Part 1. An Explainer for Information Retrieval Research
Part 2. Open Domain Complex Question Answering

Sourav Saha
CS 1833

Supervised by Mandar Mitra

Indian Statistical Institute, Kolkata

Signature _____

Date _____ / _____ / _____

Abstract

This thesis is organised in two parts. First, an explainability in Information retrieval (IR) research where we focus on the performance of the IR models. We present a toolkit I-REX to illustrate the performance and explainability of IR systems. It is an interactive interface built on top of *Lucene* and gives a white box view of any proposed method. It is implemented as a web based and as well as shell based interface to provide an intuitive explanations and performance of IR systems. The baseline retrieval models such as LM, BM25 and DFR, and a set of well-defined features enable debugging the performance of retrieval experiments such as ad-hoc IR or query expansion. Next we worked on an open domain complex factoid Question Answering (QA). Creating annotated data in QA problem requires lot of resources and it is very time consuming. The available datasets are often domain specific and most of the times created for some specific languages. Therefore we mainly focus on answering the questions in an unsupervised way. As a benchmark data we used the data provided by Lu et al. (Quest)[26]. It mainly focuses on complex questions which cannot be answered by knowledge graphs (KGs) directly. Our architecture uses corpus signals over the various documents along with the traditional QA pipeline to answer the complex questions. We proposed a set of modified evaluation protocols to overcome some serious pitfalls in the evaluation measure used in Quest. We also compared the performances of our architecture with another neural benchmark model DrQA [11]. Experiments on this benchmark datasets have shown that our model significantly outperforms Quest and DrQA. We find this very encouraging since DrQA is trained on SQuAD [32], TREC Questions [4], WebQuestion [5], WikiMovies [30] while our proposed method is unsupervised in nature.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Dr. Mandar Mitra. People like him are extremely rare in this world. He has been an inspiration for me to do good research in the field of IR. I would be forever indebted to him for his academic as well as non-academic help towards my career at ISI. This work would not be possible without his strong encouragement, motivation and support.

Many thanks to Dwaiapayan Roy for being an awesome mentor, colleague and a senior. He helped me a lot in the initial days of my IR research by answering my silly naive questions and also guiding me on *what not to do* regarding research. Thanks to Debasis Ganguly for his everlasting inspiration, enthusiasm and all the academic help I required. I would also thank other lab members Ayan da, Suchana di.

Thanks to Professor Soumen Chakrabarti for his generous help towards giving a short survey of what's going on about QA, corpus and KG across IR, NLP and AI communities.

Special thanks to Professor Dipti Prasad Mukherjee and his lab members for providing us with the GPU support, without that a lot of experiments would have never been possible.

I am indebted to the faculty members at ISI for providing the wonderful courses, slides, lecture notes and giving a friendlier teacher student relationship.

Finally, I would like to thank my friends, family for their love and support.

Contents

List of Tables

List of Figures

Chapter 1

I-REX [1]

1.1 Abstract

Providing high-level, intuitive explanations of the performance of IR systems is generally difficult due to their complexity, and the various low-level implementation details involved. We present `I-REX`, a tool built on top of Lucene, that is intended to provide a systematic view into the inner workings of retrieval models and methods (specifically query expansion). This should help researchers study, compare, understand and explain the performance of these models and methods. `I-REX` can be run either as a Web service accessible through a browser, or as a terminal-based tool with a shell-like interactive interface. In this article, we describe a session that illustrates how `I-REX` can be used to explain the observed difference in the performance of two variants of the Language Model.

1.2 Introduction

In recent times, as technology, particularly Artificial Intelligence and Machine Learning, make remarkable advances, there is a growing interest in the *explainability* of these technologies, i.e., in understanding *why* a technique performs the way it does, rather than

[1] This chapter is based on *Dwaipayan Roy, **Sourav Saha**, Mandar Mitra, Bihan Sen, Debasis Ganguly* **I-REX: A Lucene Plugin for EXplainable IR**. *28th ACM International Conference on Information and Knowledge Management (CIKM 2019),* pp. 2949–2952.

simply finding whether it works well or poorly. In Information Retrieval (IR) too, many well-known studies have provided high-level, intuitive explanations of the performance of retrieval models and methods. For example, Singhal et al. [36] explained that the traditional normalisation method used in the Vector Space Model (VSM) over-penalises long documents, leading to poor retrieval effectiveness. This in turn led to an improved term-weighting scheme within VSM. Smucker and Allan's analysis [37] provides a similar explanation for why Dirichlet smoothing works better than Jelinek-Mercer smoothing in the language modeling framework. Likewise, the axiomatic frameworks for retrieval [18] and pseudo-relevance feedback (PRF) [13] encode several intuitions in the form of axioms that can be used to easily explain the relative performances of different models and methods. As in [36], the insights thus obtained lead to improved methods for PRF. Recently, researchers have applied local linear approximation approaches for model agnostic explanations of IR models [38, 19].

A tool for gaining insights into an IR model. Because IR systems are complex, and there are numerous sources of variation, the process of analysing experimental observations in order to identify and validate high-level explanations can be both difficult and tedious. Researchers generally resort to ad hoc combinations of search systems, scripts and visualisation tools for this task. As a first step towards addressing this situation, we contribute `I-REX`, an interactive tool, implemented as an additional layer built upon Lucene's API. Two example features that `I-REX` provides are outlined below. A more complete description can be found in Section 1.4. To the best of our knowledge, no available version of Lucene provides such information in a convenient way.

1. For a given query Q, a document D, and a retrieval model $\mathcal{M}$, the tool may be used to analyse the similarity score of D with respect to Q under $\mathcal{M}$. Following [28], we view any ranking formula as a linear function of the form $S\big(D, Q\big) = \sum_{t \in D \cap Q} w\big(t, D\big)$, where $w(t, D)$ denotes the weight of query term t in document D. Our tool displays the value of $w(t, D)$ for each matching query term, along with the individual components (e.g., term frequency, collection frequency, document length) that are used to compute

$w(t, D)$, thus explaining how these contribute to the overall score for D.

A related feature can be used to analyse the similarity scores of two documents at a time, in order to understand why one of them is ranked higher than the other by a given IR model in response to a query.

2. I-REX can also display the candidate expansion terms for a query (and their weights), as determined by various standard PRF methods. Additional statistics about these terms, e.g., their occurrence counts in relevant and non-relevant documents, may be obtained to further understand how 'appropriate' these are as expansion terms.

I-REX can be run as a Web service that can be accessed via a browser. It also has a text-based front-end that can be used like an interactive shell from any terminal. A version of I-REX that provides access to several TREC ad hoc collections is available at http://irlab.isical.ac.in:8080/i-rex. The source code for the tool, along with instructions for installing and running the service, may be obtained from https://github.com/souravsaha/I-REX.

Key contributions. We believe I-REX makes the following useful contributions.

1. It provides researchers with a more interactive interface to Lucene, along with tools that help to understand or explain what is going on beneath the hood. Note that, while end-users who use Lucene as a "black box" for searching also need an interactive interface, these users' requirements are quite different from the needs of researchers.

2. I-REX aids reproducibility / repeatability by potentially providing a white box view of any proposed method. If the re-implementation of a proposed technique gives significantly different results from those reported by the original authors, the details provided by I-REX should help in identifying precisely where the results diverge.

3. I-REX also helps as a teaching tool, as it provides a look at the 'nuts and bolts' of standard IR operations like term-weighting and query expansion.

In the next section, we justify our choice of Lucene as our starting point, and explain why we believe that I-REX addresses a need that is not currently fulfilled. A summary

of commands provided within I-REX is provided in Section 1.4. Next, in Section 1.5, we present an example workflow to illustrate how the features of I-REX may prove useful to researchers. Section 1.6 concludes this article with some of our ideas for extending I-REX.

1.3 Background

The IR community has a long tradition of building and sharing open source IR systems, with SMART [8] likely being the oldest. A number of other systems have become available over the years (both [2] and [43] provide lists of such systems), with Lemur / Indri[2] and Terrier[3] being the most notable among them. These systems are naturally large and complex. Thus, it is usually difficult for others to effectively use them in the way the designers meant them to be used; implementing one's own ideas within the framework of such a system is even harder. This may explain why no single system has emerged as a standard. The lack of a standard has, in turn, become somewhat of a barrier to reproducibility, sound baselines, and fair comparisons. In such an environment, the persuasive arguments of Lucene 'evangelists' and the increasing use of Lucene in the academic community [43, 42, 2, 3] offer some hope. We thus chose to build our tool on top of Lucene. Preliminary efforts surrounding Lucene (e.g., [2]) focused simply on "how to use Lucene to perform typical IR operations (i.e. indexing, retrieval, etc.) as well as how to extend and modify Lucene to extract term statistics, implement different ranking models, etc." More recently, the Anserini system [43, 42] was created on top of Lucene with the objective of enabling its users to very easily conduct TREC-style ad hoc retrieval experiments on standard test collections (including modern web-scale collections), and to replicate competitive baselines "right out of the box."

Other systems with somewhat similar aims as I-REX include Luke[4], Clue[5], and Splainer[6]. Luke and Clue are relatively simple tools that provide, respectively, a graphical user

[2] https://www.lemurproject.org/indri.php
[3] http://terrier.org/
[4] https://code.google.com/archive/p/luke/
[5] https://github.com/javasoze/clue
[6] splainer.io

interface and a command-line application. These tools may be used for browsing and modifying Lucene indexes and posting lists, as well as searching in the index with keyword queries. Although these tools provide basic functionality like showing the term vector for a document, and performing simple interactive searches, they lack several useful features that could be useful to IR researchers. Splainer appears to be more ambitious, and claims to provide insights into existing systems based on Solr and Elasticsearch. However, it only seems to provide search-term highlighting; there does not seem to be a way to find useful expansion terms for a given query and a document collection, or to compare the rank of a document within a collection, as determined by two different retrieval models for the same query.

1.4 I-REX features

`I-REX` can be run as a Web service accessible through any browser. It also provides a shell-like interface that may be invoked from the command-line. Various default settings (e.g., the path to the Lucene index for a document collection, the retrieval model to use and its parameters) can be specified in a configuration file, and overridden through the Web interface (or using command-line options). Below, we list the commands supported by `I-REX`. The argument(s) accepted by a command are written in a constant-width font (like `this`).

- `cf` returns the collection frequency of the `term`.
- `compare` compares a pair of `documents` for the `query` and the `retrieval model` (see Fig 1.5 for an example).
- `diff` compares a `pair of .res files` — result files in TREC format (see Fig 1.2 for an example).
- `df` returns the document frequency of the `term`.
- `dl` returns the length (# tokens) of the `document`.

- `dump` dumps the textual content of the `document`.
- `dv` returns the document vector of the `document`.
- `expansion` returns candidate expansion terms for the `query`.
- `explain` explains the selection of the `document` by the `retrieval model` using the `query` (see Fig 1.3 for an example).
- `pl` returns the posting list of the `term`.
- `rank` returns the rank of the `document` for the `query`, `set of documents` and the `retrieval model`.
- `search` performs search using the `query` with the `retrieval model` provided as parameters.
- `sigtest` takes a `pair of res files`, computes standard evaluation measures (e.g., MAP) for these files, and reports whether the metrics are statistically significantly different (as determined by a paired t-test).
- `stats` returns basic statistics about the index.
- `tf` returns the term frequency of the `term` in `document`.
- `docsimilar` computes the similarity score of a pair of documents i.e how similar two documents are.

The use of basic commands like `cf, df, tf` etc. are straightforward. In the following section, the utility of the other commands is explained with examples.

Usability features.. The Web interface to `I-REX` is organised into three columns: the first column contains a menu of available commands; the middle column displays a short message about the usage of a selected command; the third column consists of a sequence of cards or panels, each of which contains the output of a command that was run earlier. These panels may be individually collapsed or expanded as required; they may also be

deleted. An entire session can be saved for later perusal. The I-REX shell makes use of apache.commons.cli, and thus provides standard convenience features such as command-line editing and navigating the command history.

1.5 Illustrative workflows

To highlight some of the important features of I-REX, we consider the problem of explaining the difference in the performance of the Language Model when Dirichlet smoothing (LM-Dir) and Jelinek-Mercer smoothing (LM-JM) are used (with their respective optimal parameter settings) on a particular test collection: the WT10G document collection and the TREC 9 topic set for this example.

Let trec9-lmdir1000.res and trec9-lmjm0.2.res denote the two result files obtained for this collection using LM-Dir and LM-JM, respectively. We start by running sigtest, as shown in Figure 1.1.[7] This internally runs trec_eval[8], and displays some standard evaluation metrics (configurable) as well as the result of a paired t-test. From the output of

```
$ sigtest trec9-lmdir1000.res trec9-lmjm0.2.res
AP      avg1 = 0.2237 avg2 = 0.1555 p-value = 0.001414
P@5     avg1 = 0.3440 avg2 = 0.2160 p-value = 0.000745
Recall  avg1 = 0.6771 avg2 = 0.5980 p-value = 0.058544
```

Figure 1.1: Use of sigtest on a pair of result files. Significance is tested for three evaluation metrics: AP, P@5, and recall at rank 1000.

sigtest, we observe that MAP and P@5 values for LM-Dir and LM-JM are significantly different. This warrants a more detailed investigation.

Next, we use diff to identify the queries that contribute most to the observed difference. A subset of the output of diff is shown in Figure 1.2. The figure displays some documents retrieved by the two models for TREC topic 472 (*Antique Appliance Restoration*). diff focuses on relevant documents that are retrieved at top ranks by the better performing model (trec9-lmdir1000.res), but are ranked poorly, or not retrieved at all,

[7] In this paper, we show the output as produced by the text-based interface for better readability. The Web interface is available at http://irlab.isical.ac.in:8080/i-rex.

[8] https://trec.nist.gov/trec_eval/

by the model with inferior performance (`trec9-lmjm0.2.res`). The complete output also includes non-relevant documents that are retrieved at top ranks by the inferior model, but are ranked poorly, or not retrieved at all, by the better model. In Figure 1.2, the

```
$ diff -f1 trec9-lmdir1000.res -f2 trec9-lmjm0.2.res
QID   DOCID         R1  SIM1   R2   SIM2 RANK-DIFF REL
472 WTX048-B17-92  239  7.23  INF   0.00   INF    1
472 WTX074-B30-101  61  8.50  720  11.48   659    1
472 WTX076-B18-261  57  8.57  451  12.19   394    1
472 WTX063-B44-202 229  7.31  316  12.41    87    1
472 WTX025-B13-23    6 10.30   81  13.94    75    1
472 WTX005-B03-27   12  9.45   36  14.88    24    1
472 WTX017-B02-111  11  9.54   31  14.98    20    1
472 WTX037-B40-290   1 10.61   16  16.55    15    1
```

Figure 1.2: A subset of the output of `diff` on a pair of result files. This identifies query-document pairs that contribute most to the performance difference.

columns R1, Sim1, R2 and Sim2 indicate ranks and similarity scores for a document as determined by the two models specified using `-f1` and `-f2`. The column 'RANK-DIFF' corresponds to R1−R2. The last column 'REL' indicates whether the document is relevant or non-relevant for the query named in the QID column. From Figure 1.2, we identify `WTX048-B17-92` and `WTX025-B13-23` as relevant documents that have noticeably better ranks for LM-Dir as compared to LM-JM.

We next run the `explain` command (see Figure 1.3) on document `WTX025-B13-23`, which is ranked sixth by LM-Dir and at 81 by LM-JM. Note that the rank of a document

```
$ explain -q "antique appliance restoration" -n
WTX025-B13-23 "lmdir 1000" "lmjm 0.2"
term    cf     idf    tf col-prob  s(M1)  s(M2)
antiqu 26750  4.8577 105 0.0000  6.7493  8.5732
restor 62932  3.9939  10 0.0001  3.5522  5.3708
rank(WTX025-B13-23, M1):     6
rank(WTX025-B13-23, M2):    81
doc-len(WTX025-B13-23):   1536
Total-Score(M1, WTX025-B13-23): 10.30154
Total-Score(M2, WTX025-B13-23): 13.94404
```

Figure 1.3: Use of `explain` on document `WTX025-B13-23` for LM-Dir and LM-JM.

retrieved by an IR system is dependent on the ranks of other documents in the collection.

Thus, a relevant document can drop to poor ranks in the ranked list generated by a model because the model selects other, non-relevant documents at top ranks.

To see the n top-ranked documents retrieved by LM-JM for a given query, we run `search -t` n ($n = 20$ by default). This command takes a number of flags that can be used to view additional information about the retrieved documents. In Figure 1.4, the self-explanatory flags `-rank`, `-score` and `-dl` have been used. The figure shows that the top-ranked documents are mostly quite short. As explained in [37, 34], LM-JM is known to preferentially retrieve short documents. We may thus attribute the relatively poor rank of `WTX025-B13-23` to its length (1536 tokens), which causes it to be retrieved behind many short, non-relevant documents.

```
$ search -q "antique appliance restoration"
         -r "lmjm 0.2" -rank -score -dl
  docid           rank    score   doclen
  WTX018-B08-186    0    19.35    207
  WTX098-B42-40     1    19.35    207
  WTX099-B40-178    2    19.30    144
  WTX099-B40-207    3    19.30    131
  WTX038-B50-342    4    17.97    234
  WTX095-B09-377    5    17.40    982
  WTX046-B50-43     6    17.27    193
  WTX063-B42-189    7    17.27    216
  WTX102-B44-212    8    17.27    230
  WTX103-B01-109    9    17.27    220
```

Figure 1.4: Use of `search` with a query and a retrieval model on the index. The `dl` flag causes document length to be displayed.

Other commands provided by `I-REX` include `compare`, which analyses the scores of a pair of documents for the same query and retrieval model. The use of the command is shown in Figure 1.5. The output suggests that the non-relevant document `WTX095-B09-377` is ranked ahead of `WTX037-B40-290`, a relevant document, because it contains the query term *applianc* (stem of *appliance*), which is missing from the relevant document.

The `expansion` command can be used to obtain a list of related terms for a given query. In Figure 1.6, the command has been used with the query *Antique Appliance Restoration*. These terms may be selectively added by a user to the query during an interactive IR

```
$ compare -n1 WTX037-B40-290 -n2 WTX095-B09-377
          -q "antique appliance restoration"
          WTX037-B40-290       WTX095-B09-377
relevance:              1                    1
rank:                  16                    5
score:            16.5561              17.4025
docLen:               201                  982
tf(antiqu):            16                    4
tf(applianc):           0                    9
tf(restor):            18                    2
score(antiqu):     8.6480               5.8781
score(applianc):      0.0               7.1870
score(restor):     7.9080               4.3373
```

Figure 1.5: Use of `compare` on a pair of documents to see the relative scores for the individual query terms.

session.

```
$ expansion "antique appliance restoration" 12
automobil       art             audio
equip           transport       air
advertis        museum          aircraft
classic         manufactur      architectur
```

Figure 1.6: Use of `expansion` to see a set of potentially related terms that can be used for use expansion.

Finally, `dv` may be used to view the complete vector for a given document. When the `-d` flag is passed, as in Figure 1.7, `dv` displays the most important or discriminative terms for the document. In this example, the score for each term in `WTX037-B40-290` (another relevant document for TREC topic 471) has been computed using the Dirichlet-smoothed language model, and the top 5 terms have been displayed. Of course, both the model and the number of terms to be displayed are configurable.

1.6 Conclusion and future work

`I-REX` provides features that would help researchers to analyse, understand and explain experimental results. Implementations of common term-weighting models (the Vector Space Model, BM25, Language Modeling) are already provided by Lucene. As deep

```
$ dv -n WTX037-B40-290 -d 5 "lmdir 1000"
term          cf    idf tf  score
minicraft     80 10.99  1  8.7039
cabinetmak   336  8.98  2  7.9716
craftwork    211  9.31  1  7.7420
holtzman     249  9.20  1  7.5772
currier      491  8.43  1  6.9005
```

Figure 1.7: Use of the command `dv` with flag `-d` to see the 5 discriminating terms of the document `WTX037-B40-290`.

and/or more complex IR models evolve, we have seen deep learning models can often be biased [17] or represent myopic results [6]. Thus, a tool like `I-REX` is likely to become even more useful in order to make sense of the massive number of word vector features, parameters and the complex interactions between them. Accordingly, the next thing on our agenda is incorporating support for analysing learning to rank retrieval systems. If `I-REX` is adopted by a reasonable number of researchers, we expect the current version to serve as a core around which additional functionality can be built.

Chapter 2

Open Domain Complex Question Answering

2.1 Introduction

Traditional Information Retrieval (IR) techniques generally address the problem of retrieving some relevant documents for a given user query. If the query corresponds to a specific and focussed information need, manually reading through various documents to find the required information may be considered to be unnecessarily time-consuming. In recent times, search engines (SEs) try to address this problem by displaying small and focussed information nuggets related to the query. Figure 2.1 shows an example: in response to the question *"Who is the founder of Kolkata?"*, Google displays a short snippet with the answer. However, most search engines usually fail to answer more complex questions. Figures 2.2 and 2.3 show examples of such questions that were not answered by Google and Bing, as of 19th June, 2020. Open-domain, complex question answering therefore remains an active area of research within the IR and Natural Language Processing (NLP) communities.

2.1.1 Background and problem statement

Prager [31] presents a comprehensive history of QA systems starting from the earliest days, and provides a detailed discussion of the QA track organised by TREC, starting in

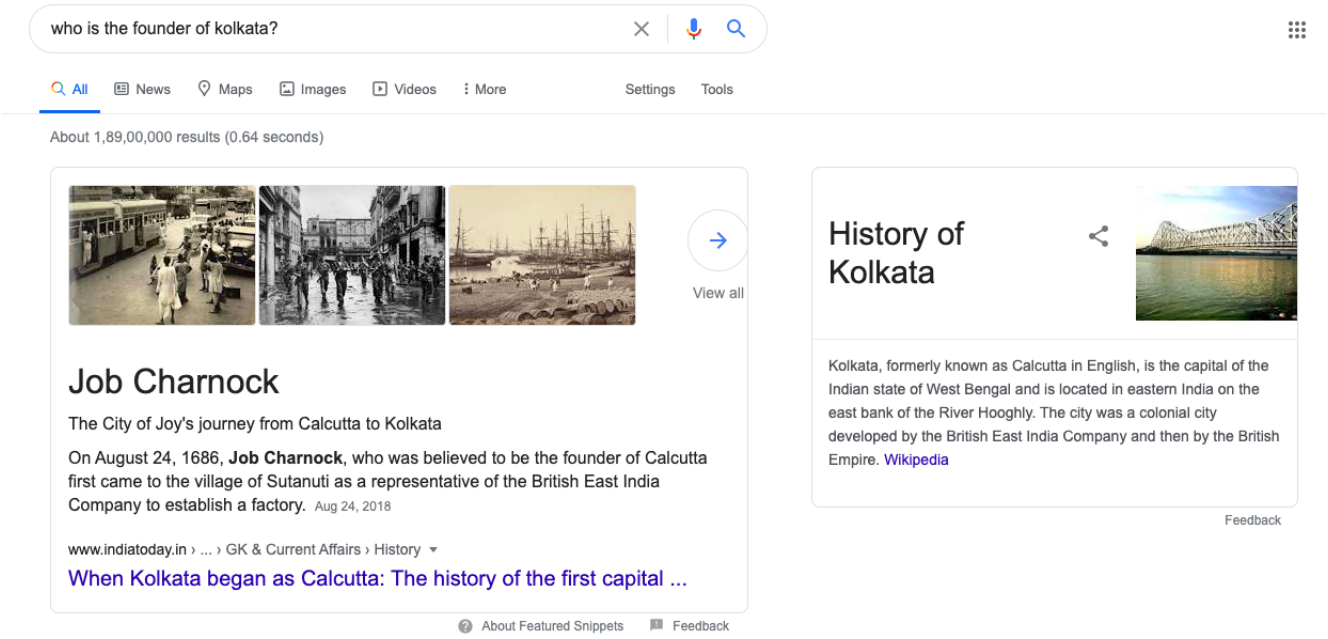

Figure 2.1: Question Answering in Search Engine.

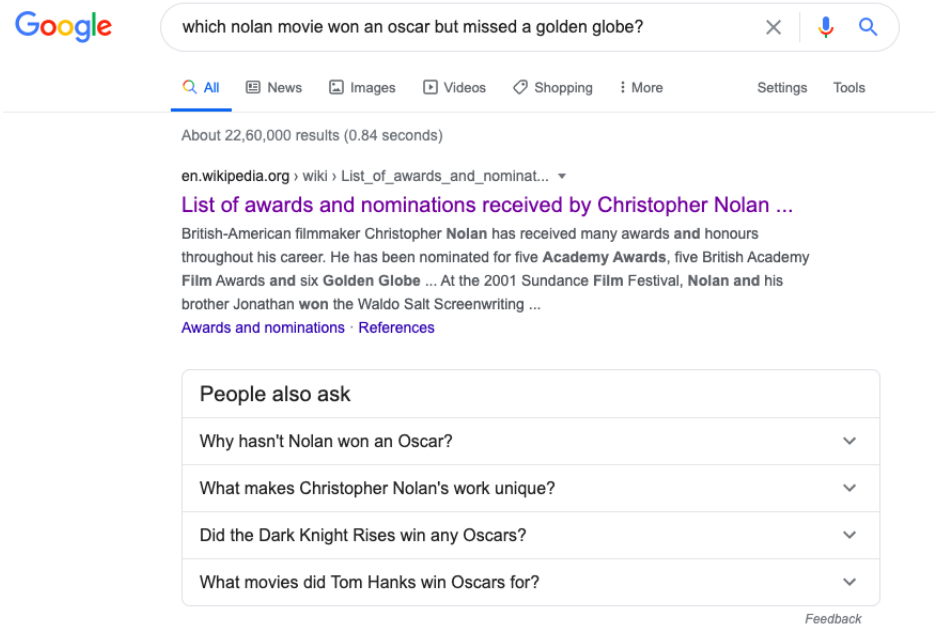

Figure 2.2: Complex Question in Google.

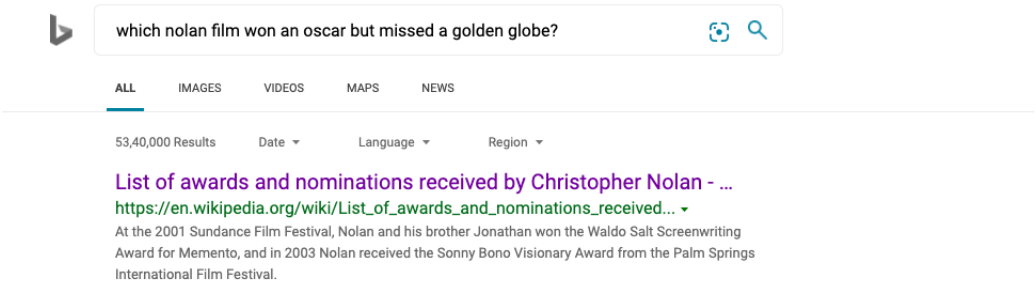

Figure 2.3: Complex Question in Bing.

1999. The TREC 1999 QA track required systems to return 50-byte (or 250-byte) long character sequences, instead of full-text documents [39]. Examples of questions asked in that track are given below.

- *Who leads the star ship Enterprise in Star Trek?*
- *Who played the part of the Godfather in the movie, "The Godfather"?*
- *What is the tallest building in Japan?*
- *Which country is Australia's largest export market?*

The track had around 200 questions, most of which have a fact-based, short answer that may be found within a single document in the collection. In the next year, 500 questions were added. The QA track continued till 2004. Over the years, newer and more challenging types of questions were added. The document set considered in TREC QA track was the Financial Times Limited (1991-1994), Congressional Record of the 103rd Congress (1993), the Federal Register (1994), Foreign Broadcast Information Service (1996), and the Los Angeles Times (1989-1990).

The development of Knowledge Graphs (KGs) such as YAGO, DBpedia, Wikipedia, provided a boost to QA research. A Knowledge Graph is a directed graph that models subject (S), predicate (P), and object (O) relations found in text. The vertex set V of a KG consists of $\{S, O\}$, while the predicates $\{P\}$ correspond to the edges. Briefly, using KGs for QA involves extracting entities from questions, and aligning them with the nodes of a KG via structured query processing. An overview of recent work on QA with KG can be found in [10].

There has been a substantial amount of recent research work in this field, but many of the best known methods are supervised in nature [45, 44, 12, 32]. Systems are trained using a large number of samples consisting of question, paragraph and answer triples. Given a test query and some text, the system tries to extract an answer using the model(s) constructed on the basis of the training data. Creating adequate quantities of labeled data can be very expensive. Yet, such expensive human annotations may be necessary when training the system to handle new types of document collections. Further, state-of-the-art methods for supervised QA generally employ Deep Learning based techniques that are computationally expensive to develop, and which need GPUs for effective deployment.

Problem statement. In this dissertation, therefore, we focus on *unsupervised*, *extractive* question answering.

- **Unsupervised QA.** In an unsupervised setting, QA involves finding an answer, given only a question Q and the corpus C [22]. Note that, modules used within an unsupervised QA pipeline could well involve supervised or transfer learning techniques for particular sub-tasks. For example, the pipeline may make use of a Part of Speech (POS) tagger that is trained on some text annotated with POS tags. However, we completely avoid the kind of training data that is required for a supervised, *end-to-end* question answering system.

- **Extractive QA.** the answers provided by the system correspond to sequences of words extracted from an existing document; no attempt is made to synthesize or generate a natural language answer.

We base our investigations on the techniques and datasets described in a very recent article on an unsupervised QA system called Quest (Lu et al., SIGIR 2019 [26]). The techniques presented in this report are both conceptually and computationally simpler, but the best results achieved by the proposed methods are significantly better than those obtained using Quest.

Chapter outline. In the next section, we describe the Quest system, focusing both on

the techniques used, as well as the dataset used for Lu et al.'s experiments. We also briefly discuss DrQA [11], another QA system that is used as a point of comparison in [26]. We believe there are certain serious drawbacks in the evaluation protocol used by Lu et al. In Section 2.3, we propose a set of modified evaluation measures that address these drawbacks. Next, in Section 2.4, we review the overall architecture of unsupervised QA systems. For each component of this overall scheme, we tried a number of different variations. These are described in Section 2.4. Experimental results are presented and discussed in Section 2.4.5. Finally, Section 2.5 provides a summary of our findings, and lists issues that we would like to study in further detail in the future.

2.2 Baseline Unsupervised QA System

The Quest dataset [26] consists of 150 complex fact-centric questions from WikiAnswers, and another 150 complex questions using emerging entities from Google Trends. Two examples are given below.

- *"Which aspiring model split with Chloe Moretz and is dating Lexi Wood?"*
- *"What movie did Russell Crowe and Denzel Washington work on together?"*

The questions are described as *complex* because the answer to a question is not found in any single document within the collection. Instead, evidence from multiple documents must be combined in order answer these questions. In the rest of this section, we provide an overview of the Quest architecture, as well as DrQA.

2.2.1 Quest

Quest focuses on complex questions which refer to multiple entities and their relationships. It finds answers by building and analysing a Quasi Knowledge Graph that is constructed using documents that are retrieved by a search engine in response to the question. Quest uses a completely unsupervised approach. A brief overview of the Quest processing pipeline is given below.

Question Pipeline

- Given a question Q, Quest starts by using Q as a keyword query to retrieve a set of documents D from the Web.
- From the 10 top-ranked documents, subject-predicate-object (SPO) triples are extracted using Open Information Extraction (OpenIE) tools [27, 21].
- It merges the extracted SPO triples from all the 10 documents to build a noisy KG. The entity nodes in the graph are then annotated with their types. For example, if the question is *"Which Nolan films won an Oscar but missed a Golden Globe?"*, the quasi-KG contains an entity node labeled *Inception*. Two nodes labeled *type* and *science thriller* are added to the graph and connected to the *Inception* node to form a chain of the form *Inception* $\rightarrow$ *type* $\rightarrow$ *science thriller*.
- Next, for each node, a similarity score is computed based on word overlap with the question. Nodes having a high similarity are designated as *cornerstones*.
- A Group Steiner Tree (GST) [20] algorithm is run on the KG to find a minimum weight spanning tree $\mathcal{T}$ containing all the *cornerstones*. More specifically they used GST-k [20, 16, 23] to obtain top-k trees for their experiments.
- From the resultant tree $\mathcal{T}$, they removed the cornerstones and rank the remaining entities based on their type and their presence in the multiple GSTs and merge the similar entities.

Figure 2.4 shows an example of the output obtained by running the GST algorithm on a quasi-KG constructed by Quest. The input question was *Who played for FC Munich and was born in Karlsruhe?* and the node *kahn* got the 1st rank.

Dataset

As mentioned above, Quest contains two sets of questions: 150 complex questions from WikiAnswers (denoted CQ-W), and another 150 complex questions from Google Trends

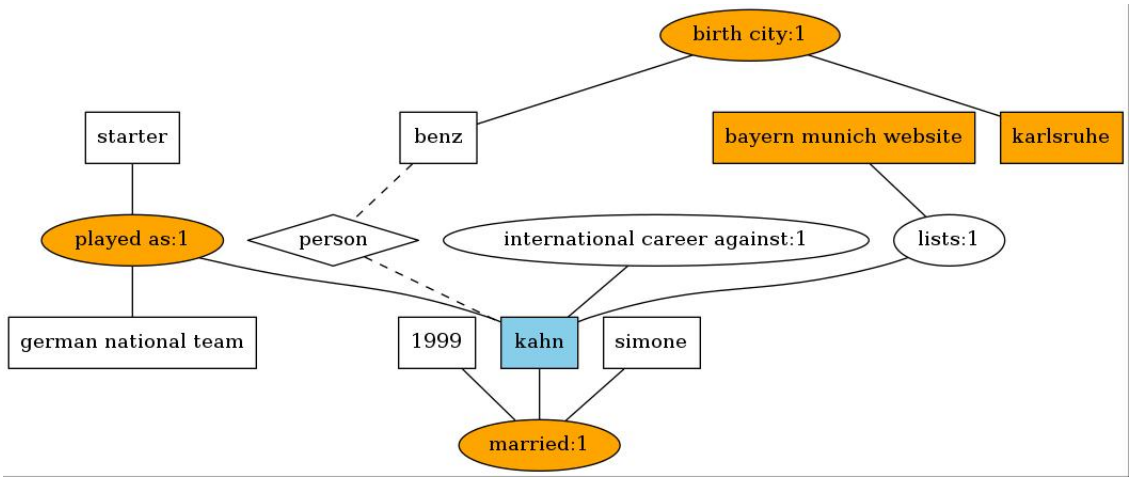

Figure 2.4: Top GST result by Quest.

(denoted CQ-T). The document corpus from which answers were extracted was constructed in one of several possible ways.

- The document collection consists of the top 10 documents retrieved by the Google Search API when the question Q is issued as the query. To ensure reproducibility and to eliminate dependencies on Google, these documents are also distributed as a part of their dataset.

- To simulate small variations in the search algorithm, additional collections were constructed using stratified sampling [40]. They took x_1% from the top 10 documents, x_2% from (0.1*x_1 + 1) to 25, x_3% from 26 to 50 documents. The settings of x_1-x_2-x_3 are: 60-30-10 (*Strata 1*), 50-40-10 (*Strata 2*), 50-30-20 (*Strata 3*), 40-40-20 (*Strata 4*), 40-30-30 (*Strata 5*).

  Because we tried a very large number of variations when formulating our approach, we used only the first document collection in our experiments.

Evaluation

Mean Reciprocal Rank (MRR), Precision@1 (P@1), and Hit@5 values are reported in [26]. MRR is regarded as the main metric. MRR and P@1 have their usual interpretations; the Hit@5 value for a query is 1 or 0 depending on whether the correct answer is found within the top 5 positions or not.

2.2.2 DrQA

DrQA [11] is used as the main point of comparison for Quest in [26]. It is a distantly supervised model, that has been trained on SQuAD [32], TREC Questions [4], WebQuestions [5] and WikiMovies [30]. DrQA has two modules.

- **Document Retriever** : This module retrieves 5 relevant articles related to each question. A TF-IDF based weighting scheme is used to compare the questions and the articles, and to rank the documents. Also they are considering bigram count between the query and document pair to improve the performances.

- **Document Reader** : It takes the retrieved documents from the previous stage and extracts all the paragraphs $\{p_1, p_2, ...p_n\}$ from those. The model encodes each paragraph p_i as a multi layered bidirectional long short term memory (LSTM) by encoding all the words present in p_i with four features the embedding of each word, exact match with the question, {part-of-speech (POS), whether it is a named entity or not, and normalized term frequence (tf)}, attention with the similar words present in the paragraph to the question. For questions q, it is using a simple recurrent neural network (RNN) on top of the word embeddings for each query words. Now to generate the start and end token of the answer they trained two classifiers to predict the two ends of the tokens for each paragraph vector p_i with the q vector. Finally take the argmax over all the paragraph present in the retrieved documents.

As described in the Quest paper they ran both of the above modules on the Quest datasets.

2.3 Proposed Evaluation Measure

The ranked lists of answers returned by Quest often contain ties, i.e., the system retrieves a number of different entities as possible answers at the *same* position of the ranked list. The authors confirmed that they did not implement any tie-breaking mechanisms. More importantly, the evaluation script used by Quest[1] does not take these ties into account

[1] `https://quest-sys.mpi-inf.mpg.de/`

when computing MRR, P@1 and Hit@5. The figures reported in [26] are therefore likely to be highly inflated over-estimates. For example, suppose that, in response to a particular question, Quest returns a list of possible answers that contains 10 different entities at rank 1. If any one of these matches the gold-standard answer, then the MRR and P@1 values (and obviously Hit@5 as well) calculated for this query is 1!

To overcome this significant drawback of Quest's evaluation method, we propose the *adjusted P@1*, *adjusted MRR*, and *adjusted Hit@5* measures, which can be calculated for lists with ties, but which are not as over-optimistic as the figures reported in [26]. These measures are essentially the expected values of the original measures under the assumption that ties are broken randomly, and all permutations of the answers at a particular rank i are equally likely.

- ***adjusted P@1*** : We take the number of correct entity retrieved at *1st* position divided by the total number of entities retrieved at the same position. Mathematically it can be defined as,

$$\frac{1}{|Q|}\sum_{q\in Q}\frac{\#\text{ of correct answers retrieved at } \textit{position 1}}{\text{total }\#\text{ of answers at } \textit{position 1}}$$

- ***adjusted MRR*** :

Rank	# entities
1	n_1
$\vdots$	$\vdots$
i	$n_i \longleftarrow$ 1st correct answer here

Let n_i denote the number of answers retrieved at rank i, and also $N_i = \sum_{j=1}^{i} n_j$; i.e., N_i denotes the total number of answers retrieved up to rank i (inclusive). Suppose at rank i, k answers are correct and $n_i - k$ many answers are incorrect.

The idea is to find the expected rank of the first correct answer. Assume that at rank i the position of *1st* correct answer is j (Figure 2.5). Now, fill jth position in 1 of k distinct ways and after jth position i.e remaining answers can permute

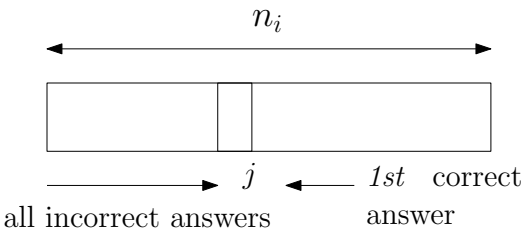

Figure 2.5: 1st correct answer at ith rank.

in $(n_i - j)!$ ways. For the first $j-1$ positions we can select it from $n_i - k$ wrong answers i.e ${}^{n_i-k}P_{j-1} ways$.

So, in total ${}^{n_i-k}P_{j-1} \times k \times (n_i - j)!$ many arrangements for the correct answer at j.

Let, r be a random variable denoting the rank of the first correct answer within the ith group. Assume P(a_j) = Probability of observing the 1st correct answer at the j th position within ith group. Expected value of the rank of the answer (within the i th group) of the 1st correct answer,

$$E(r) = \sum_{j=1}^{n_i-k+1} j \times \underbrace{\frac{{}^{n_i-k}P_{j-1} \times k \times (n_i - j)!}{n_i!}}_{\mathrm{P}(a_j)}$$

$$ExpectedCaseMRR = \frac{1}{N_{i-1} + E(r)}$$

$$BestCaseMRR = \frac{1}{N_{i-1} + 1}$$

$$WorstCaseMRR = \frac{1}{N_i - k + 1}$$

- ***adjusted Hit@5*** : Hit@5 is defined as whether a correct answer is found within top 5 positions. Let i be such that $N_{i-1} < 5$ and $N_i \geq 5$ i.e the position where the first correct answer found have more than 5 answers as shown in Figure 2.6. Note that if $N_i < 5$, Hit@5 will be 1. We don't need to do anything extra. For the former case we proceed as follows,

  Probability that at least one correct is in *1st* of m positions = 1 - Probability that all *1st*

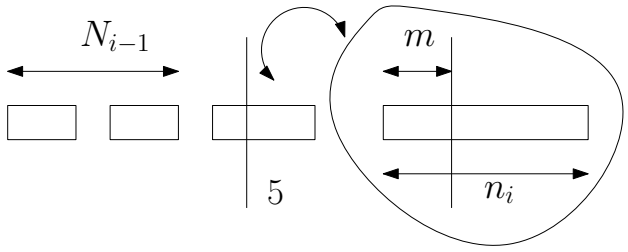

Figure 2.6: Ranked list pictorial view.

m positions are occupied by incorrect answers

$$n_i = \begin{cases} k & \text{correct} \\ n_i - k & \text{incorrect} \end{cases}$$

$$Hit@5 = 1 - \frac{^{n_i-k}P_{5-N_{i-1}} \times (N_i - 5)!}{n_i!}$$

We assume if $n_i - k < 5 - N_{i-1} \implies$ The *1st* k positions are less than the number of incorrect answers. No way the incorrect answers can occupy all the *1st* k positions. Hence we make, $^{n_i-k}P_{5-N_{i-1}} = 0$.

Corner case, if more than 5 answers are retrieved at 1st position i.e $N_1 \geq 5$ then we consider $N_0 = 0$.

2.4 Our Work

Fig 2.7 shows the general Question Answering (QA) architecture. At first, given a question $\mathcal{Q}$ usually we retrieve top k documents using various IR models. Then we are running a classifier on the questions and determine the type of it. Next we extract the named entities from the retrieved documents and apply some methods and algorithms to rank and re-rank the entities. Next section will discuss the each sub module in detail.

In our case we got the top 10 and various stratified sampled documents retrieved from Quest which we are using for our QA problem, therefore not running any retrieval to retrieve top k documents.

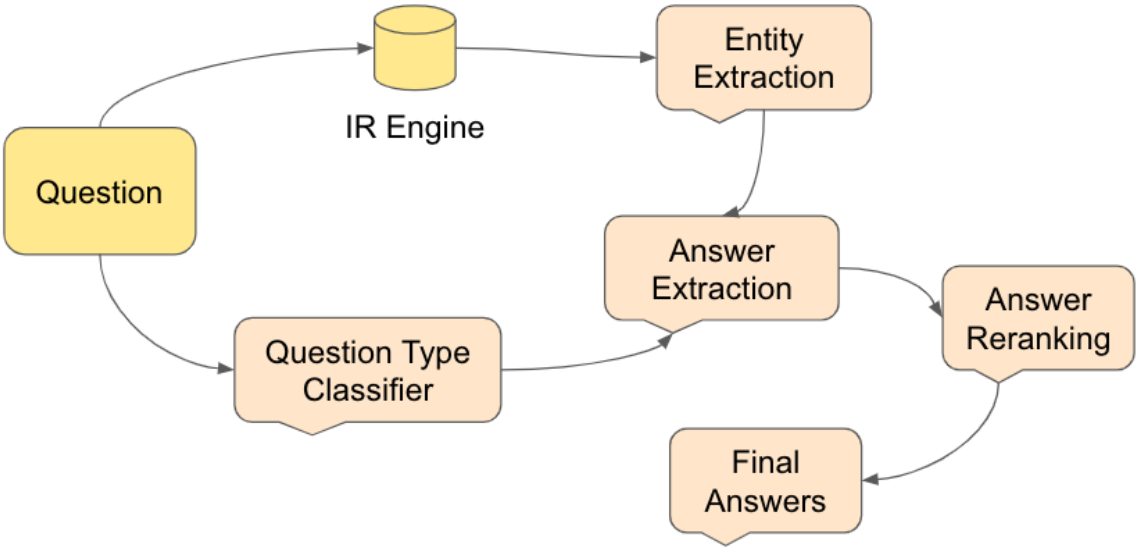

Figure 2.7: Question Answering architecture.

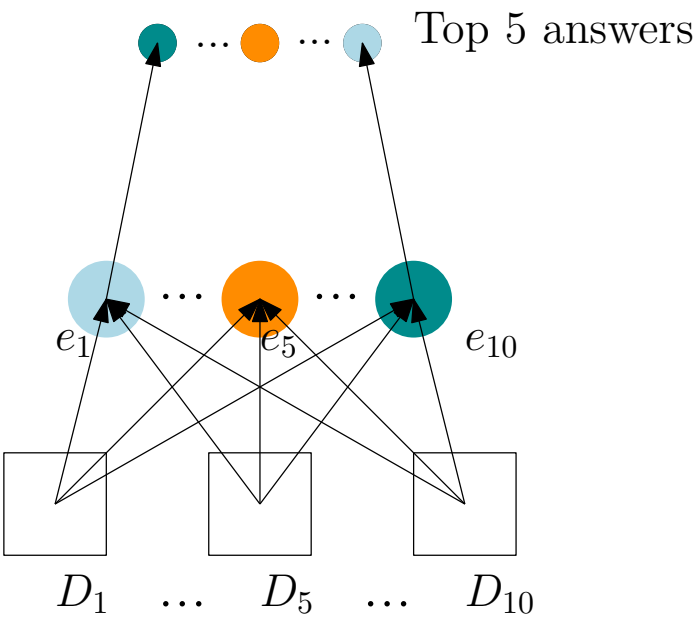

Figure 2.8: Similarity measure of entities.

2.4.1 Question Type Classifier

We used UIUC benchmark set (TREC 2004) [24] to train the question type classification module. The dataset contains 50 finer classes and 6 coarse type (Table 2.1). We are using named entities, lemmas, POS tags, syntactic dependency relation as a set of features $\mathcal{F}$ from the training dataset. The idea is each type of the question classes will have the same semantic similarities [24] and those set of features will be activated for that specific type of question class.

As, we have seen from the past literature [35] that Support Vector Machine (SVM) performs very well for this task, we ran a linear SVM with the above features. However, removing the syntactic dependency relation slightly increases the accuracy, so we removed it from the feature space $\mathcal{F}$. For the *CQ-T* and *CQ-W* we use this model to determine the coarse and fine grained type of the questions.

To avoid the engineered features we also used a recent deep learning based transfer learning architecture to classify questions. [9] We are using Google Universal Sentence Encode (USE) [9] to embed the questions into a $\mathbb{R}^d$ dimensional vector. Next we fed a simple feed forward neural network with one dense layer and a softmax layer as the output layer. We train the network with adam optimizer and relu as an activation unit. We used cross entropy as the loss function defined as follows.

$$\mathcal{L} = -\sum_{i=1}^{N}\sum_{k=1}^{K} \mathbb{I}_{i\in k} \log P(y_i)$$

Here N is the batch size, K is the number of labels at output layer i.e number of different types of coarse or fine category respectively, $\mathbb{I}_{i\in k}$ is an indicator random variable denoting the ith sample belongs to kth class, $\log P(y_i)$ is the output softmax probability. We trained two network separately for the coarse and fine grained category. Similar to initial approach we ran CQ-T, CQ-W questions on this network and predict the coarse and fine grained type of the questions.

Class	Class
ABBREV	description
abb	manner
exp	reason
ENTITY	HUMAN
animal	group
body	individual
color	title
creative	description
currency	LOCATION
dis.med.	city
event	country
food	mountain
instrument	other
lang	state
letter	NUMERIC
other	code
plant	count
product	date
religion	distance
sport	money
substance	order
symbol	other
technique	period
term	percent
vehicle	speed
word	temp
DESCRIPTION	size
definition	weight

Table 2.1: TREC 2004 coarse and fine grained labels. Upper case labels are of type coarse and lower cases are fine grained type.

2.4.2 Extracting Named Entities

First we pre-process the retrieved documents $\mathcal{D}$ by normalizing the unicode characters and then expanding the contractions present in the sentences like, *"I can't do..."* to *"I can not do..."*, etc. Named entities (NEs) capture the semantic relationship in a sentence as well as in a paragraph. Next we extract the named entities $\mathcal{E}$ present in the each

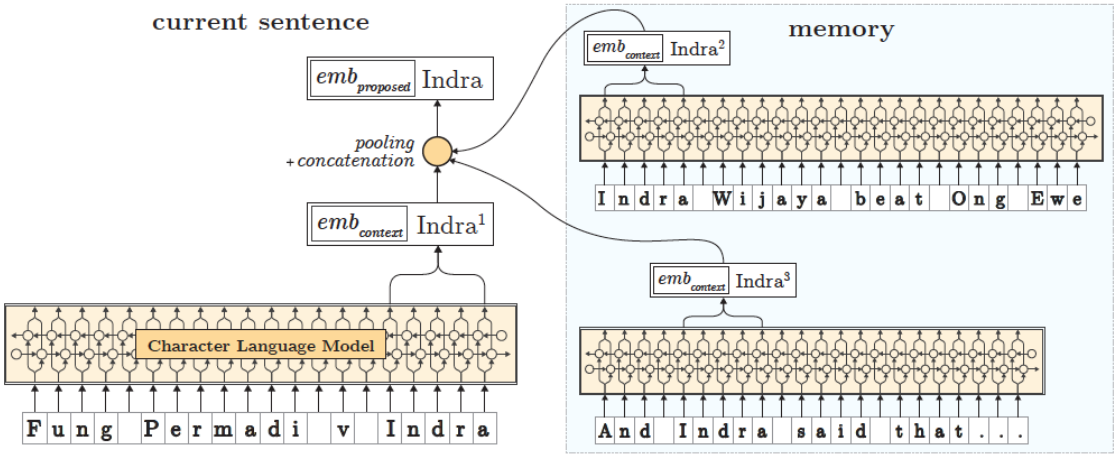

Figure 2.9: Flair embedding of words.

set of documents $\mathcal{D}$. Initially we started with spaCy[2] to harvest the named entities and later we used state-of-the-art deep learning based embeddings. Spacy has been trained on *OntoNotes 5* corpus and their entity label information is shown in the Table 2.2. We also used a recent contextualized based embedding Flair [1] to extract the down straem named entity recognition (NER) tasks. It uses a pooling based contextual string embeddings to recognize the named entities. They are using a character level contextualized embeddings for each words. Next it uses a memory to store the embeddings of each unique word tokens and take the pooling (min, max, avg) operation of this memory. It then concatenates the pooled version of embeddings with the original contextualize embeddings. It has achieved a recent state-of-the-art result and good accuracy (F1 score). In our case as compared to spaCy, entities like "...*fc barcelona*..." of type *ORG* from sentences can be captured by using Flair. Figure [1] shows an example of their model.[3] At this stage we are having a bag of entities $\mathcal{E}$ collected from the retrieved set of documents $\mathcal{D}$ which are relevant for the Question $\mathcal{Q}$ in a broader aspect.

[2]https://spacy.io/
[3]The figure is taken from [1] paper

Class	Description
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
DATE	Absolute or relative dates or periods.
TIME	Times smaller than a day.
PERCENT	Percentage, including "%".
MONEY	Monetary values, including unit.
QUANTITY	Measurements, as of weight or distance.
ORDINAL	"first", "second", etc.
CARDINAL	Numerals that do not fall under another type.

Table 2.2: Ontonotes coarse and fine grained labels. Upper case labels are of type coarse and lower cases are fine grained type.

2.4.3 Answer Extraction

Now after extracting the named entities $\mathcal{E}$ we order them based on the decreasing *document frequency* (df_l^t).

$$df_l^t = \sum_{n=1}^{k} \mathbb{I}(c(t,l) > 0)$$

$\mathbb{I}$ is an Indicator random variable and $c(t,l)$ = number times the named entity E_i with text t and label l is present in the document. Notice that the maximum value of df_l^t can be 10 as there are 10 retrieved documents in the set $\mathcal{D}$. The labels for named entities $\mathcal{E}$ for Trec 2004 benchmark data and OntoNotes 5 dataset are different. We wrote rules to map lables from Trec 2004 domain to OntoNotes 5 domain (Appendix B.1). This is bascially a many-to-many mapping. For each questions type ($\mathcal{T}$) we apply the above mapping to transform it into a OntoNotes 5 domain. We filter out the entities that are not classified

by the question type $\mathcal{T}$. Next, we pull out the sentences $\mathcal{S}$ from which the entities were curated. The idea is there may be some semantic relatedness in the question $\mathcal{Q}$ and the answer containing the sentences. To measure the semantic context we embed the input question $\mathcal{Q}$ and these sentences $\mathcal{S}$ to a vector space by using sentence embeddings from Infersent [14].

Infersent was trained on Stanford Natural Language Inference (SNLI) [7] datasets and uses a bidirectional LSTM with max pooling for learning the universal sentence embeddings. For each sentences they concatenated the hidden states of the forward and backward LSTM network and on top of it they are taking maximum (max pool) value over each dimension of the representations.

Now, we process top k many entities $\{E_1, E_2...E_k\}$ based on df_l^t as computed above and we compute score $Score(E_i)$ for each of the entity E_i in the following ways

- ***Approach 1*** We are taking average *Cosine Similarity* across all S' with Q i.e *Sim(Q,S')*, where $S' \in \mathcal{S}$ over all the retrieved documents $\mathcal{D}$.

$$Score(E_i) = \frac{1}{|S|} \sum_{S' \in S} Sim(Q, S')$$

- ***Approach 2*** For each document d $\in \mathcal{D}$ we take the max *Cosine Similarity* score of *Sim(Q,S^d)* , where S^d is the all sentences in Document d. Then we take the avg of this measure for each document.

$$Score(E_i) = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} \max_{S' \in S^d} Sim(Q, S')$$

- ***Approach 3*** We are taking maximum *Cosine Similarity* across all S' with Q i.e, *Sim(Q,S')*, where $S' \in \mathcal{S}$ over all the retrieved documents $\mathcal{D}$. It basically tries to see over the document sets the entity that we extracted out how much similar the

sentence with respect to the question.

$$Score(E_i) = \max_{S' \in S} Sim(Q, S')$$

Figure 2.8 shows the pictorial diagram of this method, where the entities e_i are pulled out from the document set $\mathcal{D}$ and ranked based on the approaches defined above. Alternatively one can use various other sentence embedding techniques like USE [9], Sentence-Bert [33] etc. However, these are very expensive requires lot of gpu computations.
We have also used Sentence-Bert as an alternative to Infersent [14] based Sentence embedding.
It uses pretrained BERT [15] and RoBERTa [25] and adds a mean pooling over the output of [CLS]-token to create a useful sentence embeddings and as discussed in [33] it works much faster than BERT [15] and gives better representation.
Resolving Coreferences : Coreference resolution refers to the problem of finding the linguistic mentions in the text that refers to the same entities to the sentences. We also resolve the coreferences from the documents by using NeuralCoref[4] model.

2.4.4 Answer Ranking

We rank the entities based on the following measure,

- ***Approach 0*** We are taking just $Score(E_i)$ obtained from previous stage.

- ***Approach 1*** We add the normalize document frequency value with the above $Score(E_i)$.

$$Rerank\_Score(E_i) = Score(E_i) + \frac{df(t,l)}{|\mathcal{D}|}$$

- ***Approach 2*** We multiply the normalize document frequency value with the above

[4]https://github.com/huggingface/neuralcoref

$Score(E_i)$.

$$Rerank\_Score(E_i) = Score(E_i) \times \frac{df(t,l)}{|\mathcal{D}|}$$

- ***Approach 3*** We take a mixture of $Score(E_i)$ and the normalize document frequency value, as shown below.

$$Rerank\_Score(E_i) = \alpha \times Score(E_i) + \beta \times \frac{df(t,l)}{|\mathcal{D}|}$$

Here α and β are the parameters to be tuned. Now retrieved the top 5 answers for each of the questions $\mathcal{Q}$.

Re-ranking : For doing re-ranking we are extracting named entities from each question $Q_i \in \mathcal{Q}$. We are using the approaches mentioned in Section 2.4.2 to do so. Next we used Entity2vector model from [41] to turn the entities in a embedding space (Figure 2.10). The idea is to use the traditional skipgram model [29] to learn the embeddings of the entities into a d dimensional vector space. The skipgram model learns by predicting the context words for each input word.

By using the above model we project the answer entities as well as question entities in the embedding space. Then for each entity $\{q_1, q_2...q_k\} \in Q_i$ compute centroid *Cosine Similarity* measure. We re-rank the answer entities $\{a_1, a_2, a_3, a_4, a_5\}$ based on the above similarity measure.

Finally we run the evaluation metric with MRR, P@1, Hit@5 as well as our proposed MRR (adjusted), P@1 (adjusted) and Hit@5 (adjusted) to avoid the over optimistic ranking of the system.

2.4.5 Result and Discussions

There are two types of evaluation protocol we considered. In the first one, we assume there is some oracle which will resolve ties present in the answers. Later we explicitly resolve the type and use our adjusted MRR, adjusted P@1 and adjusted Hit@5. Table 2.3, 2.4 shows

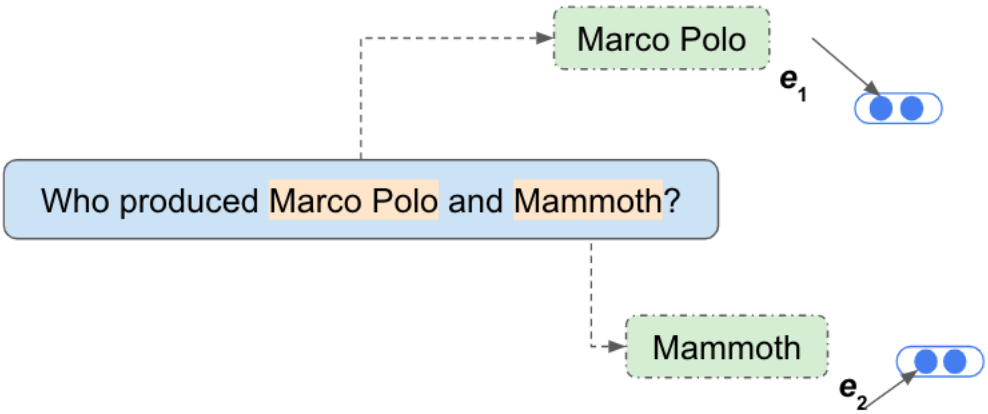

Figure 2.10: Entity extraction from questions and embedding.

the benchmark reported result and our obtained result for CQ-W and CQ-T questions respectively. The obtained benchmark results with our modified evaluation measure can be found in Table 2.5, 2.6. Some of the different methods tried out for those two complex QA sets can be found in Table 2.7, 2.8, 2.9, 2.10 for both the evaluation protocols. Result file conventions are as follows,

- Sent-Emb refers to the [14] based embedding for the sentences as described in Section 2.4.3. Sent-bert refers to the alternative embedding used as described in Section 2.4.3.

- spacy : The named entities were extracted by using spacy. Flair [1] refers to the alternate approach tried as described in Section 2.4.2.

- avg-score, avg-max-score, max-score are pointing to the approaches for scoring the entities as described in Section 2.4.3 with *Approach1*, *Approach2*, *Approach3* respectively.

- re-rank1, re-rank2, re-rank3 are referring to Section 2.4.4 ranking strategies and if no re-rank is mentioned, we are using *Approach 0* in Section 2.4.4.

- co-ref : We are resolving the coreferences as described in Section 2.4.3 and entity-re-rank is the reranking strategies adapted in Section 2.4.4 module.

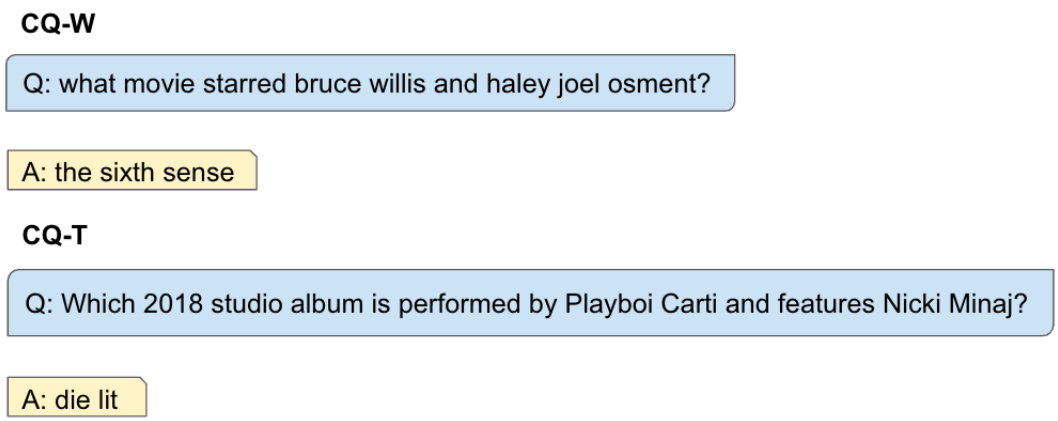

Figure 2.11: Some snippet of P@1 answer retrieval where other baselines (Quest and DrQA) fails to retrieve.

- USE (ques type) is the neural question type classifier based method as in Section 2.4.1. If no (ques type) is mentioned, we are using the old question type classifier as described in the same section.

Note that for different stratified sampling strategies on top-k retrieved documents i.e strata1, strata2, strata3, strata4 and strata5 the result do not change much. So, given the amount of computational resources and processing time it requires we are considering only top10 sampled document sets.

For, Quest the evaluation result changes with our proposed evaluation metric as it contains many ties in the ranking list. On the other hand DrQA seems to be robust in this regard. Figure 2.11 shows one example from each CQ-W and CQ-T set where our model retrieves at top position, however other baseline fails to retrieve those.

For Spacy named entities, the result shows for the entities obtained from small web corpus. We tried both the small as well as large web corpus. However, in the result it didn't give much differences. In the interests of conciseness, this section presents only the results for the techniques corresponding to Sections 2.4. Detailed results for the techniques that we tried, but which did not work very well, are not presented here. Next we compared our best performing modules with the benchmark results (Table 2.11, 2.12, 2.13, 2.14) for both the question set with the two set of evaluation protocols respectively. For CQ-W question set, approach with Sent-Emb+Flair+max-score+re-rank1 is giving best result

on our proposed evaluation measure and for old evaluation protocol approach with Sent-Emb+Flair+max-score performs best. On CQ-T question set, Sent-Emb+Flair+max-score+re-rank2 is giving best results across both the evaluation measure. We hope to do a detailed failure analysis of these techniques in future work.

	Metric		
System	MRR	P@1	Hit@5
Quest(reported)	0.355	0.268	0.376
Quest(obtained)	0.2947	0.253	0.366
DrQA(reported)	0.226	0.184	0.313
DrQA(obtained)	0.228	0.186	0.313

Table 2.3: Baseline result for CQ-W questions for Top10 data set.

	Metric		
System	MRR	P@1	Hit@5
Quest(reported)	0.467	0.394	0.531
Quest(obtained)	0.401	0.36	0.48
DrQA(reported)	0.355	0.286	0.453
DrQA(obtained)	0.355	0.286	0.453

Table 2.4: baseline result for CQ-T questions for Top10 data set.

	Metric		
System	adjusted MRR	adjusted P@1	adjusted Hit@5
Quest(obtained)	0.067	0.04	0.155
DrQA(obtained)	0.228	0.186	0.313

Table 2.5: Baseline result for CQ-W questions for Top10 data set with our proposed metric.

	Metric		
System	adjusted MRR	adjusted P@1	adjusted Hit@5
Quest(obtained)	0.065	0.043	0.154
DrQA(obtained)	0.355	0.286	0.453

Table 2.6: Baseline result for CQ-T questions for Top10 data set with our proposed metric

2.5 Conclusion and Future Work

For this dissertation, we look at the problem of unsupervised question answering. We use a very recent system (Quest) presented at SIGIR 2019 [26] as our baseline. Our best results are substantially better than those reported for Quest (an unsupervised method) and DrQA (a strong deep learning benchmark) on both the CQ-W and the CQ-T datasets. Further, in terms of processing speed, our method is faster compared to Quest. As confirmed from the Quest authors[5], it takes around 3-3.5 hours to complete one question set, i.e., 150 questions on a 256 GB server. Our model takes around 1 hour to complete the same question set on a 128GM RAM and 24 GB GPU server. As a part of future work, we plan to explore the issues listed below.

- In Entity2vec model we are getting some out of vocabulary (OOV) problem for many entities, therefore the result is not as expected and also for CQ-W most of the questions are not in capital case format. We hope to take care those for a better coverage.

- It would be great to give key question focus to the supporting words present in the questions.

- We are planning to do a detailed failure analysis for the various methods we tried out.

- Instead of converting from TREC 04 to Ontonotes 05 label, we need some automated mapping. It would be better to have a many to one or more finer grained mapping. We also observed these handwritten rules are not sometimes correct so we need to correct some of these.

[5]Personal communication

Metric	Method	Top10
MRR	Sent-Emb+spacy+avg-score	0.069
	Sent-Emb+spacy+avg-max-score	0.135
	Sent-Emb+spacy+max-score	0.386
	Sent-Emb+spacy+max-score+re_rank_1	0.333
	Sent-Emb+spacy+max-score+re_rank_2	0.331
	Sent-Emb+Flair+max-score	**0.471**
	Sent-Emb+Flair+max-score+re-rank1	0.431
	Sent-Emb+Flair+max-score+re-rank2	0.427
	Sent-bert+flair+re-rank1	0.381
	Sent-bert+flair+re-rank2	0.389
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.341
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.329
	Sent-Emb+flair+re-rank2+USE (ques type)	0.423
	Sent-bert+flair+re-rank2+entity-rerank	0.38
	Sent-Emb+spacy+max-score+co-ref	0.327
P@1	Sent-Emb+spacy+avg-score	0.04
	Sent-Emb+spacy+avg-max-score	0.06
	Sent-Emb+spacy+max-score	0.28
	Sent-Emb+spacy+max-score+re_rank_1	0.22
	Sent-Emb+spacy+max-score+re_rank_2	0.22
	Sent-Emb+Flair+max-score	**0.346**
	Sent-Emb+Flair+max-score+re-rank1	0.293
	Sent-Emb+Flair+max-score+re-rank2	0.293
	Sent-bert+flair+re-rank1	0.26
	Sent-bert+flair+re-rank2	0.273
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.22
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.22
	Sent-Emb+flair+re-rank2+USE (ques type)	0.293
	Sent-bert+flair+re-rank2+entity-rerank	0.26
	Sent-Emb+spacy+max-score+co-ref	0.2
Hit@5	Sent-Emb+spacy+avg-score	0.133
	Sent-Emb+spacy+avg-max-score	0.286
	Sent-Emb+spacy+max-score	0.56
	Sent-Emb+spacy+max-score+re_rank_1	0.54
	Sent-Emb+spacy+max-score+re_rank_2	0.54
	Sent-Emb+Flair+max-score	**0.646**
	Sent-Emb+Flair+max-score+re-rank1	0.646
	Sent-Emb+Flair+max-score+re-rank2	0.646
	Sent-bert+flair+re-rank1	0.62
	Sent-bert+flair+re-rank2	0.6
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.52
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.533
	Sent-Emb+flair+re-rank2+USE (ques type)	0.64
	Sent-bert+flair+re-rank2+entity-rerank	0.62
	Sent-Emb+spacy+max-score+co-ref	0.526

Table 2.7: Result for CQ-W data Set.

Metric	Method	Top10
MRR	Sent-Emb+spacy+avg-score	0.068
	Sent-Emb+spacy+avg-max-score	0.118
	Sent-Emb+spacy+max-score	0.342
	Sent-Emb+spacy+max-score+re_rank_1	0.377
	Sent-Emb+spacy+max-score+re_rank_2	0.377
	Sent-Emb+Flair+max-score	0.44
	Sent-Emb+Flair+max-score+re-rank1	0.441
	Sent-Emb+Flair+max-score+re-rank2	**0.442**
	Sent-bert+flair+re-rank1	0.409
	Sent-bert+flair+re-rank2	0.414
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.377
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.377
	Sent-Emb+flair+re-rank2+USE (ques type)	0.414
	Sent-bert+flair+re-rank2+entity-rerank	0.387
	Sent-Emb+spacy+max-score+co-ref	0.321
P@1	Sent-Emb+spacy+avg-score	0.04
	Sent-Emb+spacy+avg-max-score	0.067
	Sent-Emb+spacy+max-score	0.233
	Sent-Emb+spacy+max-score+re_rank_1	0.26
	Sent-Emb+spacy+max-score+re_rank_2	0.26
	Sent-Emb+Flair+max-score	0.3
	Sent-Emb+Flair+max-score+re-rank1	0.306
	Sent-Emb+Flair+max-score+re-rank2	0.306
	Sent-bert+flair+re-rank1	0.293
	Sent-bert+flair+re-rank2	**0.312**
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.26
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.26
	Sent-Emb+flair+re-rank2+USE (ques type)	0.286
	Sent-bert+flair+re-rank2+entity-rerank	0.24
	Sent-Emb+spacy+max-score+co-ref	0.206
Hit@5	Sent-Emb+spacy+avg-score	0.133
	Sent-Emb+spacy+avg-max-score	0.24
	Sent-Emb+spacy+max-score	0.52
	Sent-Emb+spacy+max-score+re_rank_1	0.54
	Sent-Emb+spacy+max-score+re_rank_2	0.54
	Sent-Emb+Flair+max-score	0.653
	Sent-Emb+Flair+max-score+re-rank1	0.646
	Sent-Emb+Flair+max-score+re-rank2	**0.646**
	Sent-bert+flair+re-rank1	0.599
	Sent-bert+flair+re-rank2	0.566
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.54
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.54
	Sent-Emb+flair+re-rank2+USE (ques type)	0.606
	Sent-bert+flair+re-rank2+entity-rerank	0.6
	Sent-Emb+spacy+max-score+co-ref	0.506

Table 2.8: Result for CQ-T data Set.

Metric	Method	Top10
adjusted MRR	Sent-Emb+spacy+avg-score	0.064
	Sent-Emb+spacy+avg-max-score	0.128
	Sent-Emb+spacy+max-score	0.307
	Sent-Emb+spacy+max-score+re_rank_1	0.324
	Sent-Emb+spacy+max-score+re_rank_2	0.322
	Sent-Emb+Flair+max-score	0.364
	Sent-Emb+Flair+max-score+re-rank1	**0.416**
	Sent-Emb+Flair+max-score+re-rank2	0.412
	Sent-bert+flair+re-rank1	0.379
	Sent-bert+flair+re-rank2	0.388
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.333
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.32
	Sent-Emb+flair+re-rank2+USE (ques type)	0.407
	Sent-bert+flair+re-rank2+entity-rerank	0.378
	Sent-Emb+spacy+max-score+co-ref	0.311
adjusted P@1	Sent-Emb+spacy+avg-score	0.036
	Sent-Emb+spacy+avg-max-score	0.063
	Sent-Emb+spacy+max-score	0.209
	Sent-Emb+spacy+max-score+re_rank_1	0.216
	Sent-Emb+spacy+max-score+re_rank_2	0.216
	Sent-Emb+Flair+max-score	0.254
	Sent-Emb+Flair+max-score+re-rank1	**0.282**
	Sent-Emb+Flair+max-score+re-rank2	0.282
	Sent-bert+flair+re-rank1	0.26
	Sent-bert+flair+re-rank2	0.273
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.216
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.216
	Sent-Emb+flair+re-rank2+USE (ques type)	0.282
	Sent-bert+flair+re-rank2+entity-rerank	0.233
	Sent-Emb+spacy+max-score+co-ref	0.186
adjusted Hit@5	Sent-Emb+spacy+avg-score	0.108
	Sent-Emb+spacy+avg-max-score	0.24
	Sent-Emb+spacy+max-score	0.488
	Sent-Emb+spacy+max-score+re_rank_1	0.526
	Sent-Emb+spacy+max-score+re_rank_2	0.526
	Sent-Emb+Flair+max-score	0.559
	Sent-Emb+Flair+max-score+re-rank1	**0.64**
	Sent-Emb+Flair+max-score+re-rank2	0.633
	Sent-bert+flair+re-rank1	0.613
	Sent-bert+flair+re-rank2	0.593
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.52
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.52
	Sent-Emb+flair+re-rank2+USE (ques type)	0.626
	Sent-bert+flair+re-rank2+entity-rerank	0.613
	Sent-Emb+spacy+max-score+co-ref	0.521

Table 2.9: Result for CQ-W data set with our new metric.

Metric	Method	Top10
adjusted MRR	Sent-Emb+spacy+avg-score	0.067
	Sent-Emb+spacy+avg-max-score	0.112
	Sent-Emb+spacy+max-score	0.288
	Sent-Emb+spacy+max-score+re_rank_1	0.372
	Sent-Emb+spacy+max-score+re_rank_2	0.372
	Sent-Emb+Flair+max-score	0.346
	Sent-Emb+Flair+max-score+re-rank1	0.433
	Sent-Emb+Flair+max-score+re-rank2	**0.433**
	Sent-bert+flair+re-rank1	0.408
	Sent-bert+flair+re-rank2	0.413
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.372
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.372
	Sent-Emb+flair+re-rank2+USE (ques type)	0.404
	Sent-bert+flair+re-rank2+entity-rerank	0.371
	Sent-Emb+spacy+max-score+co-ref	0.253
adjusted P@1	Sent-Emb+spacy+avg-score	0.04
	Sent-Emb+spacy+avg-max-score	0.06
	Sent-Emb+spacy+max-score	0.183
	Sent-Emb+spacy+max-score+re_rank_1	0.253
	Sent-Emb+spacy+max-score+re_rank_2	0.253
	Sent-Emb+Flair+max-score	0.216
	Sent-Emb+Flair+max-score+re-rank1	0.296
	Sent-Emb+Flair+max-score+re-rank2	0.296
	Sent-bert+flair+re-rank1	0.293
	Sent-bert+flair+re-rank2	**0.312**
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.253
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.253
	Sent-Emb+flair+re-rank2+USE (ques type)	0.276
	Sent-bert+flair+re-rank2+entity-rerank	0.233
	Sent-Emb+spacy+max-score+co-ref	0.145
adjusted Hit@5	Sent-Emb+spacy+avg-score	0.126
	Sent-Emb+spacy+avg-max-score	0.213
	Sent-Emb+spacy+max-score	0.466
	Sent-Emb+spacy+max-score+re_rank_1	0.526
	Sent-Emb+spacy+max-score+re_rank_2	0.526
	Sent-Emb+Flair+max-score	0.579
	Sent-Emb+Flair+max-score+re-rank1	0.646
	Sent-Emb+Flair+max-score+re-rank2	**0.646**
	Sent-bert+flair+re-rank1	0.599
	Sent-bert+flair+re-rank2	0.566
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.3, \beta = 0.7$)	0.526
	Sent-Emb+spacy+max-score+ re-rank3 ($\alpha = 0.8, \beta = 0.2$)	0.526
	Sent-Emb+flair+re-rank2+USE (ques type)	0.6
	Sent-bert+flair+re-rank2+entity-rerank	0.587
	Sent-Emb+spacy+max-score+co-ref	0.45

Table 2.10: Result for CQ-T data set with our new metric.

	Metric		
System	MRR	P@1	Hit@5
Quest(reported)	0.355	0.268	0.376
Quest(obtained)	0.2947	0.253	0.366
Dr-QA(reported)	0.226	0.184	0.313
Dr-QA(obtained)	0.228	0.186	0.313
Our Best	**0.471**	**0.346**	**0.646**

Table 2.11: Performance comparison of baseline results with our best model for CQ-W questions on Top10 data set. Evaluated with old metric.

	Metric		
System	MRR	P@1	Hit@5
Quest(reported)	**0.467**	**0.394**	0.531
Quest(obtained)	0.401	0.36	0.48
DrQA(reported)	0.355	0.286	0.453
DrQA(obtained)	0.355	0.286	0.453
Our Best	0.442	0.312	**0.646**

Table 2.12: Performance comparison of baseline results with our best model for CQ-T questions for Top10 data set. Evaluated with old metric.

	Metric		
System	adjusted MRR	adjusted P@1	adjusted Hit@5
Quest(obtained)	0.067	0.04	0.155
DrQA(obtained)	0.228	0.186	0.313
Our Best	**0.416**	**0.282**	**0.64**

Table 2.13: Performance comparison of baseline results with our best model for CQ-W questions for Top10 data set. Evaluated with our proposed metric.

	Metric		
System	adjusted MRR	adjusted P@1	adjusted Hit@5
Quest(obtained)	0.065	0.043	0.154
DrQA(obtained)	0.355	0.286	0.453
Our Best	**0.433**	**0.296**	**0.646**

Table 2.14: Performance comparison of baseline results with our best model for CQ-T questions for Top10 data set. Evaluated with our proposed metric.

Bibliography

[1] Akbik, A., Bergmann, T., and Vollgraf, R. Pooled contextualized embeddings for named entity recognition. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics* (2019), p. 724–728.

[2] Azzopardi, L., Crane, M., Fang, H., Ingersoll, G., Lin, J., Moshfeghi, Y., Scells, H., Yang, P., and Zuccon, G. The Lucene for information access and retrieval research (LIARR) workshop at SIGIR 2017. In *Proc. SIGIR* (2017), pp. 1429–1430.

[3] Azzopardi, L., Moshfeghi, Y., Halvey, M., Alkhawaldeh, R. S., Balog, K., Di Buccio, E., Ceccarelli, D., Fernández-Luna, J. M., Hull, C., Mannix, J., and Palchowdhury, S. Lucene4IR: Developing ir evaluation resources using Lucene. *SIGIR Forum 50*, 2 (Feb. 2017), 58–75.

[4] Baudiš, P., and Šedivý, J. Modeling of the question answering task in the yodaqa system. In *Proceedings of the 6th International Conference on Experimental IR Meets Multilinguality, Multimodality, and Interaction - Volume 9283* (Berlin, Heidelberg, 2015), CLEF'15, Springer-Verlag, p. 222–228.

[5] Berant, J., Chou, A., Frostig, R., and Liang, P. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (Seattle, Washington, USA, Oct. 2013), Association for Computational Linguistics, pp. 1533–1544.

[6] Bolukbasi, T., Chang, K., Zou, J. Y., Saligrama, V., and Kalai, A. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Proc. of NIPS 29*. 2016, pp. 4349–4357.

[7] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (Lisbon, Portugal, Sept. 2015), Association for Computational Linguistics, pp. 632–642.

[8] Buckley, C. Implementation of the smart information retrieval system. Tech. rep., Ithaca, NY, USA, 1985.

[9] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B., and Kurzweil, R. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (Brussels, Belgium, Nov. 2018), Association for Computational Linguistics, pp. 169–174.

[10] Chakrabarti, S. Interpretable complex question answering. In *Proceedings of The Web Conference 2020* (New York, NY, USA, 2020), WWW '20, Association for Computing Machinery, p. 2455–2457.

[11] Chen, D., Fisch, A., Weston, J., and Bordes, A. Reading Wikipedia to answer open-domain questions. In *Association for Computational Linguistics (ACL)* (2017).

[12] Clark, C., and Gardner, M. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (Melbourne, Australia, July 2018), Association for Computational Linguistics, pp. 845–855.

[13] Clinchant, S., and Gaussier, É. A theoretical analysis of prf models. In *Proc. of ICTIR '13* (2013), pp. 6–13.

[14] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing* (Copenhagen, Denmark, September 2017), Association for Computational Linguistics, pp. 670–680.

[15] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 4171–4186.

[16] Ding, B., Xu Yu, J., Wang, S., Qin, L., Zhang, X., and Lin, X. Finding top-k min-cost connected trees in databases. In *2007 IEEE 23rd International Conference on Data Engineering* (2007), pp. 836–845.

[17] Dixon, L., Li, J., Sorensen, J., Thain, N., and Vasserman, L. Measuring and mitigating unintended bias in text classification. In *Proc. of 2018 AAAI/ACM Conference on AI, Ethics, and Society* (2018), ACM, pp. 67–73.

[18] Fang, H., and Zhai, C. An exploration of axiomatic approaches to information retrieval. In *Proc. of 28th SIGIR* (2005), ACM, pp. 480–487.

[19] Fernando, Z. T., Singh, J., and Anand, A. A study on the interpretability of neural retrieval models using deepshap. In *Proc. of SIGIR'19* (2019), pp. 1005–1008.

[20] Garg, N., Konjevod, G., and Ravi, R. A polylogarithmic approximation algorithm for the group steiner tree problem. In *Proceedings of the Ninth Annual*

ACM-SIAM Symposium on Discrete Algorithms (USA, 1998), SODA '98, Society for Industrial and Applied Mathematics, p. 253–259.

[21] Kadry, A., and Dietz, L. Open relation extraction for support passage retrieval: Merit and open issues. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2017), SIGIR '17, Association for Computing Machinery, p. 1149–1152.

[22] Lewis, P., Denoyer, L., and Riedel, S. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (Florence, Italy, July 2019), Association for Computational Linguistics, pp. 4896–4910.

[23] Li, R.-H., Qin, L., Yu, J. X., and Mao, R. Efficient and progressive group steiner tree search. In *Proceedings of the 2016 International Conference on Management of Data* (New York, NY, USA, 2016), SIGMOD '16, Association for Computing Machinery, p. 91–106.

[24] Li, X., and Roth, D. Learning question classifiers. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1* (USA, 2002), COLING '02, Association for Computational Linguistics, p. 1–7.

[25] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized BERT pretraining approach. *CoRR abs/1907.11692* (2019).

[26] Lu, X., Pramanik, S., Saha Roy, R., Abujabal, A., Wang, Y., and Weikum, G. Answering complex questions by joining multi-document evidence with quasi knowledge graphs. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2019), SIGIR'19, Association for Computing Machinery, p. 105–114.

[27] Mausam, M. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (2016), IJCAI'16, AAAI Press, p. 4074–4077.

[28] Metzler, D., and B. Croft, W. Linear feature-based models for information retrieval. *Inf. Retr. 10*, 3 (June 2007), 257–274.

[29] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2* (Red Hook, NY, USA, 2013), NIPS'13, Curran Associates Inc., p. 3111–3119.

[30] Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., and Weston, J. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (Austin, Texas, Nov. 2016), Association for Computational Linguistics, pp. 1400–1409.

[31] Prager, J. Open-domain question answering. *Found. Trends Inf. Retr. 1*, 2 (Jan. 2006), 91–231.

[32] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing* (Austin, Texas, Nov. 2016), Association for Computational Linguistics, pp. 2383–2392.

[33] Reimers, N., and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (11 2019), Association for Computational Linguistics.

[34] Roy, D., Mitra, M., and Ganguly, D. To clean or not to clean: Document preprocessing and reproducibility. *J. Data and Info. Quality 10*, 4 (Oct. 2018), 18:1–18:25.

[35] Silva, J., Coheur, L., Mendes, A. C., and Wichert, A. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review 35*, 2 (Nov. 2010), 137–154.

[36] Singhal, A., Buckley, C., and Mitra, M. Pivoted Document Length Normalization. In *Proc. of SIGIR* (1996), SIGIR '96, pp. 21–29.

[37] Smucker, M., and Allan, J. An investigation of Dirichlet prior smoothing's performance advantage. Tech. rep., CIIR, U. Mass., Amherst, 2005.

[38] Verma, M., and Ganguly, D. LIRME: locally interpretable ranking model explanation. In *Proc. of SIGIR'19* (2019), pp. 1281–1284.

[39] Voorhees, E. M. The trec-8 question answering track report. In *In Proceedings of TREC-8* (1999), pp. 77–82.

[40] Voorhees, E. M. The effect of sampling strategy on inferred measures. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval* (New York, NY, USA, 2014), SIGIR '14, Association for Computing Machinery, p. 1119–1122.

[41] Yamada, I., Asai, A., Sakuma, J., Shindo, H., Takeda, H., Takefuji, Y., and Matsumoto, Y. Wikipedia2vec: An efficient toolkit for learning and visualizing the embeddings of words and entities from wikipedia. *arXiv preprint 1812.06280v3* (2020).

[42] Yang, P., Fang, H., and Lin, J. Anserini: Enabling the use of lucene for information retrieval research. In *Proc. of the 40th SIGIR* (2017), SIGIR '17, pp. 1253–1256.

[43] Yang, P., Fang, H., and Lin, J. Anserini: Reproducible ranking baselines using lucene. *J. Data and Information Quality 10*, 4 (Oct. 2018), 16:1–16:20.

[44] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2018).

[45] Zhao, C., Xiong, C., Qian, X., and Boyd-Graber, J. Complex factoid question answering with a free-text knowledge graph. In *Proceedings of The Web Conference 2020* (New York, NY, USA, 2020), WWW '20, Association for Computing Machinery, p. 1205–1216.

Appendix A

I-REX

A.1 Language Model

In the traditional language modeling, it is assumed that the query Q and the document D are being generated from the same underlying same language model θ_D. Now, given a query Q we want to know which document is the query most likely to have been drawn from. Assume, if the query contains m many terms, i.e $Q = \{q_1, q_2, ..., q_m\}$. Therefore,

$$\begin{aligned} Score(D, Q) &= P(Q \mid \theta_D) \\ &= \prod_{q \in Q} P(q \mid \theta_D) \end{aligned}$$

Estimation of θ_D from D is done using maximum likelihood estimator (MLE). For unigram language model MLE of $P(q \mid \hat{\theta}_D)$ turns out to be $\frac{tf(q,D)}{|D|}$. Where, $tf(q, D)$ is the term frequency of q in document D and $|D|$ is the document length.

A.1.1 Jelinek-Mercer smoothing (LMJM)

Now, to avoid the zero probability during MLE of unseen terms in the document D, the language model is smoothed with a background model estimated from the collection C,

as shown in below equation.

$$P(Q \mid \hat{\theta}_D) \quad = \quad \prod_{q \in Q} \lambda \frac{tf(q,D)}{|D|} + (1-\lambda)\frac{cf(q)}{|C|}$$

Here, λ is an interpolation parameter, lies from 0 to 1. $cf(q)$ is the collection frequency of the term q and $|C|$ is the size of the collection altogether.

A.1.2 Dirichlet smoothing (LMDIR)

Another variant of smoothing is called dirichlet smoothing where instead of MLE it uses Bayesian estimator which can impose a prior belief of θ_D. The final formulae is shown below,

$$P(Q \mid \hat{\theta}_D) \quad = \quad \prod_{q \in Q} \frac{tf(q,D) + \mu \frac{cf(q)}{|C|}}{|D| + \mu}$$

Here, μ is a parameter which ranges from 100 to 5000 and all the other variables are same as discussed above.

A.2 WT10g Dataset

Topic Set	Documents	# documents	# topics	#relevant-docs
TREC9, TREC10	WT10g	1,692,096	100 (451-550)	5980

Table A.1: TREC Web Corpus WT10g dataset overview

A.3 Trec Eval

trec_eval is a utility maintained by TREC used for evaluating the ad-hoc retrieval run for any result files and the corresponding relevance judgement files.

Appendix B

Question Answering

B.1 Mapping from TREC 04 Domain to Ontonotes

Trec04 Entity Type	Ontonotes Entity Type
hum	PERSON
loc	GPE, LOC, ORG
enty	NORP, FAC, PRODUCT, EVENT LANGUAGE, LAW, WORK_OF_ART
num	DATE, TIME, PERCENT, MONEY QUANTITY, ORDINAL, CARDINAL
money	MONEY
date	DATE
gr	ORG

Table B.1: TREC 04 Entity label to Ontonotes