
OpinionBot: An Intelligent Conversational Agent for Opinion Mining

Samridhi Shree Choudhary
sschoudh@cs.cmu.edu

Rama Kumar Pasumarthi
rpassumar@andrew.cmu.edu

1 Introduction

This document presents the design and implementation of an Opinion Bot, a ‘Conversational Agent’ meant to fetch diverse and succinct opinions on different topics requested by the person interacting with it. There has been a recent surge in research related to conversational agents and dialog systems. With efforts ranging from rule-based dialog systems to deep neural networks for both task-oriented and open-domain dialog agents, the primary focus has been on making dialog with these agents more relevant, coherent, informative and interesting. Opinion Mining has been an active domain of research for a long time. However, there is a significant gap between the current dialog systems and the capability to give proper opinions on an open domain of topics in a conversation. With this work, we hope to take a step in that direction.

Most of the dialog agents today (Alexa, Siri, Cortana and more) fail to provide an opinion about entities and an answer to subjective (non-factual and non-task oriented) queries. The standard replies to questions like ‘What do you think about Politics’ currently range from ‘Sorry I couldn’t find the answer to the question you asked’, ‘I don’t have an opinion on that’ or further still a result from the web search of the question asked. In order to provide the capability to answer questions as mentioned above we aim to adapt the long standing approaches in ‘Opinion Mining’ and ‘Text Summarization’ to the domain of Dialog Systems.

The following sections give an overview of the System Architecture, the dataset used and a description of the modules of the system. We also present some preliminary evaluation of our system. This is followed by the individual insights and learning gained by each of the team members regarding the project and Dialog Systems in general. We conclude with our plan to extend this chat bot further in Future Work.

2 System Architecture

Figure 1 describes the overall architecture and the control flow of the system envisioned by us. The different modules of the system are detailed below:

- **Tokenizer and Entity Recognition** - The query entered by the user is first passed to the ‘Tokenizer’ and the ‘Entity Recognition’ module. The tokenizer splits the query into separate words or tokens. These set of tokens are passed through an entity recognition engine and are filtered to get the identifiable entities. Instead of passing all the tokens or the words in the query to the indexed corpus, we pass this filtered set of entities for targeted and more relevant retrieval of opinions. If the tagger fails to identify the entities from the tokens, the raw set of tokens are processed to remove the stop words and then passed to the corpus for retrieval.
- **Corpus** - The dataset used for this project comprises of the Washington Post news articles and the comments. We scraped the Washington Post API for news articles and the associated comments. The content was stored in MongoDB ([7]) instances. The comments across all the articles in the dataset were annotated with sentiment values (Positive, Negative and Neutral) using a lexical based sentiment analyzer from [4]. These sentiment valence scores

are used to generate opinions on the entities recognized from the user query. For faster retrieval, Lucene is used to index the documents (articles and the comments). More details on Data and Indexing can be found in the Section 3. The indexer returns the sentiment annotated comments that have the highest relevance to the input entities. BM25 scoring is used to calculate the inherent relevance to the query tokens.

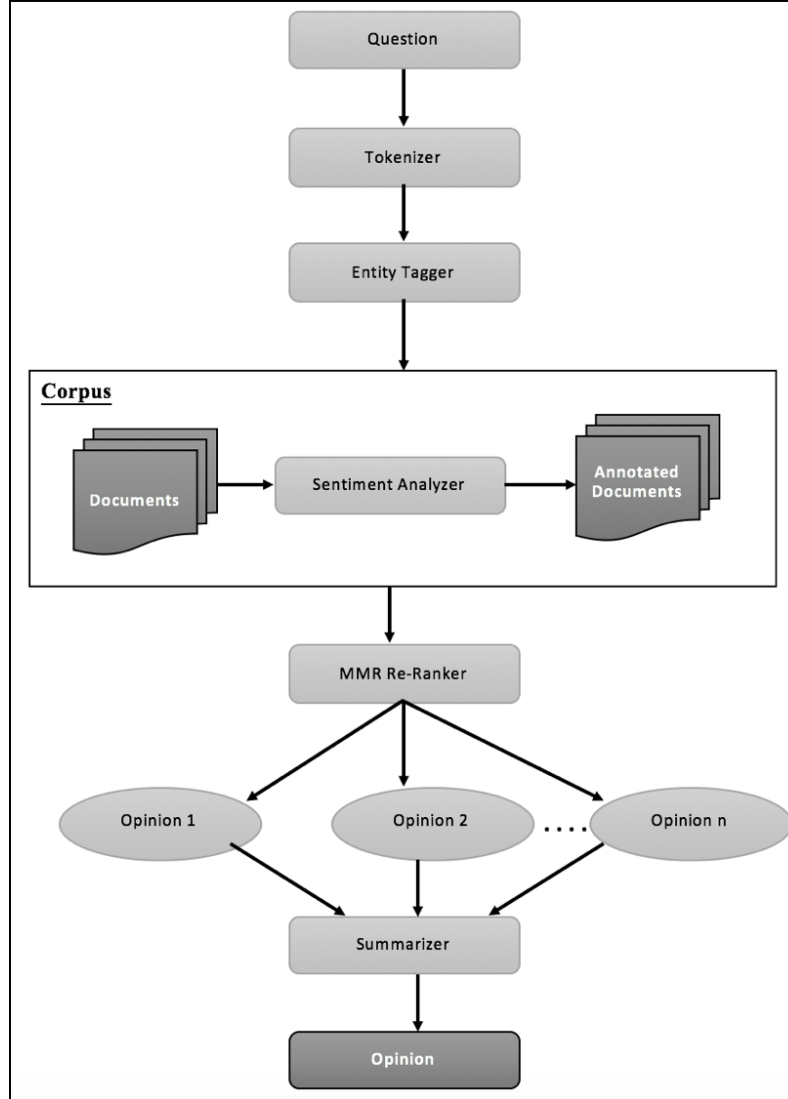


Figure 1: System Overview

- **Maximal Marginal Relevance (MMR) Reranker** - The indexer above returns a list of most relevant comments for the query passed to the system. This list is further divided into 3 separate lists, one for each sentiment value (Positive, Negative and Neutral). Each list is sorted in decreasing order of their sentiment scores. The sorted lists are passed to the MMR [3] reranker. MMR has traditionally been used to provide non-redundant or diverse multi-document summaries. In our setup, it readjusts the retrieval scores of the documents (comments in our case) in order to maintain the relevance to the original query as well to put the documents with maximal non-redundant information next to each other. Each sentiment annotated list of comments are seeded with their original BM25 relevance score that gets adjusted and re-sorted by this module. More details about the re-ranking can be found in Section 5.

- **Summarizer** - The output from the reranker produces multiple opinions for each opinion valence (Positive, Negative or Neutral). These multiple opinions are passed to the summarization module in order to get an informative, well formed opinion from multiple opinions. An 'Extractive Summarization' is performed on this list. A summary of all Positive, Negative and Neutral opinions on a topic is created and is then presented to the user depending on her/his choice of the valence. More details on the summarization module can be found in Section 6.

3 Data

We use Washington Post (WP) news articles and the associated user comments as our dataset for extracting opinions on various topics. As mentioned in the previous section, the Web API was scraped for both articles and comments. The interfaces for articles and comments were separate and hence they were collected separately and stored on two MongoDB instances. Since both had a common 'contenturl' field, we created a combined table with the articles and all the comments associated with it. This combined table was indexed and was used as the main Corpus detailed in Section 2. The JSON schema for the articles and the comments scraped are shown in Figure 2.

The database is updated and indexed everyday with the latest WP news on the web. Following are the current statistics for the dataset:

- **Number of articles** = 363994
- **Number of comments** = 2609071
- **Span of years** = 2012-2017

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "https://docs.washingtonpost.com/docs#",
  "properties": {
    "blurb": {
      "id": "/properties/blurb",
      "type": "string"
    },
    "body": {
      "id": "/properties/body",
      "type": "string"
    },
    "contenturl": {
      "id": "/properties/contenturl",
      "type": "string"
    },
    "headline": {
      "id": "/properties/headline",
      "type": "string"
    },
    "id": {
      "id": "/properties/id",
      "type": "string"
    },
    "location": {
      "id": "/properties/location",
      "type": "string"
    },
    "source": {
      "id": "/properties/source",
      "type": "string"
    }
  },
  "type": "object"
}
```

(a) JSON schema for WP articles

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "definitions": {},
  "id": "https://docs.washingtonpost.com/docs#",
  "properties": {
    "comment": {
      "id": "/properties/comment",
      "type": "string"
    },
    "compound_score": {
      "id": "/properties/compound_score",
      "type": "integer"
    },
    "contenturl": {
      "id": "/properties/contenturl",
      "type": "string"
    },
    "id": {
      "id": "/properties/id",
      "type": "string"
    },
    "negative_score": {
      "id": "/properties/negative_score",
      "type": "integer"
    },
    "neutral_score": {
      "id": "/properties/neutral_score",
      "type": "integer"
    },
    "positive_score": {
      "id": "/properties/positive_score",
      "type": "integer"
    }
  },
  "type": "object"
}
```

(b) JSON schema for WP comments

Figure 2: JSON Schemas

4 Entity Recognition and Query Pre-Processing

4.1 LSTM-CRF model for Entity Recognition

The Entity Recognition module treats entity tagging as a sequential labeling module. The LSTM-CRF architecture proposed by Lample *et. al.* [5] is used to train the tagger on CONLL-2003 dataset [8].

The data has three entity tags: *PER*, *LOC*, *ORG* and an "O" tag for words which are not part of any entity. Sentences are represented using the IOB format (*Inside*, *Outside*, *Begin*). A B-label is used to tag the first word of an entity, and all subsequent words are tagged using I-label.

The architecture comprises of a Bidirectional-LSTM (Long Short Term Memory) layer to process the sentence and capture the context for a given word. The model has a Conditional Random Field layer while decoding, to maximize both emission probabilities (based on Bi-LSTM generated features) and transmission probabilities (which look at sequence level scores).

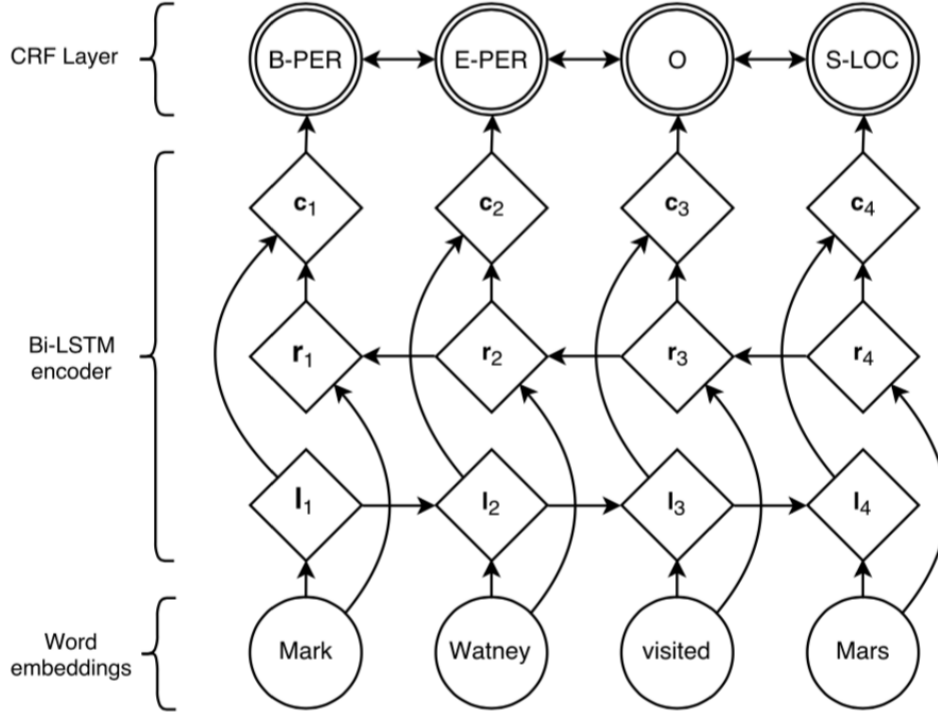


Figure 3: LSTM-CRF architecture for Entity Recognition

The entity tagger was hosted as a web service, accessible via a REST API, using flask python module and *mod_wsgi*.

4.2 Query Pre-Processing

The query pre-processing module extracts entities, if any, and retrieves most relevant modules for those entities. If no entities are detected, the entire query is used for retrieval. The pre-processor also does a basic data cleaning to remove stop words.

5 Retrieval Model

This section provides detail of the Model used in order to retrieve relevant opinions from the dataset for the query passed. There are three primary components for this model. Each component is described in the following sub-sections. The model strives to return a list of relevant, short and diverse opinions.

5.1 Data Pre-Processing

As mentioned before, the data collected by scraping the web articles and comments are stored in MongoDB collections, 'article' and 'comments', as JSON documents. Figure 4 shows the pre-

processing steps applied to the data collected. These actions facilitate in accurate and fast retrieval of opinions for the query passed. Each step is defined in more detail in the following subsections.

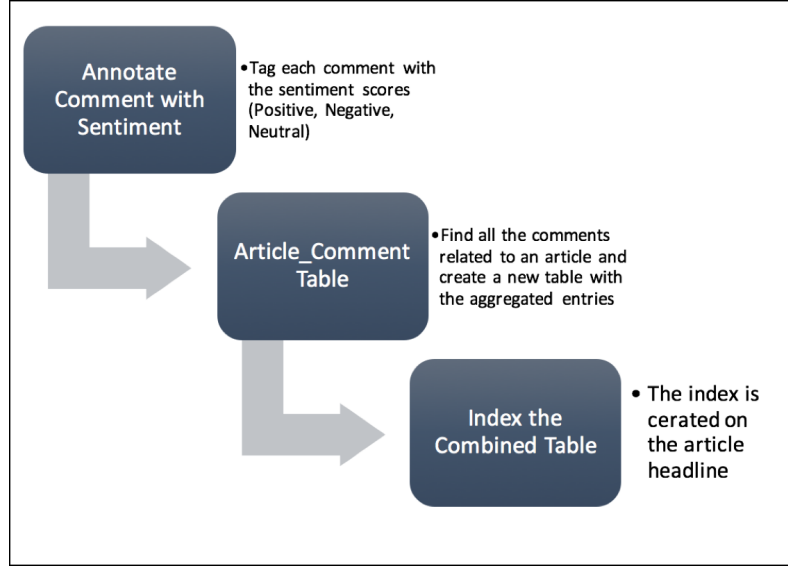


Figure 4: Data Preprocessing

5.1.1 Article-Comment Table

The web interface for collecting articles and comments are different. Therefore, they are scraped and stored separately in different Mongo collections. In order to find the comments associated with a particular news article, we make use of the shared field ‘contenturl’ between them. This field stores the URL of the article scraped. All the comments having the same ‘contenturl’ as that of an article are aggregated into a list of JSON documents. Every article is stored with its corresponding comments list in a new table. The indexing and retrieval is performed on this particular table.

5.1.2 Sentiment Annotation

Traditionally, opinion identification has been done by identifying the inherent sentiment in the text. Similarly, we annotate the user comments with sentiment values in order to classify them as expressing either a positive, a negative or a neutral opinion. We mine these sentiment-tagged comments in order to form a crowd-sourced opinion, of a certain polarity, about an entity. This helps us to retrieve information about what people think about a topic. We use the rule-based lexical sentiment analyzer as described in [4]. The tool estimates four scores for each post: ‘positive’, ‘negative’, ‘neutral’ and ‘compound’. The positive, neutral and negative score represent the proportions of the text that fall into each of these categories respectively. The compound score aggregates the overall sentiment of the post. We look at the compound score for opinion identification. For each query we retrieve the most relevant list of ‘positive’, ‘negative’ and ‘neutral’ comments present in the database.

5.1.3 Indexing

Faster responses are one of the most desirable features in a good dialog system. In order to facilitate faster and accurate retrieval we use Lucene [2] for indexing our combined table as described in section 5.1.1. Indexing in Lucene is done using tf-idf scores for the document. Instead of indexing all the fields in the combined table, we only index the ‘headline’ column. This field represents the headline for a news article. Given a query, the tokens/entities of the query are passed to the indexer that returns the most relevant documents based on its internal scoring functions.

5.2 Retrieval

Retrieval is performed on the pre-processed data. The query is processed as described in section 4.2. Once the tokens or the entities are identified, they are passed to the index created. Following are the basic steps performed in order to retrieve relevant opinions.

1. The query entities/tokens are passed to the combined-table indexer. The articles with the most relevant headlines (tf-idf frequency) to the query are returned. We retrieve top 50 articles for a given query.
2. Comments for these top articles are then fetched from the database. Longer comments are discarded. The comments containing ‘abusive’ words are also removed from the final list. Currently, abusive words are handled with a stopword list of commonly used abuses. Certain words specific to the domain of news articles are also included in the stopword list. Some examples of these words are ‘op-ed’, ‘article’, ‘author’ and so on.
3. ‘BM25’ scores, in relation to the query terms, are calculated for the filtered comments.
4. The list of comments are sorted in decreasing order of their scores and returned.
5. This list is then divided into 3 separate lists: ‘Positive’, ‘Negative’ and ‘Neutral’ comments list. The comments are therefore separated depending on their valence scores. The 3 component lists are further sorted in decreasing order of their sentiment scores and passed further on to the re-ranker module.

5.3 Re-Ranking for Diversity

In order to facilitate the creation of diverse and informative summaries, we use Maximal Marginal Relevance (MMR) as defined in [3] to rerank our sorted lists obtained as described above. MMR combines query relevance with information novelty by reducing redundancy in the information contained in multiple documents. It readjusts the relevance scores returned by our retrieval engine described above by taking into account the ‘relevant novelty’. Marginal Relevance is a linear combination of relevance measure and novelty. Therefore, a document has high marginal relevance if it is relevant to the query and has the least information similarity with the previously selected documents. The MMR engine treats each comment as a single document. It re-estimates the scores of the comments and sorts the list with maximally relevant comments. This provides a well sorted list for further summarization as described in section 6.

6 Summarizing the Opinions

6.1 Meta Level Summaries

The agent can also respond with high level summaries for any given valence. It responds with number of relevant opinions, and fraction of them which are of positive, negative and neutral valence. This would help the user to choose which valence of the opinions to query for.

6.2 Opinionated Extractive Summarization

We generate responses in two ways: retrieving an opinion with the highest combined score for polarity and generate a summarized opinion over the highest ranked opinions.

For extractive summarization, we use an approach based on TextRank, which does a keyword extraction to identify the set of words or phrases that are most informative from a given document [6]. The TextRank algorithm determines terms that are central to the document, in a manner similar to PageRank.

A graph with words/phrases is constructed, with edges representing lexical similarity, based on word co-occurrences. Centrality in this graph is used as a measure of relative importance of the phrase. The most important phrases are used to generate the summary.

7 Multi Turn Conversations

Our agent can hold multi-turn conversations, where the user is allowed to query about a topic/entity, get high level summaries, choose a polarity of the opinion and repeatedly query for different polarities. The user can switch to another query, and also give feedback in terms of usefulness .

An example multi-turn conversation is shown below. Note that this is a real example based on an interaction during evaluation of our agent.

Bot: "Hi, I am Opinion Bot"

User: "Tell me about impact of capitalism"

Bot: "Do you want a positive/negative/neutral opinion?"

User: "positive"

Bot: "conservatives love unbridled capitalism. the best solution to poverty is unbridled capitalism and economic freedom, unimpeded by government interference.." Was this useful?

User: "negative"

Bot: "bad people (is this a pleonasm?). you suggest that capitalism is only bad, not so?" Was this useful?

User: "neutral"

Bot: "my ideas of capitalism are based on observations of how corporate america conducts itself in the 21st century and the impact their actions have on the world around us.." Was this useful?

User: "Yes"

Bot: "Thank You. Would you like to know about something else?"

8 Evaluation

We evaluate on a Likert scale [1] of 1-5 along the following factors:

1. *Informativeness*: How much of the important information is conveyed
2. *Fluency*: How fluent/grammatically correct is the response
3. *Compactness*: Response contains unnecessary information
4. *Relevance*: How relevant are the response

Table 1: Evaluation

Annotator ID	Informativeness	Fluency	Compactness	Relevance
1	3	5	2	4
2	2	4	3	2
3	2	4	2	2
4	2	4	4	2
5	2.5	4	4	3.5
6	3	4	5	3
Average	2.42	4.17	3.33	2.75

8.1 Analysis

The Opinion Bot produces fluent and compact responses, as it aggregates human generated opinions and performs a summarization over them. The summarization helped in informativeness, without sacrificing fluency. While the agent was able to retrieve responses with good search relevance to the query, evaluators gave a lower score as they were looking for abstract topical relevance. In many cases, the relevance and retrieval model was able to generate very interesting and deeply insightful

(and opinionated) responses. Characterizing what constitutes a point-of-view and what are diverse points-of-view for a given topic is particularly challenging, which we plan to address in future work.

9 Contributions and Insights

We both worked on System Evaluation and incorporating the feedback into improving the system.

9.1 Samridhi - Contributions

I worked on the following:

- Scrape the Washington Post API for articles and comments.
- Pre-Process the data. This included cleaning, tagging the data with sentiment scores and storing it in Mongo Collections. The data was then indexed using Lucene.
- Build the Retrieval module with all its caveats as described in Section 5.
- Build the MMR engine for diversity.
- Get the initial pipeline ready for the chat interactions and further filtering of the abusive and news specific stopwords as and when the system was used.

9.2 Samridhi - Insights

This project was a great learning experience. Building the retrieval pipeline was helpful in understanding the subtle caveats of IR systems. I learned about indexing and summarization. Implementing MMR and comparing the results with the vanilla model helped me understand the importance of information novelty in Opinion Systems. I was exposed to the exciting area of Open Domain Dialog Systems and Opinion Mining. Both are very challenging and interesting fields. This project along with the weekly dialog reading groups, gave me an opportunity to understand the emerging domain of chat-bots and dialog agents. I also learned about creating an Alexa skill, setting up the Lambda Function and get the skill working.

Instilling the capability of human like interactions in conversational agents is a burgeoning area of research. I believe our work is a right step in that direction. Looking for opinions on different topics and entities is most natural form of interaction among a group of people. With the wealth of data and information available online, if we can build a chat bot which can not only crowd source but also provide abstractive summaries of the opinions that people hold, would make the conversations seem more humane. There are, of course, many caveats to it. Some examples that we faced while building the systems included - handling abusive statements, identifying hate speech, identifying negative and sarcastic opinions, highly polarizing opinions on sensitive topics like terrorism, politics, politicians, women equality, racial equality and so on. We realized in the process that this problem is far from solved and requires more effort and innovative ideas to handle it. Stopword lists seem like the most popular way to go but there is scope for further improvement.

It was a great collaborative experience and I look forward to working further in this domain with hope to make our opinion bot more informative, fluent and perhaps more personal to the user interacting.

9.2.1 Rama Kumar - Contribution

My work involved building:

- Built the Entity Tagger, host it as a web service accessible via REST API
- Built the extractive summarizer module, which works over a list of opinions of a given valence
- Added capability to respond with high level summaries per valence
- Added multi-turn capabilities to elicit desired sentiment, high level summaries and respond with summarized opinion.

9.3 Rama Kumar - Insights

Chat bots are the new apps, which provide a rich, human-like interaction. Discussions on topics on various perspectives and opinions is something that comes up naturally in conversations with friends and colleagues, but most chat bots currently can only function in a fact based question answering style. This project was a first step in that direction. This project was an insightful collaborative experience, where I learned to identify the appropriate modules and architecture for an opinion mining conversational agent, implement and interact with it.

Natural Language Processing and Understanding, clearly has additional challenges in conversational settings, where tasks such as entity recognition and summarization have to be fine-tuned to be suitable for succinct, interactive and informative conversations.

Understanding topical relevance, and characterizing diverse polarities of opinions, or points-of-view, are the next set of challenges that I plan to look at.

A key challenge we encountered was that the agent also needs to have an understanding of what polarity of opinion to present. It needs a careful balance between always presenting positive opinions in a monotonous way, and from presenting negatively polarized opinions which might be borderline abusive. Abuse detection in opinions is a challenging task, that can greatly help in language understanding and response generation for conversational agents.

Alexa Skills Kit provides an easy to use interface to implement conversational agents. This project was helpful in understanding how to design various components of a dialog agent, how to host them, and ensure low latency for easy interaction.

10 Conclusion and Future Work

The Opinion Bot was able to generate responses that were fluent and compact and in many cases, interesting. Making the response more relevant to user is a challenging task, as many a times, it depends on the context of the topic the user is talking about. Given that these are opinions, certain words tend to be more important, which are not necessarily captured by search engine relevance. Doing topic modelling based analysis for keywords and documents for better relevance could help.

Movie and restaurant reviews are another direction that is worth exploring. They tend to have summaries and ratings along different dimensions, which can be useful in training summarization and relevance modules.

References

- [1] I Elaine Allen and Christopher A Seaman. Likert scales and data analyses. *Quality progress*, 40(7):64, 2007.
- [2] Apache. Lucene. <https://lucene.apache.org/>, 1999. Accessed: 2017-05-12.
- [3] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM, 1998.
- [4] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.
- [5] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*, 2016.
- [6] Rada Mihalcea and Paul Tarau. Texttrank: Bringing order into texts. Association for Computational Linguistics, 2004.
- [7] MongoDB. Mongoddb. <https://www.mongodb.com/>, 2009. Accessed: 2017-05-12.

- [8] Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics, 2003.