# Research into LLM

To determine the best model for addressing the stakeholder's problem, I conducted a thorough investigation into various large language models (LLMs). This research involved exploring the models available for local use, evaluating their strengths and weaknesses, and identifying their potential for extracting relevant information from technical documents such as the "raming."

For each model, I documented the pros and cons, considering factors like performance, resource requirements, and suitability for handling architectural and engineering-related content. Additionally, I tested each model on a sample PDF to assess its accuracy in extracting key parameters. This allowed me to compare the models not only theoretically but also in practical application, providing a clear understanding of which model offers the best balance of accuracy and efficiency for our specific use case.

## Key Requirements for Models:

1. **Text Parsing and Information Extraction**: The model must be able to handle technical text, extract relevant parameters, and categorize them correctly (e.g., dimensions, material types, power ratings).
2. **Handling dutch language:** It is important that the model can recognize and handle the dutch language since the Bestek documents are written in dutch.
3. **Comprehension of Architecture and Engineering Terms**: The model must understand and process terms related to construction, architecture, and building systems.
4. **Efficiency**: Since we are using these models locally with Ollama, computational efficiency and ability to run on local hardware (with reasonable resources like a GPU) are important.
5. **Customizability**: The ability to fine-tune the model for our specific task (e.g., architecture-focused or for extracting bestek data).

## Choosing the models

In order to find which models I should test I looked into multiple sources that list the most popular language. It was also important that the model is usable with Ollama and that it can extract parameters from text. After doing this research and talking about it with the group we decided to research the following models:

* Llama3
* Gemma2
* Mistral
* Nuextract

## Testing the models

To test each of the models I created a chunk of information that I will use to extract data from. I will use each model and record the time, accuracy and analyse the results to see if it has made any mistakes. All tests will be done using a Nvidia GTX 1080 GPU running Ollama locally using Docker on Windows. I will let the model do the same task 5 times to see if there are any changes in results.

## Potential challenges

* **Asked for a value that is not in the text**

I asked for the value of the liftInstallatie which is not in the text. This will help me see if the model will hallucinate a result or will return 0 like asked.

* **Used a synonym**

For the verlichtingsinstallatie I changed it to lampinstallatie in the text to see if it understands the synonym and still returns the correct value

**The prompt that I will be using is as follows:**

"prompt": """

You are an assistant specialized in extracting data.

Your task is to extract the \*\*total cost related to different parameters\*\*

from the following text and return it in the following \*\*valid JSON format\*\* only:

{{

"value": {{

"totaalkosten noodstroomvoorziening": "<extracted\_cost>"

"totaalkosten krachtstroominstallatie": "<extracted\_cost>"

"totaalkosten verlichtingsinstallatie": "<extracted\_cost>"

"totaalkosten beveiligingsinstallatie": "<extracted\_cost>"

"totaalkosten appartementen": "<extracted\_cost>"

"totaalkosten alles": "<extracted\_cost>"

"totaalkosten liftInstallatie": "<extracted\_cost>"

}}

}}

If the total cost cannot be determined, return:

{{

"value": {{

"totaalkosten noodstroomvoorziening": "0"

"totaalkosten krachtstroominstallatie": "0"

"totaalkosten verlichtingsinstallatie": "0"

"totaalkosten beveiligingsinstallatie": "0"

"totaalkosten appartementen": "0"

"totaalkosten alles": "0"

"totaalkosten liftInstallatie": "0”

}}

}}

Please provide only the total cost and no other calculations, explanations, or extra text.

Here's the text:

"""

**The text chunk is as followed:**

The text chunk that I used is copied from one of the ‘ramings’ documents that contains a lot of the important parameters when it comes to costs. I chose this chunk because it included a lot of the parameters in a short amount of text. Making it a great test to see if the models can get the parameters and values out of there.

totaaloverzicht per hoofdstuk

totaal totaal totaal totaal totaal totaal

uren loon materiaal derden zonder met

opslagen opslagen

directe kosten

70.11.10 centrale elektrotechnische voorzieningen 0,00 uur Euro---58.646

70.11.11 noodstroomvoorziening 0,00 uur Euro---23.965

70.11.20 krachtstroominstallatie 0,00 uur Euro---14.618

70.11.30 lampeninstallatie 0,00 uur Euro---95.377

75.10.11 telematica installaties 0,00 uur Euro---1.890

75.10.12 intercom/videofoon installatie 0,00 uur Euro---5.670

75.10.21 beveiligingsinstallatie 0,00 uur Euro---1.302

80.11 transportinstallatie 0,00 uur Euro--- 145.000

appartementen/woningen 0,00 uur Euro--- 438.380

totaal 784.847,77

**If the model is fully accurate and follows the prompt the result should be:**

{{

"value": {{

"totaalkosten noodstroomvoorziening": 23.965

"totaalkosten krachtstroominstallatie": 14.618

"totaalkosten verlichtingsinstallatie": 95.377

"totaalkosten beveiligingsinstallatie": 1.302

"totaalkosten appartementen": 438.380

"totaalkosten alles": 784.847,77

}}

}}

## Llama3

**Token limit: 8000  
Parameters: 70B**

LLaMA 3 (Large Language Model Meta AI 3) is an open-source language model developed by Meta, designed to be highly efficient and powerful for natural language understanding tasks. It is available in various sizes, optimized for performance while using fewer computational resources compared to some other large models.

Testing LLaMA 2 is a good idea because it provides strong general-purpose language capabilities, making it versatile for various tasks, including document analysis. Its balance between performance and resource efficiency makes it a solid choice for extracting and understanding data from complex documents like *bestek* files.

#### Testing

**Amount of times ran:** 5  
**Average time:** 8,38 seconds **Standard deviation:** 4.43seconds **Results:**

**A screenshot of a computer

Description automatically generated**

**Evaluation and accuracy:**

The model is fully accurate on all of the values, the only problem is that it did not fully listen to the prompt given to the model. It gave a bunch of extra information that is not needed. It also instead of giving back the 6 parameters that I asked for gave back 10 parameters. This would not work for this project as it is important that the model will fully follow all instructions given.

It also sometimes hallucinated a random number for the liftinstallatie but this only happened once in the 5 times I ran it.

It did pass the synonym part and gave back the correct value.  
I also notice that everytime I retry the prompt it gives back a slightly different answer when it comes to the structure of the results.

**Pros**:

* Relatively fast for its size (8,38 seconds average).
* Accurate in extracting values, passing the synonym recognition test.

**Cons**:

* Inconsistent with following instructions, returning more parameters than requested (10 instead of 6).
* Tends to generate extra, unnecessary information that wasn't part of the prompt.
* Occasional hallucinations (e.g., inventing a number for "liftinstallatie").

## Mistral

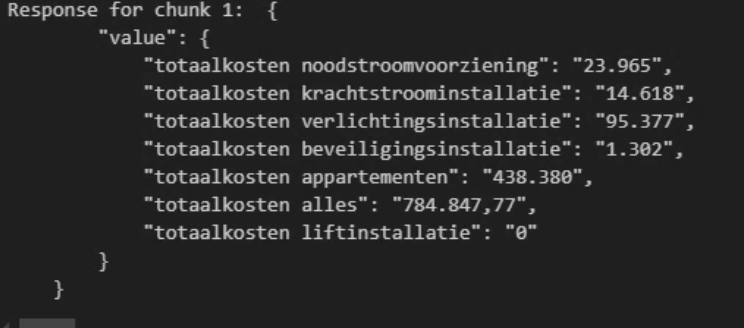
**Token limit: 4096  
Parameters: 7.25B**

Mistral is a new open-source large language model (LLM) known for its efficiency and performance. It is designed to be lightweight while delivering results comparable to much larger models, thanks to optimized architectures and training techniques. Mistral models often have fewer parameters but can still perform complex tasks, making them resource-efficient and cost-effective.

Testing Mistral is a good idea because it provides high-quality outputs while being faster and requiring less computational power. This makes it ideal for integration into projects where speed and efficiency are critical, such as extracting data from architectural documents.

#### Testing

**Amount of times ran:** 5  
**Average time:** 4,78 seconds **Standard deviation:** 0,19 seconds **Results:**

**  
Evaluation and accuracy:**

The results that mistral gave were very impresive. Not only did it do exactly as stated in the prompt, it also got it 100% correct everytime I ran the test. It fully passed both of the challenges without any issues.

**Pros**:

* Fastest among the tested models (4,78 seconds average).
* Fully follows instructions, delivering exactly what is requested.
* 100% accuracy across all tests, including synonym recognition.

**Cons**:

* None observed.

## Nuextract

**Token limit: 2000  
Parameters: 3.8B**

NuExtract is a version of [phi-3-mini](https://huggingface.co/microsoft/Phi-3-mini-4k-instruct), fine-tuned on a private high-quality synthetic dataset for information extraction. This model is purely extractive, so all text output by the model is present as is in the original text.

#### Testing

**Amount of times ran:** 5  
**Average time:** 6.44 Seconds **Standard deviation:** 0.29 Seconds **Results:**

**A screen shot of a computer

Description automatically generated  
Evaluation and accuracy:**

Nuextract was really accurate and did not make any mistakes when it came to the values. It also recognized that there was no liftinstallatie in the text and therefore didn’t return it as an answer. What it didn’t do well was recognizing the synonyms, it did not return anything for the verlichtingsinstallatie. For this reason this model will be unusable this synonym recognition is really important for the end product.

**Pros**:

* Very fast (6.44 seconds average).
* Accurate with extracting values, and appropriately ignores non-existent values (e.g., not returning "liftinstallatie" when it's not present).

**Cons**:

* Struggles with synonym recognition, failing to detect alternative terms like "verlichtingsinstallatie."

## Gemma2

**Token limit: 8192  
Parameters: 9B**

#### Testing

**Amount of times ran:** 5  
**Average time:** 15.94 seconds **Standard deviation:** 2.93 seconds  
**Results:**

**A screenshot of a computer

Description automatically generated**

**Evaluation and accuracy:**

Gemma2 is a very accurate llm on the data that I am using. It gave the correct answer every single time that I ran the prompt. It was really slow compared to the other models, and also had some of the same issues as the Llama3 model. The issue was that it did not fully listen to the prompt and added random texts as answers next to the json.

**Pros**:

* Accurate in extracting values, delivering correct answers consistently.

**Cons**:

* Slowest among the tested models (15,94 seconds average).
* Similar to LLaMA 3, it sometimes fails to fully adhere to the prompt, providing extra, irrelevant information alongside the desired output.

## Conclusion

After evaluating the different models, **Mistral** stands out as the top performer. It combines speed, accuracy, and the ability to strictly follow instructions, completing tasks in just 5.4 seconds on average. Unlike other models, Mistral consistently delivers exactly what is requested, without adding unnecessary information or hallucinating values. It also excels in handling synonym recognition, making it highly reliable for complex data extraction tasks. This level of precision and efficiency makes Mistral the best choice for projects requiring both speed and strict adherence to instructions, outperforming larger models like LLaMA 3 and Gemma2, which struggled with following prompts consistently. Overall, Mistral offers an ideal balance of performance and resource efficiency.

## Sources:

<https://ai.meta.com/blog/meta-llama-3/>

<https://www.llama.com/>

<https://ollama.com/library/mistral>

<https://ollama.com/library/nuextract>

https://ollama.com/library/gemma2