icoxfog417/awesome-text-summarization

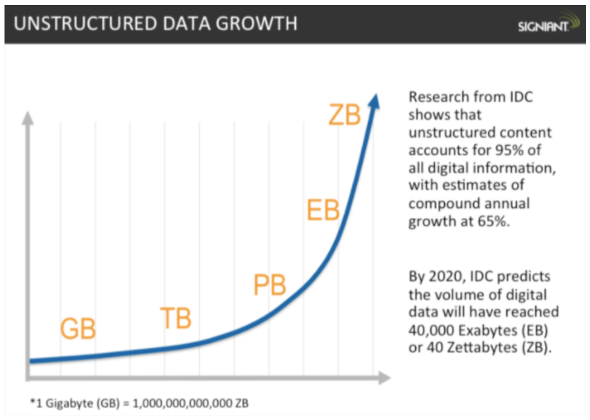
README.md

The guide to tackle with the Text Summarization.

- Motivation
- Task Definition
- Basic Approach
  - Extractive
  - Abstractive
- Evaluation
- Resources
  - Datasets
  - Libraries
  - Articles
  - Papers

Motivation

To take the appropriate action, we need latest information.
But on the contrary, the amount of the information is more and more growing. There are many categories of information (economy, sports, health, technology...) and also there are many sources (news site, blog, SNS...).



from THE HISTORICAL GROWTH OF DATA: WHY WE NEED A FASTER TRANSFER SOLUTION FOR LARGE DATA SETS

So to make an automatically & accurate summaries feature will helps us to **understand** the topics and **shorten the time** to do it.

Task Definition

Basically, we can regard the "summarization" as the "function" its input is document and output is summary. And its input & output type helps us to categorize the multiple summarization tasks.

- Single document summarization
  - *summary = summarize(document)*

- Multi-document summarization
    - *summary = summarize(document_1, document_2, ...)*

We can take the query to add the viewpoint of summarization.

- Query focused summarization
    - *summary = summarize(document, query)*

This type of summarization is called "Query focused summarization" on the contrary to the "Generic summarization". Especially, a type that set the viewpoint to the "difference" (update) is called "Update summarization".

- Update summarization
    - *summary = summarize(document, previous_document_or_summary)*

And the *"summary"* itself has some variety.

- Indicative summary
    - It looks like a summary of the book. This summary describes what kinds of the story, but not tell all of the stories especially its ends (so indicative summary has only partial information).
- Informative summary
    - In contrast to the indicative summary, the informative summary includes full information of the document.
- Keyword summary
    - Not the text, but the words or phrases from the input document.
- Headline summary
    - Only one line summary.

Discussion

`Generic summarization` is really useful? Sparck Jones argued that summarization should not be done in a vacuum, but rather done according to the purpose of summarization (2). She argued that generic summarization is not necessary and in fact, wrong-headed. On the other hand, the headlines and 3-line summaries in the newspaper helps us.

Basic Approach

There are mainly two ways to make the summary. Extractive and Abstractive.

Extractive

- Select relevant phrases of the input document and concatenate them to form a summary (like "copy-and-paste").
    - Pros: They are quite robust since they use existing natural-language phrases that are taken straight from the input.
    - Cons: But they lack in flexibility since they cannot use novel words or connectors. They also cannot paraphrase like people sometimes do.

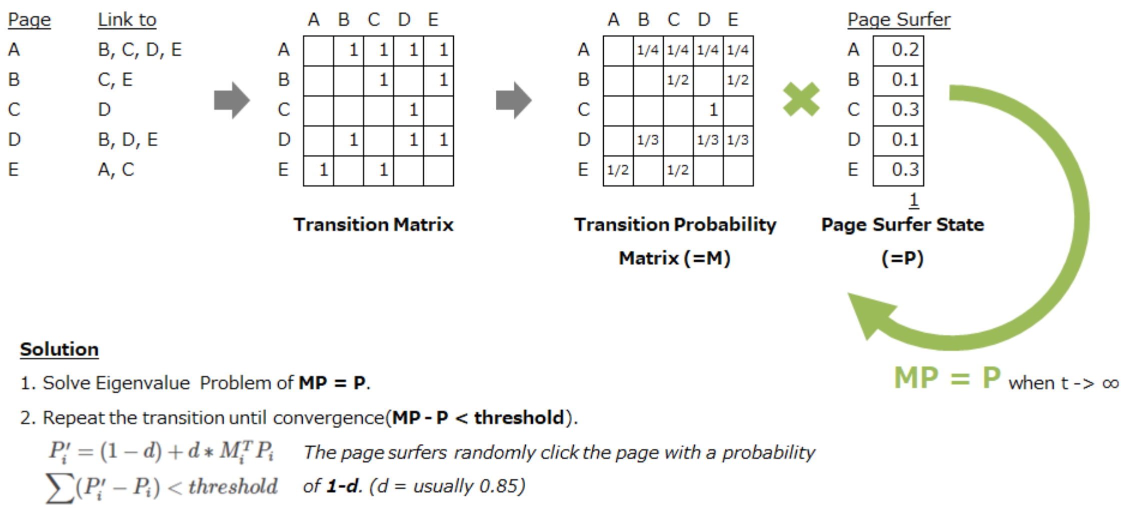
Now I show the some categories of extractive summarization.

Graph Base

The graph base model makes the graph from the document, then summarize it by considering the relation between the nodes (text-unit). `TextRank` is the typical graph based method.

TextRank

TextRank is based on PageRank algorithm that is used on Google Search Engine. Its base concept is "The linked page is good, much more if it from many linked page". The links between the pages are expressed by matrix (like Round-robin table). We can convert this matrix to transition probability matrix by dividing the sum of links in each page. And the page surfer moves the page according to this matrix.



Page	Link to
A	B, C, D, E
B	C, E
C	D
D	B, D, E
E	A, C

Transition Matrix

	A	B	C	D	E
A		1	1	1	1
B			1		1
C				1	
D		1		1	1
E	1		1		

Transition Probability Matrix (=M)

	A	B	C	D	E
A		1/4	1/4	1/4	1/4
B			1/2		1/2
C				1	
D		1/3		1/3	1/3
E	1/2		1/2		

Page Surfer State (=P)

Page Surfer	
A	0.2
B	0.1
C	0.3
D	0.1
E	0.3

Solution

1. Solve Eigenvalue Problem of **MP = P**.
2. Repeat the transition until convergence(**MP - P < threshold**).

$P_i' = (1-d) + d * M_i^T P_i$ *The page surfers randomly click the page with a probability*

$\sum(P_i' - P_i) < threshold$ *of **1-d**. (d = usually 0.85)*

Page Rank Algorithm

TextRank regards words or sentences as pages on the PageRank. So when you use the TextRank, following points are important.

- Define the "text units" and add them as the nodes in the graph.
- Define the "relation" between the text units and add them as the edges in the graph.
  - You can set the weight of the edge also.

Then, solve the graph by PageRank algorithm. LexRank uses the sentence as node and the similarity as relation/weight (similarity is calculated by IDF-modified Cosine similarity).

If you want to use TextRank, following tools support TextRank.

- gensim
- pytextrank

Feature Base

The feature base model extracts the features of sentence, then evaluate its importance. Here is the representative research.

Sentence Extraction Based Single Document Summarization

In this paper, following features are used.

- Position of the sentence in input document
- Presence of the verb in the sentence
- Length of the sentence

- Term frequency
- Named entity tag NE
- Font style

...etc. All the features are accumulated as the score.

$$Score(l,w) = \Pi_i f_i(w)$$
$$Score(l) = \sum_i Score(l, w_i)$$
$$Score(l) = Score(l) + ((No.ofcoreferences) \times SPW(l-1))$$
$$SPW(l) = \frac{Score(l)}{length(l)}$$

The `No.of coreferences` are the number of pronouns to previous sentence. It is simply calculated by counting the pronouns occurred in the first half of the sentence. So the Score represents the reference to the previous sentence.

Now we can evaluate each sentences. Next is selecting the sentence to avoid the duplicate of the information. In this paper, the same word between the new and selected sentence is considered. And the refinement to connect the selected sentences are executed.

Luhn's Algorithm is also feature base. It evaluates the "significance" of the word that is calculated from the frequency.

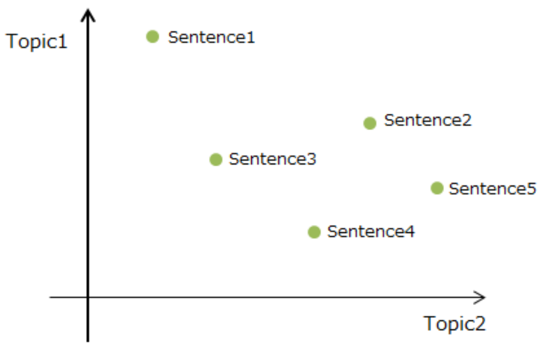
You can try feature base text summarization by TextTeaser (PyTeaser is available for Python user).

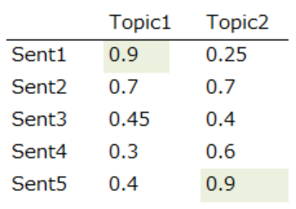
Topic Base

The topic base model calculates the topic of the document and evaluate each sentences by what kinds of topics are included (the "main" topic is highly evaluated when scoring the sentence).

Latent Semantic Analysis (LSA) is usually used to detect the topic. It's based on SVD (Singular Value Decomposition).
The following paper is good starting point to overview the LSA(Topic) base summarization.

Text summarization using Latent Semantic Analysis





	Topic1	Topic2
Sent1	0.9	0.25
Sent2	0.7	0.7
Sent3	0.45	0.4
Sent4	0.3	0.6
Sent5	0.4	0.9

The simple LSA base sentence selection

There are many variations the way to calculate & select the sentence according to the SVD value. To select the sentence by the topic(=V, eigenvectors/principal axes) and its score is most simple method.

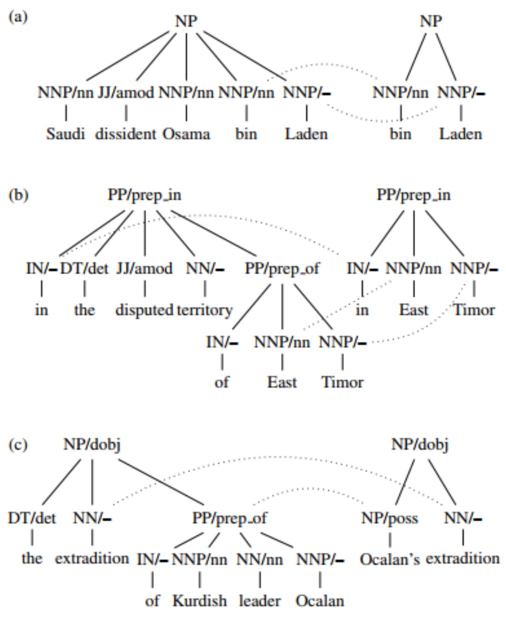
If you want to use LSA, gensim supports it.

- gensim models.lsimodel

Grammer Base

The grammer base model parses the text and constructs a grammatical structure, then select/reorder substructures.

Title Generation with Quasi-Synchronous Grammar



This model can produce meaningful "paraphrase" based on the grammatical structure. For example, above image shows the phrase "in the disputed territory of East Timor" is converted to "in East Timor". To analyze grammatical structure is useful to reconstruct the phrase with keeping its meaning.

Abstractive

- Generate a summary that keeps original intent. It's just like humans do.

- Pros: They can use words that were not in the original input. It enables to make more fluent and natural summaries.
- Cons: But it is also a much harder problem as you now require the model to generate coherent phrases and connectors.

Extractive & Abstractive is not conflicting ways. You can use both to generate the summary. And there are a way collaborate with human.

- Aided Summarization
  - Combines automatic methods with human input.
  - Computer suggests important information from the document, and the human decide to use it or not. It uses information retrieval, and text mining way.

The beginning of the abstractive summarization, Banko et al. (2000) suggest to use machine translatation model to abstractive summarization model. As like the machine translation model converts a source language text to a target one, the summarization system converts a source document to a target summary.

Nowadays, encoder-decoder model that is one of the neural network models is mainly used in machine translation. So this model is also widely used in abstractive summarization model. The summarization model that used encoder-decoder model first achieved state-of-the-art on the two sentence-level summarization dataset, DUC-2004 and Gigaword.

If you want to try the encoder-decoder summarization model, tensorflow offers basic model.

- Text summarization with TensorFlow

Encoder-Decoder Model

The encoder-decoder model is composed of encoder and decoder like its name. The encoder converts an input document to a latent representation (vector), and the decoder generates a summary by using it.

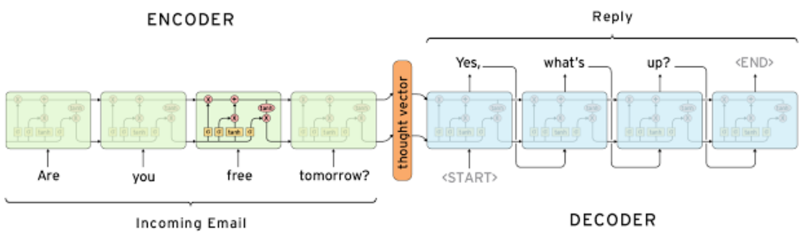


Diagram by Chris Olah

- from Computer, respond to this email*

But the encoder-decoder model is not the silver bullet. There are many remaining issues are there.

- How to set the focus on the important sentence, keyword.
- How to handle the novel/rare (but important) word in source document.
- How to handle the long document.
- Want to make more human-readable summary.
- Want to use large vocabulary.

Researches based on Encoder-Decoder Model

A Neural Attention Model for Sentence Summarization

- *How to set the focus on the important sentence, keyword.?*
    - use Attention (sec 3.2)
- *How to handle the novel/rare (but important) word in source document.*
    - add n-gram match term to the loss function (sec 5)
- Other features
    - use 1D convolution to capture the local context
    - use beam-search to generate summary
- Implementation

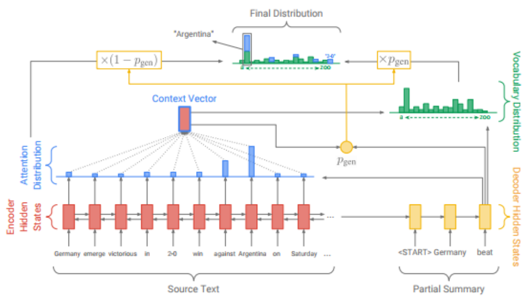
Abstractive Text Summarization Using Sequence-to-Sequence RNNs and Beyond

- *How to set the focus on the important sentence, keyword.*
    - use enhanced feature such as POS, Named Entity tag, TF, IDF (sec 2.2)
- *How to handle the novel/rare (but important) word in source document.*
    - switch the decoder(generate word) and pointer(copy from original text). (sec 2.3)
- *How to handle the long document.*
    - use sentence level attention (sec 2.4)
- *Want to use large vocabulary.*
    - use subset of vocabulary on the training (sec 2.1, please refer On Using Very Large Target Vocabulary for Neural Machine Translation)

Get To The Point: Summarization with Pointer-Generator Networks

- *How to set the focus on the important sentence, keyword.*

    - use Attention (sec 2.1)
- *How to handle the novel/rare (but important) word in source document.*
    - switch the decoder(generator) and pointer network (by `p_gen` probability).
    - combine the distribution of vocabulary and attention with `p_gen` and (1 - `p_gen`) weight (please refer the following picture).
- Implementation



A Deep Reinforced Model for Abstractive Summarization

- *How to set the focus on the important sentence, keyword.*
    - use intra-temporal attention (attention over specific parts of the encoded input sequence) (sec 2.1)
- How to handle the novel/rare (but important) word in source document.
    - use pointer network to copy input token instead of generating it. (sec 2.3)
- *Want to make more human-readable summary.*
    - use reinforcement learning (ROUGE-optimized RL) with supervised learning. (sec 3.2)

- Other features
    - use intra-decoder attention (attention to decoded context) to supress the repeat of the same phrases. (sec 2.2)
    - constrain duplicate trigram to avoid repetition. (sec 2.5)

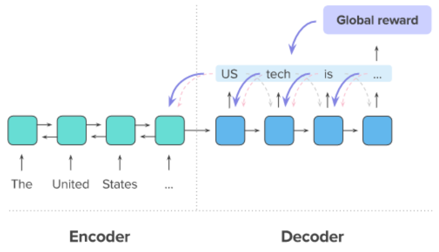
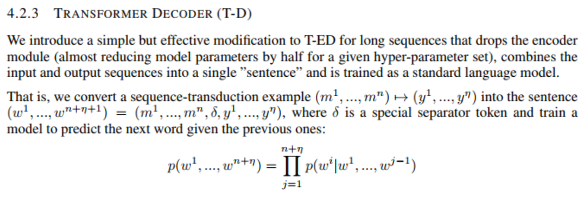


Figure 8: Combination of supervised learning (in red) and reinforcement learning (in purple), showing how our model can learn both local and global rewards and optimize both for readability and overall ROUGE score.

from [Your tldr by an ai: a deep reinforced model for abstractive summarization]

[Generating Wikipedia by Summarizing Long Sequences]

- *How to set the focus on the important sentence, keyword.*
    - use the encoder-less self-attention network (sec 4.2.3~). it concatenates input & output, and predict the next token from the previous sequence.
- *How to handle the long document.*
    - use the extractive model to extract tokens from long document first, then execute the abstractive model.



4.2.3 TRANSFORMER DECODER (T-D)

We introduce a simple but effective modification to T-ED for long sequences that drops the encoder module (almost reducing model parameters by half for a given hyper-parameter set), combines the input and output sequences into a single "sentence" and is trained as a standard language model.

That is, we convert a sequence-transduction example $(m^1, ..., m^n) \mapsto (y^1, ..., y^\eta)$ into the sentence $(w^1, ..., w^{n+\eta+1}) = (m^1, ..., m^n, \delta, y^1, ..., y^\eta)$, where δ is a special separator token and train a model to predict the next word given the previous ones:

$$p(w^1, ..., w^{n+\eta}) = \prod_{j=1}^{n+\eta} p(w^i | w^1, ..., w^{j-1})$$

Evaluation

[ROUGE-N]

Rouge-N is a word N-gram count that matche between the model and the gold summary. It is similart to the "recall" because it evaluates the covering rate of gold summary, and not consider the not included n-gram in it.

ROUGE-1 and ROUGE-2 is usually used. The ROUGE-1 means word base, so its order is not regarded. So "apple pen" and "pen apple" is same ROUGE-1 score. But if ROUGE-2, "apple pen" becomes single entity so "apple pen" and "pen apple" does not match. If you increase the ROUGE-"N" count, finally evaluates completely match or not.

[BLEU]

BLEU is a modified form of "precision", that used in machine translation evaluation usually. BLEU is basically calculated on the n-gram co-occerance between the generated summary and the gold (You don't need to specify the "n" unlike ROUGE).

Resources

Datasets

- DUC 2004
- Opinosis Dataset - Topic related review sentences
- 17 Timelines
- Legal Case Reports Data Set
- Annotated English Gigaword

Libraries

- gensim
  - `gensim.summarization` offers TextRank summarization
  - `gensim models.lsimodel` offers topic model
- pytextrank
- TextTeaser
  - PyTeaser for Python user
- TensorFlow summarization
- sumeval
  - Calculate ROUGE and BLEU score

Articles

- Wikipedia
  - Automatic summarization
- Blogs
  - Text summarization with TensorFlow
  - Your tl;dr by an ai: a deep reinforced model for abstractive summarization

Papers

Overview

1. A. Nenkova, and K. McKeown, "Automatic summarization,". Foundations and Trends in Information Retrieval, 5(2-3):103–233, 2011.
2. K. Sparck Jones, “Automatic summarizing: factors and directions,”. Advances in Automatic Text Summarization, pp. 1–12, MIT Press, 1998.

Extractive Summarization

1. R. Mihalcea, and P. Tarau, "Textrank: Bringing order into texts,". In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, 2004.
2. G. Erkan, and D. R. Radev, "LexRank: graph-based lexical centrality as salience in text summarization,". Journal of Artificial Intelligence Research, v.22 n.1, p.457-479, July 2004.
3. J. Jagadeesh, P. Pingali, and V. Varma, "Sentence Extraction Based Single Document Summarization", Workshop on Document Summarization, 19th and 20th March, 2005.
4. P.H. Luhn, "Automatic creation of literature abstracts,". IBM Journal, pages 159-165, 1958.
5. M. G. Ozsoy, F. N. Alpaslan, and I. Cicekli, "Text summarization using Latent Semantic Analysis,". Proceedings of the 23rd International Conference on Computational Linguistics, vol. 37, pp. 405-417, aug 2011.
6. K. Woodsend, Y. Feng, and M. Lapata, "Title generation with quasi-synchronous grammar,". Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, p.513-523, October 09-11, 2010

Abstractive Summarization

1. M. Banko, V. O. Mittal, and M. J. Witbrock, "Headline Generation Based on Statistical Translation,". In Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, pages 318–325. Association for Computational Linguistics, 2000.
2. A. M. Rush, S. Chopra, and J. Weston, "A Neural Attention Model for Abstractive Sentence Summarization,". In EMNLP, 2015.
    - GitHub
3. S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks,". In North American Chapter of the Association for Computational Linguistics, 2016.
4. R. Nallapati, B. Zhou, C. dos Santos, C. Gulcehre, and B. Xiang, "Abstractive text summarization using sequence-to-sequence RNNs and beyond,". In Computational Natural Language Learning, 2016.
5. S. Jean, K. Cho, R. Memisevic, and Yoshua Bengio. "On using very large target vocabulary for neural machine translation,". CoRR, abs/1412.2007. 2014.
6. A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointergenerator networks,". In ACL, 2017.
    - GitHub
7. R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization,". arXiv preprint arXiv:1705.04304, 2017.
8. P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer, "Generating Wikipedia by Summarizing Long Sequences,". arXiv preprint arXiv:1801.10198, 2018.