CMPE493 Introduction to Information Retrieval | Term Project

Spring 2019

Question Answering System in Turkish

Group 3

Cemal Burak Aygün | Baran Kılıç | Atıf Emre Yüksel

Introduction

Problem: Implementing a question answering system that can answer geography related questions asked in Turkish free-text. The corpus of the system consists of the paragraphs taken from the high school geography books that are used by Turkish Education Ministry.

Solution consists of two steps:

1. Retrieving the most related paragraph(s) for a given question
2. Extracting the answer from those paragraph(s)

Dataset Analysis

Dataset consists of two parts:

- A corpus of 3806 paragraphs
- 1794 question-answer pairs

Some observations:

- Dataset needs Unicode normalization
- Some paragraphs include “soft hyphen” character (u‘\xad’)
- Some paragraphs include references to the materials (table, image, diagram, etc.) such as (Görsel 1.2), (Tablo 3.3)
- Some numbers include dot (.) as the thousands separator

Text Preprocessing & Tokenization

- Unicode normalization to NFC
  - unicodedata.normalize('NFC', text)

C+◌̧ → Ç

- Removal of “soft hyphen” character (u‘\xad’)
  - re.sub(u'\xad', '', text)

Soft hy\xadphen | Soft hyphen | Soft hy-
phen

- Removal of references to the materials (table, image, diagram, etc.)
  - re.sub(r' \((Fotoğraf|Görsel|Grafik|Harita|Resim|Şekil|Şema|Tablo).+?\)', '', text)
- Removal of dot (‘.’) as the thousands separator
  - re.sub(r'(?<=\d)\.(?=\d)', '', text)

1.000 → 1000

- Removal of circumflexes (^) from some letters
  - text.translate(str.maketrans("âîôû", "aiou"))

rüzgâr → rüzgar

- Tokenization includes splitting text from any characters other than Turkish letters and digits, and lowercasing

Text Preprocessing & Tokenization

- We tried some lemmatization and/or stemming libraries for Turkish words. However, the results got worse.
- Geography domain contains some rare words that lemmatizer could not handle.
  - Vertisol → ver
  - Podzol → po
  - Mesozoyik → me
  - Paleozoyik → pal
  - Epirojenez → e

The lemma versions of the words above are generated using https://github.com/akoksal/Turkish-Lemmatizer .

Part 1: Related Paragraphs Methodology

Paragraphs are scored based on their similarity to a given question and *top-k* paragraphs with the highest scores are chosen.

6 similarity metrics are considered:

- Jaccard Score (JAC): Jaccard value of <question terms> and <paragraph terms> (with stopwords removal)
- Bigrams Score (BIG): Word Bigrams, <# of common bigrams> / <# of question bigrams>
- Term Frequency Score (FRQ): <# of question tokens that appear in paragraph> / <# of question tokens> (with stopwords removal)
- Longest Common Subsequence Score (LCS): <# of words in longest common subsequence> / <# of words in question>
- Cosine Score (COS): *Tf-Idf* based vectors of question terms (with stopwords removal)
- TF-IDF Score (TFI): Sum of *tf-idf* scores of question terms in paragraph (with stopwords removal)

Note that all the metrics except *TFI* produces a real number in the range [0, 1].

Part 1: Related Paragraphs

Methodology

Related paragraph accuracies (for top-1 highest scored paragraph) on train dataset (1794 data):

Jaccard Score (JAC)	53.62%
Bigrams Score (BIG)	73.19%
Term Frequency Score (FRQ)	76.37%
Longest Common Subsequence Score (LCS)	76.48%
Cosine Score (COS)	79.77%
TF-IDF Score (TFI)	81.27%

Experiments with the linear combinations of these metrics yield the following model:

5*<JAC> + 3*<BIG> + 0*<FRQ> + 4*<LCS> + 8*<COS> + 0.28*<TFI>

Top-1 Accuracy	83.56%
Top-2 Accuracy	89.19%
Top-3 Accuracy	90.86%
Top-5 Accuracy	92.47%
Top-10 Accuracy	94.20%

Part 1: Related Paragraphs

Results

Net-Score: 5*<JAC> + 3*<BIG> + 4*<LCS> + 8*<COS> + 0.28*<TFI>

We considered retrieving *top-k* related paragraphs (according to the net-score above) for a given question and then trying to find the most related one among those *k*.

However, we couldn't beat the accuracy score of retrieving directly the most related paragraph (top-1).

Summary of Part 1: Net-score for each of the paragraphs in the corpus is calculated and the one with the highest score is returned as the related paragraph of the question.

Accuracy Score on
Train Dataset (1794 data)

83.56%

Accuracy Score on
Test Dataset (489 data)

79.76%

Part 2: Question Answering Methodology

Producing an answer for a given question consists of 3 steps:

1. Retrieving *top-3* most related paragraphs (from part 1)
   Experiments yield the best results when *top-3* paragraphs are considered.
   (Related paragraph accuracy for top-3 is 90.86%)

2. Finding the most similar sentence among the sentences of these 3 paragraphs
   The sentence which includes the answer to a question contains most of the terms of that question. Based on this observation, sentences are scored according to the number of common character bigrams and the one with the highest score is chosen as the answer sentence of the question. With this approach, accuracy of finding the sentence that contains the answer to a question is about **80%**.

3. Extracting the answer from the most similar sentence

Part 2: Question Answering Methodology

Extraction of the Answer from a Given Sentence

A rule based approach is adopted. Some observations about questions:

- Numeric Type: Questions that contain one of “kaç”, “tarihte”, “hangi (yıl|sene)”, “ne kadar”, “ne zaman” have a numeric answer. (e.g., 1995)
  - Concatenate all the numbers in the sentence with a space and return it as the answer
- “Hangi Çağ” Type: Questions that contain “hangi çağ” have an answer in the form “<X> Çağ” (e.g., Orta Çağ)
  - Find the first occurrence of “çağ” and return it along with the token just before it.
- Verb Type: Questions that contain “nasıl” have an answer which is the last word of the sentence (e.g., arttırır)
  - Return the last token of the sentence as the answer

Part 2: Question Answering Methodology

Extraction of the Answer from a Given Sentence

Before the rules on the previous slide are applied, sentence is trimmed down:

- First of all, tokens of the sentence that are the same as any of the question terms are removed. Here, “sameness” of two tokens is measured as follows: If the ratio <# of common characters from the beginning> / <length of the longest token> is greater than a threshold (e.g., 0.5), they are considered as the same. Since Turkish is an agglutinative language, we consider the beginning of the tokens.

nem and neolitik	=>	|{n, e}| / |{n, e, o, l, i, t, i, k}| = 2/8 = 0.25 (not same)
nem and nemli	=>	|{n, e, m}| / |{n, e, m, l, i}| = 3/5 = 0.6 (same)

- If all the tokens of the sentence gets removed on the previous step, only the tokens that are the *exact* same as any of the question terms are removed.

Part 2: Question Answering

Results

If none of the rules is matched with the question or a matched rule returns an empty answer, trimmed sentence is returned as the answer.
In case of the trimmed sentence is empty, the original sentence is returned as the answer.

Summary of Part 2: *Top-3* most related paragraphs for a given question is retrieved, the most related sentence among these 3 paragraphs is selected and the answer is extracted from that sentence based on some predefined rules.

Accuracy Score on Train Dataset (1794 data)	Jaccard Score on Train Dataset (1794 data)	Accuracy Score on Test Dataset (489 data)	Jaccard Score on Test Dataset (489 data)
11.26%	28.75%	11.46%	29.96%

Evaluation & Conclusion

We don’t encounter overfitting problem on test data even we use rule-based model. However, rules must be defined clearly in order to get general patterns in not only training data.

We can try Siamese networks and BERT for sentence similarity task as a deep learning approach.

Comprehensive **lookup table** can be constructed for detecting the characteristic words of different subtopics in geography. Example question:

S2413: Gelişen 8 Ülke örgütünün kurulduğu toplantı hangi şehirde yapılmıştır?
C1909: İstanbul
Related Sentence: [8607] Gelişen 8 Ülke örgütü, 1997 yılında İstanbul’da yapılan bir toplantıda kurulmuştur.

şehir: {ankara, istanbul, izmir, manisa, …}
mevsim: {sonbahar, kış, ilkbahar, yaz }

Mueller, J and Thyagarajan, A. Siamese Recurrent Architectures for Learning Sentence Similarity. Proceedings of the 30th AAAI Conference.

Thanks for listening!

Acknowledgements:
Turkish stopwords list from Stopwords ISO (https://github.com/stopwords-iso) repository is utilized in this project.