PROJECT REPORT

MULTI DOCUMENT SUMMARIZATION OF NEWS ARTICLES USING DEEP LEARNING

Introduction

Automatic summarization is a complex NLP task. There are two types of automatic summarization techniques widely used. They are Extractive Summarization and Abstractive Summarization.

In Extractive Summarization, the system extracts the objects(read sentences) from collection, without modifying the object themselves. However, in Abstaction Summarization the text is condensed to generate a summary, which requires natural language generation. Extractive summarization is a more common technique then abstractive summarization because it is a more easier task.

Here, I have divided the given task in three parts:

- Crawling the news articles
- Clustering them on the basis of semantic similarity
- Generating Summary

1. Crawling the News Articles

I crawled the news articles from “The Hindu” newspaper. For this I used the Scrapy Framework. It is and elaborate framework which allows us to design crawler according to our need and customize which information we need to store from the page by specifying there XML or HTML selectors. It also allows to store the crawled pages in form of a list json objects which can be used later for processing.Here, around 34,000 articles were crawled.

2. Clustering News Articles on the basis of Semantic Similarity

To cluster the News Articles on the basis of semantic similarity I used gensim to find similarity among articles and then applied K-means clustering to find similarity. First, the stopword removal and stemming was done. The they were given as input to gensim to construct the similarity matrix. Since, K-means clustering requires matrix to be in memory and a 34,000*34,000 matrix is difficult to handle, I reduced it to 5000*5000 i.e. we just took 5000 articles for clustering. And 2000 articles to construct background corpus for gensim.

Using Kmeans clustering I constructed 100 clusters of documents based on semantic similarity.

3. Automatic Summarization of News Articles

This was the main and the toughest task in the whole project. I have tried to do extractive

summarization for a set of documents through a selection of sailent sentences. There are two major things to be done in case of extractive summarization i.e. sentence selection and sentence ranking. In sentence ranking, we give a score to each sentence on the basis of some features associsted with words and sentences. In sentence selection, we select the sentence on basis of score and on the basis of redundancy among sentences.

Using Recursive Neural Networks for ranking has three major benefits:

• It transforms sentence ranking into a hierarchical regression task. Therefore more supervision knowledge can be taken into account.

• It is capable of automatically learning additional ranking features for a sentence as well as the constituents over the parsing tree.

• It provides consistent ranking scores from words to sentences, which makes more accurate sentence selection method possible.

Ranking Framework

The ranking for sentences is done in the following way:

1. We compute the sailency score for each sentence using it's parse tree on the basis of ROUGE-1 and ROUGE-2.For a pre-terminal node(i.e. directly linked to words), we just use the ROUGE-1(R_1) score. For an upper node, the combination of ROUGE-1 (R_1) and ROUGE-2 (R_2) scores is adopted to measure its importance. The scoring formula is given by:

$$s(n) = \begin{cases} R_1(n), n \in N_w \\ \alpha R_1(n) + (1-\alpha) R_2(n), n \in N - N_w \end{cases}$$

where,N ={n} is the set of whole non-terminal nodes and N_w is the set of pre-terminal nodes.Here, α is set at 0.5. When there are multiple references, we choose the maximal value. This sailency score is later used to calculate errors using the Cross Entropy function which is defined as :

$$CE(s(n), \hat{s}(n)) = \\ -\left(s(n) \ln \hat{s}(n) + (1 - s(n)) \ln(1 - \hat{s}(n))\right)$$

where ŝ(n) is sailency score calculated by our ReNN.

2. Now, the next step involves designing the ReNN. Our, ReNN doesn't take word embeddings as input. Instead, it takes word and sentence wise features as input. The features are :

Feature	Description
TF	Term frequency over the cluster.
IDF	Total document number in the datasets, divided by the frequency of documents which contains this word.
CF	The frequency of documents which contains this word in the current cluster.
POS	A 4-dimension binary vector indicates whether the word is a noun, a verb, an adjective or an adverb. If the word has another part-of-speech, the vector is all-zero.
Named entity	A binary value equals one iff the output of the named entity classifier from CoreNLP is not empty.
Number	A binary value denotes whether the word is a number.
Sentence length	The maximal length of sentences owning the word.
Sentence TF	The maximal TF score of sentences owning the word.
Sentence CF	The maximal CF score of sentences owning the word.
Sentence IDF	The maximal IDF score of sentences owning the word.
Sentence sub-sentences	The maximal sub-sentence count of sentences owning the word. A sub-sentence means the node label is "S" or "@S" in the parsing tree.
Sentence depth	The maximal parsing tree depth of sentences owning the word.
Position	The position of the sentence. Supposing there are M sentences in the document, for the i_{th} sentence, the position feature is computed as $1 - (i-1)/(M-1)$.
Length	The number of words in the sentence.
Sub-sentences	Sub-sentence count of the parsing tree.
Depth	The root depth of the parsing tree.
Averaged TF	The mean TF values of words in the sentence, divided by the sentence length.
Averaged IDF	The mean word IDF values in the sentence, divided by the sentence length.
Averaged CF	The mean word CF values in the sentence, divided by the sentence length.
POS ratio	The numbers of nouns, verbs, adjectives and adverbs in the sentence, divided by the sentence length respectively.
Named entity ratio	The number of named entities, divided by the sentence length.
Number ratio	The number of digits, divided by the sentence length.
Stopword ratio	The number of stopwords, divided by the sentence length. We just use the stopword list of ROUGE.

The top 12 features are word features. They are calculated for each word. The other features seprated by a double line are sentence features.They are calculated for each sentence. These features are normalized to have value between [0,1].

3. These features are used at two different stages in the ReNN. First, a projection layer is added to transform raw features f_w to hidden features f_h.

$$f_h = HT(f_w \text{ X } W_p)$$

where W_p has dimensions $k_w X k_h$ (k_w is the number of word features and k_h is the number of hidden features)

4. Here, now we introduce two regression matrices Wr2 and Wr3 for word-level and sentence level regression respectively. Because, in practice it has been seen that raw features improve performance greatly over learned features.Now, we can find the sailence score for each case i.e. pre-terminal nodes(Nw), root nodes(Ns), and the rest internal nodes(Ni) according to the formula :

$$\hat{s}(n) = \begin{cases} \sigma(l(n) \times W_{r1} + f_w \times W_{r2}), n \in N_w \\ \sigma(l(n) \times W_{r1}), n \in N_i \\ \sigma(l(n) \times W_{r1} + f_s \times W_{r3}), n \in N_s \end{cases}$$

where $l(n) = a_t([l(c1),l(c2)]XW_t)$.

5. In the forward propagation step, the word-level features f_w are first projected into hidden features f_h. Then,the hidden features are used as inputs of RNN to compute all the upper node representation l(n). Finally, different kinds of regression are conducted over the tree.In the backward propagation

step, weights W_p ,W_t and W $r_{1\sim3}$ are updated with the guide of all the supervisions. Using CE to measure errors and HT as the activation function, this step can be computed as fast as linear models.

In this way the ReNN provides ranking scores for both words and sentences.

Sentence Selection

For sentence selection I used the greedy approach. In this approach we select the most sailent sentence according to the sailency score. In each step we select the sentence which has the highest sailency score and it's similarity with any sentence in the group is less than a given threshold.

References :

- https://en.wikipedia.org/wiki/Automatic_summarization
- https://www.semanticscholar.org/paper/Ranking-with-Recursive-Neural-Networks-and-Its-Cao-Wei/0ce17d4d41b1291e942ec3bf6c102f3ae3034b3d/pdf
- http://stanfordnlp.github.io/CoreNLP/
- *https://github.com/dasmith/stanford-**corenlp-python***