


09.Text Summarization

Tags: [#permanent](#)

Description:

Theme: [Mestrado ITA](#) , [CM 219 - Processamento de Linguagem Natural \(NLP\)](#), [01.1.Machine Learning for NLP](#), [\(10\) Redes Neurais Artificiais\(RNA\)](#), [02.Deep Learning for NLP](#), [Recurrent Neural Network \(RNN\)](#), [Feedback notation and Feedforward Computation](#), [Backpropagation through time\(BPTT\) & Truncated BPTT](#), [Text To Vector Representation](#), [Word Embeddings](#), [Long- Short-Term Memory Units \(LSTM\)](#), [Bidirectional RNN](#), [Encoder and Decoder](#), [Attention Mechanism](#), [03.Transformers](#), [04.Generative AI And LLM Models](#), [05.Langchain](#), [06.Langchain Gen Ai](#), [07.LLM With LCEL](#)

ID: 20250809212337

Brief Description

In LangChain, **text summarization** refers to the process of condensing one or more documents into a shorter form while preserving the key ideas, and it can be implemented through different summarization chains. The main techniques include:

1. Stuff

- All the text chunks are loaded into the LLM prompt at once.
- This is simple and works for small inputs within the model's context window.
- Limitation: Becomes impractical for large documents because of token limits.

2. Map-Reduce

- **Map step:** Each chunk of the document is summarized individually (parallelizable).
- **Reduce step:** The summaries from the map step are combined into a final summary (can be iterative if still too large).
- This technique is scalable and handles large inputs better, at the cost of potentially losing some cross-chunk context.

3. Refine

- The model starts with an initial summary from the first chunk, then iteratively refines it by incorporating each subsequent chunk.
- This approach retains context across the entire document and allows progressive improvement, but it is slower than map-reduce.

4. Stuff Document Chain

- A specialized LangChain chain for the "stuff" approach, where all the documents are concatenated with a prompt template before being passed to the LLM.

5. Map-Reduce Document Chain

- Implements the map-reduce approach, often used when dealing with large sets of documents or long transcripts.

6. Refine Document Chain

- Implements the refine approach, preserving an evolving context across chunks.

In practice:

- Use **Stuff** for short inputs (e.g., single-page PDFs).
- Use **Map-Reduce** when documents exceed the model's context window.
- Use **Refine** when the flow or narrative between chunks matters and you want the summary to integrate information progressively.

Stuff Document Chain Summarization

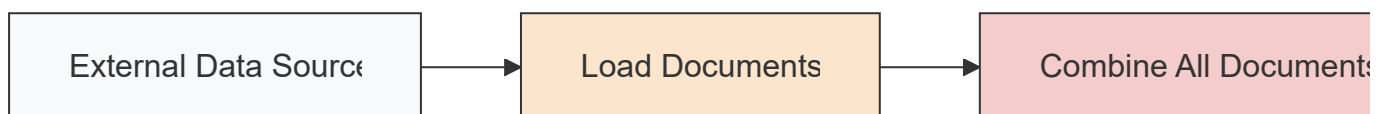
The **Stuff Document Chain** is one of the most basic summarization techniques in LangChain.

It operates by taking external data sources—such as a PDF file—and loading their contents into a **prompt template**. The entire content is then passed to a **Large Language Model (LLM)** to generate a comprehensive summary in a single step.

In this approach, if the source contains multiple documents (e.g., 10 separate PDFs), all their contents are concatenated and inserted directly into the prompt template. The LLM processes this combined text to produce a single summarized output.

Challenges:

The main limitation of the Stuff method is **context size**. Since LLMs have a maximum token limit, summarization becomes impractical when the combined size of the documents exceeds the model's context window. This can result in truncation or loss of important information.



Map-Reduce Summarization

The **Map-Reduce** summarization technique is one of the most widely used methods for handling large-scale text summarization tasks in LangChain.

In this approach, when the external data source contains multiple documents, the content is first **split into smaller chunks**. Each chunk is then passed individually into a **prompt**

template, and a **Large Language Model (LLM)** generates a **partial summary** for that chunk—this is the **Map step**.

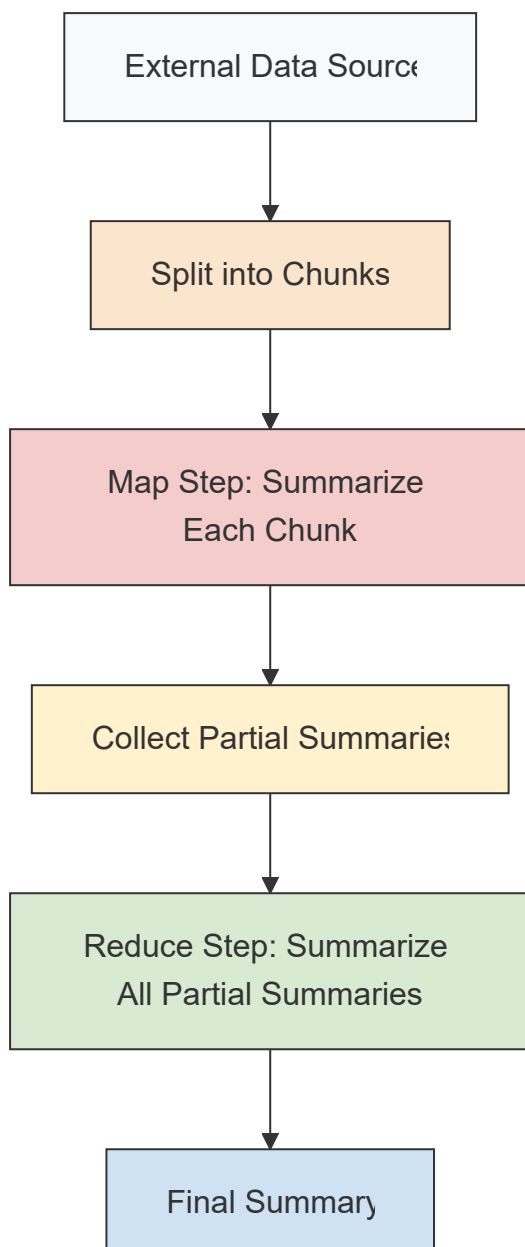
After all chunks are summarized, their partial summaries are **combined and summarized again** to produce the **final condensed summary**—this is the **Reduce step**.

Advantages:

- Capable of processing very large documents that exceed the LLM's context window.
- Easily parallelizable in the Map step for efficiency.

Challenges:

- Risk of losing **cross-chunk context** since chunks are summarized independently before the Reduce step.



Refine Chain Summarization

The **Refine Chain** is a summarization technique in LangChain that processes large documents by **iteratively building and improving a summary**.

Like the **Map-Reduce** method, the source document is first **split into smaller chunks**. Each chunk is passed to a **Large Language Model (LLM)** along with a **prompt template** to generate a **summary of that chunk**.

The difference lies in how subsequent chunks are processed:

- For the **first chunk**, the LLM produces an **initial summary**.
- For the **second and subsequent chunks**, the LLM is provided with **both the new chunk and the previously generated summary**.
- This allows the LLM to **refine and expand** the summary with each iteration, maintaining continuity and preserving context across all chunks.

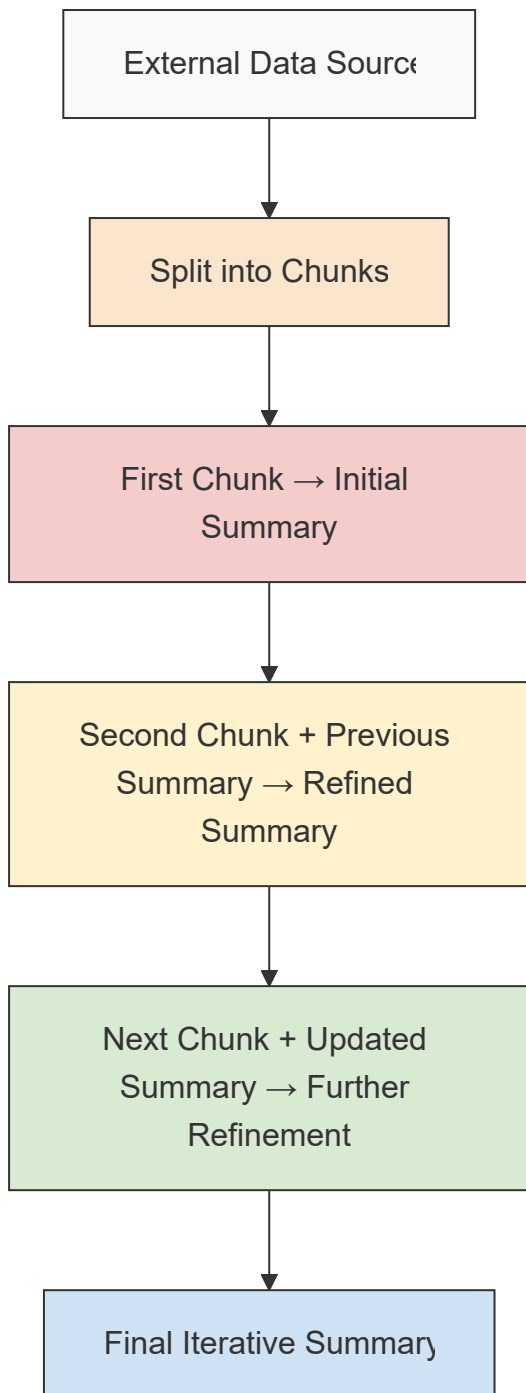
Advantages:

- Preserves cross-chunk context better than Map-Reduce.
- Produces a cohesive and progressively refined summary.

Challenges:

- Slower than Map-Reduce, as each chunk depends on the result of the previous step.
- Less parallelizable due to sequential processing.

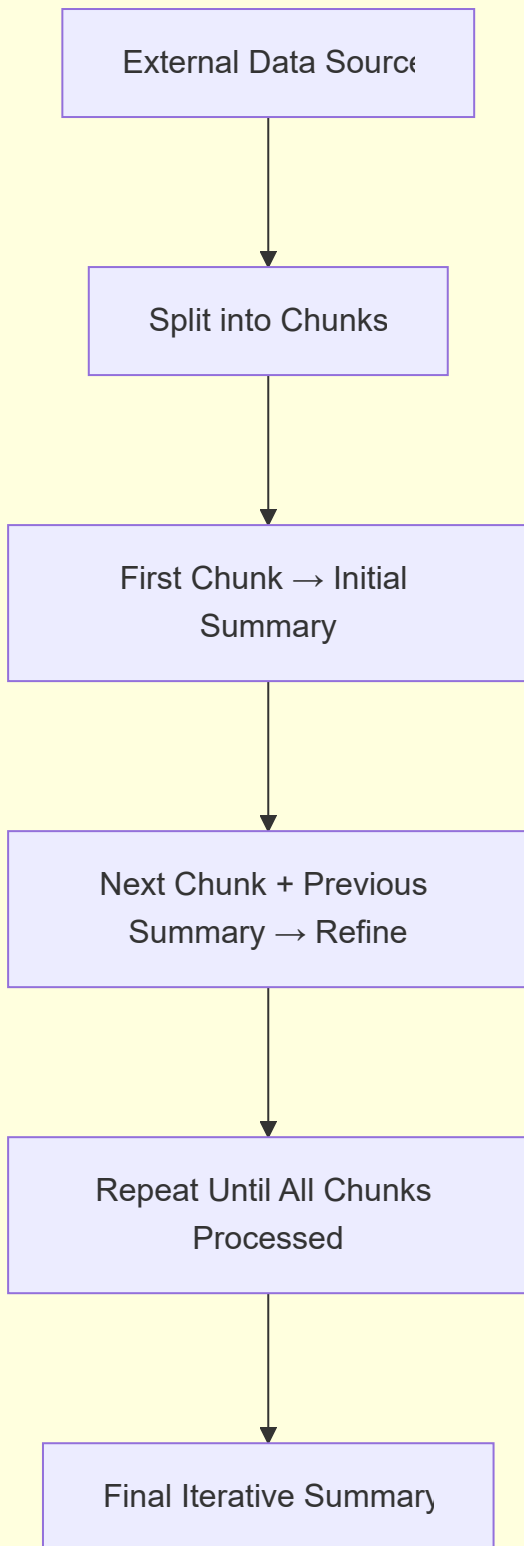
Mermaid Graph



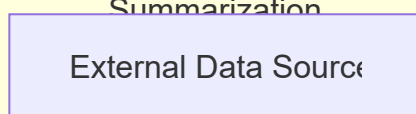
Comparison of LangChain Summarization Techniques

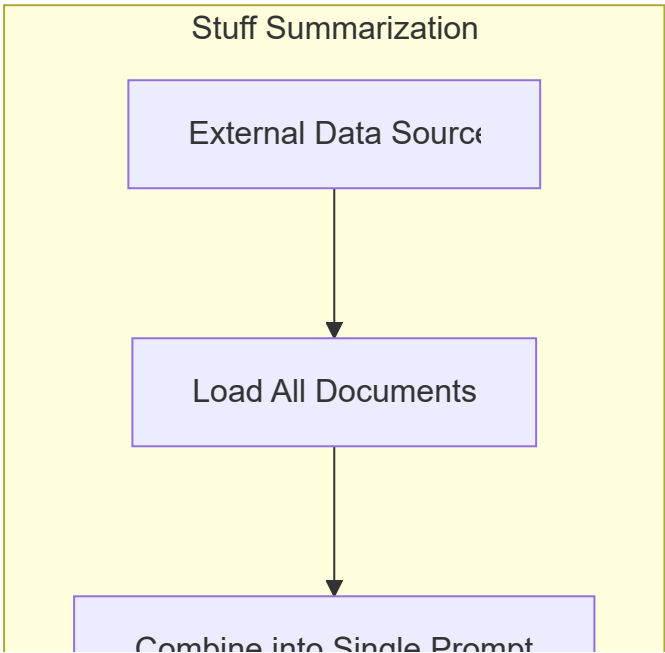
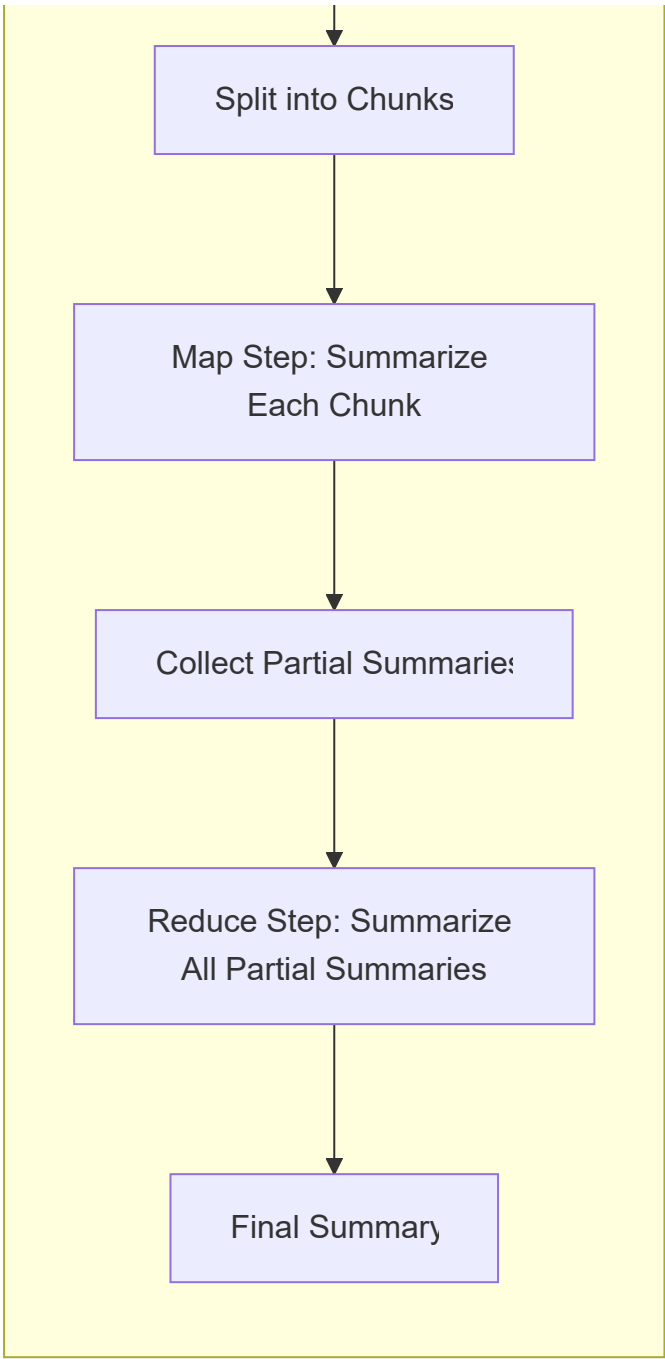
Mermaid Comparative Diagram

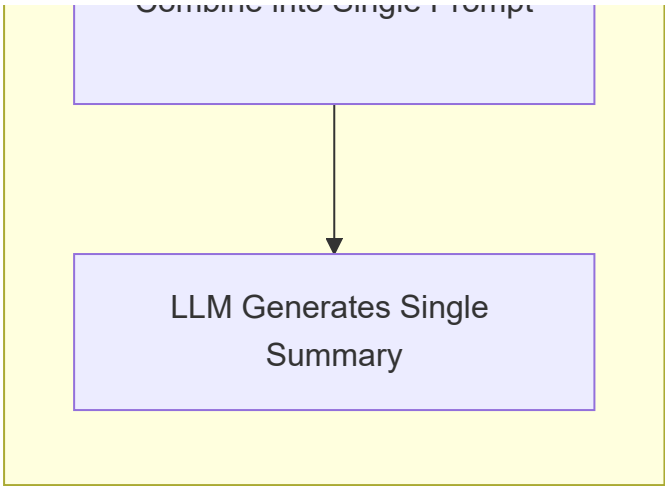
Refine Summarization



Map-Reduce Summarization







Comparison Table

Feature	Stuff	Map-Reduce	Refine
Process	Concatenate all documents into a single prompt and summarize once.	Summarize chunks independently, then combine summaries.	Summarize chunks sequentially, refining the summary with each step.
Strengths	Simple and fast for small inputs.	Handles large inputs; parallelizable.	Maintains narrative/context across chunks.
Weaknesses	Limited by context window; not scalable.	May lose cross-chunk context.	Slower; not parallelizable.
Best Use Case	Small, single-document summaries.	Large, unstructured datasets.	Long narratives or context-dependent texts.

Summary Structure

he best structure for a summary is the **Inverted Pyramid**, and yes, the approach differs slightly for large versus small content.

The Inverted Pyramid structure prioritizes information, presenting the most crucial elements first, followed by supporting details, and finally, background information. Think of it like a news article: you get the main takeaway immediately. 📝

Summarizing Small Content

For smaller pieces like an article, a short video, or a single chapter, a concise, single-paragraph summary is usually enough.

A great structure for this is:

1. **Main Idea:** Start with a single sentence that captures the central point or thesis of the original content.¹ Ask yourself: "What is the one thing the author wants me to know?"
2. **Key Supporting Points:** Add 2-3 sentences that outline the main evidence, arguments, or examples used to support that main idea.
3. **Conclusion/Implication:** End with a final sentence that explains the conclusion or significance of the information.²

Example (summarizing an article about bees):

A recent study reveals that pesticide use is the primary driver of declining bee populations, not habitat loss as previously thought. The research tracked several bee colonies and found that those exposed to common neonicotinoid pesticides showed a 70% reduction in queen production and foraging ability. These findings highlight the urgent need for stricter agricultural regulations to protect these essential pollinators.

Summarizing Large Content

For larger content like a book, a long report, or a research paper, you need a more detailed, multi-paragraph summary (often called an **executive summary** or **abstract**).

This structure expands on the simple model:

1. **Introduction (1 paragraph):** State the overall subject, the author's purpose or question, and the main thesis or conclusion. This is essentially the "small content" summary described above.
2. **Body Paragraphs (2-4 paragraphs):** Dedicate one paragraph to each major section, theme, or argument of the original work. Summarize the key points and evidence for each individual section. This gives the reader a sense of the original's structure and flow.
3. **Conclusion (1 paragraph):** Restate the overall thesis in a new way and briefly mention the original work's final conclusions, recommendations, or implications.

General Tips for Any Summary

No matter the length, a good summary always follows these rules:

- **Be Objective:** Don't include your own opinions or interpretations. Stick to the author's message.
- **Use Your Own Words:** Paraphrase the content to show you understand it. Avoid copying sentences directly.
- **Be Concise:** Eliminate filler words, repeated ideas, and minor details.³ A summary should be significantly shorter than the original.

- **Be Accurate:** Make sure your summary faithfully represents the original content's main ideas and context.
-

References

[Ollama](#)

[A Complete Guide to LLMs-based Autonomous Agents](#)

- [LangChain Docs: Multi-Vector Retriever](#)
- [LangChain Docs: Parent Document Retriever](#)
- LangChain Documentation: https://python.langchain.com/docs/get_started/introduction
- LangSmith Introduction: <https://docs.smith.langchain.com/>
- LangServe Documentation: <https://python.langchain.com/docs/langserve>

[How ChatGPT is Trained](#)

[Improving Language Understanding by Generative Pre-Training](#)

[the best blog jalammar](#)

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., & Polosukhin, I. (2017). *Attention Is All You Need*. In *Advances in Neural Information Processing Systems*, 30.

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). "Attention Is All You Need." *NeurIPS* 30. <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *NAACL*. <https://aclanthology.org/N19-1423>

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). *Layer Normalization*. arXiv:1607.06450. arxiv.org

Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. Proceedings of ICML. medium.com

"Build Better Deep Learning Models with Batch and Layer Normalization" (Pinecone, 2023). pinecone.io

- [ANN - Churn Classifier project](#)
- **1. Schmidt, R. M. (2019).** *Recurrent Neural Networks (RNNs): A gentle introduction and overview*. arXiv preprint arXiv:1912.05911. Available at: <https://arxiv.org/abs/1912.05911>

- **2. Sutskever, I., Vinyals, O., & Le, Q. V. (2014).** *Sequence to sequence learning with neural networks*. arXiv preprint arXiv:1409.3215. Available at: <https://arxiv.org/abs/1409.3215>
- **3. Zen, H., & Sak, H. (2015).** *Unidirectional long short-term memory recurrent neural network with recurrent output layer for low-latency speech synthesis*. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4470–4474). Available at: <https://research.google.com/pubs/archive/43266.pdf>
- Bengio, Y., Simard, P., & Frasconi, P. (1994). *Learning long-term dependencies with gradient descent is difficult*. *IEEE Transactions on Neural Networks*, 5(2), 157–166. <https://doi.org/10.1109/72.279181>
- Graves, A., Mohamed, A. R., & Hinton, G. (2013). *Speech recognition with deep recurrent neural networks*. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6645–6649. <https://doi.org/10.1109/ICASSP.2013.6638947>
- Hochreiter, S., & Schmidhuber, J. (1997). *Long short-term memory*. *Neural Computation*, 9(8), 1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). *Recurrent neural network based language model*. In *Interspeech*, 1045–1048.
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to sequence learning with neural networks*. In *Advances in Neural Information Processing Systems (NeurIPS)*, 3104–3112. https://papers.nips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf
- Elman, J. L. (1990). *Finding structure in time*. *Cognitive Science*, 14(2), 179–211. https://doi.org/10.1207/s15516709cog1402_1
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning internal representations by error propagation*. In *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1, pp. 318–362).
- Werbos, P. J. (1990). *Backpropagation through time: what it does and how to do it*. *Proceedings of the IEEE*, 78(10), 1550–1560. <https://doi.org/10.1109/5.58337>
- Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). [Learning Word Vectors for Sentiment Analysis](#). *The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*.
- **Chen, G. (2016).** *A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation*. arXiv preprint [arXiv:1610.02583](https://arxiv.org/abs/1610.02583)
- Cho et al. (2014) *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation*
- [Christopher Olah's "Understanding LSTM"](#)
- **Kyunghyun Cho (2014).** [Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation](#)

- Kim, Z., Oh, H., Kim, HG. *et al.* Modeling long-term human activeness using recurrent neural networks for biometric data. *BMC Med Inform Decis Mak* **17** (Suppl 1), 57 (2017). <https://doi.org/10.1186/s12911-017-0453-1>
- M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," in *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673-2681, Nov. 1997, doi: 10.1109/78.650093.
- Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). **BLEU: a Method for Automatic Evaluation of Machine Translation**. *Proceedings of the 40th Annual Meeting of the ACL*, 311-318.
- Chen, B., & Cherry, C. (2014). **A Systematic Comparison of Smoothing Techniques for Sentence-Level BLEU**. *Proceedings of the Ninth Workshop on Statistical Machine Translation (WMT)*, 362-367.
- Post, M. (2018). **A Call for Clarity in Reporting BLEU Scores**. *Proceedings of the Third Conference on Machine Translation (WMT)*, 186-191
- Bahdanau, D., Cho, K., & Bengio, Y. (2015). **Neural Machine Translation by Jointly Learning to Align and Translate**. *ICLR*.
- Luong, M., Pham, H., & Manning, C. D. (2015). **Effective Approaches to Attention-based Neural Machine Translation**. *EMNLP*.
- Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). **BLEU: a Method for Automatic Evaluation of Machine Translation**. *ACL*.

```
@article{article,
author = {Zhao, Wei and He, Ting and Xu, Li},
year = {2021},
month = {03},
pages = {1-1},
title = {Enhancing Local Dependencies for Transformer-Based Text-to-Speech via Hybrid
Lightweight Convolution},
volume = {PP},
journal = {IEEE Access},
doi = {10.1109/ACCESS.2021.3065736}
}
```