



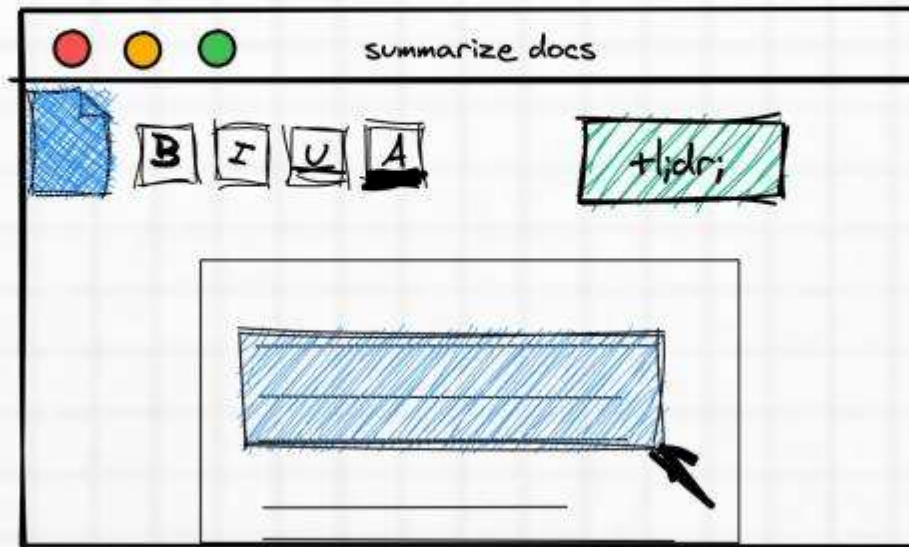
Add a text summarizer to Google Docs using Natural Language Processing Cloud's Text Summarization API

June 7, 2021

Article by [Rafał Rybnik](#), Head of Software Development at Instytut Badań Pollster

Whether you're a writer, data scientist or just skimming through sources to get a job done, reading longer texts to pluck out crumbs of information can be quite exhausting. Automating such elements of your work allows you to focus on the creative side of things.

Text Summarization



Unless stated otherwise, all pictures in the article are by the author.

Text Summarization

Text summarization is the technique of extracting key informational elements of a voluminous text. Manual text summarization is a difficult and time-expensive task, so Natural Language Processing and machine learning algorithms became popular to automate it.

There are ready-made solutions on the market, whether in the form of libraries or ready-made tools for end-users.

In this article, we will prepare our own tailored solution, which at the same time does not require advanced knowledge of data science.

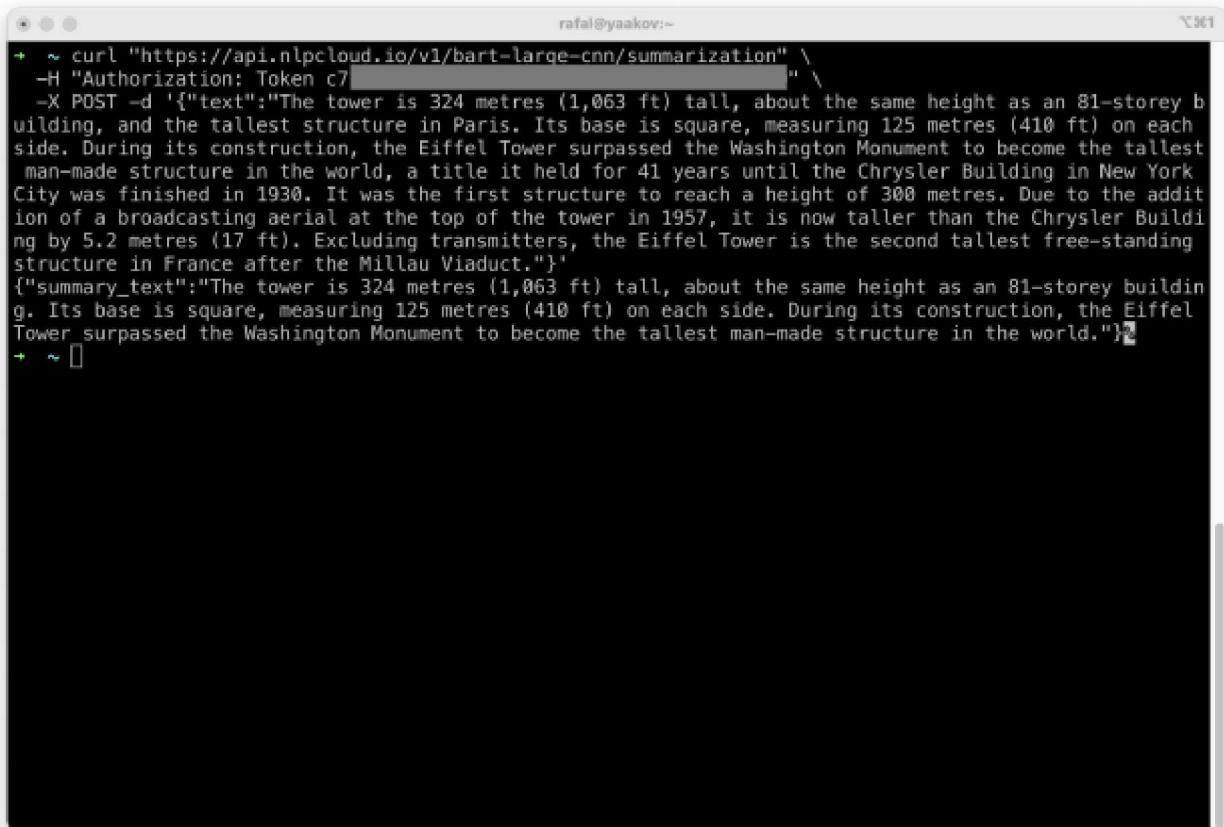
NLP Cloud

NLP Cloud is a provider of multiple APIs for text processing using machine learning models. One of these is the text summarizer, which looks promising in terms of simple implementation.

The summarizer provided by NLP Cloud is abstractive, which means that new sentences might be generated and parts with low information-noise ratio are

removed.

Let's see an example:

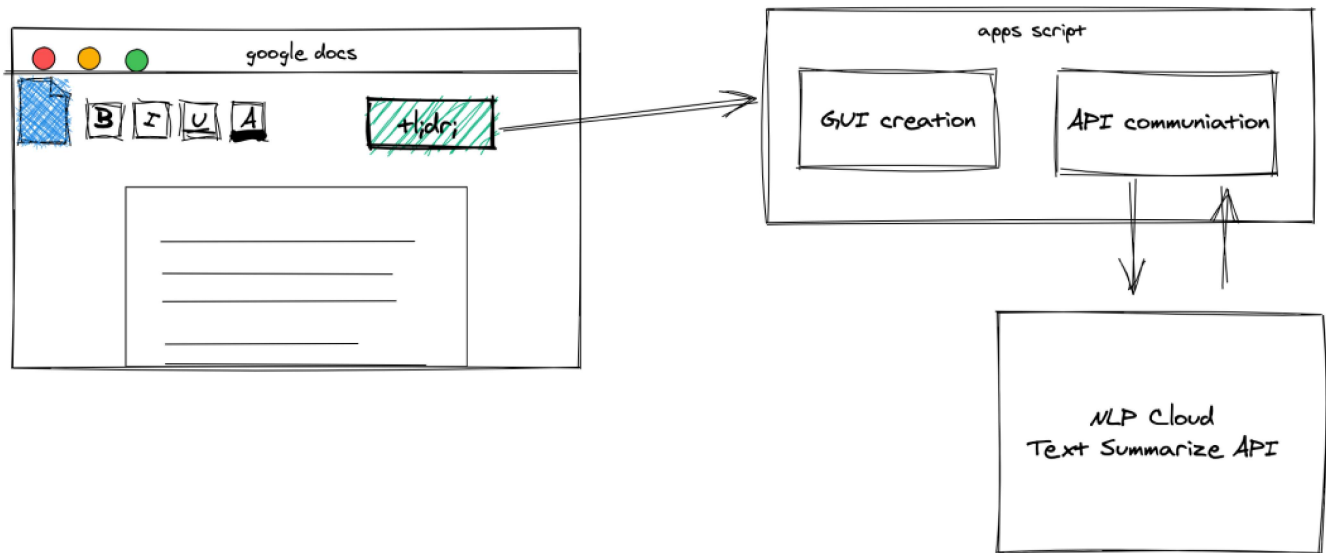


```
rafael@yaakov:~  
+ ~ curl "https://api.nlpcloud.io/v1/bart-large-cnn/summarization" \br/>  -H "Authorization: Token c7[REDACTED]" \br/>  -X POST -d '{"text":"The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building, and the tallest structure in Paris. Its base is square, measuring 125 metres (410 ft) on each side. During its construction, the Eiffel Tower surpassed the Washington Monument to become the tallest man-made structure in the world, a title it held for 41 years until the Chrysler Building in New York City was finished in 1930. It was the first structure to reach a height of 300 metres. Due to the addition of a broadcasting aerial at the top of the tower in 1957, it is now taller than the Chrysler Building by 5.2 metres (17 ft). Excluding transmitters, the Eiffel Tower is the second tallest free-standing structure in France after the Millau Viaduct."}'  
{"summary_text":"The tower is 324 metres (1,063 ft) tall, about the same height as an 81-storey building. Its base is square, measuring 125 metres (410 ft) on each side. During its construction, the Eiffel Tower surpassed the Washington Monument to become the tallest man-made structure in the world."}  
+ ~
```

We pass a block of text, and the model returns a summary. But operating in a console is not very convenient. So, let's make Google Docs summarize the selected text fragment this way.

Extending Google Docs

Our goal is to create a convenient menu so that text summarization happens automatically from within Google Docs.



This is how our project is structured. Using Apps Script, we will extend the GUI with a button that will trigger functions that communicate with the NLP Cloud API and then insert the result below.

We will start by preparing an add-on for the menu.

Making Menu

Using Google Apps Scripts to adding custom functionalities to Google Docs is fairly easy. You can customize the user interface with new menus, dialog boxes and sidebars. To create a script, after opening Google Docs, select Tools -> Script editor.

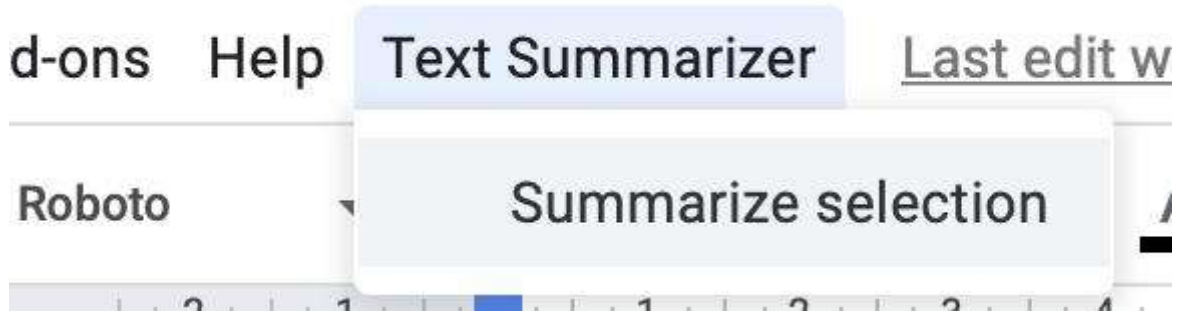
All the interface elements should be added in this function: `onOpen`. It is run after opening the document and allows us to add the menu.

```
function onOpen() {
  var ui = DocumentApp.getUi();

  ui.createMenu('Text Summarizer')
    .addItem('Summarize selection', 'summarizeSelection')
    .addToUi();
}

function summarizeSelection() {
  // summarization function
}
```

After saving script and refreshing Docs, you should find new menu element “Text Summarizer”.



We must get the selected text and then implement the following function:

```
summarizeSelection
```

Get Selection

The tricky part is getting currently selected text and passing it to functions. It is possible thanks to this function: `getSelection`.

```
DocumentApp.getActiveDocument().getSelection();
```

However, this function returns not only highlighted part of the documents, but a whole paragraph in which selections resides. That’s why we create more complex `getSelectedText` function:

```
function getSelectedText() {
  var document = DocumentApp.getActiveDocument();
  var body = document.getBody();
  var text = "";

  var selection = document.getSelection();
  var parent = false;
  var insertPoint

  if (selection) {
    var elements = selection.getRangeElements();

    if(elements[0].isPartial()) {
      text = elements[0].getElement().asText().getText();
      text = text.substring(elements[0].getStartOffset(),elements[0].getEndOffset());
      parent = elements[0].getElement().getParent();
    }
  }
}
```

```

    } else {
    text = elements.map( function(element) {
        parent = element.getElement().getParent();
        return element.getElement().asText().getText();
    });
    }
    // Logger.log(text);
    if (parent) {
    insertPoint = body.getChildIndex(parent);
    }
}
return [text, insertPoint];
}

```

The function simply takes the entire paragraph and trims it down to just the selected fragment. It also returns the index of the paragraph so you know where to insert the summary.

Now, let's send the text to the API and parse the result.

External APIs

We'll use the following service to make API requests directly: `UrlFetch`. Text Summarization API request requires authorization through token. To get it, sign up at nlpcloud.io (the free plan will do just fine).

The API being requested returns a raw JSON response for a request.

Remember that all text processed by this script are sent to external API.

```

function summarizeSelection() {
    var document = DocumentApp.getActiveDocument();
    var selectedText = getSelectedText();
    var text = selectedText[0];
    var insertPoint = selectedText[1];

    if (text) {
        DocumentApp.getUi().alert(text);
        var url = 'https://api.nlpcloud.io/v1/bart-large-cnn/summarization

        var response = UrlFetchApp.fetch(

```

```

url,
{
  'method': 'POST',
  'contentType': 'application/json',
  'headers':{
    'Authorization': 'Token c714fb961e7f6ef1336ab7f501f4d842f2dc23f',
  },
  'payload': JSON.stringify({"text":text}),
  'muteHttpExceptions': true
}
);

try {
  summary_text = JSON.parse(response.getContentText())['summary_text'];
  DocumentApp.getUi().alert(summary_text);

} catch (e) {
  DocumentApp.getUi().alert('Something went wrong 😞');
}

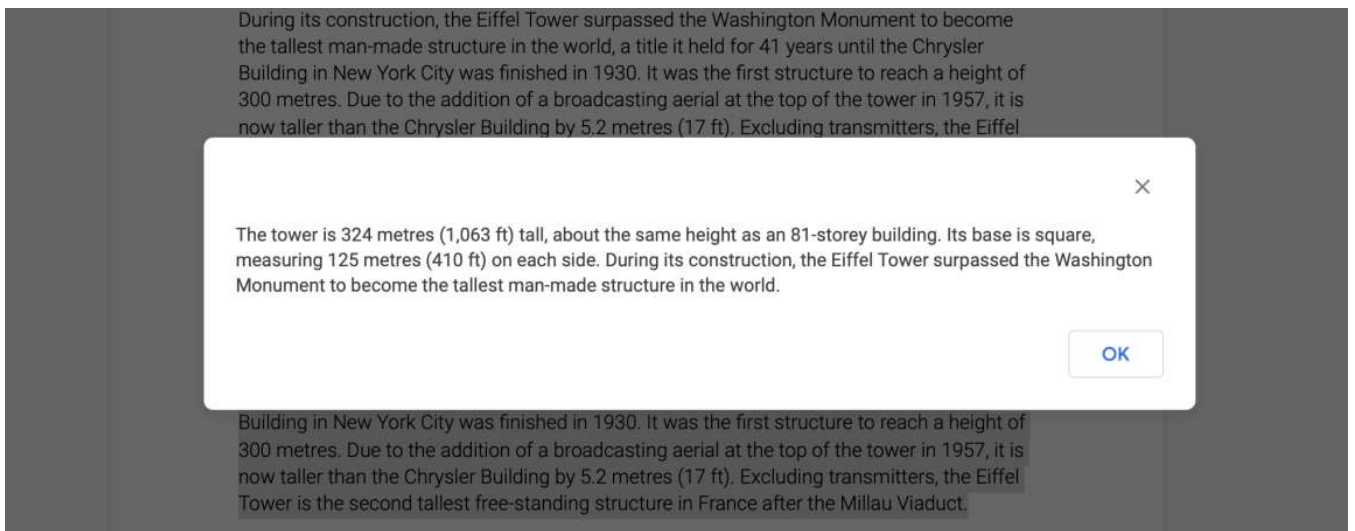
} else {
  DocumentApp.getUi().alert('You must select some text!');
}

}

```

To test the functionality, just select a text fragment in Docs and choose Text Summarization -> Summarize selection from the Docs menu.

After a moment of processing, you should see a pop-up with the result, which you can even copy.



Writing response to document

Finally, we can insert an API-generated summary directly into the document. For ease of distinction, bold the text of the summary. That's why the following function also returns the paragraph index of the selected text:

`getSelectedText` .

This way we know exactly where to tell Apps Script to insert the new text.

```
function summarizeSelection() {
  var document = DocumentApp.getActiveDocument();
  var selectedText = getSelectedText();
  var text = selectedText[0];
  var insertPoint = selectedText[1];

  if (text) {
    DocumentApp.getUi().alert(text);
    var url = 'https://api.nlpcloud.io/v1/bart-large-cnn/summarization';

    var response = UrlFetchApp.fetch(
      url,
      {
        'method': 'POST',
        'contentType': 'application/json',
        'headers': {
          'Authorization': 'Token c714fb961e7f6ef1336ab7f501f4d842f2dc23f',
        },
        'payload': JSON.stringify({"text":text}),
        'muteHttpExceptions': true
      }
    );
  }
}
```



```
);
```

```
try {  
  summary_text = JSON.parse(response.getContentText())['summary_text']  
  DocumentApp.getUi().alert(summary_text);
```

```
} catch (e) {  
  DocumentApp.getUi().alert('Something went wrong 😞');  
}
```

```
//////////
```

```
var body = document.getBody();
```

```
var summaryParagraph = body.insertParagraph(insertPoint+1, summary.  
summaryParagraph.setBold(true);
```

```
//////////
```

```
} else {  
  DocumentApp.getUi().alert('You must select some text!');  
}
```

```
}
```



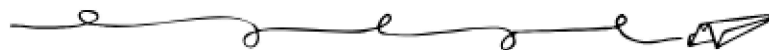
Let's test the final version:

Google Docs text summarization



Takeaways

Building an application that extends Google Docs functionality based on external APIs is an interesting alternative to building large systems to automate inconvenient work steps. Moreover, individual components can be replaced, depending on needs and specific skills. For example, our solution can be extended with emotion detection or text classification (also available in NLP Cloud). You can also prepare your own API or simple functions directly in Apps Script.



Thank you for reading. I hope you enjoyed reading as much as I enjoyed writing this for you.



[Playground](#)

[Contact and support](#)

[Documentation](#)

[Pricing](#)

[Press \(English\)](#)

[Presse \(Français\)](#)

[Presse \(Deutsch\)](#)

[Use Cases](#) 

[Blog Post Generation](#)

[Classification](#)

[Chatbot/Conversational
AI](#)

[Code Generation](#)

[Embeddings](#)

[Grammar and Spelling
Correction](#)

[Headline Generation](#)

[Intent Classification](#)

[Keyword and Keyphrase
Extraction](#)

[Language Detection](#)

[Lemmatization](#)

[Named Entity
Recognition \(NER\)](#)

[Paraphrasing](#)

[Part-Of-Speech tagging](#)

[Product Description
and Ad Generation](#)

[Question Answering](#)

[Privacy
Policy](#)

[Service
Level
Agreement](#)



Semantic Similarity

Sentiment and Emotion
Analysis

Summarization

Text Generation with
GPT-J

Tokenization

Translation

Blog



Introduction to the
NLP Cloud API with the
Python client

Designing a
classification Natural
Language Processing
API with FastAPI and
Transformers

Adding a text
summarizer to Google
Docs

Contextual ad
targeting using text
classification

Natural Language
Processing
introduction: what is
Natural Language
Processing?

Google Cloud Natural
Language VS NLP Cloud

GPT-3 open-source
alternatives: GPT-J
and GPT-Neo

Effectively using GPT-
J and GPT-Neo with
few-shot learning

Zero-shot learning for
text classification

Fine-tuning GPT-J

How to build a chatbot
with GPT-3/GPT-J

How to speed up
inference of Natural
Language Processing
Transformers