PREPROCESS

Preprocess splits documents into optimal chunks of text for use in language model tasks. If you want to learn more about Preprocess check out preprocess.co

GETTING STARTED

You should have already received an API key, if not please write to us at support@preprocess.co.

If you want to first give a try to the solution without implementing code, please have a look at the playground section.

Preprocessing is a time-intensive task, for this reason, the API is asynchronous. The response to the API call will confirm the document has been received correctly, and when the chunking is completed the result will be sent to the indicated webhook. If you are not in the conditions to set up a webhook we got you covered.

REQUEST

The request for chunking a document:

cURL
cURL curl --location

```
--request POST 'https://chunk.ing' \
--header 'Content-Type: multipart/form-data' \
--header 'x-api-key: your_api_key' \
--form 'file=@"/your_file.ext"'
```

- [header] **x-api-key:** the 32-character API Key token. If you don't have one yet, please reach out to support at support@preprocess.co.
- [form] **file**: the file to be uploaded as binary in the request form. Maximum file size 30 MB. Allowed file types: pdf, doc, docx, ppt, pptx, xls, xlsx, odt, ods, odp, eml, html, plain text

Optional query parameters:

- **webhook**: the URL to be called after chunking, where the result will be posted (GETTING THE RESULT BY WEBHOOK). If you prefer to get the result through an HTTP request (GETTING THE RESULT BY HTTP REQUEST), omit this parameter.
- **repeat_title**: [true, false] [default false]
  If set to true, each chunk will include the title of the parent paragraph/section.
- **keep_header**: [true, false] [default true]
  If set to false, the content of the headers will be removed. Headers may include page numbers, document titles, section titles, paragraph titles, and fixed layout elements.
- **smart_header**: [true, false] [default true]
  If set to true, only relevant titles will be included in the chunks, while other information will be removed. Relevant

titles are those that should be part of the body of the page as a title.
If set to false, only the keep_header parameter will be considered. If keep_header is false, the smart_header parameter will be ignored.

- **keep_footer**: [true, false] [default false]
  If set to true, the content of the footers will be included in the chunks. Footers may include page numbers, footnotes, and fixed layout elements.
- **image_text**: [true, false] [default false]
  If set to true, the text contained in the images will be added to the chunks.
- **table_output_format**: ["text", "markdown", "html"] [default "text"]
  Specifies the output format for tables.
- **repeat_table_header**: [true, false] [default false]
  If set to true, and if tables are split across multiple chunks, each chunk will include the table row header.
- **language**:
  The language in ISO 639-1 format. If not provided, the system will automatically identify it.
- **merge**: [true, false] [default false]
  If set to true, short chunks will be merged with others to maximize the chunk length.

RESPONSE

cURL

```
cURL
curl --location
--request POST 'https://chunk.ing/?repeat_title=true&webhook=https%3A%2F%2Fwebhook.url' \
--header 'Content-Type: multipart/form-data' \
--header 'x-api-key: your_api_key' \
--form 'file=@"/your_file.ext"'
```

The response to the above curl request:

JSON in case of success

```
{
  "status": "OK",
  "success": true,
  "data": {"process":{"id": string }},
  "message": "Chunking started."
}
```

JSON in case of error

```
{
  "status": "ERROR",
  "success": false,
  "error": …object with additional data…,
  "message": "Unsupported file extension"
}
```

```
{
  "status": "ERROR",
  "success": false,
  "error": …object with additional data…,
  "message": "File too large, the allowed max size is 30MB"
}
```

```
{
  "status": "ERROR",
  "success": false,
  "error": …object with additional data…,
  "message": "Unknown error, please try again later"
}
```

In case of success the response will provide you with an ID that you can use to keep track of the different jobs you started.

GETTING THE RESULT BY WEBHOOK

If you set up a webhook in the chunking request, Preprocess will call the URL provided in the webhook parameter when the chunking of your document is finished. The webhook request will be a POST in JSON format:

JSON in case of success

```
{
  "status": "OK",
  "success": true,
  "message": "The file has been chunked successfully."
```

```
  "info": {
    "file":{"name": string }
  },
  "data": {
    "process":{
      "id": string
    },
    "detected_language": string ,
    "chunks": [
      "first chunk …",
      "second chunk …",
      …
    ]
  }
}
```

JSON in case of error

```
{
  "status": "ERROR",
  "success": false,
  "error": …object with additional data…,
  "message": "An error happened during the chunking of your document,
please try again later"
}
```

Some details on the data you will receive:

- **data > process > id**: the ID you received in the response of the chunking request.
- **data > detected_language**: the language you provided in the chunking request, or the one automatically identified.

- **data > chunks**: the list of chunks the document has been divided into.
- **info > file > name**: the name of the file you uploaded.

GETTING THE RESULT BY HTTP REQUEST

If you have not provided a webhook, you can get the result via an API request. This endpoint will provide you with the information related to the ID of the chunking request you initiated. Once the chunking is completed, the endpoint will return the result in the response.

cURL

```
cURL
curl --location
--request GET 'https://chunk.ing/get_result?id=id_of_the_chunking_request' \
--header 'x-api-key: your_api_key'
```

Depending on the status of the chunking process you will get different responses:

JSON in case of chunking not finished yet

```
{
  "status": "OK",
  "success": false,
```

```
  "message": "The file chunking has not finished yet."
}
```

JSON in case of success

```
{
  "status": "OK",
  "success": true,
  "message": "The file has been chunked successfully."
  "info": {
    "file":{"name": string }
  },
  "data": {
    "process":{
      "id": string
    },
    "detected_language": string ,
    "chunks": [
      "first chunk …",
      "second chunk …",
      …
    ]
  }
}
```

JSON in case of error

```
{
  "status": "ERROR",
  "success": false,
  "error": …object with additional data…,
  "message": "An error happened during the chunking of your document,
please try again later"
}
```

Please keep in mind that every file and result will be deleted 24 hours after the chunking request.

Some details on the data you will receive:

- **success**: false if the process is not yet finished, true otherwise.
- **data > process > id**: the ID you received in the response of the chunking request.
- **data > detected_language**: the language you provided in the chunking request, or the one automatically identified.
- **data > chunks**: the list of chunks the document has been divided into.
- **info > file > name**: the name of the file you uploaded.

EXAMPLES

PYTHON

Coming soon!

LANGCHAIN

If you want to integrate Preprocess results in your Langchain flow just load the chunks as Documents using the following snippet.

Langchain example

```
from langchain.document_loaders import JSONLoader

loader = JSONLoader(file_path="result.json", jq_schema='.data.chunks[]')
data = loader.load()
```

In the above example “result.json” is a JSON file containing the information posted in the webhook you provided.

Alternatively, you can execute the above logic on the webhook you provided that received the results of the chunking process.

LLAMAINDEX

If you want to integrate Preprocess results in your llamaindex flow just load the chunks as Documents using the following snippet.

llamaindex example

```
import json
from llama_index.readers.schema.base import Documents
from llama_index import VectorStoreIndex

with open('result.json', 'r') as f:
    result = json.load(f)

index = VectorStoreIndex([])
for chunk in result['data']['chunks']:
    index.insert(Document(text=chunk, extra_info={}))
query_engine = index.as_query_engine()
```

In the above example “result.json” is a JSON file containing the information posted in the webhook you provided.

PLAYGROUND

We have released a playground to easily test our solution. Go to https://playground.chunk.ing insert your API key, upload a file, and see the results.

SUPPORT

For any doubt, issue, or request, please write at support@preprocess.co , we are at your disposal.