Machine Learning Techniques for Text

Module 7: Summarizing Wikipedia Articles

Dr. Nikos Tsourakis

Course outline

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- Module 0: Python Crash Course
- Module 1: Intro to Machine Learning
- Module 2: Detecting Spam Emails
- Module 3: Classifying Topics of Newsgroup Posts
- Module 4: Extracting Sentiments from Product Reviews
- Module 5: Recommending Music Titles
- Module 6: Teaching Machines to Translate
- **Module 7: Summarizing Wikipedia Articles**
- Module 8: Detecting Hateful and Offensive Language
- Module 9: Generating Text in Chatbots
- Module 10: Clustering Speech-to-Text Transcriptions

Overview

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- As more content is available online, the less easy it is to discover and consume the most important information efficiently
- Automatically extracting the gist of longer texts into an accurate summary and thus eliminating irrelevant content is urgently needed
- This module introduces another challenging topic in natural language processing (NLP) and demystifies methods for ***text summarization***
  - To implement pertinent systems, we exploit data coming from the web
  - Besides the standard text summarization methods, we delve into a state-of-the-art architecture that provides exceptional performance in many real-world applications
  - The specific topology extends the ***sequence-to-sequence*** (seq2seq) architectures we have already discussed and combines many concepts encountered throughout the seminar
  - Finally, we discuss the metrics to assess the performance of relevant systems

Module objectives

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

After completing this module, you should be able to:

- Discussing different techniques for text summarization
- Applying web crawling and data scraping
- Understanding related web technologies
- Implementing state-of-the-art architectures for text summarization
- Evaluating relevant systems using the appropriate metrics

Machine Learning Techniques for Text

Section 1: Understanding text summarization

Introduction

- People have an inherent tendency to reduce their information load by attending to specific parts of their visual stimuli
- It is, therefore, not a surprise that many stakeholders seize the opportunity to offer products that alleviate this load
- The current module focuses on ***text summarization***, which is the process of condensing a piece of text into a shorter version while preserving critical information and the overall meaning of the original
- Similar to machine translation, the task is not a simple text-to-text transformation because context is essential to fluently pass the intended message in the output summary

Introduction

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- In school, we were often asked to summarize large documents and demonstrate our capacity to understand and extract the most valuable information from the text
- Some students would highlight specific parts of the original document and include verbatim reproductions of these fragments in the output summary
- Others aimed to comprehend the content more deeply and identify the most crucial information. Then, they had to formulate sentences from scratch that conveyed the original meaning
- Based on this analogy, there are two types of text summarization: ***extractive*** and ***abstractive***

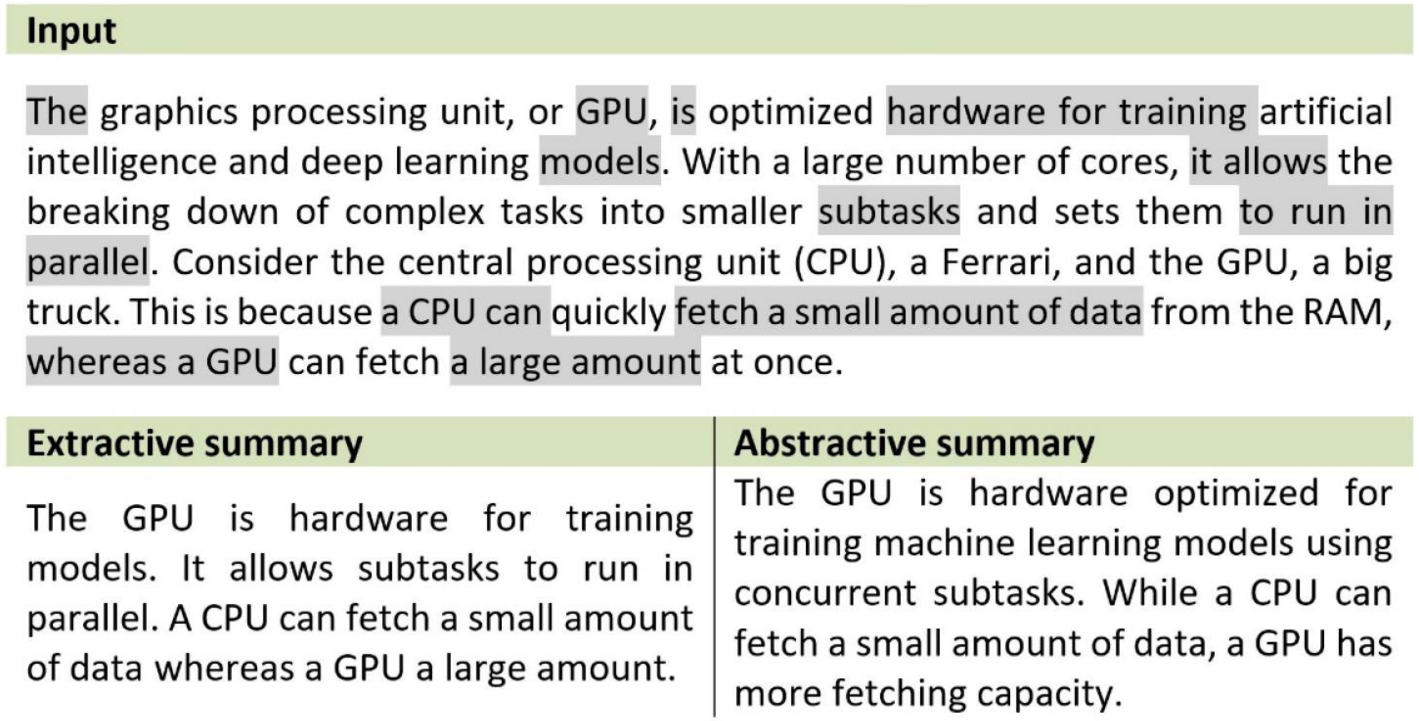
Introduction

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- The highlighted text in the input is used for the ***extractive summary***
  - … but it’s unclear how we chose it in the first place
- The formulation in the ***abstractive summary*** is even more obscure



Input

The graphics processing unit, or GPU, is optimized hardware for training artificial intelligence and deep learning models. With a large number of cores, it allows the breaking down of complex tasks into smaller subtasks and sets them to run in parallel. Consider the central processing unit (CPU), a Ferrari, and the GPU, a big truck. This is because a CPU can quickly fetch a small amount of data from the RAM, whereas a GPU can fetch a large amount at once.

Extractive summary	**Abstractive summary**
The GPU is hardware for training models. It allows subtasks to run in parallel. A CPU can fetch a small amount of data whereas a GPU a large amount.	The GPU is hardware optimized for training machine learning models using concurrent subtasks. While a CPU can fetch a small amount of data, a GPU has more fetching capacity.

Machine Learning Techniques for Text

Section 2: Understanding web scraping

Introduction

‹packt›

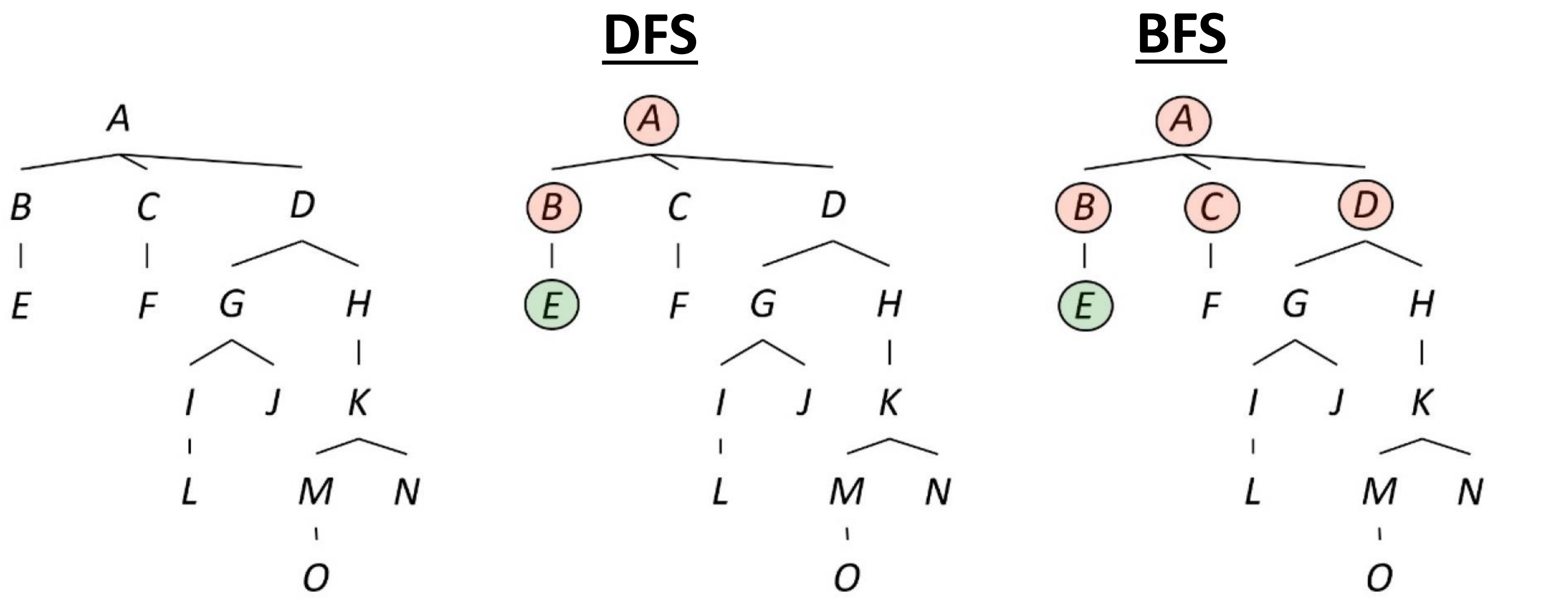
Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- Suitable corpora are rarely available for free, and it's the data scientist's primary responsibility to harvest them
- The ***world wide web*** (WWW) is a goldmine where we can resort to finding or augmenting our datasets using ***web scraping***, the process of collecting and parsing raw data from the web
- For this task to succeed, ***web crawlers*** are used to retrieve the requested content
- These are also known as ***spiders*** because they crawl all over the web, just as real spiders crawl on their spiderwebs

The crawling process

- Implementing a web crawler typically dictates choosing a traversal strategy, and the two most prominent options are the ***depth-first*** (DFS) and ***breadth-first*** (BFS) algorithms



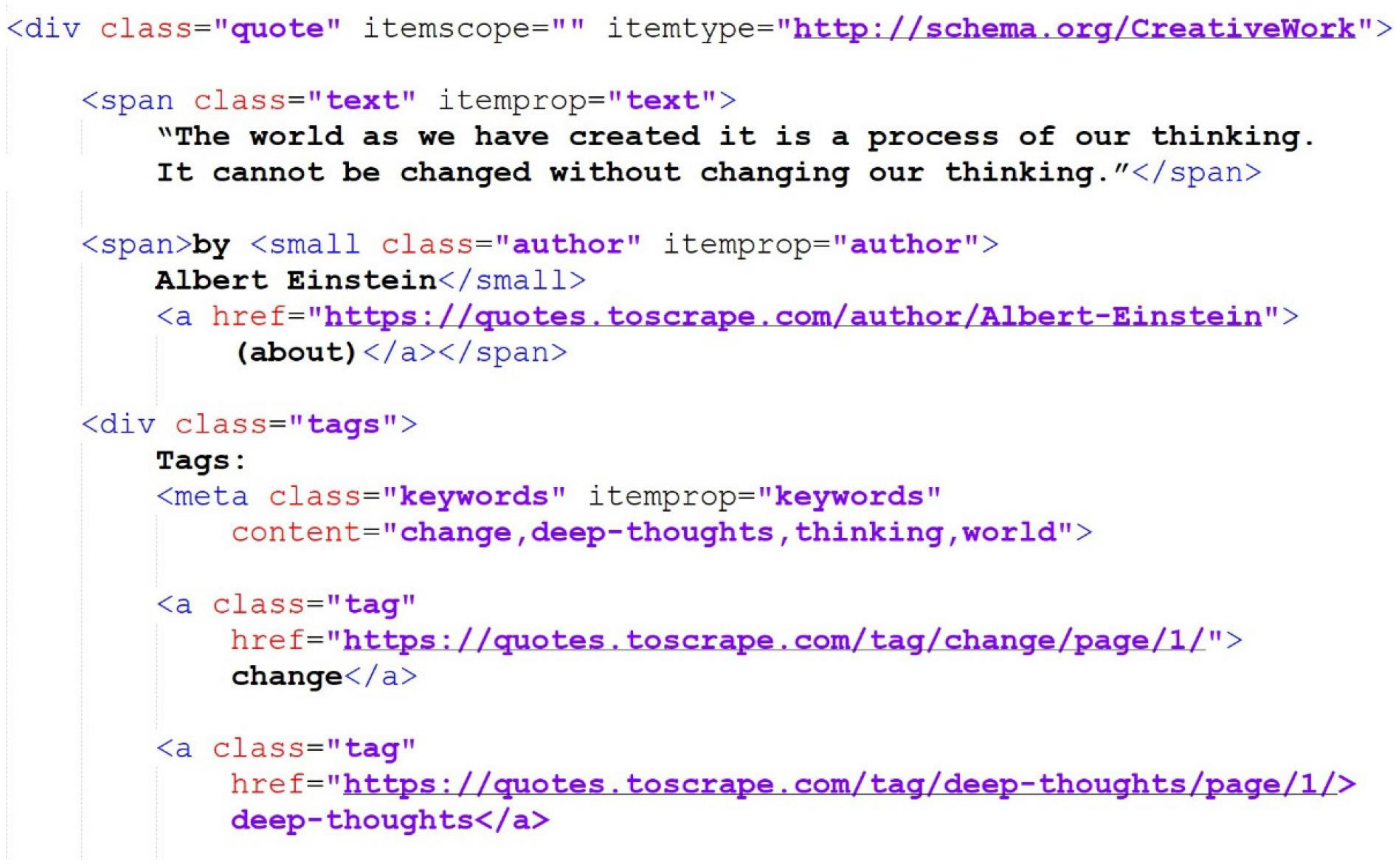
ABECFDGILJHKMON **ABCDEFGHIJKLMNO**

The crawling process

- The web scraping task is far from easy, as websites come in different shapes and forms and it’s in the self interest of the site to block the scraping process to avoid overwhelming the server with requests
- Additionally, the ***completely automated public turing test to tell computers and humans apart*** (CAPTCHA), despite being annoying for end users, helps verify that the request came from an actual human, not an internet bot CAPTCHAs are one of the most popular anti-scraping techniques available
- As a way to assist the scraping process, many websites include a file called ***robots.txt*** that explicitly dictates which parts of the site cannot be crawled

HTML code

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation



```html
<div class="quote" itemscope="" itemtype="http://schema.org/CreativeWork">

    <span class="text" itemprop="text">
        “The world as we have created it is a process of our thinking.
        It cannot be changed without changing our thinking.”</span>

    <span>by <small class="author" itemprop="author">
        Albert Einstein</small>
        <a href="https://quotes.toscrape.com/author/Albert-Einstein">
            (about)</a></span>

    <div class="tags">
        Tags:
        <meta class="keywords" itemprop="keywords"
            content="change,deep-thoughts,thinking,world">

        <a class="tag"
            href="https://quotes.toscrape.com/tag/change/page/1/">
            change</a>

        <a class="tag"
            href="https://quotes.toscrape.com/tag/deep-thoughts/page/1/>
            deep-thoughts</a>
```

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Let’s practice!

Tasks

- Web scraping

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-07/quote-scraper.ipynb

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Let’s practice!

Tasks

- Web scraping

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-07/wikipedia-scraper.ipynb

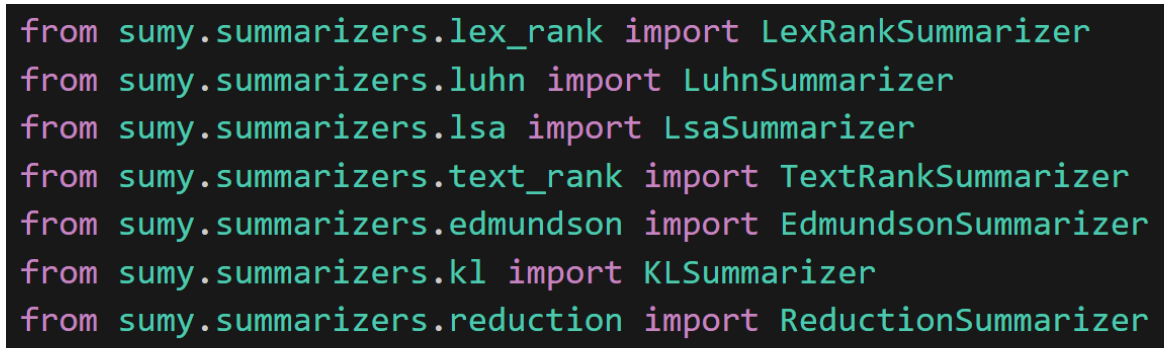
Machine Learning Techniques for Text

Section 3: Performing extractive summarization

Introduction

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- ***Extractive summarization*** identifies important words or phrases and stitches them together to produce a condensed version of the original text
- Use ***Sumy*** - module for automatic summarization of text documents and HTML pages



```
from sumy.summarizers.lex_rank import LexRankSummarizer
from sumy.summarizers.luhn import LuhnSummarizer
from sumy.summarizers.lsa import LsaSummarizer
from sumy.summarizers.text_rank import TextRankSummarizer
from sumy.summarizers.edmundson import EdmundsonSummarizer
from sumy.summarizers.kl import KLSummarizer
from sumy.summarizers.reduction import ReductionSummarizer
```

- A good starting point is the following link:
  - https://miso-belica.github.io/sumy/summarizators.html

Let’s practice!

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Tasks

- Web scraping
- Extractive summarization

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-07/text-summarization.ipynb

Machine Learning Techniques for Text

Section 3: Performing abstractive summarization

<packt>

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Introduction

- ***Abstractive summarization*** generates novel sentences by rephrasing the reference and introducing new text
- This task is quite challenging, and for this reason, more sophisticated methods are required
- This section adopts a step-by-step approach to present pertinent concepts and techniques
- Ultimately, we glue all the pieces together in a state-of-the-art model for abstractive summarization

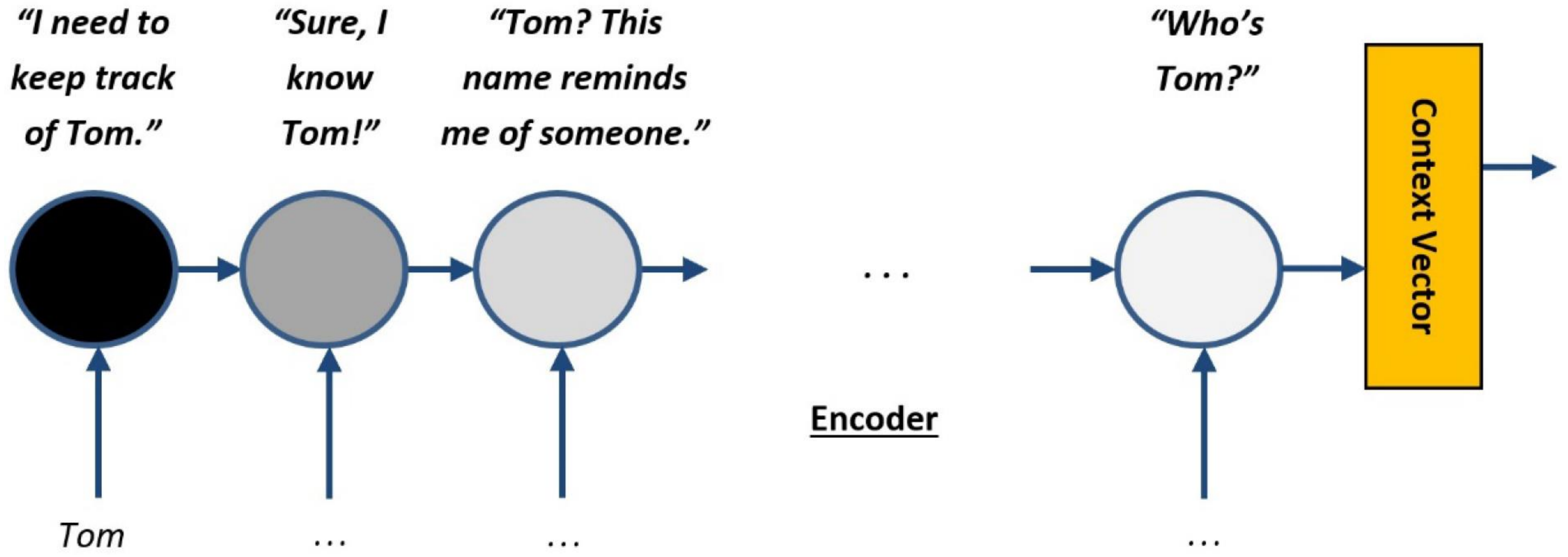
Attention mechanism

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- The ***encoder-decoder seq2seq*** architecture suitable for translating sentences from a source language to a target one
- The complete input is encoded in a ***context vector*** used by the ***decoder*** to produce a translation
- In actual human communications, we tend to listen to the whole sentence before responding
- The context vector represents this process; it crams the whole input into a single vector
- But as humans tend to forget important information, so can seq2seq models
- ***Long Short-Term Memory*** (LSTM) units and ***Gated Recurrent Units*** (GRUs), tend to remember information for longer periods

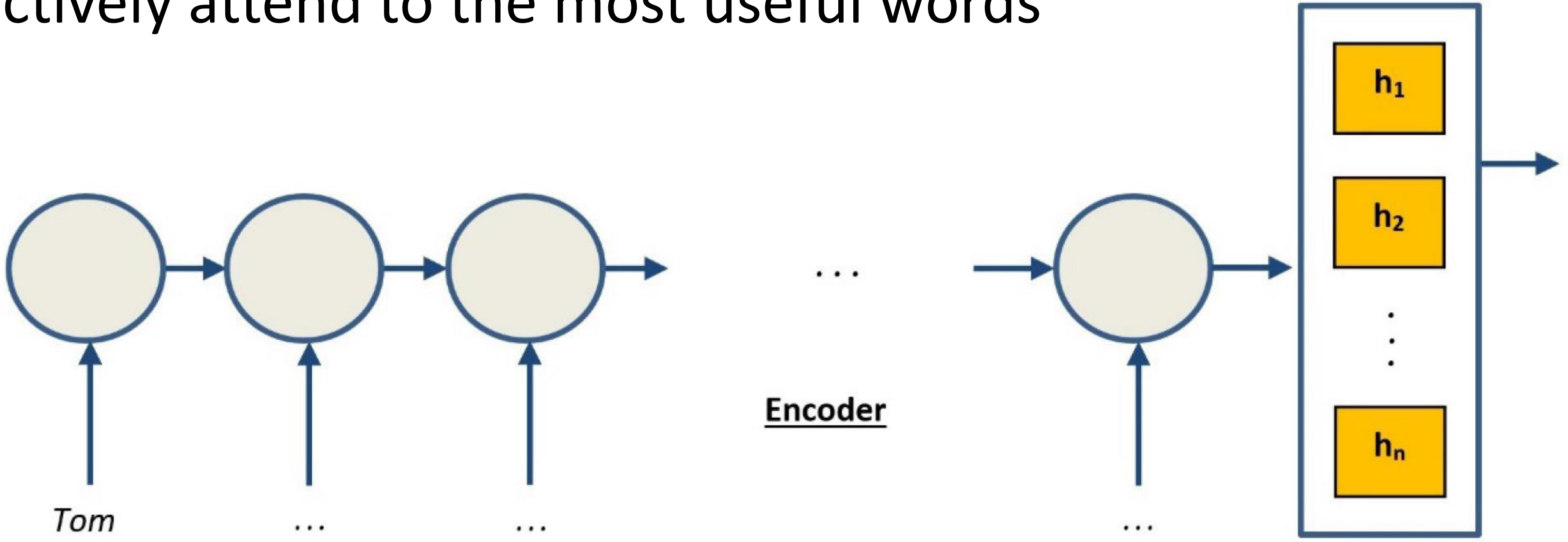
Attention mechanism

- However, when the input sequences become sufficiently large, even these networks cease to include all important information in their context vector
- The words' influence at the beginning of the input becomes smaller and smaller after the consecutive updates in the intermediate hidden states of the encoder



Attention mechanism

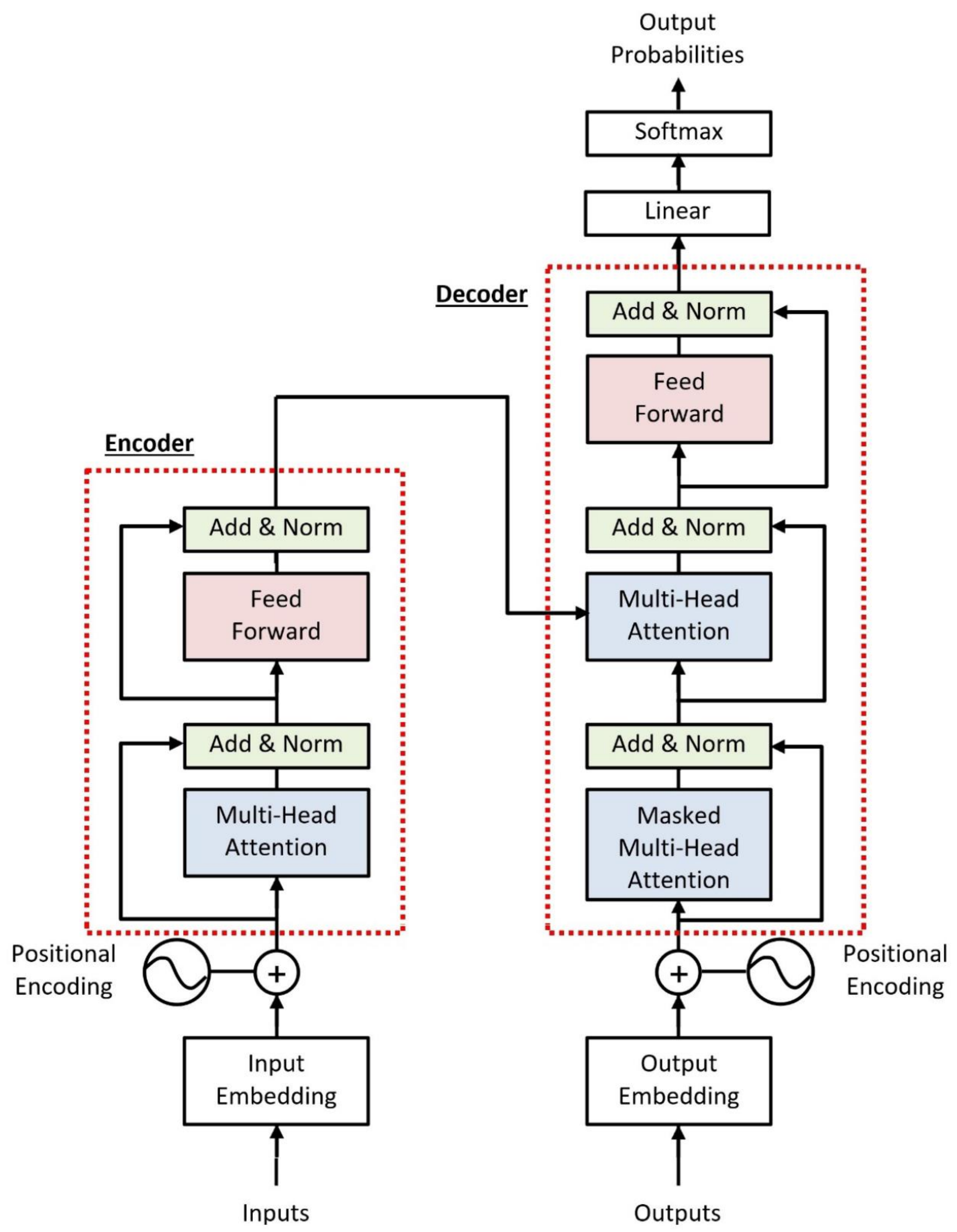
- A better approach is to use all hidden states and weigh individual words in the input sequence according to their impact on the target
- This approach is the basic idea behind the ***attention mechanism***
- The encoder passes all hidden states to the decoder which can selectively attend to the most useful words

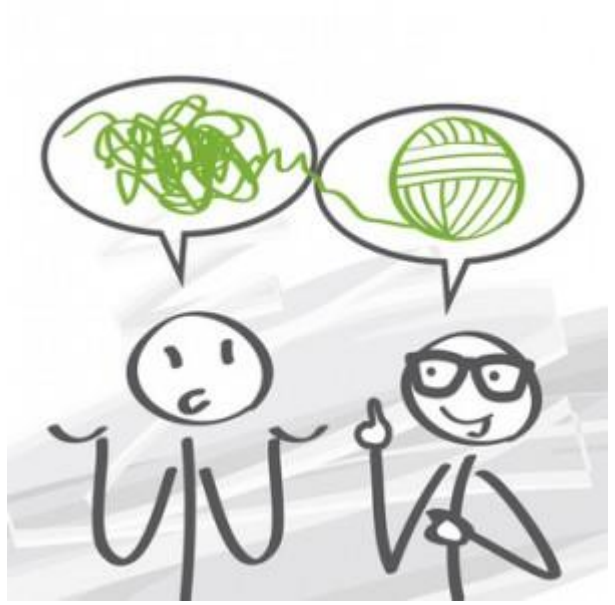


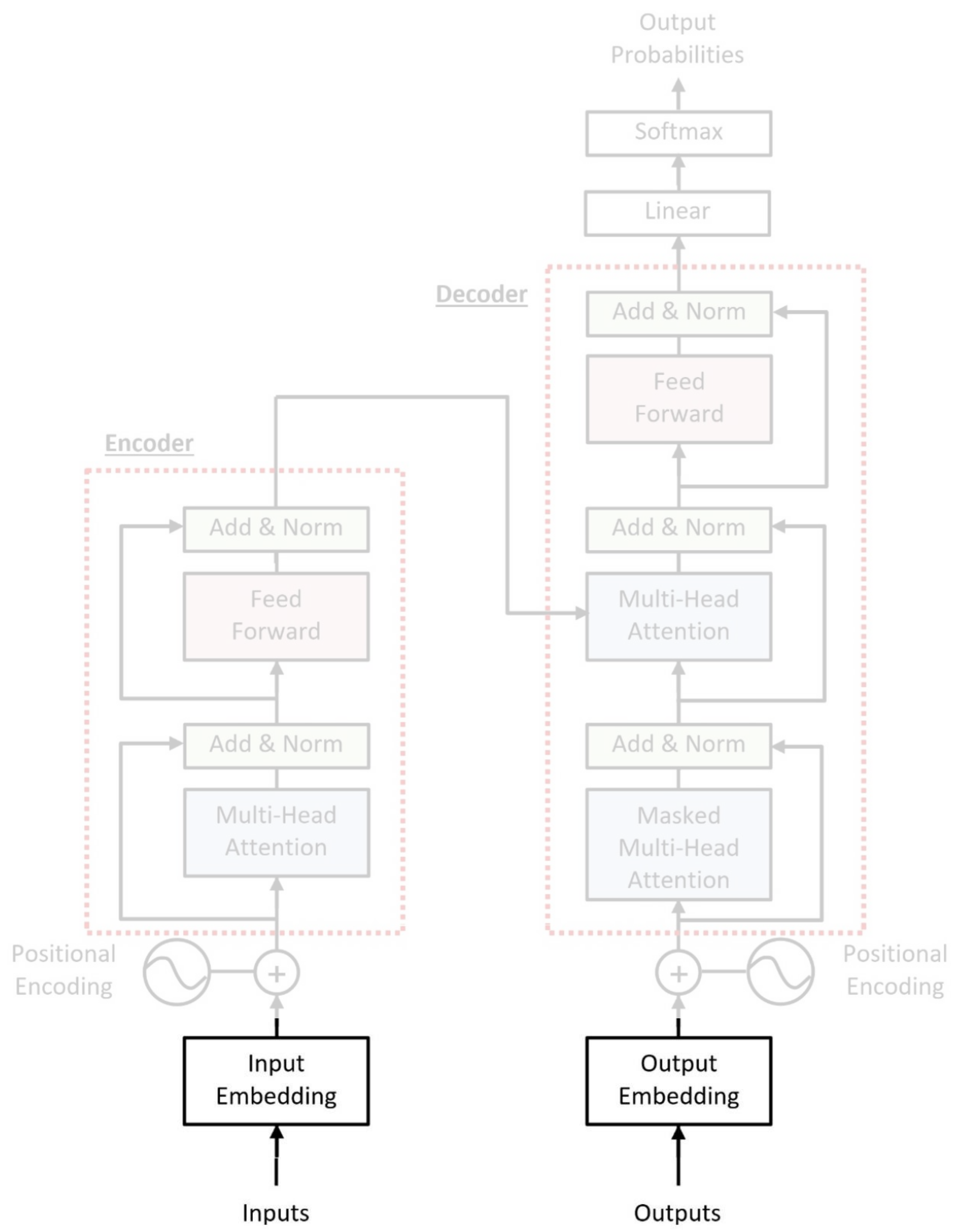
The transformer model

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- Imagine a team of editors that each specializes in different aspects of a manuscript, collectively working to condense lengthy texts into concise summaries
- The editors focus on various positions in input sequences, paying attention to specific details and collaborate to ensure the final summary captures the essence of the original text
- The ***transformer model*** is inspired by the collaborative dynamics of the team of editors, employing parallel processing and selective attention mechanisms to maintain the essence of the original content
- The transformer model replaces the recurrent layers most commonly used in encoder-decoder architectures with multi-headed self-attention to boost the performance of ***deep neural networks*** (DNNs)


Output Probabilities
Softmax
Linear
Decoder
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Encoder
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs



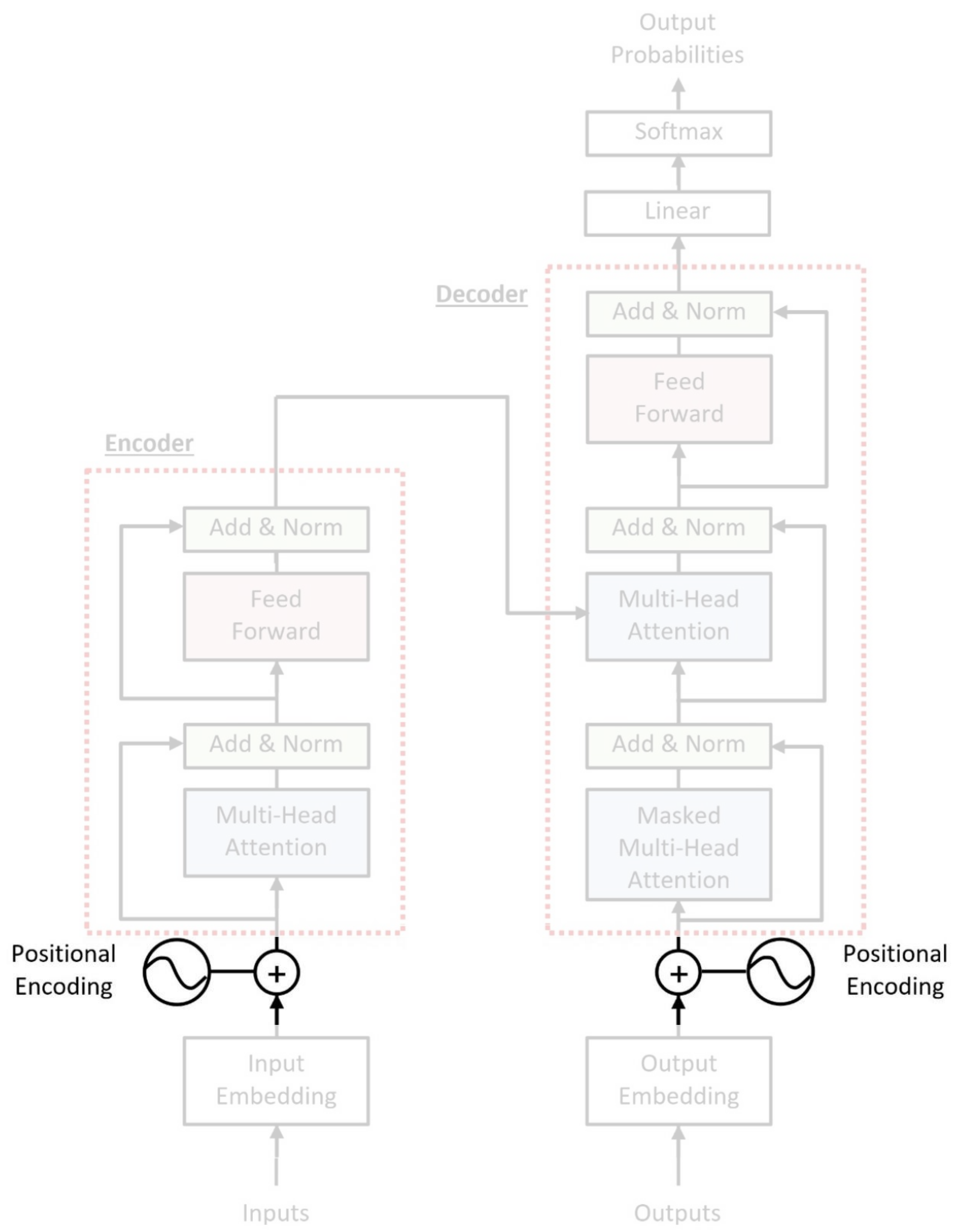


<u>Problem</u>

Represent the input/output sentences

<u>Solution</u>

Use word embedding



Problem

Identical words placed in different positions can change the sentence's meaning entirely

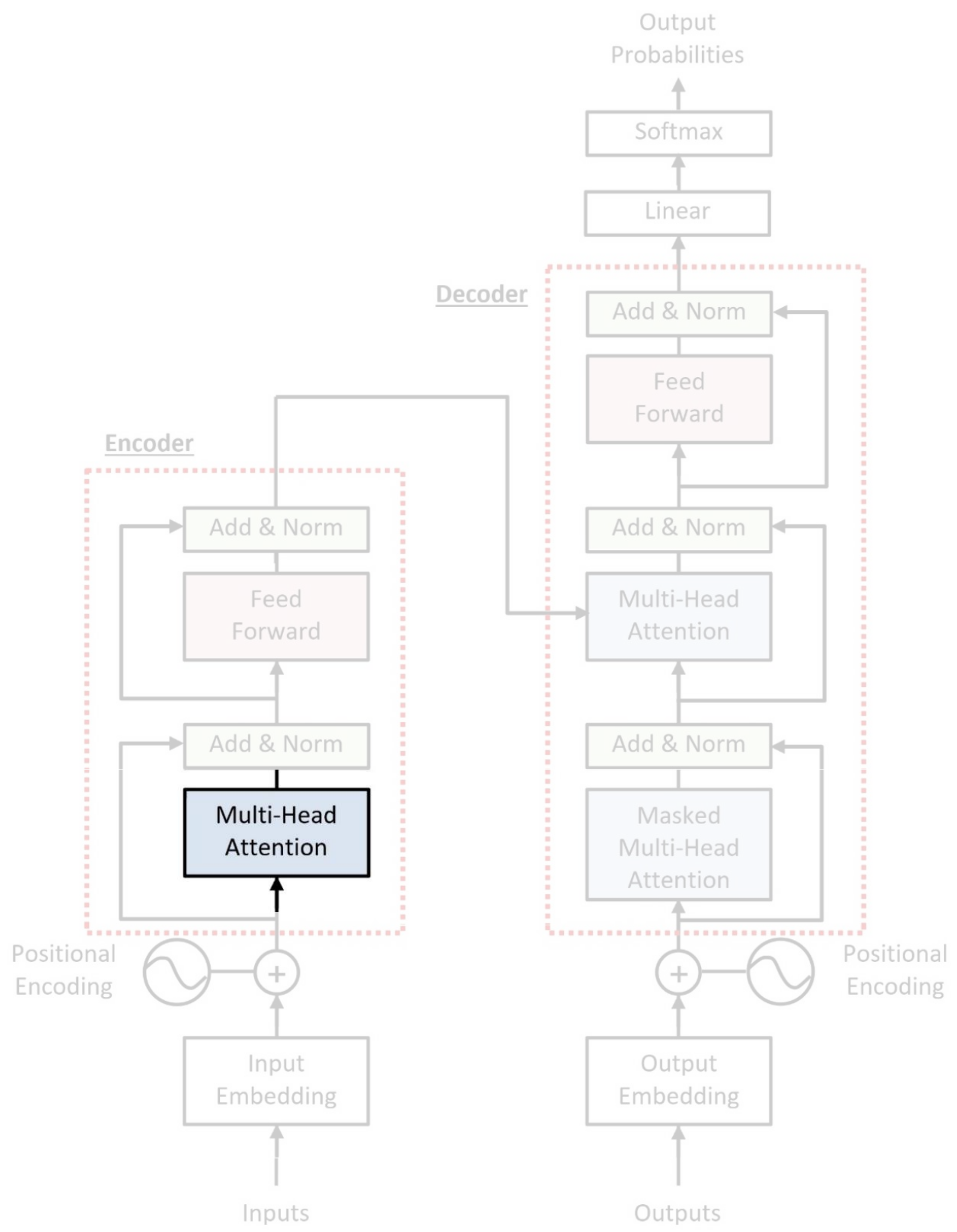
For example:

Paul bit a dog versus ***A dog bit Paul***

Solution

Use ***positional encoding*** that adds the necessary spatial information about each word in a vector of the same size as the embeddings one

After calculating the positional encoding vector, we add it to the embedding vector, which now includes an injected pattern with spatial information for the words

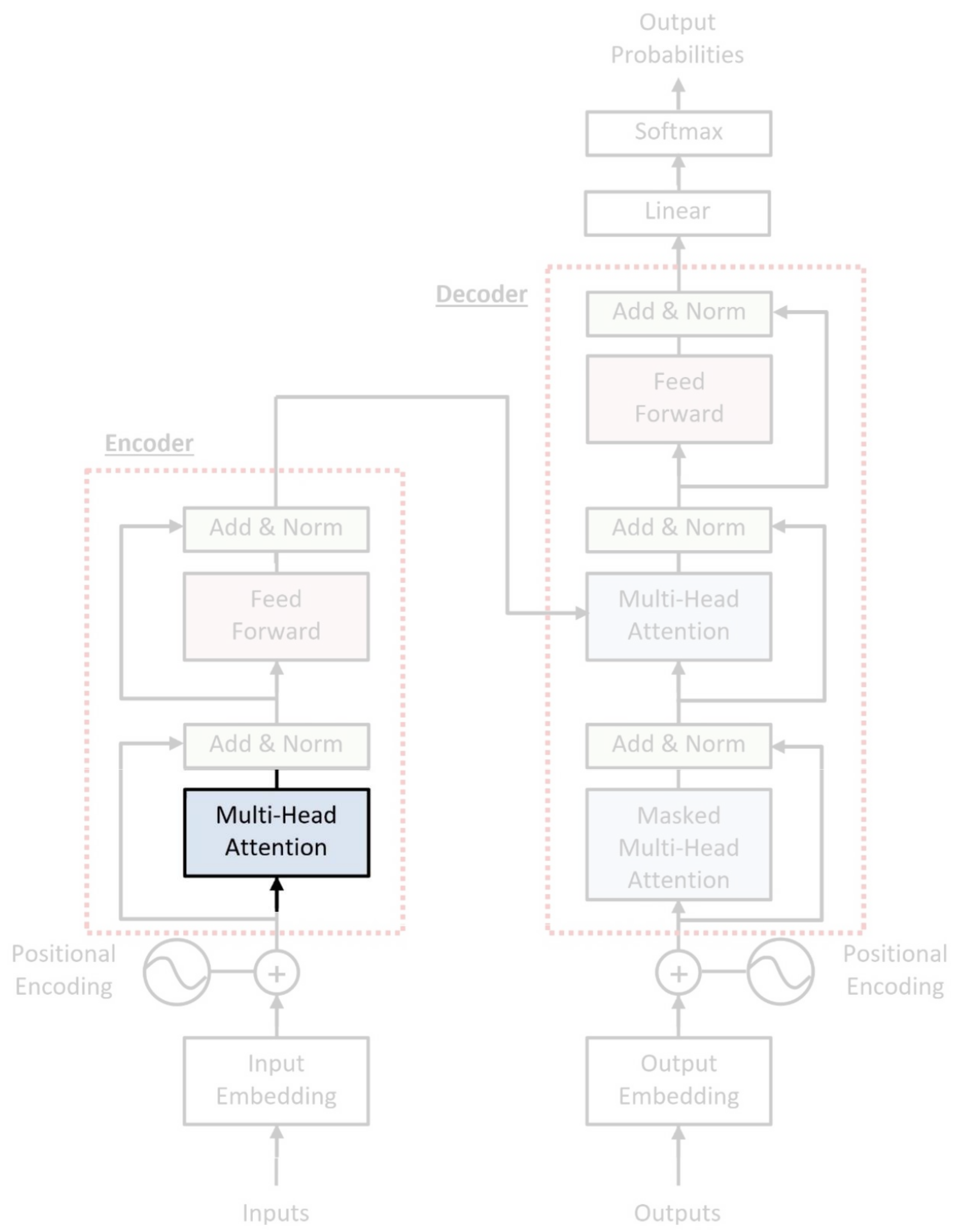


Problem

Having a rich representation of the input at our disposal, we need a component that can attend to the most critical information

Solution

Use ***self-attention*** that allows the inputs to interact with each other and the model learns where to pay attention



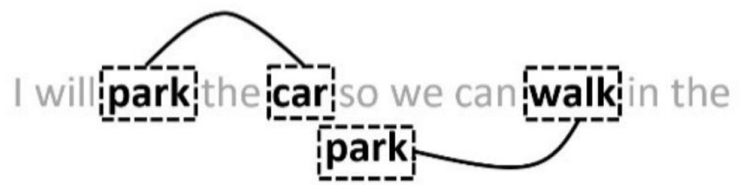
Simple attention

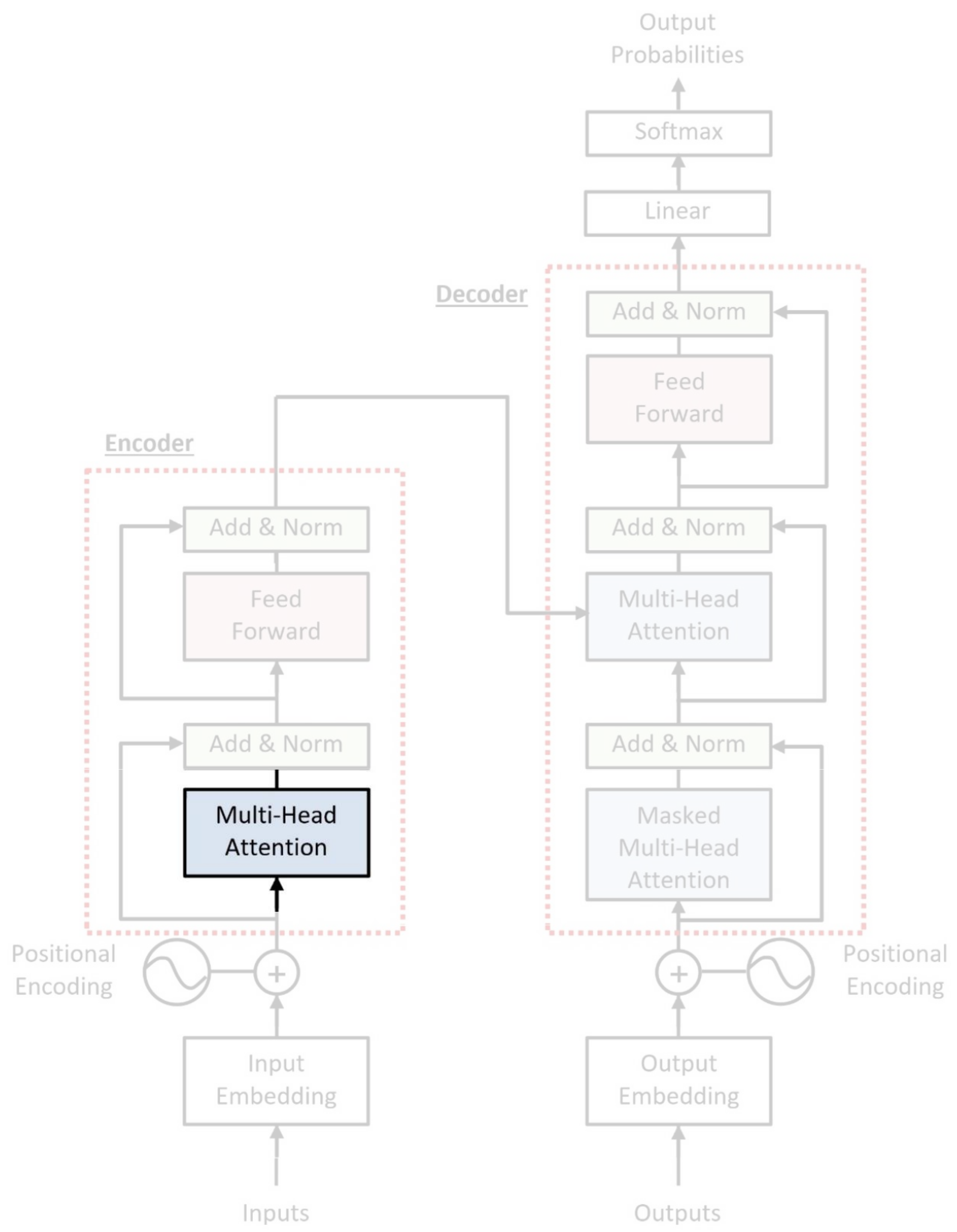
Who framed Roger Rabbit?

Judge Doom is a fictional character who appears as the main **antagonist** in the 1988 film Who Framed **Roger Rabbit**, portrayed by Christopher Lloyd.

Self-attention

We can *park* a car and we can walk in a *park*.



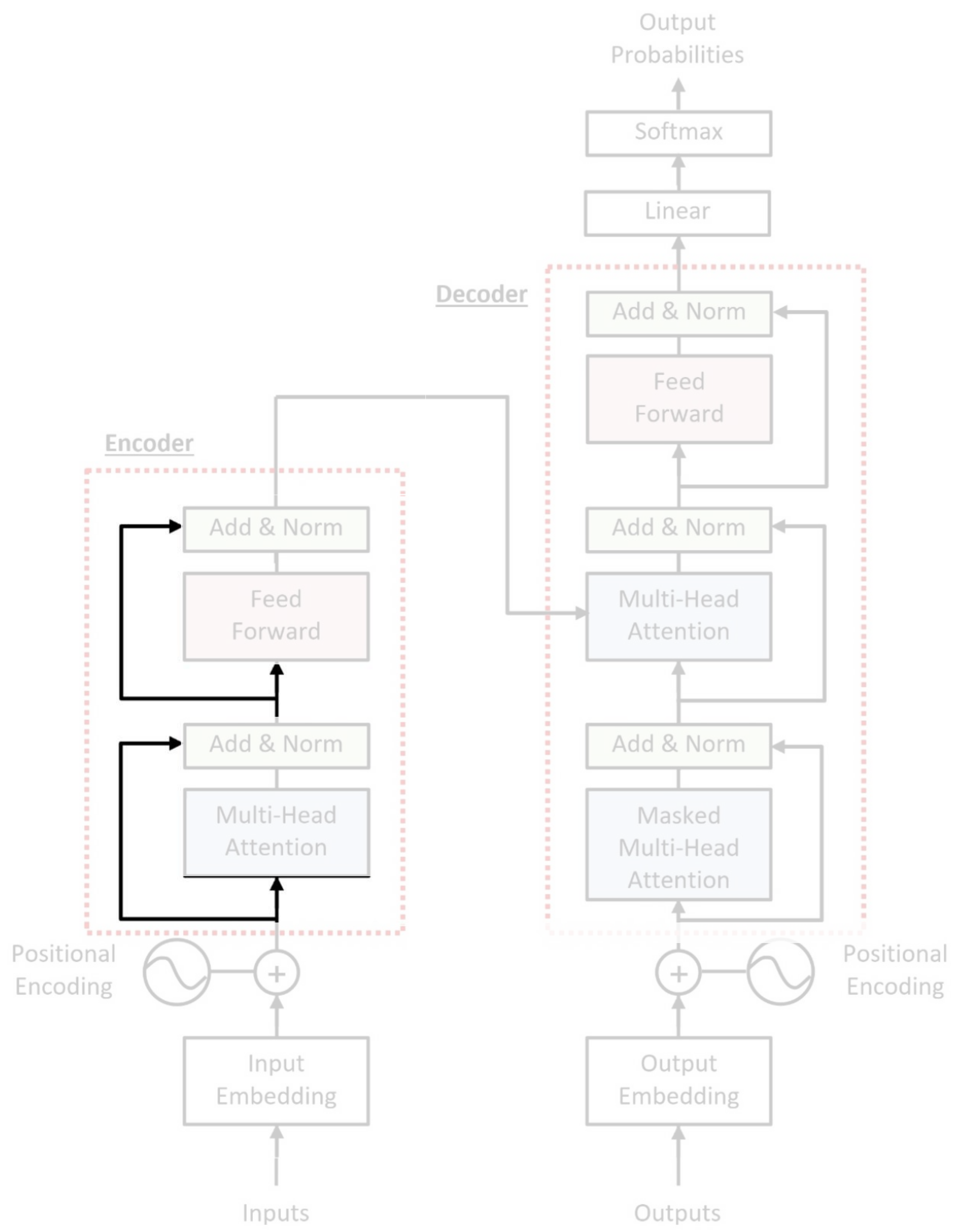


Problem

Applying a single attention filter is not enough to attend to all critical information

Solution

Use ***multi-head attention*** by stacking more than one attention component together to focus on different parts of the input

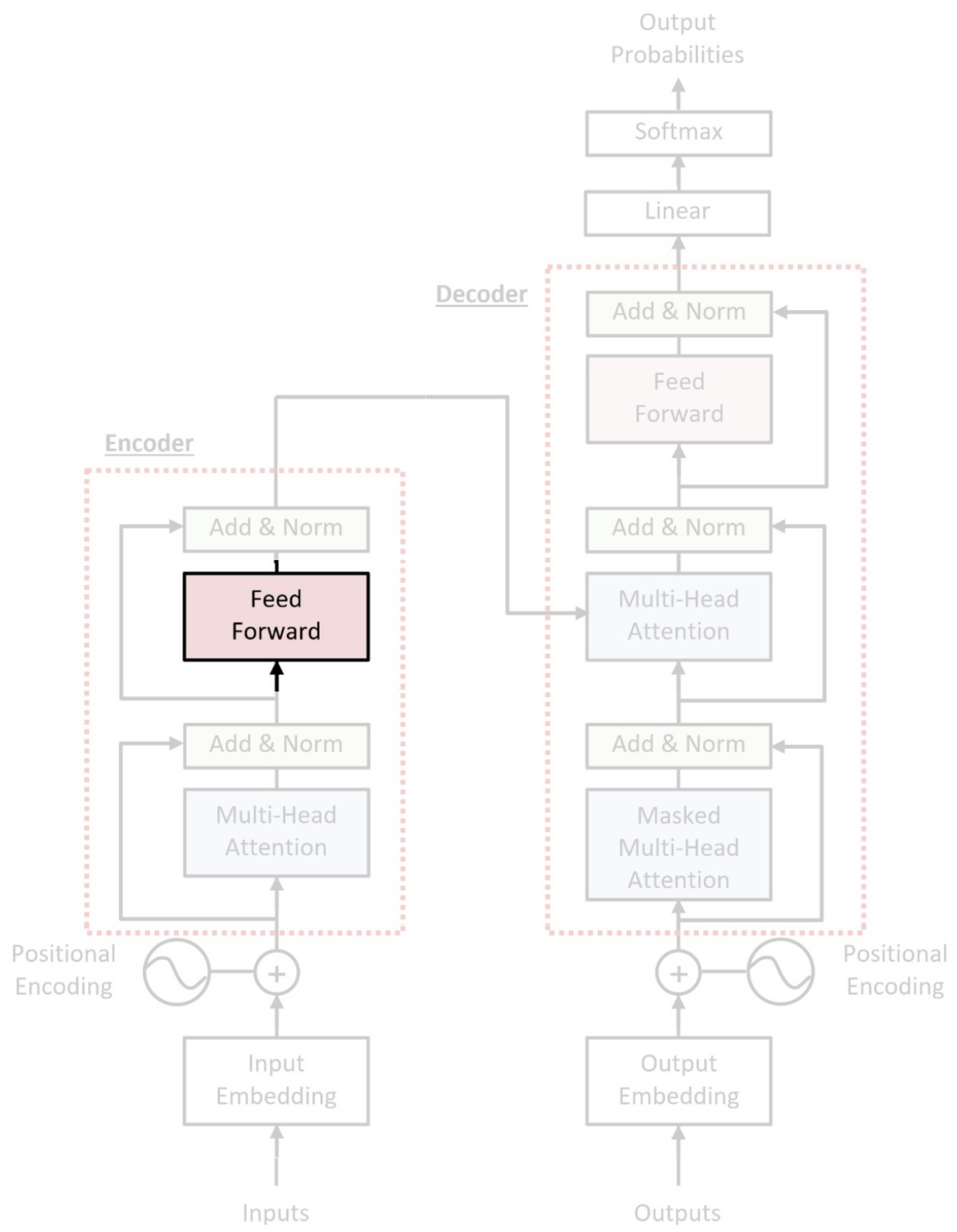


Problem

The more processing that happens to the data, the more information is lost about its original form. Preserving possible previous versions of the data becomes a necessity

Solution

Use ***residual connections*** that provides alternative paths for data to reach the latter parts of the network by bypassing some layers

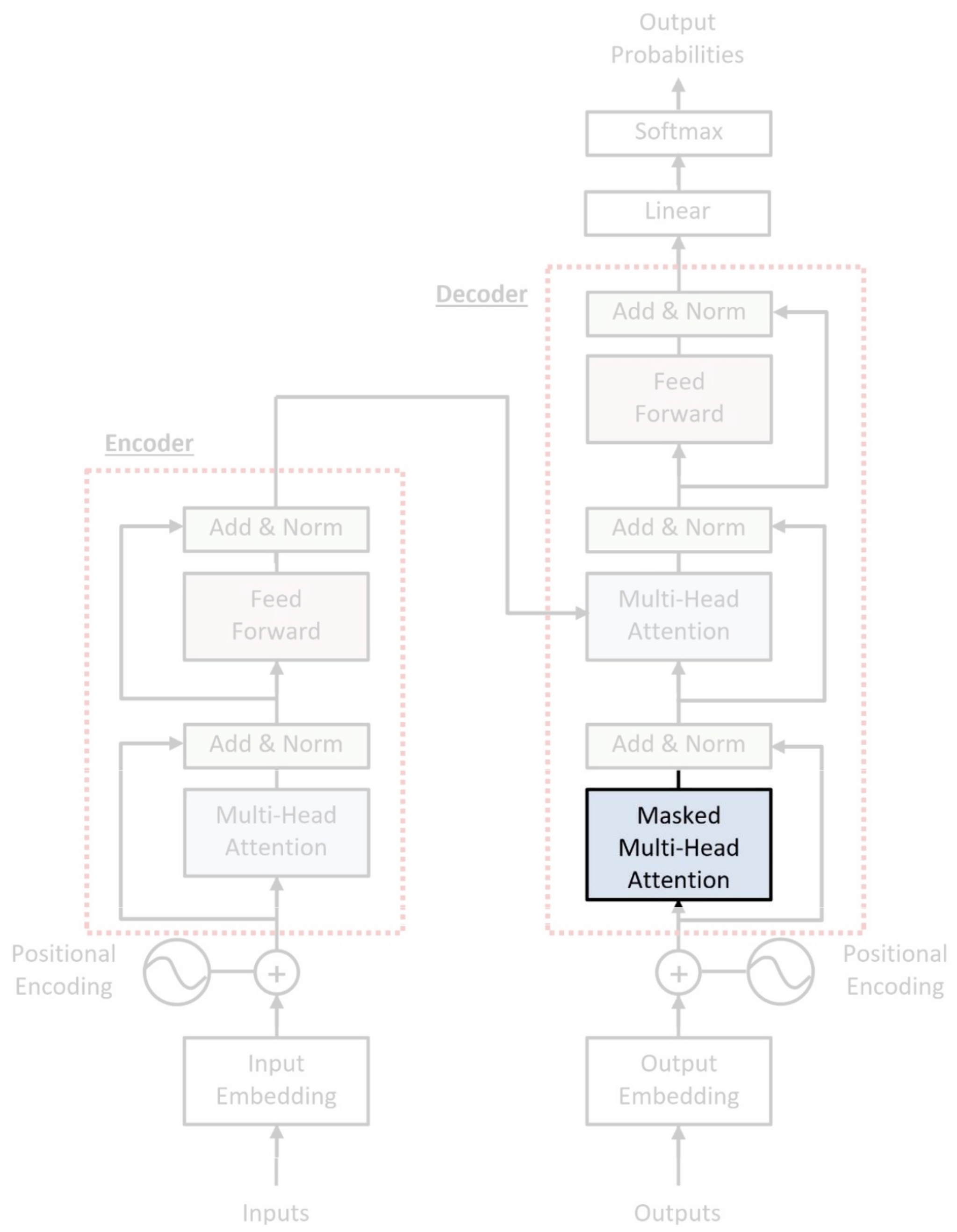


Problem

Capture more linguistic patterns, and provide richer representations

Solution

Use a fully connected ***feed forward network*** to process the attention output further



Problem

During the training phase, we do not want the decoder's attention mechanism to access tokens (words) that are not yet predicted

Solution

To prevent computing attention scores for future words use ***masking***

Let’s practice!

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Tasks

- Web scraping
- Extractive summarization
- Abstractive summarization

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-07/text-summarization-transformer.ipynb

Machine Learning Techniques for Text

Section 4: Measuring summarization performance

The ROUGE score

‹packt›
Machine Learning Techniques for Text
Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

- The ***Recall-Oriented Understudy for Gisting Evaluation*** (ROUGE) score assesses the performance of text summarization systems
- It works by comparing an automatically produced summary against a human reference summary using ***n-grams***
- In that sense, it is symmetrical to the ***BLEU score***
- Additionally, ROUGE is a set of metrics rather than a single one
- They all assign a numerical score to a summary that tells us how good it is compared to a reference
- ***ROUGE-N*** measures the overlap of unigrams, bigrams, trigrams, and higher-order n-grams, where ***N*** represents the n-gram order
  - Thus, for ***ROUGE-1***, we would measure the match rate of unigrams

The ROUGE score

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Reference: *"Trying is the first step towards your great success."*

Prediction: *"Trying is really the first step towards failure."*

- The calculations for ***ROUGE-2*** are based on recall, precision, and F-score

$$ROUGE2_{precision} = \frac{\#common\_bigrams}{\#bigrams\_prediction} = \frac{4}{7} = 57\%$$

$$ROUGE2_{recall} = \frac{\#common\_bigrams}{\#bigrams\_reference} = \frac{4}{8} = 0.5\%$$

$$F\text{–}score = 2 * \frac{ROUGE2_{precision} * ROUGE2_{recall}}{ROUGE2_{precision} + ROUGE2_{recall}} = 2 * \frac{0.57 * 0.5}{0.57 + 0.5} = 0.53\%$$

The ROUGE score

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Reference: *"Trying is the first step towards your great success."*

Prediction: *"Trying is really the first step towards failure."*

- ***ROUGE-L*** measures the ***longest common subsequence*** (LCS) between our prediction and reference
  - A subsequence appears in the same relative order (not necessarily contiguous)

$$ROUGEL_{precision} = \frac{LCS\_length}{\#words\_prediction} = \frac{6}{8} = 75\%$$

$$ROUGEL_{recall} = \frac{LCS\_length}{\#words\_reference} = \frac{6}{9} = 67\%$$

$$F{-}score = 2 * \frac{ROUGEL_{precision} * ROUGEL_{recall}}{ROUGEL_{precision} + ROUGEL_{recall}} = 2 * \frac{0.75 * 0.67}{0.75 + 0.67} = 71\%$$

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Let’s practice!

Tasks

- Web scraping
- Extractive summarization
- Abstractive summarization
- Performance

https://colab.research.google.com/github/PacktPublishing/Machine-Learning-Techniques-for-Text/blob/main/chapter-07/text-summarization-transformer.ipynb

Key takeaways

‹packt›

Machine Learning Techniques for Text

Apply modern techniques with Python for text processing, dimensionality reduction, classification, and evaluation

Visualizations

- Knowledge graphs

ML concepts

- Attention

ML algorithms & models

- Transformers

Performance metrics

- ROUGE score

Tools

- Web crawling and scraping

Machine Learning Techniques for Text

Questions?