11-791 Design and Engineering of Intelligent Information System &
11-693 Software Methods for Biotechnology
Homework 3

Engineering and Error Analysis with UIMA

Hao Zhang

haoz1

1. Error Analysis for Task 1.

Query 10 is the only query whose Reciprocal Rank is one. The Mean Reci procal Rank for the baseline system is 0.475, which means averagely the r elevant document is ranked at the second position in the list.

By going through the 20 queries and their ranked documents. I summarize d the Classify error as follows:

Error Type	Query ID
irrelevant documents contains some keywords but when take into account the length they will be ranked higher.	1, 2, 4, 8, 18
stemmer not applied	3, 13, 16
punctuations is not separated from w ord in each token	5, 12, 14
semantic meanings (like the words of the answer do not repeat the words i n the query)	6, 9, 15, 17, 20
others	7, 11, 19

To improve the baseline system the following can be applied:
(1) Better tokenizer.
Instead of tokenizing with respect to white space, we could get rid of punct uations and convert uppercase to lowercase. This tokenization yields a MR R of 0.637
(2) Using stemmer.
I tried to use StanfordNLP stemmer but there are some compatible bugs a re not fixed yet.

Now query 3, 6, 8, 9, 10, 14, 20 rank the relevant document at the top one position.

2. Design new approach for vector space retrieval mod el

2.1 General idea

In the documents.txt, the query sentence implies related semantic meanin g with relevant document. For example, "What is the Keystone State?" car ry the semantics that is related to "They call it the Keystone State, and in this unpredictable election year, Pennsylvania is living up to its name.". This can be viewed as paraphrase between query and relevant se ntence. So the task here is to find the best sentence which carry semantic ally related meanings. The model used in Task 1 is based on bag-of-word s features and therefore the structure information in the sentence can not b e taken into account. Stanford NLP group published a paper in 2011 on NI PS where they tried to detect paraphrase with recursive auto-encoders. Th eir approach can be viewed as a alternative approach for our situation, we can incorporate compositional semantic information in our model to help re trieval. Some useful package are provided along with the paper and can b e used in our design. Unfortunately, part of their implementation needs MA TLAB, which does not fit in our Java development. Pre-processing is cond ucted to finish Non-Java part in the algorithm.

2.2 Type design

I design MyDocuments type by extending Documents type from the Type system in Task 1. Besides the feature inherited from Documents, Feature Vector is added to store sentence representation. FeatureVector is an inst ance of *uima.cas.FloatArray*. The dimension of FeatureVector is set to 10 0.

2.3 CollectionReader

CollectionReader is design to read in documents.txt and features.txt in ord er to initialize CAS for later processing. It simply reads in a line in docume nts.txt and a line in features.txt then concat them together as in text content for a CAS.

2.4 Analysis Engine

The main job of analysis engine is to to fill in FeatureVector and other information for a annotation (defined by MyDocuments). It reads in the text initialized in CollectionReader then parse the text to extract each attribute (qID, rel, feature representation). Finally it adds the annotation into index list.

2.5 Collection consumer.

Collection consumer is the key part for retrieval. It processes each annotation in CAS and save necessary information in global object. After going through all the CAS it will call method "collectionProcessComplete", where it use the information stored in the global object to rank documents for each query. In my pipeline, both query and document are represented as a 100 dimensional real value vetor, which is extracted with external tools (MATLAB + StanfordNLP package). The measure can be Cosine distance, Euclidean distance and so on. Based on the ranking the most possible document can be retrieved.

2.6 Feature Extraction for sentence

The core algorithm is described in the following paper:

Dynamic Pooling And Unfolding Recursive Autoencoders For Paraphrase Detection 2011 NIPS

The basic idea is to first parse each sentence to get its parse tree. Then based on the structure of parse tree, two words are collapsed to a composed representation by a recursive neural network. Start from the bottom of the tree, all the words in the sentence can be collapsed into one representation in the root node, which is used as representation for the sentence.

2.7 Result

Using 100 dimension feature vector as representation and Cosine distanc e as metric, my pipeline can yields MRR = 0.554.

I suspect the reason that this model is not as good as the one where we u se better tokenizer may be the following:
1). The recursive neural network model is not powerful enough to capture t he semantic information for some sentence.

2). We are not using the best measure here. The Cosine distance used he re might not distinguish the information embedded in the 100 dimensional representation. Therefore I tired other distance measure:
(a)
    Euclidean distance:
    Under L2 measure, MRR = 0.600
(b)
    KL divergence between feature vector after softmax function
    In this case the MRR = 0.625

3). The metric should be learned, not defined as some sort of distance. Th e basic idea is we are defining what type of measure we will use in retrieval . But the true measure (or mapping) between query representation and do cuments representation is unknown. Intuitively, there should be a "QA" ma pping from query representation to document representation. If we have m ore samples we can develop a mapping function and try to learn the param eters. Then during retrieval, we can see which document have a better mat ch to the mapping of the query. For example, assume our query is a vector represented as Q, and the relevant document is a vector A. We could as sume a linear mapping from Q to A: QM = A, where M is a matrix to be lear ned. From the training data we can solve the parameter matrix M. For a ne w coming query Q_new, we calculate Q_newM, then compare which docu ment candidate is most close to the mapping. To measure the distance we only need to be consistent with the measure (objective) used in training.