OCR and Document Understanding

Document OCR processor

The Document OCR (Optical Character Recognition) processor, powered by our latest OCR model `mistral-ocr-latest`, enables you to extract text and structured content from PDF documents.

Key features:

- Extracts text content while maintaining document structure and hierarchy
- Preserves formatting like headers, paragraphs, lists and tables
- Returns results in markdown format for easy parsing and rendering
- Handles complex layouts including multi-column text and mixed content
- Processes documents at scale with high accuracy
- Supports multiple document formats including PDF, images, and uploaded documents

The OCR processor returns both the extracted text content and metadata about the document structure, making it easy to work with the recognized content programmatically.

OCR with PDF

python typescript curl

```typescript
import { Mistral } from '@mistralai/mistralai';

const apiKey = process.env.MISTRAL_API_KEY;
const client = new Mistral({apiKey: apiKey});

const ocrResponse = await client.ocr.process({
    model: "mistral-ocr-latest",
    document: {
        type: "document_url",
        documentUrl: "https://arxiv.org/pdf/2201.04234"
    },
    includeImageBase64: true
});
```

Or passing a Base64 encoded pdf:

```typescript
import { Mistral } from '@mistralai/mistralai';
import fs from 'fs';
```

```
async function encodePdf(pdfPath) {
    try {
        // Read the PDF file as a buffer
        const pdfBuffer = fs.readFileSync(pdfPath);

        // Convert the buffer to a Base64-encoded string
        const base64Pdf = pdfBuffer.toString('base64');
        return base64Pdf;
    } catch (error) {
        console.error(`Error: ${error}`);
        return null;
    }
}

const pdfPath = "path_to_your_pdf.pdf";

const base64Pdf = await encodePdf(pdfPath);

const apiKey = process.env.MISTRAL_API_KEY;
const client = new Mistral({ apiKey: apiKey });

try {
    const ocrResponse = await client.ocr.process({
        model: "mistral-ocr-latest",
        document: {
            type: "document_url",
            documentUrl: "data:application/pdf;base64,"+base64Pdf
        },
        includeImageBase64: true
    });
    console.log(ocrResponse);
} catch (error) {
    console.error("Error processing OCR:", error);
}
```

▸ **Example output:**

OCR with uploaded PDF

You can also upload a PDF file and get the OCR results from the uploaded PDF.

Upload a file

python typescript curl

```
import { Mistral } from '@mistralai/mistralai';
import fs from 'fs';
```

```
const apiKey = process.env.MISTRAL_API_KEY;

const client = new Mistral({apiKey: apiKey});

const uploadedFile = fs.readFileSync('uploaded_file.pdf');
const uploadedPdf = await client.files.upload({
    file: {
        fileName: "uploaded_file.pdf",
        content: uploadedFile,
    },
    purpose: "ocr"
});
```

Retrieve File

python typescript curl

```
const retrievedFile = await client.files.retrieve({
    fileId: uploadedPdf.id
});
```

```
id='00edaf84-95b0-45db-8f83-f71138491f23' object='file' size_bytes=3749788
created_at=1741023462 filename='uploaded_file.pdf' purpose='ocr'
sample_type='ocr_input' source='upload' deleted=False num_lines=None
```

Get signed URL

python typescript curl

```
const signedUrl = await client.files.getSignedUrl({
    fileId: uploadedPdf.id,
});
```

Get OCR results

python typescript curl

```
import { Mistral } from '@mistralai/mistralai';

const apiKey = process.env.MISTRAL_API_KEY;
const client = new Mistral({apiKey: apiKey});

const ocrResponse = await client.ocr.process({
    model: "mistral-ocr-latest",
```

```
    document: {
        type: "document_url",
        documentUrl: signedUrl.url,
    }
});
```

OCR with image

python typescript curl

```
import { Mistral } from '@mistralai/mistralai';

const apiKey = process.env.MISTRAL_API_KEY;
const client = new Mistral({apiKey: apiKey});

const ocrResponse = await client.ocr.process({
    model: "mistral-ocr-latest",
    document: {
        type: "image_url",
        imageUrl:
"https://raw.githubusercontent.com/mistralai/cookbook/refs/heads/main/mistral/ocr/receipt.
    }
});
```

Or passing a Base64 encoded image:

```
import { Mistral } from '@mistralai/mistralai';
import fs from 'fs';

async function encodeImage(imagePath) {
    try {
        // Read the image file as a buffer
        const imageBuffer = fs.readFileSync(imagePath);

        // Convert the buffer to a Base64-encoded string
        const base64Image = imageBuffer.toString('base64');
        return base64Image;
    } catch (error) {
        console.error(`Error: ${error}`);
        return null;
    }
}

const imagePath = "path_to_your_image.jpg";

const base64Image = await encodeImage(imagePath);

const apiKey = process.env.MISTRAL_API_KEY;
const client = new Mistral({ apiKey: apiKey });
```

```
try {
    const ocrResponse = await client.ocr.process({
        model: "mistral-ocr-latest",
        document: {
            type: "image_url",
            imageUrl: "" + base64Image
        },
        includeImageBase64: true
    });
    console.log(ocrResponse);
} catch (error) {
    console.error("Error processing OCR:", error);
}
```

Document understanding

The Document understanding capability combines OCR with large language model capabilities to enable natural language interaction with document content. This allows you to extract information and insights from documents by asking questions in natural language.

The workflow consists of two main steps:

1. Document Processing: OCR extracts text, structure, and formatting, creating a machine-readable version of the document.

2. Language Model Understanding: The extracted document content is analyzed by a large language model. You can ask questions or request information in natural language. The model understands context and relationships within the document and can provide relevant answers based on the document content.

Key capabilities:

- Question answering about specific document content
- Information extraction and summarization
- Document analysis and insights
- Multi-document queries and comparisons
- Context-aware responses that consider the full document

Common use cases:

- Analyzing research papers and technical documents
- Extracting information from business documents
- Processing legal documents and contracts
- Building document Q&A applications
- Automating document-based workflows

The examples below show how to interact with a PDF document using natural language:

python typescript curl

```typescript
import { Mistral } from "@mistralai/mistralai";
// import fs from 'fs';

// Retrieve the API key from environment variables
const apiKey = process.env["MISTRAL_API_KEY"];

const client = new Mistral({
  apiKey: apiKey,
});

// If local document, upload and retrieve the signed url
// const uploaded_file = fs.readFileSync('uploaded_file.pdf');
// const uploaded_pdf = await client.files.upload({
//     file: {
//         fileName: "uploaded_file.pdf",
//         content: uploaded_file,
//     },
//     purpose: "ocr"
// });
// const signedUrl = await client.files.getSignedUrl({
//     fileId: uploaded_pdf.id,
// });

const chatResponse = await client.chat.complete({
  model: "mistral-small-latest",
  messages: [
    {
      role: "user",
      content: [
        {
          type: "text",
          text: "what is the last sentence in the document",
        },
        {
          type: "document_url",
          documentUrl: "https://arxiv.org/pdf/1805.04770",
          // documentUrl: signedUrl.url
        },
      ],
    },
  ],
});

console.log("JSON:", chatResponse.choices[0].message.content);
```

Cookbooks

For more information and guides on how to make use of OCR and leverage document understanding, we have the following cookbooks:

- Tool Use and Document Understanding
- Batch OCR
- Structured OCR

FAQ

Are there any limits regarding the OCR API?

Yes, there are certain limitations for the OCR API. Uploaded document files must not exceed 50 MB in size and should be no longer than 1,000 pages.