DHBW
Duale Hochschule
Baden-Württemberg

Datensammlung und Datenstruktur

Dokumentation Big Data Projekt

des Studienganges Informatik

an der Dualen Hochschule Baden-Württemberg Stuttgart

von

Luca Holder

22.11.2025

Bearbeitungszeitraum	04.11.2024-22.11.2024
Matrikelnummern, Kurs	5234642, TINF22C
Gutachter*in der Dualen Hochschule	Marcel Mittelstädt

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Datensammlung

Dieses Kapitel beschreibt die Herangehensweise der Datensammlung.

Die benötigten Daten lassen sich mit der API „Magic: The Gathering API“ abrufen, die Informationen über die Karten und Sets von dem Fantasy-Kartenspiel „Magic: The Gathering“. (vgl. [1])

1.1 Datenstruktur

Die Daten sind nach dem collection_job_mtg im JSON-Format verfügbar. Sie liegen somit als Sem-strukturierte Daten vor, da das JSON-Format zwar Struktur bietet, nicht aber auf der Ebene von strukturierten Daten wie Tabellen. Die genaue Struktur ist in der folgenden Abbildung sichtbar.


```
{
    "card": {
      "name": "Ankh of Mishra",
      "manaCost": "{2}",
      "cmc": 2.0,
      "type": "Artifact",
      "types": [
        "Artifact"
      ],
      "rarity": "Rare",
      "set": "LEA",
      "setName": "Limited Edition Alpha",
      "text": "Whenever a land enters the battlefield, Ankh of Mishra deals 2 damage to that land's controller.",
      "artist": "Amy Weber",
      "number": "230",
      "layout": "normal",
      "multiverseid": "1",
      "imageUrl": "http://gatherer.wizards.com/Handlers/Image.ashx?multiverseid=1&type=card",
      "rulings": [
        {
          "date": "2004-10-04",
          "text": "It determines the land’s controller at the time the ability resolves. If the land leaves the
battlefield before the ability resolves, the land’s last controller before it left is used."
        },
        {
          "date": "2004-10-04",
          "text": "This triggers on any land entering the battlefield. This includes playing a land or putting a land onto
the battlefield using a spell or ability."
        }
      ],
      "printings": [
        "2ED",
        "30A",
        "3ED",
        "4BB",
        "4ED",
        "5ED",
        "6ED",
        "CED",
        "CEI",
        "FBB",
        "LEA",
        "LEB",
        "ME1",
        "SUM",
```

```
      "VMA"
    ],
    "originalText": "Ankh does 2 damage to anyone who puts a new land into play.",
    "originalType": "Continuous Artifact",
    "legalities": [
      {
        "format": "Commander",
        "legality": "Legal"
      },
      {
        "format": "Duel",
        "legality": "Legal"
      },
      {
        "format": "Legacy",
        "legality": "Legal"
      },
      {
        "format": "Oathbreaker",
        "legality": "Legal"
      },
      {
        "format": "Oldschool",
        "legality": "Legal"
      },
      {
        "format": "Predh",
        "legality": "Legal"
      },
      {
        "format": "Premodern",
        "legality": "Legal"
      },
      {
        "format": "Vintage",
        "legality": "Legal"
      }
    ],
    "id": "33b9ca30-0296-52b7-a8e2-7d4715404b0d"
  }
}
```


Abbildung 1: eigene Abbildung, Datenstruktur der Karten

Die Attribute sind dabei, wie die nachfolgende Abbildung zeigt: name, layout, mana_cost, cmc, colors, color_identity, names, type, supertypes, subtypes, types, rarity, text, flavor, artist, number, power, toughness, loyalty, multiverse_id, variations, watermark, border, timeshifted, hand, life, release_date, starter, printings, original_text, original_type, source, image_url, set, set_name, id, legalities, rulings, foreign_names.

```
class Card(object):
    RESOURCE = 'cards'

    def __init__(self, response_dict={}):
        self.name = response_dict.get('name')
        self.layout = response_dict.get('layout')
        self.mana_cost = response_dict.get('manaCost')
        self.cmc = response_dict.get('cmc')
        self.colors = response_dict.get('colors')
        self.color_identity = response_dict.get('colorIdentity')
        self.names = response_dict.get('names')
        self.type = response_dict.get('type')
        self.supertypes = response_dict.get('supertypes')
        self.subtypes = response_dict.get('subtypes')
        self.types = response_dict.get('types')
        self.rarity = response_dict.get('rarity')
        self.text = response_dict.get('text')
        self.flavor = response_dict.get('flavor')
        self.artist = response_dict.get('artist')
        self.number = response_dict.get('number')
        self.power = response_dict.get('power')
        self.toughness = response_dict.get('toughness')
        self.loyalty = response_dict.get('loyalty')
        self.multiverse_id = response_dict.get('multiverseid')
        self.variations = response_dict.get('variations')
        self.watermark = response_dict.get('watermark')
        self.border = response_dict.get('border')
        self.timeshifted = response_dict.get('timeshifted')
        self.hand = response_dict.get('hand')
        self.life = response_dict.get('life')
        self.release_date = response_dict.get('releaseDate')
        self.starter = response_dict.get('starter')
        self.printings = response_dict.get('printings')
        self.original_text = response_dict.get('originalText')
        self.original_type = response_dict.get('originalType')
        self.source = response_dict.get('source')
        self.image_url = response_dict.get('imageUrl')
        self.set = response_dict.get('set')
        self.set_name = response_dict.get('setName')
        self.id = response_dict.get('id')
        self.legalities = response_dict.get('legalities')
        self.rulings = response_dict.get('rulings')
        self.foreign_names = response_dict.get('foreignNames')
```

Abbildung 2: eigene Abbildung, Attribute der MTG-Daten

1.2 Datenabfrage

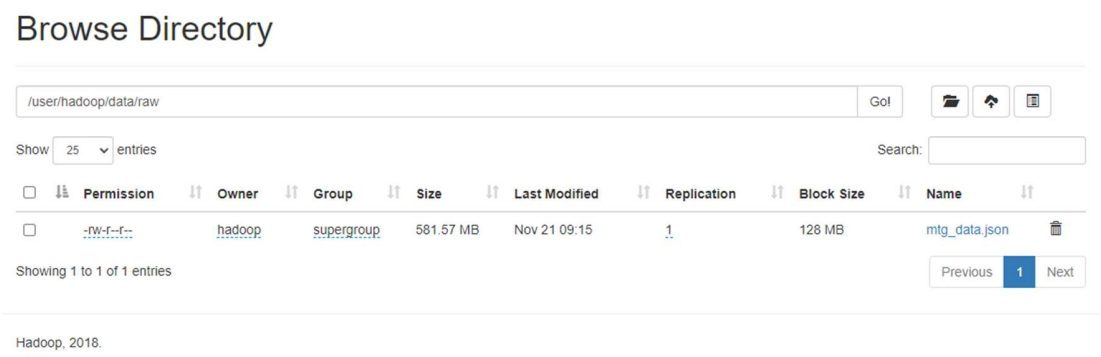
Die Dokumentation der API weist auf SDKs hin, die die Datenabfragen vereinfachen sollen. Zudem ist die Datenabfrage über die API auf maximale 5000 Abfragen pro Stunde sowie auf maximal 100 Anfragen pro 100 Karten eingeschränkt wird. (vgl. [1]) Aus diesem Grund wird eine SDK zur Abfrage verwendet.

Erstellt wird die Abfrage in Python, dabei wird das Python SDK verwendet, dass mit dem Befehl „pip install mtgsdk“ installiert werden kann.

```
All card data has been loaded in 384.38 seconds.
Time since last heartbeat(0.03 s) < heartrate(5.0 s), sleeping for 4.972679 s
Time since last heartbeat(0.05 s) < heartrate(5.0 s), sleeping for 4.952313 s
Time since last heartbeat(0.03 s) < heartrate(5.0 s), sleeping for 4.974349 s
Time since last heartbeat(0.02 s) < heartrate(5.0 s), sleeping for 4.984048 s
Time since last heartbeat(0.03 s) < heartrate(5.0 s), sleeping for 4.967826 s
Time since last heartbeat(0.02 s) < heartrate(5.0 s), sleeping for 4.98203 s
All card data has been saved as JSON to /home/airflow/raw_data/mtg_data.json in 29.71 seconds.
Time since last heartbeat(0.03 s) < heartrate(5.0 s), sleeping for 4.972811 s
Deleted /user/hadoop/data/raw/mtg_data.json
Old data deleted from HDFS raw directory.
Time since last heartbeat(0.03 s) < heartrate(5.0 s), sleeping for 4.972612 s
Data successfully uploaded to HDFS at /user/hadoop/data/raw/mtg_data.json.
All card data has been saved as JSON to HDFS in 6.81 seconds.
```

Abbildung 3: Screenshot, Konsolenausgabe der Ausführungszeiten

Bei der Abfrage aller Karten (durchgeführt am 21. November 2024 auf einer Google VM → n2-standard-2) betrug die Ausführungszeit für die Abfrage aller Karten 384,38 Sekunden, das ungefähr 6min und 40 Sekunden entspricht. Das Überführen der Daten in JSON dauerte ca. 30 Sekunden. Die Speicherung der JSON-Datei in den JDFS dauerte ca. 7 Sekunden.



Die Datei hatte am Ende eine Größe von Size: 581.57 MB und 93.643 Karten.

1.3 Automatisierung der Datensammlung

Die Datensammlung wird mit einem Python-Skript durchgeführt. Dieser Job läuft unter dem Namen collect_job_mtg und ruft alle Karten der API über die Python SDK auf und speichert diese als JSON-Datei ab. Anschließend wird über das Skript die Datei in den HDFS in das Verzeichnis „raw“ als JSON-Datei gespeichert, wie die Prüfungsleistung erfordert.

Literaturverzeichnis

[1] „MTG API Docs“, Magic: The Gathering API Documentation. Zugegriffen: 5. November 2024. [Online]. Verfügbar unter: https://docs.magicthegathering.io/