FIRE 2017

https://sites.google.com/view/fire2017irled

Track 1: Catch Phrases extraction

Introduction

Catchphrases are short phrases from within the text of the document. Catchphrases can be extracted by selecting certain portions from the text of the document. A set of legal documents (Indian Supreme Court decisions) will be provided. For a few of these documents (training set), the catchphrases (gold standard) will also be provided. These catchphrases have been obtained from a well-known legal search system Manupatra (www.manupatra.co.in), which employs legal experts to annotate case documents with catchphrases. The rest of the documents will be used as the test set. The participants will be expected to extract the catchphrases for the documents in the test set.

Dataset

- Train_docs: txt files of full length cases: case_<i>_statement.txt (i= 0 to 99)
- Train_catches: txt files of catch phrases: case_<i>_catchwords.txt (i= 0 to 99)
- Test_docs: txt files of full length cases: case_<i>_statement.txt (i= 100 to 399)

Observations

- Catch phrases number per case varies from 1 to 200.
- Catch phrase can be ambiguous like ‘Ad’.

Approach: Custom NER

- IOB Preparation: Each case statement in CONNELL IOB format. All catch phrase words for the case are embedded as B-I, and rest as O. Steps could be as follows:
    - Copy case file as iob file.
    - For each multi-word catch phrase in the list:
        - Iob_string = ‘B ‘ + ‘I ‘ * (len(catch) -1)
        - For single word, it will be ‘B’, for 2 words: ‘B I’, 3 words ‘B I I’ and so on.
        - Iob_text.replace(catch,Iob_string) # so all the catch phrase words will be replaced by BI*.
        - All remaining words in iob_text could be replaced by ‘O’ (‘Outside’ tag)
        - So, iob_text will be all IOB tags.
        - All words in original text will have IOB tag in iob_text file. Make list of both.
        - Zip both lists together to make tuples list, one on each line, like:
        - (We, ‘O’)
        - (are, ’O’)
        - (the, ‘B’)
        - (world, ‘I’)
    - IOB file will go as a (x,y) file.
- Training:
    - With CRF or LSTM: Given X-Y as training pairs, build sequence model.
    - CRF can accommodate additional features per word, like POS, word shape, etc. in LSTM its feature less.
- train.csv and test.csv has words, pos and IoB tags as a single sequence.

Implementation

- Input: X, Y training pairs are like below.
    - X is column/list of dict2id of all words from cases, in sequence
    - Y is column/list of id for any of outcomes (I, O, B) for each of X words respectively.
    - Train_x = []
    - Train_y = []
    - Test_x = []
- Modelling:
    - Split Train_x and Train_y as 70-30% for cross validation and get 4 arrays: trainX, trainY, testX and testY. Mind well, that these all 4 are from Training set, not from Test set.
    - Reshape to be fed into LSTM: Insert 1 in the middle space, like (samples, 1, features=1)
    - LSTM(4, input_shape(1,1)). Second "1" is words at a time.
    - model.compile(loss='mean_squared_error', optimizer='adam')
    - model.fit(trainX, trainY, batch_size=1, verbose=2)
    - trainPredict = model.predict(trainX)
    - testPredict = model.predict(testX)
- Testing:
    - trainScore = math.sqrt(mean_squared_error(trainY, trainPredict))
    - testScore = math.sqrt(mean_squared_error(testY, testPredict))
    - Now test the original test word list and predict their outcome values
    - Go back to test corpus, predict NER tag for each word in each document and collect the Tag sequences.

Track 2: Document Similarity

Introduction

For the precedent retrieval task, two sets of documents shall be provided:

1. Current cases: A set of cases for which the prior cases have to be retrieved.
2. Prior cases: For each "current case", we have obtained a set of prior cases that were actually cited in the case decision. These cited prior cases are present in the second set of documents along with other (not cited) documents.

For each document in the first set, the participants are to form a list of documents from the second set in a way that the cited prior cases are ranked higher than the other (not cited) documents.

Dataset

- Current_Cases: txt files of full length cases: current_case_<i>.txt (i= 1 to 200)
- Prior_Cases: txt files of full length cases: prior_case_<i>.txt (i= 1 to 2000)

Observations

- Current cases includes citations in masked form, like [?CITATION?]. Objective is to find out those.
- Found actual current_case_0001.txt at http://courtnic.nic.in/supremecourt/temp/sc%202668408p.txt
- Its citations are:

    - AIR 1958 SC 398: prior_case_0780.txt
    - AIR 1963 SC 1895: prior_case_0272.txt
    - Mohd. Yunus v. Mohd. Mustaqim & Others (1983) 4 SCC 566, AIR 38, 1984 SCR (1) 211: prior_case_0744.txt
    - Laxmikant Revchand Bhojwani & Another v. Pratapsing Mohansingh Pardeshi (1995) 6 SCC 576
    - Rena Drego (Mrs.) v. Lalchand Soni & Others (1998) 3 SCC 341
    - Virendra Kashinath Ravat & Another v. Vinayak N. Joshi & Others (1999) 1 SCC 47
    - State of Punjab & Others v. Mohabir Singh etc.etc. (1996) 1 SCC 609
    - R. Sai Bharathi v. J. Jayalalitha & Others (2004) 2 SCC 9
- All seem related to 'Article 227'.
- In current_case_002.txt: there are two Articles quoted.

Approach

- Find all the Article numbers from case under query using regex (it can be "Article", 'Art.', etc)
- Search all prior cases having these articles.
- Collect prior cases by doc2vec top 10 similarity as well.
- Give out unique ones from union of regex based as well as doc2vec based