Final Report for AN004 Technical Training Knowledge Architecture
FAA COE TTHP: 16-C-TTHP-UA-010

Prepared by:

Dr. Chien-Chung Chan (PI)
Dr. Shengyong Wang (Co-PI)
Dr. Chen Ling (Co-PI)
Mr. Sai Prajeeth Annamgari
Ms. Lakshmi Prasanna Lolla

The University of Akron
Akron, OH 44325

July 2018

Table of Contents

List of Figures

Technical Training Knowledge Architecture

Abstract

We have studied modern question answering systems and found that IR-based question answering systems can provide an effective knowledge architecture for representing technical training course materials and can work as an effective search engine to support training. In addition, we have implemented a web-based subject search system using Apache open source software. Our web-based system allows users to query for a subject of interest, and it will identify a list of documents ordered by degree of relevancy to the subject. In addition, it will also identify topics related to the subject ordered by frequency of occurrence. Our system demonstrates the feasibility and benefits of transforming existing training course content into a repository searchable by subjects of interest.

1. Introduction

Advances in computing technologies have transformed the world into a digitally connected World Wide Web filled with information generated by all types of media. It becomes a challenge to tap into this tremendous web of information without being drawn into it. Search engines are lifesavers, and they have become essential tools for retrieving digital information from the web and a variety of other sources. They also have changed the way of obtaining new knowledge in this internet-connected world.

Research and developments in information retrieval (IR) and natural language processing (NLP) facilitate the development of search engines. Enhanced by research in machine learning (ML) and artificial intelligence (AI), they are also fundamental to the development of question answering (QA) systems such as IBM's Watson, Apple's Siri, Amazon's Alexa, etc.

Question answering systems are complex IR systems. There are two major paradigms of question answering systems: IR-based question answering and knowledge-based question answering. The IR-based systems are also called text-based QA systems, since retrieval of an answer for a given question is based on textual similarity matching. The basic idea is to identify

the answer type for each given query, then it will try to retrieve the most likely answers from indexed documents based on textual similarity matching. The knowledge-based systems, built on top of textual similarity matching, need one more step of transforming queries and the content of a knowledge base into certain knowledge representation. For a given question, it will build a semantic representation from the query first, and then perform a retrieval by matching it to the content of a knowledge base. In addition, it is possible to develop a hybrid system of the two paradigms such as IBM's Watson QA system [1, 2].

In this project, we have studied a knowledge-based QA system developed by MIT called START [3]. It uses a semantic representation scheme called T-expression, which consists of a triplet of <subject, relation, object>. START extracts knowledge from documents by transforming sentences in the documents into T-expressions. Therefore, the knowledge base is a collection of T-expressions extracted from source documents by applying a set of transformation rules or patterns. It is a simple and elegant representation scheme. However, we have identified some challenges when applying T-expressions to our sample training materials.

We looked into the ADL (Advanced Distributed Learning) framework proposed by DoD [4, 5, 6]. It turns out that the main focus of the ADL framework is for cutting-edge Learning Management System specifications, and there is no specification for developing QA systems yet.

An important goal of this project is to develop a prototype QA system to support training. Since knowledge-based QA systems, such as MIT's START, need to create a knowledge base from training documents based on certain semantic representation schemes, it is beyond the scope of our work. In addition, there is no readily available open source software for developing knowledge-based QA systems at this time. Therefore, we have adopted the IR-based QA approach to implement our prototype system using the Apache Lucene/Solr open source software [7]. Our system will allow users to query for a subject of interest related to the training materials, and it will identify a list of documents ordered by degree of relevancy to the subject. In addition, it will also identify topics related to the subject ordered by frequency of occurrence.

In the following section, we will review the basic structure of an IR-based QA system. In Section 3, we briefly review the MIT START system and present some challenging issues we have observed. The idea of topic modeling is reviewed in Section 4. Section 5 presents the implementation of our prototype system along with a running example. The conclusion is given in Section 6. A list of features that can further improve the performance of our system is given in

Section 7 as future work, followed by references.

2. IR-Based QA Systems

The advancement of open source software has provided many high-quality platforms for developing IR-based QA systems. One of the most popular platforms is the Apache Lucene/Solr [7]. Lucene is a Java library of tools for indexing documents to support information search and retrieval, and Solr is built on top of Lucene to provide web-based accessibility. To construct full-fledged QA systems, we also need to apply NLP and AI tools for sentence detection and named-entity recognition when creating an index from documents, and to parse input queries and determine answer type when processing questions. These tasks can be implemented using the Apache OpenNLP platform [8]. A typical workflow of an IR-based QA system is shown in Figure 1 [9]. There are two sides to the workflow. The process for indexing documents is shown on the left-hand side, and the right-hand side shows the flow of processing a question. In the figure, applicable tools are marked inside the parentheses. The query processing has five steps:

- Parse (chunk) the input query into tokens.
- Determine the answer type so that, when we search, we can find candidates that will best answer the question.
- Generate the Lucene/Solr query using a combination of the parsed query and the answer type.
- Execute the search to determine candidate passages that might contain the answer.
- Rank the answers and return the results.

On the document processing side, the goal is to create an index to facilitate retrieval. There are three steps:

- Sentence detection: Determine the boundary of a sentence in the document.
- Named-entity recognition: Classify nouns or phrases into semantic categories such as person, location, date, etc.
- Indexing: Create an index based on useful terms. The created index is used to search for an answer when processing input queries.

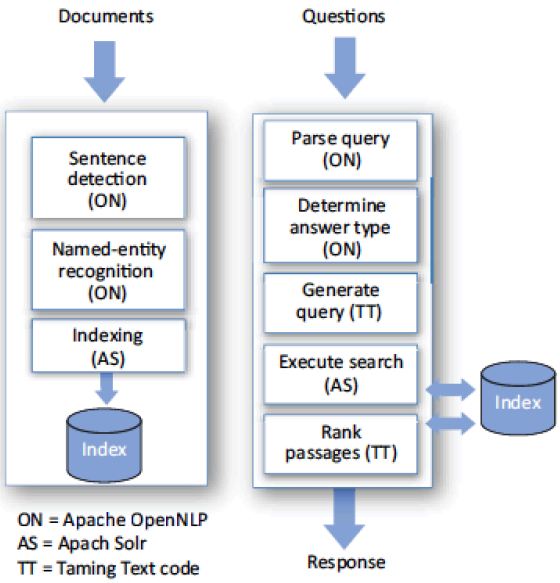

Figure 1. The structure of an IR-based QA system.

3. SynTactic Analysis using Reversible Transformations (START) system

The START system was developed by Artificial Intelligence Laboratory at Massachusetts Institute of Technology [3]. The motive of the START system was to provide an efficient natural language processing (NLP) system for efficient indexing and retrieval of information. Figure 2 shows the working of the START system which can be accessed using the link: http://start.csail.mit.edu/index.php. The user can enter a query like "Why is the sky blue?" in the search bar and get an efficient answer to the question. The START system parses incoming questions, matches the queries against its knowledge base, and presents the appropriate information segments to the user. In this way, START provides untrained users with speedy access.

It has two modules: the understanding module and the generating module. The understanding module indexes and understands the English text whereas the generating module produces English text from the knowledge base. In START's implementation, sentences are divided into units called kernel sentences. The building blocks for a kernel sentence include a noun template, verb template, and an adverb template. The kernel sentences must be converted into Ternary expressions (T-expressions) represented as a triplet of the form: <subject, relation, object>.

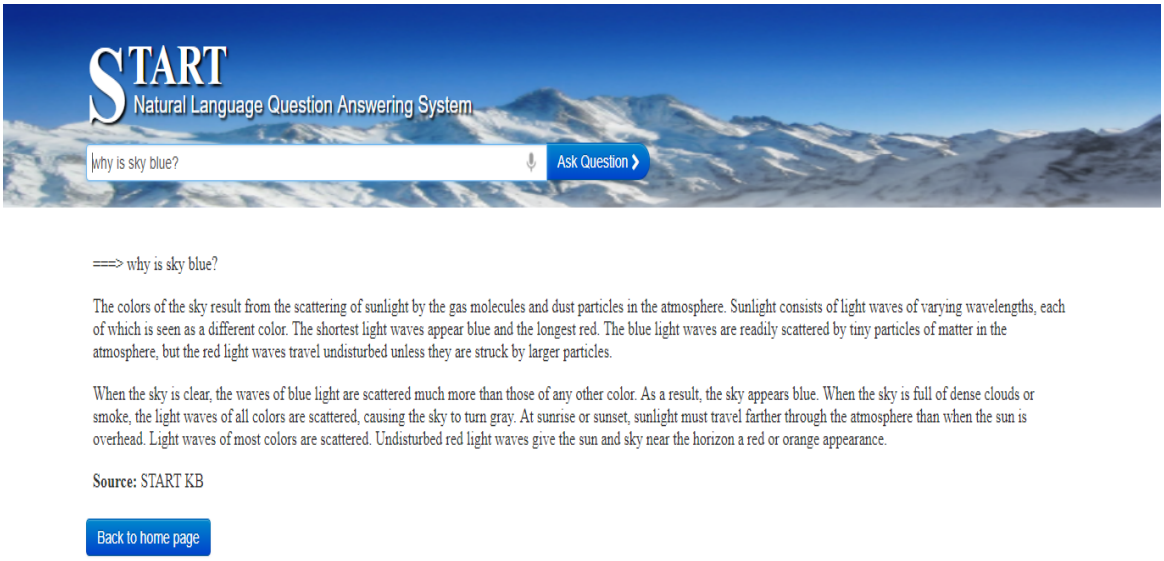

Figure 2. How the START system works

Let us consider the following example:

i. Jane will recognize Paul tomorrow.

ii. Yesterday Jane could have recognized Paul.

iii. Paul was recognized by Jane.

The above example represents three kernel sentences which are also known as building blocks. They are represented using different tenses but the sentences resolve to have the same meaning when transform into T-expressions. These kernel sentences are converted into T-expressions, shown below, which is a triplet of subject (Jane), relation (recognize), and an object (Paul).

<subject, relation, object> => <Jane recognize Paul>

The example below shows the representation of T-expressions based on a specific course number (50148) from our sample data set along with conflicts we have observed during our implementation. We have manually selected a few sentences, known as kernel sentences, to represent them in the form of Ternary expressions and to effectively identify the training subjects. Some challenges we have faced with identifying the T-expressions for our dataset are given in the following.

Case 1: parsing query “The assistant controller position is still staffed at some of the facilities with Areas of Specialization that handle international traffic.”

Possible transformation in T-expression format:

<subject, relation, object> => <assistant controller position, staffed, facilities>

Conflict: Ambiguity in identifying the Parts-Of-Speech in the sentences. The parts-of-speech for “staff” can be noun or verb, so it can be either a subject or a relation.

Case 2: parsing query “The number of sectors with nonradar airspace varies significantly from facility to facility.”

Possible transformation in T-expression format:

<subject, relation, object> => <number of sectors, nonradar airspace, facility>

Conflict: Long sentences in training material make it challenging to identify the subject.

Case 3: parsing query “Successful completion of stage I depends partly on academics and partly on adequately demonstrated performance in the simulation lab.”

Possible transformation in T-expression format:

<subject, relation, object> => <completion of stage I, academics and performance, simulation lab>

Conflict: Compound relations such as academics and performance.

Case 4: parsing query “Actively participate in training to achieve certification.”

Possible transformation in T-expression format:

<subject, relation, object> => <training, participate, certification>

Conflict: A sentence may contain multiple subjects.

4. Topic Modeling

Recently, some systems have allowed retrieving documents in terms of “topics” instead of words or phrases alone. The difference is that a “topic” is represented by a set of words co-occurring frequently enough in a collection of documents such that words appearing in the set are considered

as belonging to the same topic. The problem of topic modeling for a collection of documents is how to identify sets of words that co-occur frequently. Topic modeling is used to identify meaningful patterns and attempts to retain the semantic meaning in a given vocabulary. These models extract various topics from texts. They assume that any text is a combination of multiple words from possible baskets of words where each basket corresponds to a particular topic. If the condition is true, it becomes possible to deconstruct a text into the probable baskets from where the words first came. This process is repeated over and over again until it settles on the most likely distribution of words into baskets, which are called topics.

For example, if someone is interested in documents related to "green energy," a textual-based retrieval system will retrieve documents containing the exact phrase "green energy." It is clear that the topic of green energy is more than the phrase alone. In a topic-driven retrieval system, it will be able to retrieve documents related to the topic of green energy such as solar energy, wind energy, etc. which are considered relevant to green energy.

One of the most popular mechanisms in topic modeling is the Latent Dirichlet Allocation (LDA) algorithm [10]. It is a three-level hierarchical Bayesian model where every item of a group of items is modeled into a set of topics. Each topic is again modeled based on a set of topic probabilities. LDA formally defines a topic to be a distribution over a fixed vocabulary and belongs to a field of probabilistic modeling.

MALLET is a popular Java implementation of the LDA algorithm [11]. MALLET is a Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications to text. It is an open source AI tool developed by the University of Massachusetts, Amherst (UMass) [12].

MALLET provides command line interface. Figure 3 shows some basic commands that can be used to perform topic modeling by (1) importing the data, (2) preprocessing the data, and (3) generating a given number of topics. The first command in the figure sets the current directory where MALLET tool has been installed. The second command is used to import a data set, giving a destination name for the output file, and to perform preprocessing by removing the stop words and retaining the sequence in the text documents. The third command is used to train the topics/model by considering the input data file. The fourth command specifies the number of topics to be generated along with the output state and the keys for the output.

```
1) cd mallet-2.0.8

2) mallet import-dir --input sample-data\web\en --output tutorial.mallet --keep-sequence --remove-stopwords

3) mallet train-topics  --input tutorial.mallet

4) mallet train-topics  --input tutorial.mallet --num-topics 20 --output-state topic-state.gz --output-topic-keys
   tutorial_keys.txt --output-doc-topics tutorial_compostion.txt
```

Figure 3. MALLET commands for performing topic modeling

Figure 4 shows the result of applying MALLET topic modeling to a sample lesson file with the name "ILP01-En Route Qualification Training Overview" within course 50148. We considered a total of five topics (0-4) to be generated given a text file as an input.

```
0	route facility continued area primary simulation radar-associate required plan nonradar academy varies set
	successful terminal performance progress secondary total academics
1	airspace ojt controller position iii overview team flm certification generally length pass nonradar areas
	airports function target certify commonly problems
2	training stage page radar cpc control number evaluation department qualification map ojtis roles familiarization
	assistant include positions make ojti supervision
3	scenarios facilities controller note traffic developmental controllers aircraft instructional consists on-the-job
	aspects test conducted associate sectors cont'd teach describe part
4	stages lesson specialization hours page called initial review completion order materials skill artcc assignment
	consists purpose continue perform certification requires
```

Figure 4. Topic modeling on a text document from Course 50148

4.1. Observations based on Topic Modeling

We have observed some challenges in applying MALLET topic modeling to generate a topic index while implementing our current system. The following are the limitations which we have encountered:

- We have randomly selected the number of topics to be five and found that some of the relevant topics in the input document were not covered in the topics generated. It is very time-consuming to determine the proper number of topics to be generated and its effectiveness.

- It is also challenging to determine which words should be considered as stop words when preprocessing the documents, because this is application domain dependent.
- Topics can also be defined by N-gram words. For example, in the case of 2-gram topics, each topic is a set of two-word phrases. The issue is determining how many words should be used, and it becomes very time-consuming when the value for N becomes larger.

5. Current System

The goal of this project is to identify effective knowledge representation for training materials and to develop a prototype system to support training. After evaluating knowledge-based question answering and IR-based answering system, we believe the IR-based approach will provide more effective support for training. In addition, subject- or topic-driven retrieval can provide a more effective search for training materials. Therefore, we have implemented a subject-driven IR-based search engine to demonstrate the effective support for training by using open source software.

Our system is implemented by the following process:

- Collect course materials
- Text extraction and indexing
- Topics generation
- Input subject
- List of related documents to the given subject
- List of related topics to the given subject

Our sample data set is taken from the course materials of an FAA air traffic controller training course: En route stage 1 course: 50148. The training course files are extracted and converted into text files by using Apacha's Tika software [13]. Then, we built two levels of indices from the extracted text files: One index is at the document level, and the other is at the subject level. The indices were created using Apache Lucene/Solr tools. Since the topics generated by MALLET are static in nature and have the limitations as discussed in Section 4, we have decided to construct the subject index dynamically using keywords extracted from each input query. We used a window of 20 words before and after a keyword as a topic. Then, the topic words are used to retrieval documents with a certain degree of relevancy measured by Lucene's built-in similarity measure.

From the related documents, we then retrieve related topics based on frequency of occurrence.

In summary, our system will allow users to enter a query for a subject of interest, and it will identify a list of documents ordered by degree of relevancy to the subject. In addition, it will also identify topics related to the subject ordered by frequency of occurrence.

5.1. Implementation

We have developed a web-based interface which can be used to interact with our subject-based search system. The tools we have used to implement the application include Java Server Pages (JSP), HTML, CSS, JavaScript, Ajax, and JQuery. Figure 5 below shows the current system interface where users can log in with a username and password, then they can access the subject-based search system to run a query and get relevant lessons along with the related topics.

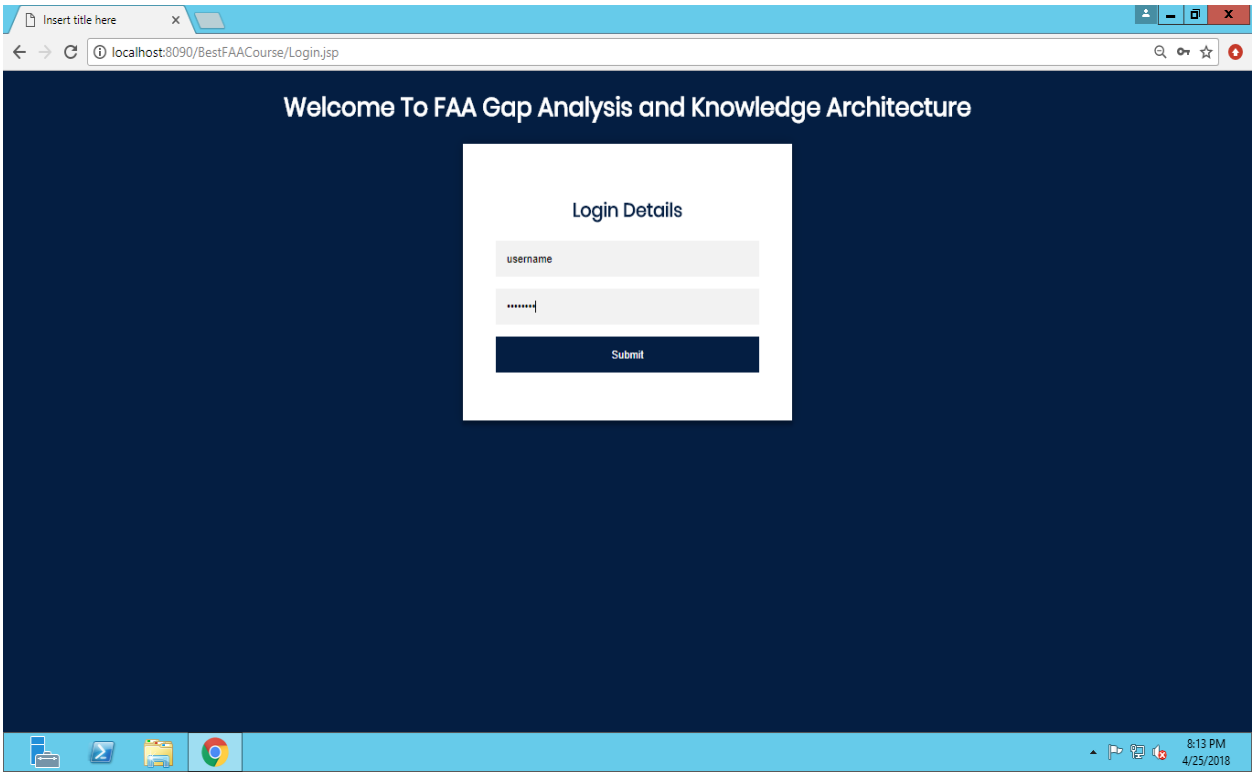

Figure 5. User interface of the current system

Figure 6 displays the home page that appears once a user logs in to the current system. The user can select the "Knowledge Architecture" tab to enter a query and get a list of related documents and topics related to the user's query. Figure 7 displays the Knowledge Architecture page, where the user can choose to search topics, select help for user instructions on how to use the system, go back to the home page, or successfully log out of the system.

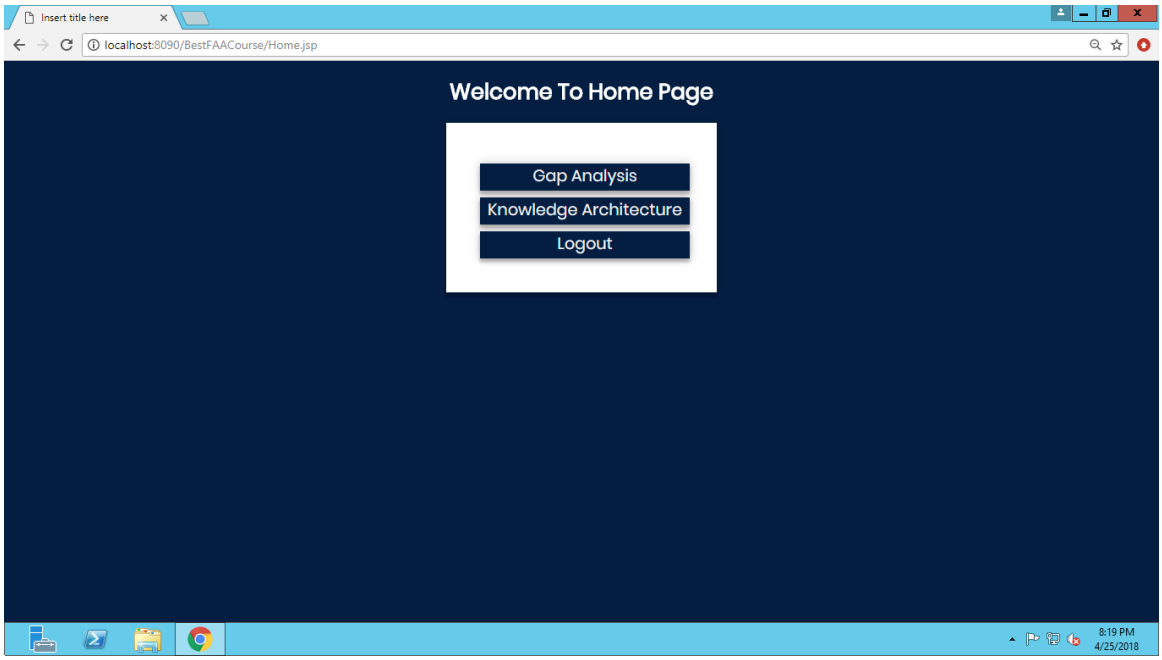

Figure 6. Home page after the user logs in

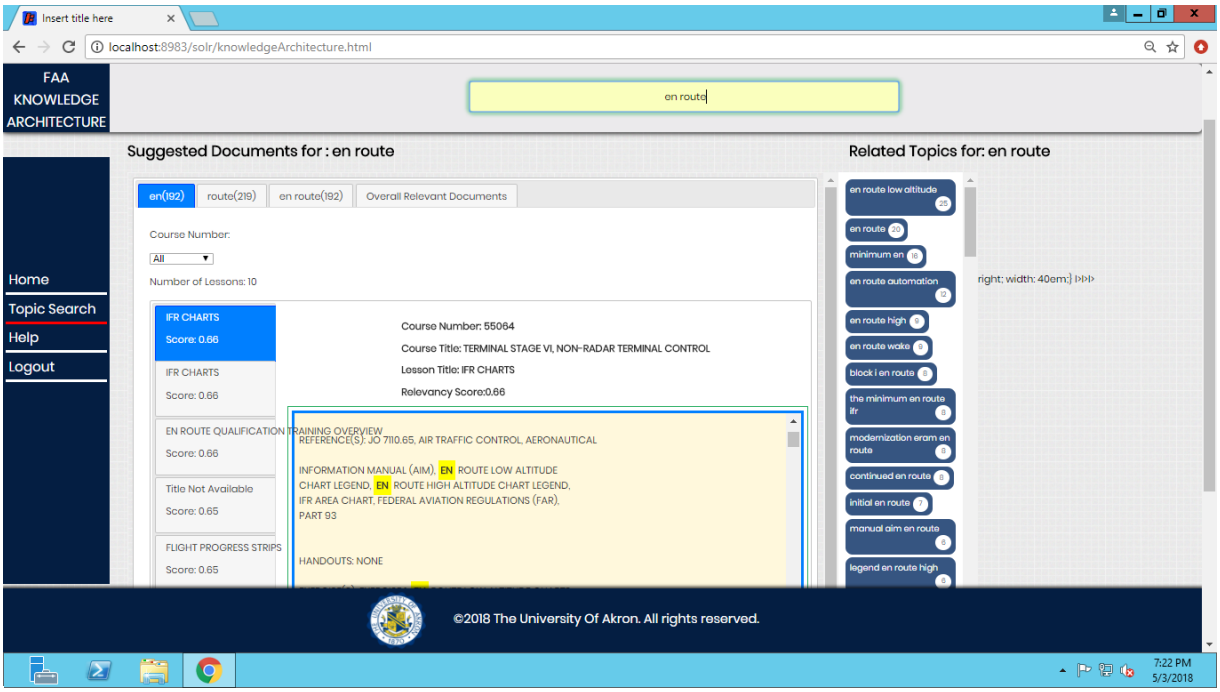

Figure 7. Results after executing a user's query

As seen in Figure 7, we consider for example that the user has entered "en route" as an input query. Once the query is executed, the user can see a list of suggested documents for "en route," and he/she can also select a desired course from the dropdown provided for courses. The relevancy score, calculated using the tf-idf (term frequency–inverse document frequency) measure indicating the relevancy of a document in the corpus, is used to retrieve the sentences in a document. It is displayed along with the documents, where the user can choose the most relevant document related to the search based on the relevancy score. Under the "Overall Related Documents" tab, the score is calculated based on the possible combinations of the user query in a document. Each document has a separate score, and finally an aggregated score is calculated based on the related documents. Combinations of words based on the user query are given the highest priority. For example, if we consider "air traffic" as an input query, the possible combinations include: air, traffic, air traffic. Here, the relevancy score for all three combinations is calculated and the combination "air traffic" is given the highest priority.

The sentences are retrieved using Apache Solr by providing a window size of 20 characters, to be included before and after the query phrase. The frequency of each individual term is calculated from the retrieved collection of documents and then this frequency score is displayed

along with the respective topics as shown in the Figure 7. The user can also select one of the topics from the related topics for “en route” to gather more relevant information on the query he/she has entered. Therefore, given a training subject, our system will return suggested lesson documents and related topics along with a choice to select a desired course and easy access within the interface.

6. Conclusion

In this project, we have studied two modern paradigms for developing question answering systems. We found that an IR-based QA system is a good knowledge architecture to support effective training. It can provide user-friendly features as found in modern search engines. In addition, enhanced with topic-driven retrieval mechanisms, it can support subject-driven search of training topics. The system is also scalable, and easily accessible through a web-based interface. And all the components are implemented using open source software. Thus, our system has demonstrated great potential to support effective training by modern information retrieval and natural language processing.

7. Future Work

The current prototype system has implemented major indexing components to support the subject-driven search of related topics of interest. It is not yet a full-fledged question answering system. In addition, there exists different types of course materials. The following is a list of tasks that could be pursued in the next phase of development:

- Support topic search from different types of course materials, such as annotated videos and audio.
- Extract structured information from course content details such as the lesson titles, authors, course numbers, etc.
- Use AI tools to further refine related topic identification.
- Extend the system into a full-fledged IR-based question answering system.

8. References

[1] David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A. Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building Watson: An overview of the DeepQA project. *AI magazine*, 31(3):59–79, 2010.

[2] Alfio Gliozzo, Or Biran, Siddharth Patwardhan, and Kathleen McKeown. Semantic technologies in IBM Watson. *Association for Computational Linguistics*, page 85, 2013.

[3] Boris Katz, "Using English for Indexing and Retrieving", Proceedings of the first RIAO Conference on User-Oriented Context-Based Text and Image Handling (RIAO'88), 1988.

[4] Office of the Deputy Under Secretary of Defense for Readiness (ODUSD(R)), Readiness and Training (April 30, 1999). Department of Defense Strategic Plan for Advanced Distributed Learning (Report to the 106th Congress).

[5] "Learning on Demand: ADL and the Future of e-Learning", Advanced Distributed Learning Initiative (2010), Washington, DC: Advanced Distributed Learning Initiative.

[6] ADL website. (accessed July 1, 2017). http://www.adlnet.org.

[7] The Apache Solr/Lucene: http://lucene.apache.org/solr/

[8] The Apache OpenNLP: https://opennlp.apache.org

[9] Grant S. Ingersoll, Thomas S. Morton, and Andrew L. Farris. *Taming Text: How to Find, Organize, and Manipulate It*. Manning Publications, Dec. 28, 2012, ISBN-13: 978-1-933988-38-2

[10] David M. Blei, Andrew Y. Ng, Michael I. Jordan, "Latent Dirichlet Allocation", Journal of Machine Learning Research 3.

[11] Shawn Graham, Scott Weingart and Ian Milligan, "Getting Started with Topic Modeling and MALLET", The Programming Historian, February 9, 2012.

[12] MAchine Learning for LanguagE Toolkit, UMASS AMHERST.

[13] Apache Tika: https://tika.apache.org