one hand concept hierarchies (e.g. for movie categorization and navigation purposes) and a sufficient set of individuals that can be used to describe movies. This enables user-friendly presentation of movie descriptions in the appropriate detail and the ability to describe movies taking advantage of the rich selection of the controlled vocabulary in MO. In this website they made an attempt to provide concepts and individuals especially for genre types.

Also, the work on Intelligent search engines by Madhu G ,Dr. A Govardhan et. All gives a detailed explanation on sematic search which gave a deeper insight into the use of search engines, their various generations developed over period of time and the application of semantic technologies over the searching.

III. PROPOSED SOLUTION

The different data sets used and the approach are explained in this section

A. DATASETS

The data sets were considered form multiple sources such as Wikipedia, dbpedia, Amazon.However, the main dataset that has been employed is the one from CMU (Carnegie Mellon University).It has several movie plot summaries extracted from Wikipedia and aligned metadata extracted from Freebase, including many other features like Movie box office revenue, genre, release date, runtime, and language. It includes Character names and aligned information about the actors who portray them, including gender and estimated age at the time of the movie's release. However, we used only the data related to movie plot summaries.

B. ARCHITECHTURE

The system architecture is a shown in the figure. The user who is interested to know about the movies that can probably match his interests can just type the lines of the story that he wishes to watch in the search engine. This query is eventually processed and runs the Scala code on the data that is stored in the local machine which is initiated by spark. The request to get the answer which is the movie name in out case is handled by spark which runs the code upon the data stored in the local disk. After the movie names are retrieved they are sent back as response to the user interface where the user can view the names of the movies.

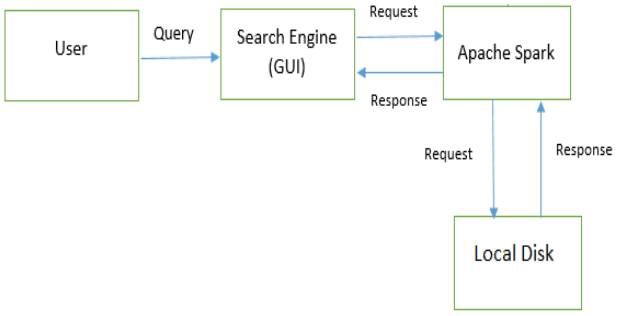

Fig. 1. Architecture

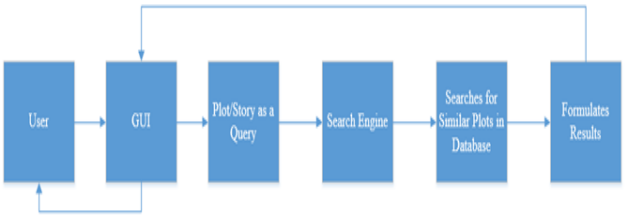

Fig. 3. Work Flow

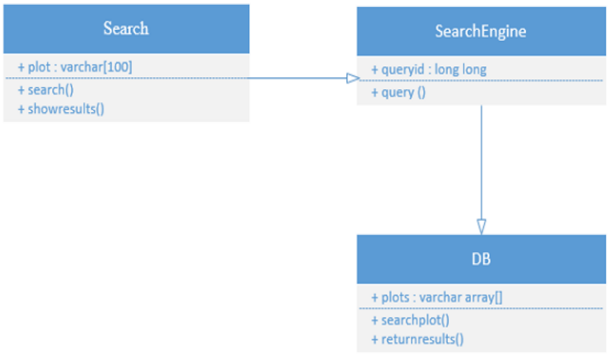

Fig. 2. Class Diagram

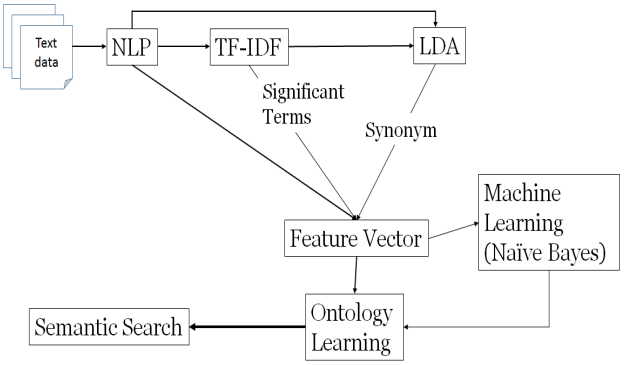

Fig. 4. Implementation

C. UML DIAGRAM

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modelling. It is used both for general conceptual modelling of the systematics of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed. A simple UML class diagram is as shown in the figure. Search, SearchEngine, DB are the classes. Here plot, queryid, plots are the properties of these classes respectively. The methods used in each class are also shown. For instance search() ans showresults() are the methods of the Search class. Similarly the other others are also shown.

IV. IMPLEMENTATION

The overall abstract workflow employed and the implementation are explained in this section

A. WORK FLOW

The working at an abstract level is a s shown in the figure. The user interacts with GUI and gives his inputs which are the plots, stories or may be part of them of any movie of their imagination or choice. This is then queried on the data stored in the disk. After the successful execution of the query the results are generated giving the required movie names.

B. IMPLEMENTATION

The detailed work flow is as shown in the figure. The basic idea is to train the system based on the data that we have and then test the incoming data to get the result. In pursuit of achieving this there are several stages involved.

Initially a large corpus of text data is taken. However, it has lot of unimportant words and has to be processed before it can be fed to the system to get some useful results. The training and testing are explained as follows.

1) CORENLP PIPELINE: Initially all the corpus is collected and given to the stages like CoreNLP Pipeline followed by a creation of Nave Bayes model. The model obtained from the Nave Bayes model is used for Ontology creation. We used Stanford CoreNLP in the project. Stanford CoreNLP provides a set of natural language analysis tools. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, and mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities, indicate sentiment, extract open-class relations between mentions, etc. Stanford CoreNLP is an integrated

framework. Its goal is to make it very easy to apply a bunch of linguistic analysis tools to a piece of text.

In this coreNLP pipeline the text is first tokenized i.e. split into words and tagged with parts of speech and later grouped based on the noun and verb phrases. After splitting the words, they are converted onto lemmas which are the root forms of the words. This eliminates superfluous and unwanted forms of the words. They are parsed to from the parse tree. Later this model is given to stop word remover which eliminates the most frequent words but unimportant words such as the, and etc. We personally used an additional regular expression to filter out further.

2) TF-IDF: After removing unwanted words the lemmas are given to TF-IDF model. tfidf, short for term frequencyinverse document frequency, which is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval and text mining. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general. Thus if a word appear more in a document it is considered important. On the other hand , if a word is repeated many times among different documents in a given corpus then it is considered. Thus the product of these two eliminates the less important and returns the important words. In this project after running the TF-IDF program each lemma is obtained as output along with its score .Greater the score , more important the word.

3) FEATURE VECTOR: The TF-IDF output is then fed to Nave Bayes Classifier. However machine learning algorithms accept only numeric input. Hence the TF-IDF output is converted into feature vectors. These features vectors are given as input to the Nave Bayesian Classifier. The features are defined based on the requirements of the user. A feature vector is an n-dimensional vector of numerical features that represent some object.

4) NAVE BAYESIAN CLASSIFIER: Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables i.e. the features in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers. Feature vectors are often combined with weights using a dot product in order to construct a linear predictor function that is used to determine a score for making a prediction.

Then the data gets classified into different categories. In our project we classified the data into categories where each category represents a movie and the content of the each category is the plot of the movie.

5) ONTOLOGY: An ontology is developed based on the result of the Nave base Classifier. Ontology is a field which studies the methods and methodologies for building ontologies: formal representations of a set of concepts within a domain and the relationships between those concepts. These are realized through RDFs and owl files where owl is the schema

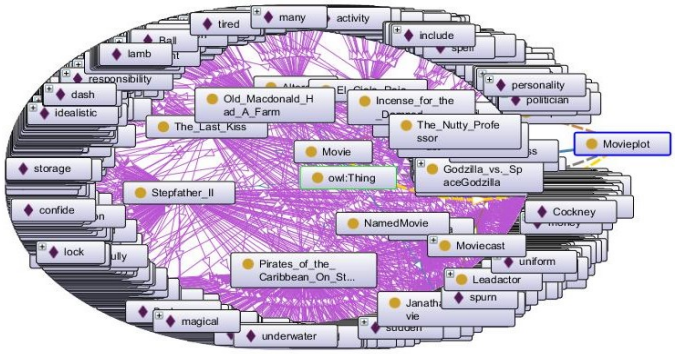

Fig. 5. Class Diagram

RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a triple). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications. This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes.

In our project the ontology is created with owl thing as the super class. The classes represent the movie names which have plots contained in them as subclasses and the characters, roles , actors etc. are present as individuals, entities etc. After building the ontology it is saved as rdf. This completes the training phase.

The ontology visualization with protg is as shown in the figure

V. TESTING

Now the data that the user gives is considered as test data. The task is to take the input text that the user gives and return a movie based on the text given. Thus the given text is made to go through the usual pipeline of coreNLP followed by TF-IDF. This is then given to LDA model.

A. LDA

Latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. In LDA, each document may be viewed as a mixture of various topics. A topic has probabilities of generating various words . A topic is not strongly defined, neither semantically nor epistemologically. It is identified on the basis of supervised labeling and (manual) pruning on the basis of their likelihood of co-occurrence. A lexical word may occur in several topics with a different probability, however, with a different typical set of neighboring words in each topic. The task is to identify the topics whose number is given by the user.

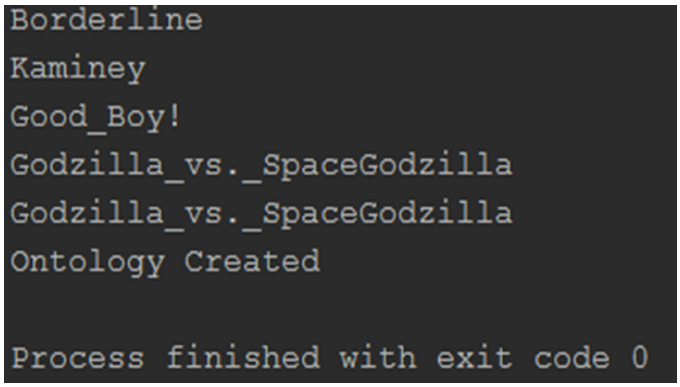

Fig. 6. Movie Names Displayed at the Output

After identifying the topics out of the given text, the categories are searched for the similarity using cosine similarity. The movies which match the content are returned which are then displayed to the user. The number of results displayed depends on the number of topics chosen.

VI. Evaluation

The Search Engine designed in this project was evaluated by calculating Accuracy, Precision, Recall and F-Measure.Here is the fraction of retrieved instances that are relevant, Recall is the fraction of relevant instances that are retrieved and F-Measure is the measurement of Test's Accuracy.

TABLE I
Evaluation Table

Metrics	Value
Accuracy	0.316868102
Weighted Precision	0.192720832
Weighted Recall	0.316868102
Weighted F-Measure	0.226257881

VII. Results

As discussed in the above sections, the results of the search engine are movie names that may be related to the plot that was entered by the user as a search query.

VIII. Future Work

Our Future Work for the project includes including various other domains for the search engine and adding a dynamic web-crawler which crawls the web for related plots and displaying the results dynamically in real time.

IX. CONCLUSION

In today's world, accuracy and precision matters. a semantic search delivers these results by understanding its users and delivering accurate and quicker content relevant to the query provided by the users. Our Project is a small concept which can be expanded across several domains for delivering precise and accurate results.

References

[1] Onan, Aytu, Serdar Korukolu, and Hasan Bulut. "Ensemble of keyword extraction methods and classifiers in text classification." Expert Systems with Applications 57 (2016): 232-247.
[2] Bamman, David, Brendan O'Connor, and Noah A. Smith. "Learning latent personas of film characters." Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL). 2014.
[3] Madhu, G., Dr A. Govardhan, and Dr TV Rajinikanth. "Intelligent semantic web search engines: a brief survey." arXiv preprint arXiv:1102.0831 (2011).
[4] Kraska, Tim, et al. "MLbase: A Distributed Machine-learning System." CIDR. Vol. 1. 2013. Madhu, G., Dr A. Govardhan, and Dr TV Rajinikanth. "Intelligent semantic web search engines: a brief survey." arXiv preprint arXiv:1102.0831 (2011).
[5] Shaikh, A. J., and V. L. Kolhe. "Framework for web content mining using semantic search and natural language queries." Computational Intelligence and Computing Research (ICCIC), 2013 IEEE International Conference on. IEEE, 2013.
[6] http://jmcauley.ucsd.edu/data/amazon/links.html
[7] http://nlp.stanford.edu/nlp/
[8] https://en.wikipedia.org/
[9] http://wiki.dbpedia.org/
[10] http://spark.apache.org/
[11] https://petscan.wmflabs.org/
[12] http://stanfordnlp.github.io/CoreNLP/
[13] http://vowl.visualdataweb.org/
[14] http://www.movieontology.org/
[15] https://www.w3.org/RDF/
[16] http://protegewiki.stanford.edu/wiki/ProtegeDesktopUserDocs
[17] http://www.cs.cmu.edu/ ark/personas/