Research Reviews

1. *Outline*

For every research paper reviewed, write 4 paragraphs outlining: the aim of the paper, the methodology used by the authors in detail, experiments carried out, evaluation of the results, conclusions.

1. Extractive Text Summarization Using Deep Learning – Serry

This paper proposes an extractive text summarization approach utilizing a deep learning model. The approach involves a mixture of engineering and extracting lower level features based on methods that are available in wider literature, latent features are then extracted using a restricted bolts man machine to enhance the quality of the generated summaries without any loss of information. Results are compared to existing approaches that utilize RBM’s

The proposed approach involves multiple phases, the first phase *‘Preprocessing’* involves segmenting documents in the corpus into paragraphs to keep track which sentence belongs to which paragraph and the position of a sentence in its respective paragraph. Paragraphs are further divided into sentences followed by breaking down each sentence into words that are lemmatized, stop words are also removed from sentences. Lastly, each word is tagged with respect to its grammatical identifier i.e. noun, verb…etc.

the second phase *‘Feature Extraction’* involves engineering features that would be utilized to enhance overall quality of summaries generated. The features selected by the author are as follows: **number of thematic words** the 10 most frequently recurring words in a text, **sentence position**, **sentence length, sentence position relative to paragraph**, this comes from an observation by the author that a new discussion begins at the start of each paragraph. The next feature proposed is **number of proper nouns per sentence**, **number of numerals**, **number of named entities**, **TF-ISF** which is like the TF-IDF weighting scheme but tailored to one document only. Lastly **sentence-centroid similarity**, the sentences with the highest TF-ISF score is considered a centroid and all other sentences are compared to it.

The last phase is *‘Feature Enhancement’* by the end of the second phase a sentence-feature matrix is generated. Each sentence is represented as feature vector of length 9. Each sentence vector is then given as input to an RBM with 1 hidden layer and 1 visible layer where each sentence feature is multiplied by the learned weights in the RBM resulting in a more enhanced sentence-feature matrix. The RBM is trained for each new document. The last step is the generation of a summary, the sentence feature vectors are summed to give a final score for each sentence. Sentences with the highest score are used for the summary.

For experimentation, the author obtained factual reports from several domains such as health, technology. The approach mentioned above was applied on each report and the generated summaries were compared to those produced by human experts. The evaluation of results is carried out based on the following metrics: F-measure, recall, precision. The results were then compared to an ‘existing approach’ that uses RBMs to indicate a fair improvement in performance.

In Conclusion, the paper provides a reasonable approach to summarization. However, the author remarks that existing approaches utilize 2 or more layers. He further remarks that 1 hidden layer would suffice given the size of the training data available. Many such ‘given conclusions’ are to be observed through out the paper without clear evidence. Furthermore, the existing approaches he mentions are no where to be found in the paper making it difficult to evaluate the results.

Bruce’s Review of above paper:

Abstract,

Samidha et al. propose an Extractive Text summarization model using RBM in combination with fuzzy logic to generate a meaningful losses summary of large singular text documents. The model takes several word and sentence features which are used to generate two summaries with RBM and Fuzzy logic these summaries are then combined to make the final summary.

Introduction,

This paper reiterates H. P. Luhn’s work on summarization, that words with a median frequency would be the most important for generating the summary, that the starting and end 7% of the document have the most relevant sentences and that short sentences tend to have less information than long sentences.

Models,

The documents are input as a .txt file, the text is then pre-processed. First the sentences are segmented and their positions in the document is taken down. These sentences are further split by tokenization into words, then the stop words and punctuation are removed.

The next step is feature extraction based on evaluating the relevance of the sentences based on several calculations.

The position of the sentence in the document is ranked based on a calculation metric:

If (first or last sentence) = 1

Else = (Total number of sentences- position of sentence)/ Total number of sentences

After the sentences are ranked the length feature score is calculated:

Length = Words in sentence/Words in the longest sentence

The next score is a calculation of the number of tokens in a sentence that are numeric:

Numeric tokens = numeric token in sentence/tokens in sentence

The next calculate metric is a variation of Term frequency-inverse document frequency, since the model is based on a single document the metric is Term frequency - Inverse Sentence Frequency (TF-ISF) calculated by:

TF-ISF =(Log(isf)\*(tf))/len

With the TF-ISF the cos similarity between a sentence and the sentence with the largest TF-ISF (centroid) is calculated:

cosSimilarity =cos(sentence,centroid)

=(sentence\*centroid)/(||senence||\*||centroid||)

The Bi-grams and Tri-grams are calculated using the NLTK libraries methods.

The proper noun counts relative to the sentence:

proper noun score = Proper nouns/ sentence length

The last feature metric is a count of thematic words:

Thematic word= thematic words in the sentence/total of thematic words

The features are then combined to form a sentence feature matrix, as a result each sentence will have nine features. The matrix is then normalized by dividing the values by the largest value which is then fed into the RBM.

RBM’s are a variant of Boltzmann Machines (BM) which are stochastic generative networks, in BM’s each note is connected to the other with symmetric connections, In a RBM model there is a clear division of visible and hidden nodes where the visible notes are only connected to nodes in the next layer and not parallel nodes. The RBM and BM do not process inputs linearly instead nodes are reset changing their states based on a Boltzmann distribution until the entire network is in equilibrium where the probability distribution has converged across the network.

In the RBM this equilibrium is achieved through forward and backward passes from and to the visible nodes, a random bias is added to the hidden nodes on the forward and backward pass but it is only added to the visible node on the backward pass.

For the forward pass, the following equation is used to determine the probability whether a node will activate:

p(𝑆i |𝑠j ) = σ(∑ sj × wij + bi )

Where Si is the following node and sj is the preceding node, and the sigmoid equation is:

𝜎(𝑥) = 1/(1+e- 𝑥)

For the backward pass a different equation is used to calculate the values for the same sigmoid equation:

p(𝑠j |𝑆i ) = σ(∑mi=1 sj × wij + bi )

Through these equations the values of the nodes inputs are predicted this is known as Gibbs Sampling. The differences between the input values and the new values are used to gain the training loss using contrasted difference calculated by the following equations:

wij new = wij old + (LR × Δw)

Where LR is the learning rate, Δw the change in weights as given by the difference of the probabilities of the node outputs:

Δw = (𝑠j ⊗ 𝑝(𝑆i|𝑠j)− 𝑠j′ ⊗ 𝑝(𝑆i ′|𝑠j ′)

Once the network has reached equilibrium then an improved feature matrix is returned as an output.

With this improved feature matrix, the RBM based summary can be derived. Using this matrix, the values of these improved features are summed together to give a sentence score. To generate the summary the first sentence of the document is used regardless of its score then the remaining sentences are ranked by score the upper 50% are taken for the summary and then sorted according to their position in the original document to give the first summary.

The fuzzy logic summary is generated by converting the previous feature scores into percentages these percentages are then sorted using triangular membership functions: HIGH, MEDIUM and LOW. Fuzzy logic IF-THEN rules are applied to this set to de fuzz the sentences into new categories of Important, Average and Unimportant the Important are then used to create the second summary by sorting then into their occurrence in the original document.

To generate the final summary sentences common to both are taken from both, the remaining uncommon sentences are then sorted by their score and the top 50% are taken. These sentences like before are then sorted into their original position in the document.

Experiments,

Two main experimental paths were taken a control and the proposed method, the control would the RBM generated summary to act as a baseline to the results gained from the proposed method if it improved upon the RBM summary. To evaluate the results the ROUGE metric was used, using the precision, recall and F measure to compare the two summaries.

Evaluation

Ten documents were summarized, and the evaluation results were compared between the two from the precision, recall and F-measure the new method showed better results by a few decimal points.

Conclusions

While the results showed that the proposed method was better there was no review of what the contents of the documents to be evaluated were and how the size of the document, its sentences might affect the summarization techniques or how the nine features contribute to the model.

1. Text summarization using unsupervised deep learning – Bruce

Yousefi et al use an Auto Encoder (AE) a type of unsupervised deep learning neural network to refine the features in the term frequencies of a document for summarization, using local and global vocabularies. The paper investigates the effect of adding noise to the term frequency before processing it with the encoder, creating a set of AEs’ called the Ensemble Noisy Auto Encoder (ENAE). This ensemble adds random noise to the input term frequencies this changes the network from a feed forward model to a stochastic run model. AE’s with local vocabularies are useful in finding the most important features and the ENAE will improve this. This model is run on a corpus of emails and evaluated using the ROUGE and ROUGE2 metrics which are specific to auto summarization and NLP applications.

Introduction,

It is difficult to understand the relevance of a large document so a condensed writeup can be particularly helpful, summaries require unbiased reviewers to read the document which can take time. The implementation of automated summarization on large corpora of documents can help in the dissemination of information across the internet by reducing time searching vast sources of information.

The two types of text summarization are abstractive and extractive, abstractive focused on the individual words and re generates texts and content while extractive instead ranks sentences by prominence. Research in automated summarization has come to the consensus that summarization models work best on documents with similar topical content, or more general, less effective models can be created instead. These models can be adjusted to summarize based on a query or to work on a single document of a collection of documents, the paper proposes a query based extractive single document summarization technique.

Deep learning was used in this model as it showed promise in NLP applications, the proposed unsupervised model finds latent representations of a given query and sentences in the document. The main aim of the model is organizing the features and extracting important features from low level features without any human interaction. Unlike NLP implementations the proposed model avoids sparsity in its text representation by created a local vocabulary to reconstruct the input text and by adding random noise to the word representation vector.

Running a single noisy AE would give a ranking of the most important sentences to gain a more accurate ranking multiple AE runs with different randomized noise components are performed and the component summaries are cross referenced against each other and the most popular sentences are used in the final summary.

Previously unsupervised and supervised deep learning have been used in summarization implementations both abstractive and extractive, at first several neural network implementations were used with promising results: Shallow neural nets, Multilayer perceptron’s with fuzzy logic, Recurrent Neural Network (RNNs), convolutional neural network (CNN), Feed forward neural networks.

Yousefi et al.’s implementation focused on the reconstructive ability of AE, the inputs are corrupted with random noise this corruption is undone by the network and this leads to dependencies and importance of the different segments of the input being revealed. Core features of this implementation that differentiates it from standard De-noising AE implementations are:

1. A small amount of random noise is added to all inputs, Standard AE’s use a random zero mask.

2. The resulting output is the same as the input

3.Noise is added to the training and test data while in standard implementations test data is not manipulated.

Models:

The AE neural network is feed forward network, the main feature of this network is the bottleneck in it hidden layer, its input and output layers have the same number of nodes and the network replicates its input as its output. What makes this network interesting is that its hidden late has a bottleneck a layer where the number of neurones is much smaller than that of any other layer, (Neuron counts in the Layers drop closer to the bottle next and increase after it) this makes the network recreate the input from sparse features and through this finding the most important features. This mean that the output is a very close approximation that it is recognisable, but not exact.

Their model has two training stages Pre-training and fine tuning, in the pre-training stage a separate neural network is used to generate training weights for the AE network. The paper uses a Restricted Boltzmann Machine (RBM) to generate weights, a two-layer neural network the single hidden layer of the RBM helps reduce features. Several RBMs were run (Gaussian–Bernoulli, Bernoulli–Bernoulli, Noisy Rectified Linear Unit (NReLU)) as a stack where the outputs of one RMM is fed into the next, then back propagation was used to refine the weights.

Their implementation uses a custom representation for the words in the corpus to reduce the sparsity of representations like bag of words a local vocabulary is proposed only terms in a single document are considered with the vocabularies being the same size across the corpus this reduces sparsity in the vocabulary.

Experiments:

The concept of the model is that a single encoder is run for a single document and the ensemble results from each AE run is fused together to create a model for the corpus this fusion changes the model from a feed forward network to a stochastic model.

This ensemble network was run on two e-mail corpora that are specifically focused on summarization training, Summarization and Keyword Extraction from Emails (SKE) and the BC3 from British Columbia University. The queries used on the emails were either the subject of the email and for those that did not have a subject keyword that are seen through the dataset are used.

To get a base line tf-idf for the equivalent documents was carried out and different vocabulary sizes were used (1000, 60, 5%, 6%). The experimental AE had the structure of 140,40,30,10 in the hidden layer with 10 being the bottleneck and the reverse going to the decoder. The aim was that the first layer would being around double the size of the smallest vocabulary, the next layers were added as a deeper AE was seen to be more effective than shallow networks in previous research.

Evaluation:

Using the ROUGE and ROUGE2 metrics for evaluations they provide baselines for n-gram overlap for the auto summarization results of the different models based on the number of sentences included in a summary.

The Local vocabulary term frequency (Ltf) showed lower results than tf-idf of large vocabulary sets (above 5%) but the AE of Ltf had better results than both.

From the results it is seen that ENAE models have lower results than tf-idf when single sentences are used in the summary with an increase in sentences the results are much better with runs having uniform noise added scoring higher than gaussian noise.

When key value summarization is performed the results are similar ENAE uniform has the highest scores across all models, and the ENAE models being on average equal to or better than tf-idf the ENAE models show better results with larger summaries.

When compared to ROUGE results of abstractive text summarization models the Ltf and Ltf-AE models do not score as high as their tf-idf equivalents, this is a sharp contrast to the other experiments results.

The paper also evaluates the error values of the various models that were tested.

Conclusions

The paper suggests a new method of auto summarizing documents using focused keywords and subjects to generate the summary. The evaluation of the model and number of experiments was remarkably in depth, one experiment that I felt should have been done is a generalized summary instead of topic based one.

1. Abstractive Text Summarization – Tamer

Abstract

Neural networks models have shown a lot of progress despite they possessed some challenges like unnatural and repetitive. . This paper takes an abstractive neural network model and suggest 6 improvements to improve its summarization output, it explains the model they considered and the metrics they use to measure the suggest improvements ,

Baseline Attention Model

Abstractive Text Summarization is an Encoding – Decoder implementation, where the text is encoded into hidden layer and then the decoder decodes the hidden layer to produce the summary

Both encoder and Decoder use LSTM Recurrent Neural Network with Attention Mechanism that calculates the importance of each input , creating context vector that is used to generate probability distribution over the target vocabulary

Metrics

Similarity will be used as a matric, If target summary is absent, they will use Topic Modeling to measure the similarity

If they have Target summary exists, they will use ROUGE Metric where ROUGE-N measure N-Gram similarity , ROUGE-L which measure Sentence level similarity and ROUGE-S which is Skip-gram Similarity

Suggested Improvements

• Large Vocabulary

o Use more linguistic rich features for the imput like POS (Part of speech), named-Entity and TF-IDF

• Hierarchical Attention

o Use of -Bi-Directional RNN on the word and Sentence level

• Pointer Generator Network

o Solves the out of vocab problem (UNK) by copying from (Pointing) the source

• Coverage Mechanism

o This technique solve the problem of Repetitive Attention by penalizing Attention on repetive words making the model focus on words that was not covered, thus “Coverage”

• Intra-Attention on Decoder Output

o Same like Coverage Mechanism but consider also Decoder output , this avoids repeating words that has been generated already

• Learning From Mistakes using reinforced learning

o The model output is compared to the reference summary using ROUGE metric , we iterate using Reinforced learning till we get a high score ROUGE score

Our Findings

To prepare for this classwork we have went through several research papers, we found that most of the papers ,including this one, focus on the technology and not the user , none of them suggested any improvements based on the user preferences, without any customization to the user location, mother tongue or preferences . we believe text summarization should incorporate user preferences and the context and adjust the model accordingly