


 Albert0i / mariadb-vss-chinese

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [...](#)

 [main](#) [mariadb-vss-chinese / README.md](#) 









Albert0i write more

7e64bef · 1 minute ago



269 lines (204 loc) · 10.7 KB

[Preview](#) [Code](#) [Blame](#)

[Raw](#)    

Vector Semantic Search in Chinese using MariaDB

"the ridiculous confusion which, in some circumstances, can determine the course of a man's life."

The Castle by Franz Kafka

Prologue

- [Vector Semantic Search using MariaDB \(Part 1/3\)](#)
- [Vector Semantic Search using MariaDB \(Part 2/3\)](#)
- [Vector Semantic Search using MariaDB \(Part 3/3\)](#)

I. Choosing a Model

Previously we used VSS on English data. *Every tool would render useless if it can not work on one's native language.* We need to do the same thing on Chinese data, typically [traditional Chinese](#). To be able to extract semantic, we need a different model as it is suggested by AI.

give me the name of model to be used with node-llama-cpp to extract semantic of traditional Chinese which can be downloaded from hugging face website

You can use **DeepSeek-V2-7B-Chat** or **Chinese-Llama-2-7B** for semantic extraction in Traditional Chinese with `node-llama-cpp`. These models are available on Hugging Face and optimized for Chinese NLP tasks. ¹ ²

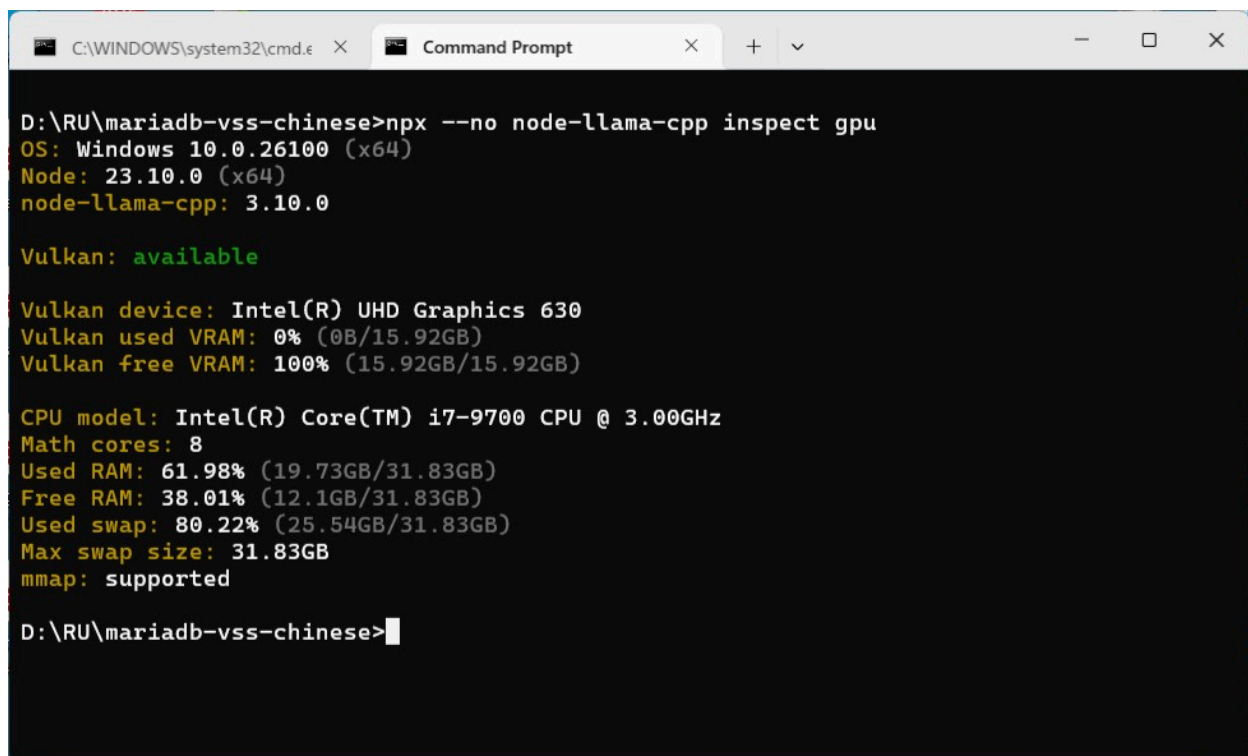
Would you like help setting up `node-llama-cpp` with one of these models? 🤖🚀

¹ huggingface.co

² www.danielcorin.com

I head into [Hugging Face](https://huggingface.co) website and search for [Chinese-Llama-2-7B](https://huggingface.co/mistral-community/chinese-llama-2-7b) model and inspect my computer with:

```
npx --no node-llama-cpp inspect gpu
```



```
D:\RU\mariadb-vss-chinese>npx --no node-llama-cpp inspect gpu
OS: Windows 10.0.26100 (x64)
Node: 23.10.0 (x64)
node-llama-cpp: 3.10.0

Vulkan: available

Vulkan device: Intel(R) UHD Graphics 630
Vulkan used VRAM: 0% (0B/15.92GB)
Vulkan free VRAM: 100% (15.92GB/15.92GB)

CPU model: Intel(R) Core(TM) i7-9700 CPU @ 3.00GHz
Math cores: 8
Used RAM: 61.98% (19.73GB/31.83GB)
Free RAM: 38.01% (12.1GB/31.83GB)
Used swap: 80.22% (25.54GB/31.83GB)
Max swap size: 31.83GB
mmap: supported

D:\RU\mariadb-vss-chinese>
```

If the machine you plan to run this model on doesn't have a GPU, you'd probably want to use a small model that can run on a CPU with decent performance.

If you have a GPU, the amount of VRAM you have will determine the size of the model you can run. Ideally, you'd want to fit the entire model in the VRAM to use only the GPU and achieve maximum performance. If the model requires more memory than the available VRAM, parts of it will be offloaded to the RAM and be evaluated using the CPU, significantly reducing the efficiency and speed of inference.

To get a more accurate estimation of how well a model will run:

```
npx --no node-llama-cpp inspect estimate ./src/models/ggml-model-q2_k.gguf
```



```
D:\RU\mariadb-vss-chinese>npx --no node-llama-cpp inspect estimate ./src/models/ggml-model-q2_k.gguf
File: D:\RU\mariadb-vss-chinese\src\models\ggml-model-q2_k.gguf
GPU info      Type: Vulkan   VRAM: 15.92GB
              Name: Intel(R) UHD Graphics 630
Model info    Type: llama MOSTLY_Q2_K   Size: 2.73GB
              Train context size: 4.1K
Resolved config 100% compatibility   Context size: 4.1K
              GPU layers: 33/33   VRAM usage: 5.05GB
              RAM usage: 178.88MB
With flash attention 100% compatibility   Context size: 4.1K
              GPU layers: 33/33   VRAM usage: 4.79GB
              RAM usage: 178.88MB   Flash attention: enabled
D:\RU\mariadb-vss-chinese>
```

```
C:\WINDOWS\system32\cmd.exe X Command Prompt X + v - □ X

D:\RU\mariadb-vss-chinese>npx --no node-llama-cpp inspect estimate ./src/models/ggml-
model-q4_k.gguf
File: D:\RU\mariadb-vss-chinese\src\models\ggml-model-q4_k.gguf
GPU info      Type: Vulkan   VRAM: 15.92GB
              Name: Intel(R) UHD Graphics 630
Model info    Type: llama MOSTLY_Q4_K_M   Size: 3.92GB
              Train context size: 4.1K

Resolved config 100% compatibility   Context size: 4.1K
                GPU layers: 33/33   VRAM usage: 6.19GB
                RAM usage: 229.5MB
With flash attention 100% compatibility   Context size: 4.1K
                    GPU layers: 33/33   VRAM usage: 5.93GB
                    RAM usage: 229.5MB   Flash attention: enabled

D:\RU\mariadb-vss-chinese>
```

```
C:\WINDOWS\system32\cmd.exe X Command Prompt X + v - □ X

D:\RU\mariadb-vss-chinese>npx --no node-llama-cpp inspect estimate ./src/models/ggml-
model-q8_0.gguf
File: D:\RU\mariadb-vss-chinese\src\models\ggml-model-q8_0.gguf
GPU info      Type: Vulkan   VRAM: 15.92GB
              Name: Intel(R) UHD Graphics 630
Model info    Type: llama MOSTLY_Q8_0   Size: 6.86GB
              Train context size: 4.1K

Resolved config 100% compatibility   Context size: 4.1K
                GPU layers: 33/33   VRAM usage: 9.02GB
                RAM usage: 337.5MB
With flash attention 100% compatibility   Context size: 4.1K
                    GPU layers: 33/33   VRAM usage: 8.76GB
                    RAM usage: 337.5MB   Flash attention: enabled

D:\RU\mariadb-vss-chinese>
```

```

D:\RU\mariadb-vss-chinese>npx --no node-llama-cpp inspect estimate ./src/models/ggml-
model-f16.gguf
File: D:\RU\mariadb-vss-chinese\src\models\ggml-model-f16.gguf
GPU info      Type: Vulkan   VRAM: 15.92GB
              Name: Intel(R) UHD Graphics 630
Model info    Type: llama MOSTLY_F16   Size: 12.91GB
              Train context size: 4.1K

Resolved config 89% compatibility   Context size: 4.1K
                GPU layers: 32/33   VRAM usage: 14.45GB
                RAM usage: 972.02MB
With flash attention 89% compatibility   Context size: 4.1K
                    GPU layers: 32/33   VRAM usage: 14.19GB
                    RAM usage: 972.02MB   Flash attention: enabled

D:\RU\mariadb-vss-chinese>

```

Suggestion from AI

Your GPU can handle models up to around 15GB VRAM usage, but for efficient performance, models should stay below 80% of total VRAM (~12GB). If you want bigger models, you may need a GPU with more memory or offload layers to CPU RAM (slower but possible).

1. ggml-model-q2_k.gguf (2.73G)
2. ggml-model-q4_k.gguf (3.92G)
3. ggml-model-q8_0.gguf (6.85G)
4. ggml-model-f16.gguf (12.9G)

To strike a balance between speed and accuracy, I opt `ggml-model-q4_k.gguf` (3.92G) instead of `ggml-model-q8_0.gguf` (6.85G). A short comparison is in [here](#). Both of them generate vector of 4096 dimensions.

I have the AI generated 1,200 sentences in traditional Chinese and put it in an array for sake of simplicity. A more realistic scenario would load the data from [Oracle](#) or by parsing a [CSV](#) file, for example.

It is *crucial* to choose the right model in the first place, change of model involves re-creating all vector embeddings may take hours or even days.

II. Using Embedding

First of all, create a table in target database:

```
USE vss;

CREATE TABLE documents (
  id          INT AUTO_INCREMENT PRIMARY KEY,

  textChi     VARCHAR(512) NOT NULL,
  visited     INT DEFAULT 0,
  embedding    VECTOR(4096) NOT NULL,

  createdAt   TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  updatedAt   TIMESTAMP,
  updateIdent INT DEFAULT 0,

  UNIQUE (textChi),
  VECTOR INDEX (embedding) M=16 DISTANCE=cosine
);
```

Note:

1. Number of dimensions is determined by model output;
2. Only one vector index per table and can not be NULL;
3. `M` — Larger values mean slower SELECTs and INSERTs, larger index size and higher memory consumption but more accurate results. The valid range is from `3` to `200`.
4. `DISTANCE` — Distance function to build the vector index for. Searches using a different distance function will not be able to use a vector index. Valid values are `cosine` and `euclidean` (the default).

Next, install Prisma, pull in model and generate client code.

```
model documents {
  id          Int           @id @default(autoincrement())
  textChi     String        @unique(map: "textChi")
  @db.VarChar(512)
  visited     Int?          @default(0)
  embedding    Unsupported("vector(4096)")
  createdAt   DateTime?     @default(now())
  @db.Timestamp(0)
  updatedAt   DateTime?     @db.Timestamp(0)
  updateIdent Int?          @default(0)

  @@index([embedding], map: "embedding")
}
```

The template to generate vector embedding is from [here](#):



```
import {fileURLToPath} from "url";
import path from "path";
import {getLlama, LlamaEmbedding} from "node-llama-cpp";

const __dirname = path.dirname(
  fileURLToPath(import.meta.url)
);

const llama = await getLlama();
const model = await llama.loadModel({
  modelPath: path.join(__dirname, "bge-small-en-v1.5-q8_0.gguf")
});
const context = await model.createEmbeddingContext();

async function embedDocuments(documents: readonly string[]) {
  const embeddings = new Map<string, LlamaEmbedding>();

  await Promise.all(
    documents.map(async (document) => {
      const embedding = await context.getEmbeddingFor(document);
      embeddings.set(document, embedding);

      console.debug(
        `${embeddings.size}/${documents.length} documents
embedded`
      );
    })
  );

  return embeddings;
}

function findSimilarDocuments(
  embedding: LlamaEmbedding,
  documentEmbeddings: Map<string, LlamaEmbedding>
) {
  const similarities = new Map<string, number>();
  for (const [otherDocument, otherDocumentEmbedding] of
documentEmbeddings)
    similarities.set(
      otherDocument,
      embedding.calculateCosineSimilarity(otherDocumentEmbedding)
    );

  return Array.from(similarities.keys())
    .sort((a, b) => similarities.get(b)! - similarities.get(a)!);
}

const documentEmbeddings = await embedDocuments([
  "The sky is clear and blue today",
  "I love eating pizza with extra cheese",
  "Dogs love to play fetch with their owners",
```



```

    "The capital of France is Paris",
    "Drinking water is important for staying hydrated",
    "Mount Everest is the tallest mountain in the world",
    "A warm cup of tea is perfect for a cold winter day",
    "Painting is a form of creative expression",
    "Not all the things that shine are made of gold",
    "Cleaning the house is a good way to keep it tidy"
  ]);

const query = "What is the tallest mountain on Earth?";
const queryEmbedding = await context.getEmbeddingFor(query);

const similarDocuments = findSimilarDocuments(
  queryEmbedding,
  documentEmbeddings
);
const topSimilarDocument = similarDocuments[0];

console.log("query:", query);
console.log("Document:", topSimilarDocument);

```

Which is a good start!

III. Seeding

A cheap trick is used to implement `UPSERT` in MariaDB, which is described in [INSERT ON DUPLICATE KEY UPDATE](#). An `UPSERT` prevents from inserting duplicated entry and let's keep track of the duplication.

```

export async function addDocument(document) {
  const { vector } = await
context.getEmbeddingFor(removeWords(document));

  // Add new document
  return await prisma.$executeRaw`
      INSERT INTO documents (textChi, embedding)
      VALUES( ${document},
      VEC_FromText(${JSON.stringify(vector)}) )
      ON DUPLICATE KEY
      UPDATE updateIdent = updateIdent + 1;
    `;
}

```

Run command to seed database:

```
npx prisma db seed
```



```

C:\WINDOWS\system32\cmd. x + v
D:\RU\mariadb-vss-chinese>npx prisma db seed
Environment variables loaded from .env
Running seed command `node prisma/seed.js` ...
[node-llama-cpp] load: special_eos_id is not in special_eog_ids - the tokenizer config may be incorrect
Document 1: 今天的天空晴朗且蔚藍
Document 2: 我喜歡吃加了額外起司的披薩
Document 3: 狗狗喜歡和主人玩接球遊戲
Document 4: 法國的首都是巴黎
Document 5: 喝水對保持身體水分很重要
Document 6: 聖母峰是世界上最高的山
Document 7: 寒冷的冬天來一杯溫暖的茶最合適
Document 8: 繪畫是一種創意表達的方式
Document 9: 並非所有閃亮的東西都是黃金製成
Document 10: 打掃房子是保持整潔的好方法
Document 11: 早晨的陽光透過窗戶照進房間
Document 12: 雨後的空氣格外清新
Document 13: 夜晚的星空閃爍著微光
Document 14: 我喜歡在春天欣賞盛開的櫻花
Document 15: 閱讀一本好書能讓人心靈沉靜
Document 16: 貓咪喜歡窩在陽光下打盹
Document 17: 音樂能夠療癒人的心靈
Document 18: 旅行是探索世界的美好方式
Document 19: 咖啡的香氣讓人感覺放鬆

```

Depending on the machine, it may take hours or even days... After that verify if there is duplicated entry with:

```
SELECT * FROM documents WHERE updateIdent <> 0;
```



IV. Finding the documentns

Most of the code remains the same, we only add update to `visited` field so that we can check to see search results.

```

export async function findSimilarDocuments(document, limit = 3) {
  const { vector } = await
  context.getEmbeddingFor(removeWords(document));

  // Find similar documents
  const docs = await prisma.$queryRaw`
    SELECT textChi,
           VEC_DISTANCE_COSINE(
             embedding,
             VEC_FromText(${JSON.stringify(vector)})
           ) AS distance,
           id
    FROM documents
    ORDER BY 2 ASC
    LIMIT ${limit} OFFSET 0;
  `;

  // Update `visited` field

```



```

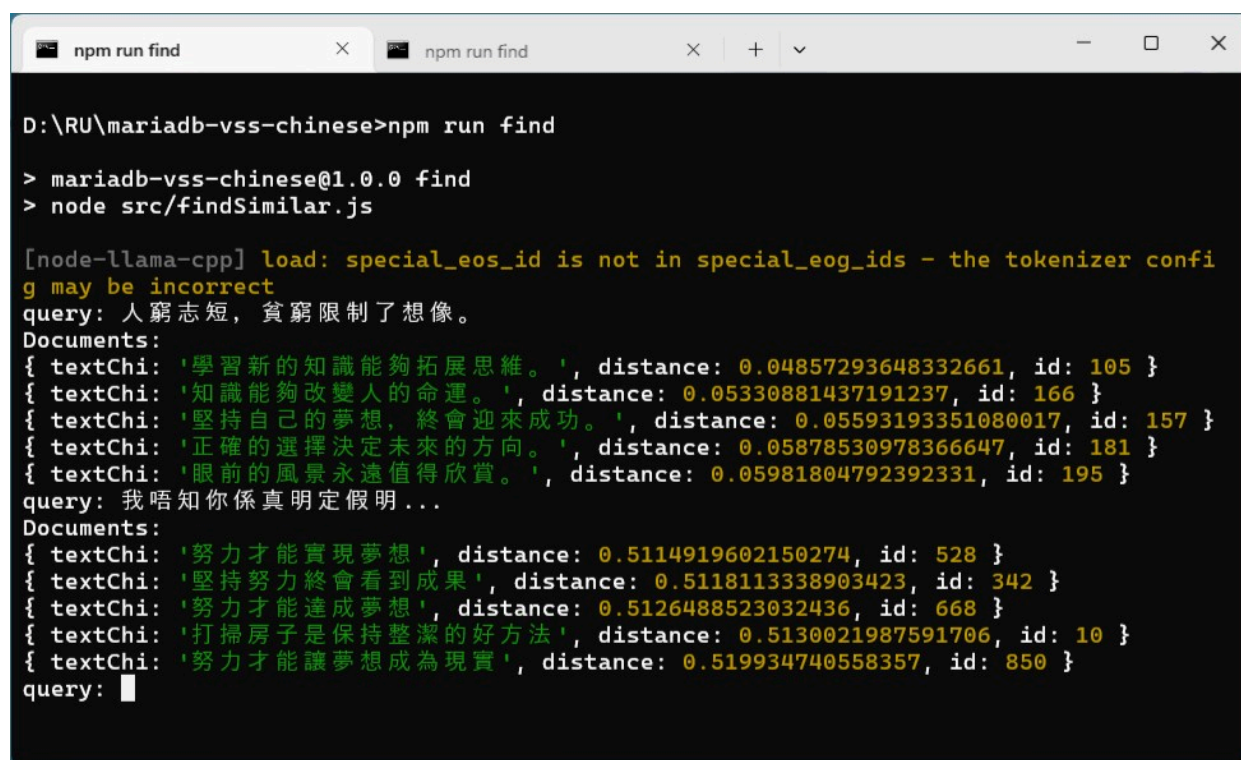
const promises = []; // Collect promises
docs.forEach(doc => {
    promises.push(prisma.$executeRaw`
        UPDATE documents
        SET visited = visited + 1,
            updatedAt = Now(),
            updateIdent = updateIdent + 1
        WHERE id=${doc.id}
    `);
})
await Promise.all(promises); // Resolve all at once

return docs
}

```

Run command to find the documents:

```
npm run find
```



```

D:\RU\mariadb-vss-chinese>npm run find

> mariadb-vss-chinese@1.0.0 find
> node src/findSimilar.js

[node-llama-cpp] load: special_eos_id is not in special_eog_ids - the tokenizer config may be incorrect
query: 人窮志短，貧窮限制了想像。
Documents:
{ textChi: '學習新的知識能夠拓展思維。', distance: 0.04857293648332661, id: 105 }
{ textChi: '知識能夠改變人的命運。', distance: 0.05330881437191237, id: 166 }
{ textChi: '堅持自己的夢想，終會迎來成功。', distance: 0.05593193351080017, id: 157 }
{ textChi: '正確的選擇決定未來的方向。', distance: 0.05878530978366647, id: 181 }
{ textChi: '眼前的風景永遠值得欣賞。', distance: 0.05981804792392331, id: 195 }
query: 我唔知你係真明定假明...
Documents:
{ textChi: '努力才能實現夢想', distance: 0.5114919602150274, id: 528 }
{ textChi: '堅持努力終會看到成果', distance: 0.5118113338903423, id: 342 }
{ textChi: '努力才能達成夢想', distance: 0.5126488523032436, id: 668 }
{ textChi: '打掃房子是保持整潔的好方法', distance: 0.5130021987591706, id: 10 }
{ textChi: '努力才能讓夢想成為現實', distance: 0.519934740558357, id: 850 }
query: 

```

Verify search results with:

```
SELECT * FROM documents WHERE visited <> 0;
```

documents 1 x

SELECT * FROM documents WHERE visited <| Enter a SQL expression to filter results (use Ctrl+Space)

Grid	id	textChi	visite	embedding	createdAt	updatedAt	ui
1	10	打掃房子是保持整潔的好方法	1	â =ÖL ÿMIN> s4?ü_K¼áÜ ÿ N-?§ü³>.	2025-06-18 16:42:06.000	2025-06-19 12:44:47.000	1
2	105	學習新的知識能夠拓展思維。	2	âlw¼ DÉ¼ \$ª?Ç >X Á? :Ç¼d ¶¼øÉÉ?..	2025-06-18 16:57:24.000	2025-06-19 12:44:22.000	2
3	157	堅持自己的夢想，終會迎來成功。	2	ð=miÖ<¼J&?Ö»¹ ·>²Á çýüè>à:ü?..	2025-06-18 17:05:40.000	2025-06-19 12:44:22.000	2
4	166	知識能夠改變人的命運。	2	˘ ð>£K > ©t?µ ÿ>øuh?j «¼ x¼>@'Ø?	2025-06-18 17:07:05.000	2025-06-19 12:44:22.000	2
5	181	正確的選擇決定未來的方向。	2	'ää=e>r= P?Ä ¼¼:¼s? É ¼Ö5 ?g ¥?...	2025-06-18 17:09:26.000	2025-06-19 12:44:22.000	2
6	195	眼前的風景永遠值得欣賞。	2	Ék °mA ½Äè ?² ¼kÁ ?¼¶?Ö Æ¼x ú?..	2025-06-18 17:11:38.000	2025-06-19 12:44:22.000	2
7	342	堅持努力終會看到成果	1	¼¼¼ ¼é °¼Y¼X¼¼"YÖ? £6¼ w~¼j"y? 7 ¼.	2025-06-18 17:33:07.000	2025-06-19 12:44:47.000	1
8	528	努力才能實現夢想	1	ÜGä½c@¶¼Ä ð>O p? j8¼i²¼t \$K?hs ¼	2025-06-18 17:59:55.000	2025-06-19 12:44:47.000	1
9	668	努力才能達成夢想	1	· ¼ E®¼ i>ç« @¼ l¼= ¼ll?Ä½;...	2025-06-18 18:20:36.000	2025-06-19 12:44:47.000	1
10	850	努力才能讓夢想成為現實	1	â ä¼gÜµ¼ ä ?i-a? ?º¼ x¼ ð>ðÉ«>...	2025-06-18 18:48:48.000	2025-06-19 12:44:47.000	1

Refresh Save Cancel Export data 9999 10 10 row(s) fetched - 0.006s, on 2025-06-19

V. Bibliography

- 1. [Choosing a Model](#)
- 2. [Using Embedding](#)
- 3. [Raw queries](#)
- 4. [CRUD](#)
- 5. [The Castle by Franz Kafka](#)

Epilogue

"I am certainly ignorant, but facts are facts, which is very sad for me but also advantageous, since an ignorant man will dare to do more, so I will happily go about in my ignorance with what I am sure are its unfortunate consequences for a little longer, as long as my strength allows. "

The Castle by Franz Kafka

EOF (2025/06/20)