Marcin Szwed

Projekt - Movies

08.06.2022

Wstęp

Celem projektu było utworzenie aplikacji pozwalającej na wyszukiwanie informacji o filmach oraz osobach z nimi związanych zebranych w lokalnej bazie danych z możliwością interpretacji zapytań sformułowanych w języku naturalnym.
Zawartość danych odpowiada części bazy The Movie Database (https://www.themoviedb.org/?language=pl).

Podstawowe funkcje aplikacji to:

- ładowanie danych z bazy danych TMDB
- wyszukiwanie filmów i osób na podstawie różnych zdefiniowanych kryteriów oraz zapytań sformułowanych w języku naturalnym

Jednym z założeń było także wyodrębnienie z tekstów (recenzje filmów, opis fabuły filmów) tzw. Named Entities, czyli rozpoznawanych przez narzędzia NLP terminów wraz z przypisaną kategorią, np. (Batman, PERSON), (Google, ORG).

Technologie

1. W projekcie wykorzystano relacyjną bazę danych PostgreSQL.
2. Aplikacja została zaimplementowana w języku Python.
3. W celu dostępu do danych i manipulacji danymi wykorzystano bibliotekę ORM SQLAlchemy
4. Do wyodrębniania named entities oraz rozpoznawania fraz języka naturalnego użyto biblioteki Spacy
5. Interfejs zbudowano z zastosowaniem platformy Dash.

Źródło danych

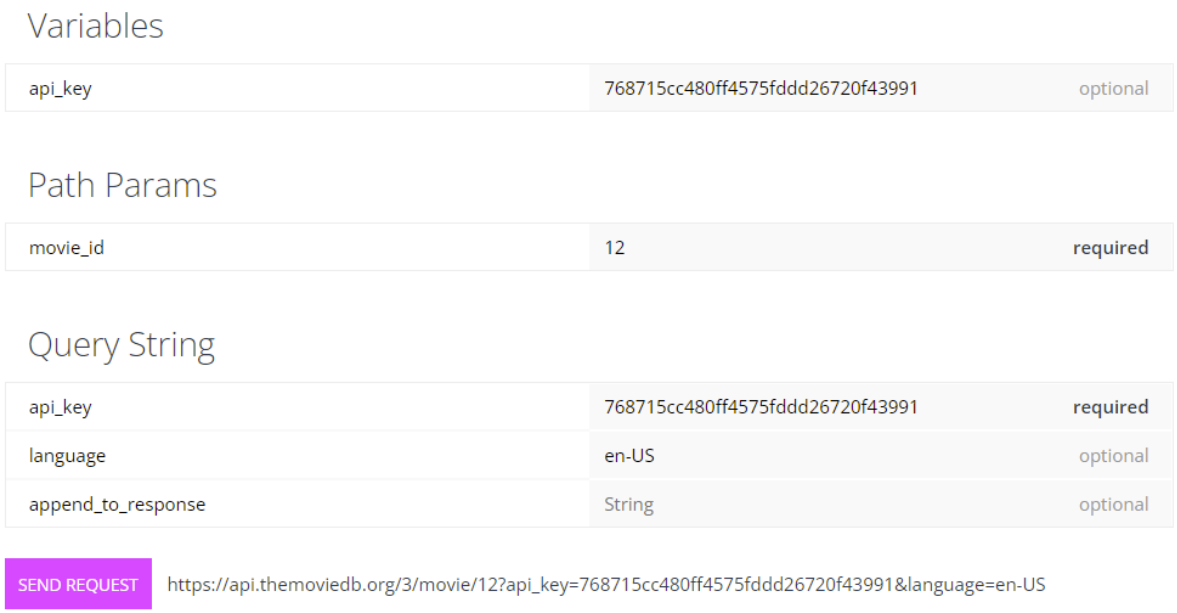
Źródłem danych jest TMDB API (https://developers.themoviedb.org/3/getting-started/introduction). W celu jego wykorzystania należy zarejestrować się na stronie. Następnie otrzymuje się klucz, który jest później używany podczas wykonania kwerend. Interfejs TMDB API jest interfejsem RESTowym i zwraca wyniki w formacie JSON.

Na rys.1 przedstawiono interfejs pozwalający na testowanie zapytań (przycisk na stronie nie działa), a na rysunku 2 sformatowany przykład odpowiedzi systemu.

W aplikacji wykorzystano następujące zapytania:

- dane o indywidualnych filmach
- dane o indywidualnych osobach
- dane credits - zbiór powiązań między filmem, a osobami. Jako jedynie zapytanie zwracane są wszystkie powiązania dla danego filmu (rys.3)
- informacje o firmach produkujących film companies
- lista recenzji dla danego filmu.

Poza credits i recenzjami nie jest możliwe pobranie informacji o grupie elementów, np. grupie filmów. Podczas zapytania należy dostarczyć ID elementu. Na stronie TMDB można znaleźć także uaktualniane codziennie listy ID dla poszczególnych obiektów. W bazie znajduje się około 700000 filmów i około 2.4 mln osób.



Rys. 1 - interfejs TMDB API umożliwiający wysyłanie zapytań

```
{
   "adult":false,
   "backdrop_path":"/sWWwq8MsfI3Zrn4Zz0HENnhp70J.jpg",
   "belongs_to_collection":{
      "id":137697,
      "name":"Finding Nemo Collection",
      "poster_path":"/xwggrEugjcJDuabIWvK2CpmK91z.jpg",
      "backdrop_path":"/2hC8HHRUvwRljYKIcQDMyMbLlxz.jpg"
   },
   "budget":94000000,
   "genres":[
      {
         "id":16,
         "name":"Animation"
      },
      {
         "id":10751,
         "name":"Family"
      }
   ],
   "homepage":"http://movies.disney.com/finding-nemo",
   "id":12,
   "imdb_id":"tt0266543",
   "original_language":"en",
   "original_title":"Finding Nemo",
   "overview":"Nemo, an adventurous young clownfish, is unexpectedly taken from his Great Barri
   "popularity":113.343,
   "poster_path":"/eHuGQ10FUzK1mdOY69wF5pGgEf5.jpg",
   "production_companies":[
      {
         "id":3,
         "logo_path":"/1TjvGVDMYsj6JBxOAkUHpPEwLf7.png",
         "name":"Pixar",
         "origin_country":"US"
      }
   ],
   "production_countries":[
      {
         "iso_3166_1":"US",
         "name":"United States of America"
      }
   ],

   "release_date":"2003-05-30",
   "revenue":940335536,
   "runtime":100,
   "spoken_languages":[
      {
         "english_name":"English",
         "iso_639_1":"en",
         "name":"English"
      }
   ],
   "status":"Released",
   "tagline":"There are 3.7 trillion fish in the ocean. They're looking for one.",
   "title":"Finding Nemo",
   "video":false,
   "vote_average":7.8,
   "vote_count":16382
}
```


Rys. 2 - I część przykładowego wyniku zapytania (dla filmu o ID 12)

```
{
  "id":12,
  "cast":[
    {
      "adult":false,
      "gender":2,
      "id":13,
      "known_for_department":"Acting",
      "name":"Albert Brooks",
      "original_name":"Albert Brooks",
      "popularity":8.761,
      "profile_path":"/8iDSGu5l93N7benjf6b3AysBore.jpg",
      "cast_id":8,
      "character":"Marlin (voice)",
      "credit_id":"52fe420ec3a36847f8000679",
      "order":0
    },
    {
      "adult":false,
      "gender":1,
      "id":14,
      "known_for_department":"Acting",
      "name":"Ellen DeGeneres",
      "original_name":"Ellen DeGeneres",
      "popularity":4.516,
      "profile_path":"/z8IEEid4z63CBlJtxrTKEfsW7NA.jpg",
      "cast_id":9,
      "character":"Dory (voice)",
      "credit_id":"52fe420ec3a36847f800067d",
      "order":1
    },
    {
      "adult":false,
      "gender":2,
      "id":12,
      "known_for_department":"Acting",
      "name":"Alexander Gould",
      "original_name":"Alexander Gould",
      "popularity":10.005,
      "profile_path":"/fe4mUSp0XotA6Ku4Bs69Q9o2lqU.jpg",
      "cast_id":7,
      "character":"Nemo (voice)",
      "credit_id":"52fe420ec3a36847f8000675",
      "order":2
    },
```


Rys. 3 - fragment rezultatu zapytania credits dla filmu o ID 12

Projekt bazy danych

Zaprojektowano relacyjną bazę danych po analizie zawartości zwracanych plików JSON. Należy zwrócić uwagę, że pliki JSON zawierają częściowe informacje o powiązanych obiektach, np. w przypadku filmów dołączone są gatunki o gatunkach i firmach producentów. Natomiast w przypadku credits (powiązania między osobami i filmami) informacje o popularności osób.

Na rysunku 4 przedstawiono diagram bazy danych.
Podstawowe tabele to:

- Movies
- People
- Credits
- Credit_types - określa rolę osoby: cast - aktor, crew - członek ekipy produkującej film
- Credit_departments - określa dział przy któreym dana osoba pracuje (np. art, sound, production, writing, directing)
- Credit_jobs - określa pełnioną funkcję (np. director, costume design, sound assistant)
- Genres
- Companies
- Countries
- Named_entities
- Reviews
- Keywords
- oraz wiele tabel asocjatywnych.

Należy zwrócić uwagę, że w pierwszej wersji bazy danych kluczem głównym dla credit types, departments i jobs była liczba całkowita i dodano ograniczenie unique na wartość, ale ze względu na ograniczenia SQLAlchemy zdecydowano usunąć wartość całkowitą jako element tabeli, stąd kluczem głównym jest tekst, a kluczem obcym w tabeli credits jest ten sam tekst.

Tabela Credits zawiera połączone dane dwóch klas cast i crew różniących się polami. Z tego powodu w zależności od klasy niektóre z nich mogą mieć wartość NULL. Implementacja odpowiada więc realizacji dziedziczenia typu *Single Table*.

Przy projektowaniu bazy danych zdefiniowano szereg indeksów, były to klucze obce, oraz sprowadzone do małych liter wartości tekstowe. Definicje tabel oraz indeksów są dostępne w pliku creatable.sql.

Rys. 4 - diagram bazy danych (dostępny także jako plik diagram.png)

Implementacja

Implementacja została zrealizowana w czterech etapach.

1. Zaimplementowano klasy w języku Python odpowiadających istniejącym tabelom w bazie danych.
2. Napisano funkcje, które pobierają informacje z TMDB API w postaci plików JSONa i zapisują w bazie danych.
3. Napisano kwerendy umożliwiające odczyt zawartości bazy danych.
4. Napisano prosty interfejs webowy.

Implementacja klas

Zaimplementowano klasy posługując się typowymi wzorcami dla SQLAlchemy. Np. dla tabeli Movies zdefiniowanej jak na rys. 5 odpowiadająca jej definicja klasy na rsy. 6.

W kodzie widoczne są także powiązania (relationship), najczęściej są to powiązania asocjatywne typu many to many.

```sql
CREATE TABLE Movies
(
    tmdb_id bigint PRIMARY KEY,
    imdb_id text,
    budget numeric,
    homepage text,
    original_language text,
    original_title text,
    overview text,
    imdb_popularity numeric,
    poster_path text,
    release_date date,
    revenue numeric,
    runtime bigint,
    status text,
    tagline text,
    title text,
    vote_average numeric,
    vote_count int
);
```

rys. 5 - create table movies

```python
class Movie(Base):
    __tablename__="movies"
    # __table_args__ = {'extend_existing': True}
    tmdb_id=Column(Integer, primary_key=True)
    imdb_id=Column(String)
    budget=Column(Integer)
    homepage=Column(String)
    original_language=Column(String)
    original_title=Column(String)
    overview=Column(String)
    imdb_popularity=Column(Numeric)
    poster_path=Column(String)
    release_date=Column(String)
    revenue=Column(Integer)
    runtime=Column(Integer)
    status=Column(String)
    tagline=Column(String)
    title=Column(String)
    vote_average=Column(Numeric)
    vote_count=Column(Numeric)
    genres=relationship("Genre",
secondary=movies_to_genres_association_table,
back_populates="movies")
    countries = relationship("Country",
secondary=movies_to_countries_association_table,
back_populates="movies")
    companies = relationship("Company",
secondary=movies_to_companies_association_table,
back_populates="movies")
```

```
    named_entities = relationship("NamedEntity",
secondary='movies_to_named_entities', back_populates="movies")
    credits= relationship("Credit", back_populates="movies")
    keywords = relationship("Keyword",
secondary=keywords_to_movies_association_table,
back_populates="movies")
    reviews = relationship("Review", back_populates="movies")
```

Rys. 6 - kod klasy Movie

Ładowanie danych

Dane ładowano w następującej kolejności:

1. Pobrano identyfikatory 30000 filmów.
2. Filmy ładowano w 100-elementowych wsadach (zaimplementowano specjalną klasę BatchGenerator). Podczas przetwarzania filmów wydzielono named entities.
3. Zapisywano je do bazy danych za pomocą funkcji SQLAlchemy merge(). Konieczne było użycie trybu merge, a nie prostego dodawania, ze względu na to, że wraz z filmami ładowały i zapisywały się informacje o gatunkach i firmach producentów.
4. Na podstawie zgromadzonych w bazie informacji o filmach załadowano i dołączono brakujące informacje o firmach (22581 firm).
5. Na podstawie zgromadzonych id filmów załadowano informacje credits (1552614).
6. Również na podstawie id filmów załadowano informacje reviews (12766). Podczas przetwarzania reviews wydzielono named entities.
7. Na podstawie załadowanych wraz z credits id osób częściowo załadowano brakujące informacje o osobach. W bazie są niepełne informacje o około 0.5 mln osobach oraz 239200 pełnych rekordów. Później po zaimplementowaniu interfejsu webowego stwierdzono, że można wykorzystać zdjęcia osób i dlatego załadowano adresy zdjęć osób.

Czas przetwarzania pojedynczego 100 elementowego wsadu wynosił od 40 do 60 s. Wydaje się, że głównym ograniczeniem jest przepustowość interfejsu TMDB API.

Funkcje dostępu do danych

Zaimplementowano około 20 kwerend realizujących dostęp do bazy.

```
def select_movie_by_id(session, tmdb_id):
    query = session.query(Movie)
    query=query.filter(Movie.tmdb_id==tmdb_id)
    result=query.all()
    return result
```

Rys. 7 - przykładowa kwerenda 1

SELECT movies.tmdb_id AS movies_tmdb_id, movies.imdb_id AS movies_imdb_id, movies.budget AS movies_budget, movies.homepage AS movies_homepage, movies.original_language AS movies_original_language, movies.original_title AS movies_original_title, movies.overview AS movies_overview, movies.imdb_popularity AS movies_imdb_popularity, movies.poster_path AS movies_poster_path, movies.release_date AS movies_release_date, movies.revenue AS movies_revenue, movies.runtime AS movies_runtime, movies.status AS movies_status, movies.tagline AS movies_tagline, movies.title AS movies_title, movies.vote_average AS movies_vote_average, movies.vote_count AS movies_vote_count
FROM movies
WHERE movies.tmdb_id = %(tmdb_id_1)s

Rys. 8 - wygenerowany automatycznie odpowiednik w języku SQL

Zestaw funkcji SQLAlchemy pozwala na dość złożone manipulacje danymi, przykładem tego może być kwerenda na rys. 9, która pozwala wyszukiwać filmy po części tytułu, oraz opcjonalnie dacie premiery w zadanym przedziale.

```
def select_movies_by_title(session, title, date_from=None, date_to=None, match_title=False):
    query = session.query(Movie)
    title=f'%{title.lower()}%'
    if match_title is True:
        query=query.filter(Movie.title.match(title))
    else:
        query=query.filter(sqlalchemy.func.lower(Movie.title).like(title))
    if date_from is not None:
        query=query.filter(Movie.release_date>=date_from)
    if(date_to is not None):
        query=query.filter(Movie.release_date<=date_to)
    result=query.order_by(Movie.vote_count.desc()).all()
    return result
```

Rys. 9 - przykładowa kwerenda 2

Przetwarzania języka naturalnego

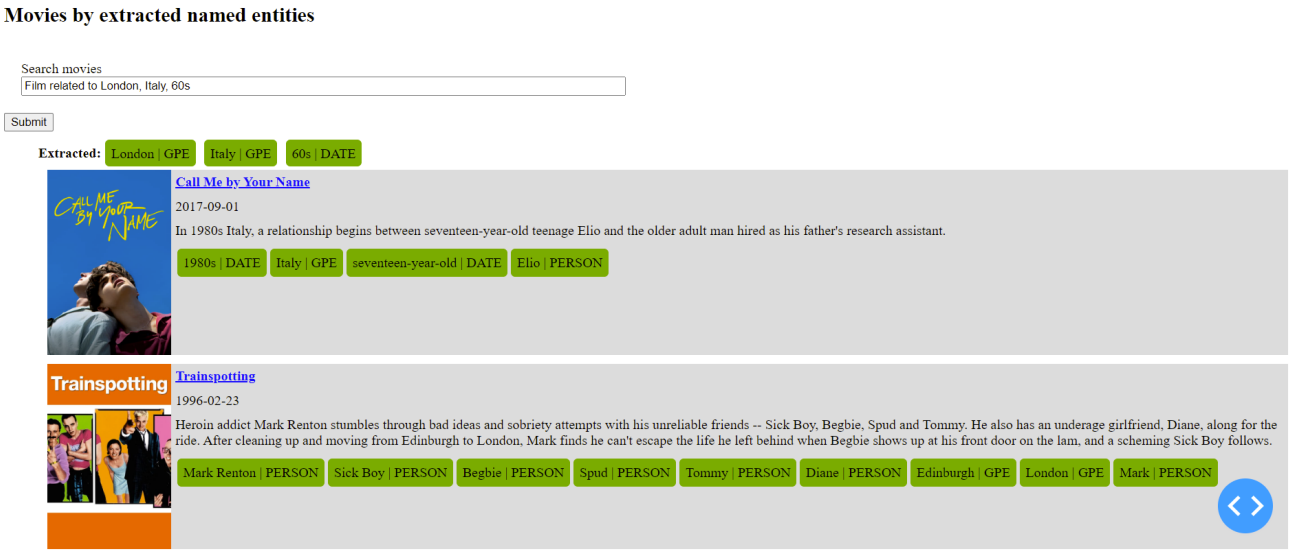
Elementami przetwarzania języka naturalnego jest identyfikacja Named Entities oraz próba dopasowania do wzorców zapytań w języku naturalnym.

Identyfikacja Named Entites

Na rysunku 10 pokazano przykład wyekstrahowanych Named Entities. Po lewej stronie podany jest tekst, a po prawej etykieta.
Named Entities były wyodrębniane zarówno z opisu fabuły filmu jak i z recenzji.
Możliwe jest wykonywanie kwerend dla których named entities są wprowadzane jawnie, albo też rozpoznawane w tekście.

236	hundreds	CARDINAL
237	European	NORP
238	Weeks	DATE
239	Fortress Camp	FAC
240	Robert	PERSON
241	Sasha	PERSON
242	Balzary	PERSON
243	Belgian	NORP
244	Hercule Poirot	PERSON
245	Nile	LOC
246	Egyptian	NORP
247	Four	CARDINAL
248	Avery	PERSON
249	Larson	PERSON

Rys. 10 - przykładowe Named Entities



Rys. 11 - named entities wyekstrahowane z tekstu oraz odpowiadające im filmy

Dopasowanie wzorców i ekstrakcja argumentów

Skorzystano z oferowanego przez bibliotekę spacy mechanizmu definiowania wzorców z tekstu, mających postać ciągu symboli, wraz z podanymi atrybutami (np. named entity, forma podstawowa wyrazu (np. forma podstawowa movies → movie, a went → go)).

Na rysunku 12 pokazano wzorzec frazy, która może zostać dopasowana do zdań:

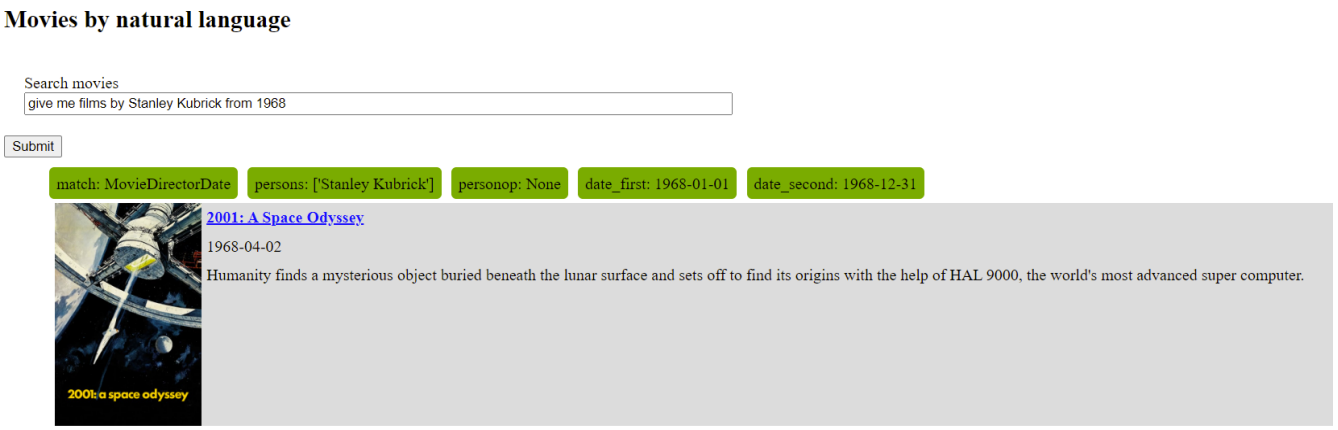
- “find movies directed by Roman Polanski in 70s”
- “give me films by Stanley Kubrick from 1968”

Dopasowanie zaczyna się od dowolnej liczby symboli, po którym pojawia się wyraz, którego formą podstawową jest movie lub film, dalej dowolna liczba symboli, opcjonalnie wyraz directed, wymagany wyraz by, named entity typu PERSON, dowolna liczba symboli, oraz data.

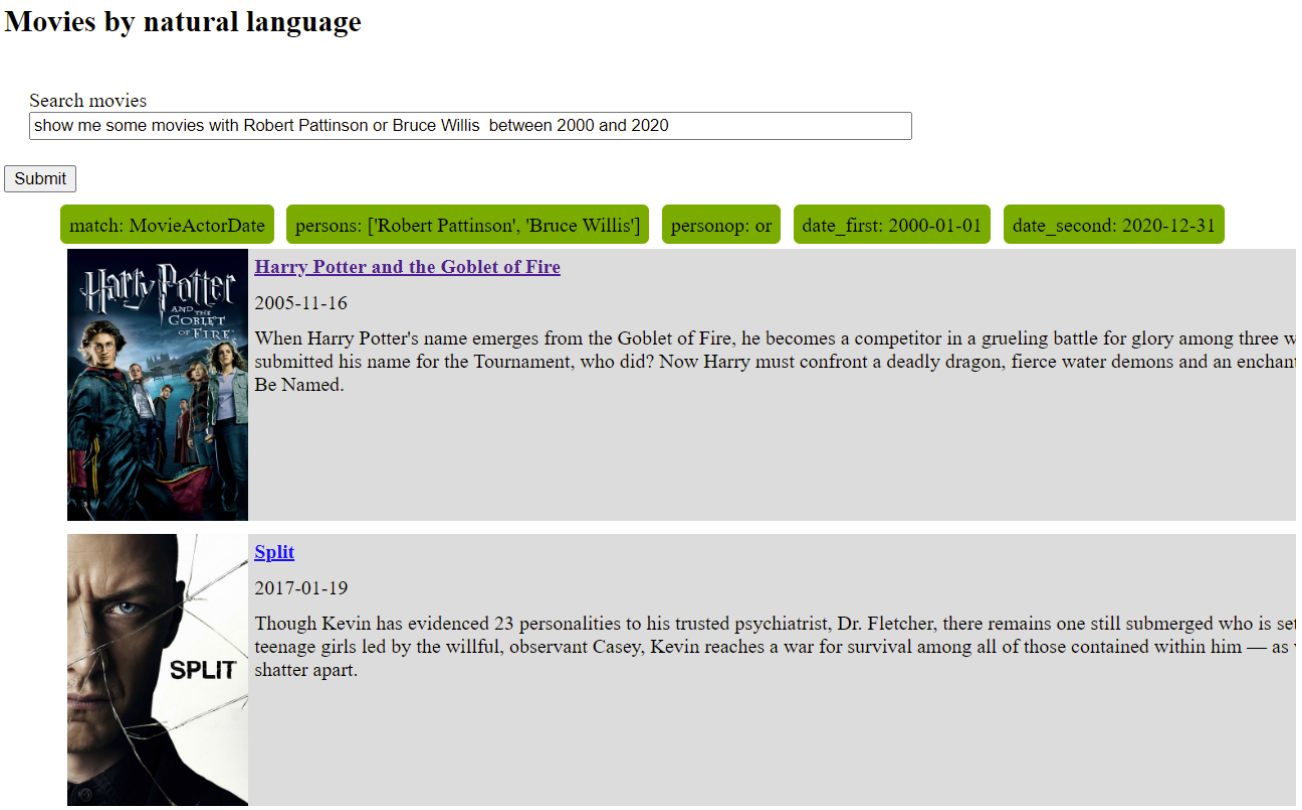
```
class MovieDirectorDate:
    def __init__(self):
        self.pattern = [[{"OP": "*"},
                         {"LEMMA": {"IN": ["movie", "film"]}},
                         {"OP": "*"},
                         {"LEMMA": "directed", "OP": "?"},
                         {"LEMMA": "by"},
                         {"ENT_TYPE": "PERSON"},
                         {"OP": "*"},
                         {"ENT_TYPE": "DATE"},
                         ],
```

Rys. 12 - wzorzec frazy

Po stwierdzeniu dopasowania (dopasowań może być więcej jeżeli wymienionych będzie np. kilka osób), ze zdania ekstrahowane są osoby oraz zakres dat. Odniesienia do czasu, takie jak 60s zamienione są na dwie daty od 01.01.1960 do 31.12.1969. Następnie wykonywana jest zidentyfikowana kwerenda.



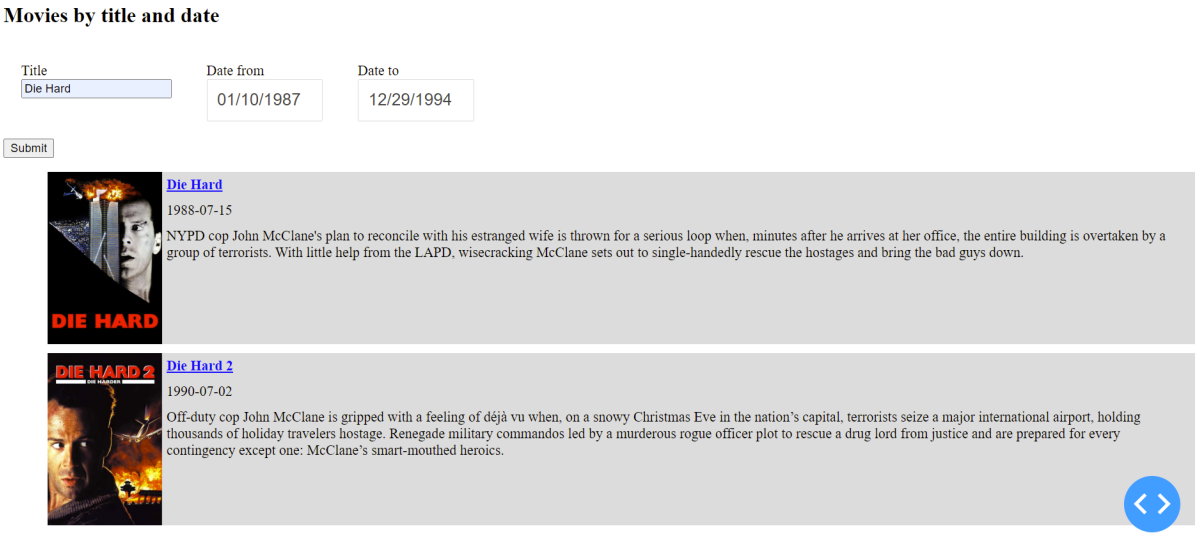
Rys. 13 - wyszukiwanie na podstawie języka naturalnego 1



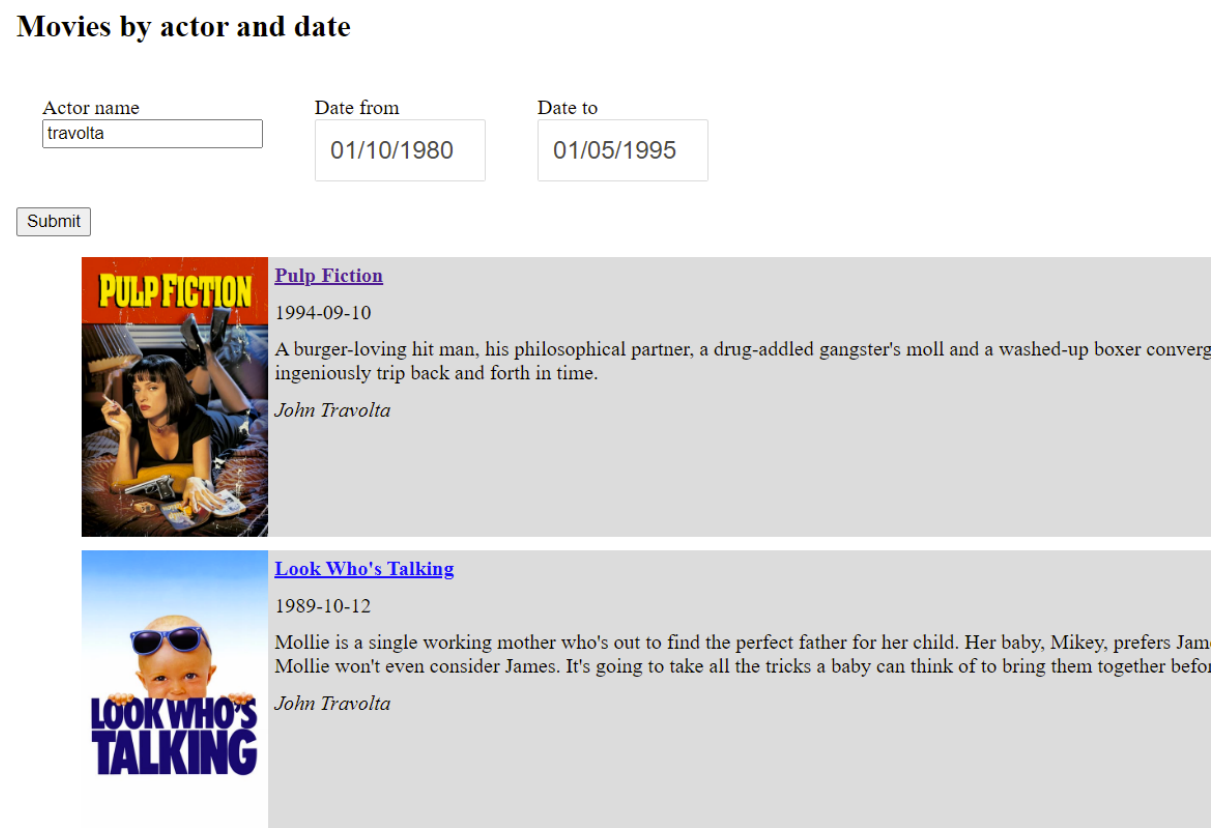
Rys. 14 - wyszukiwanie na podstawie języka naturalnego 2

Interfejs webowy

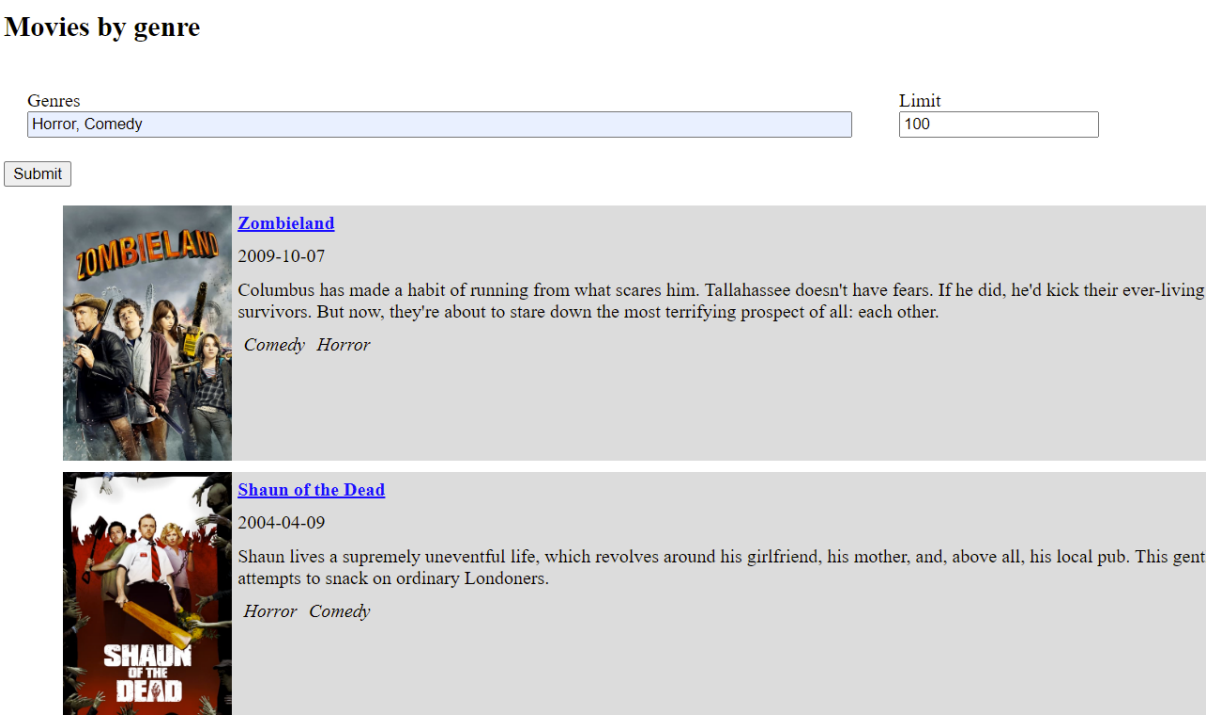
Interfejs webowy został zaimplementowany z użyciem biblioteki Dash. Na kolejnych rysunkach przedstawione zostaną przykładowe formularze do realizacji zapytań wraz z ich wynikami.



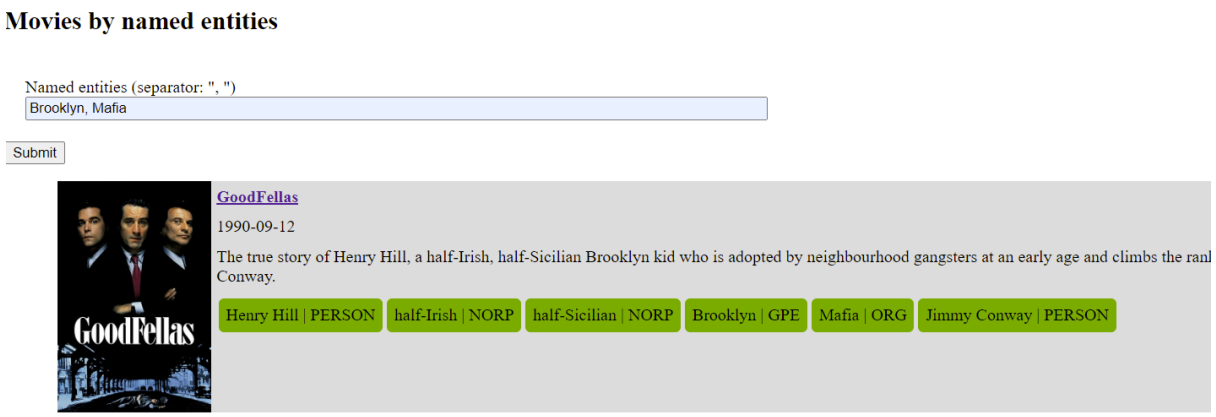
Rys. 15 - movies by title and date



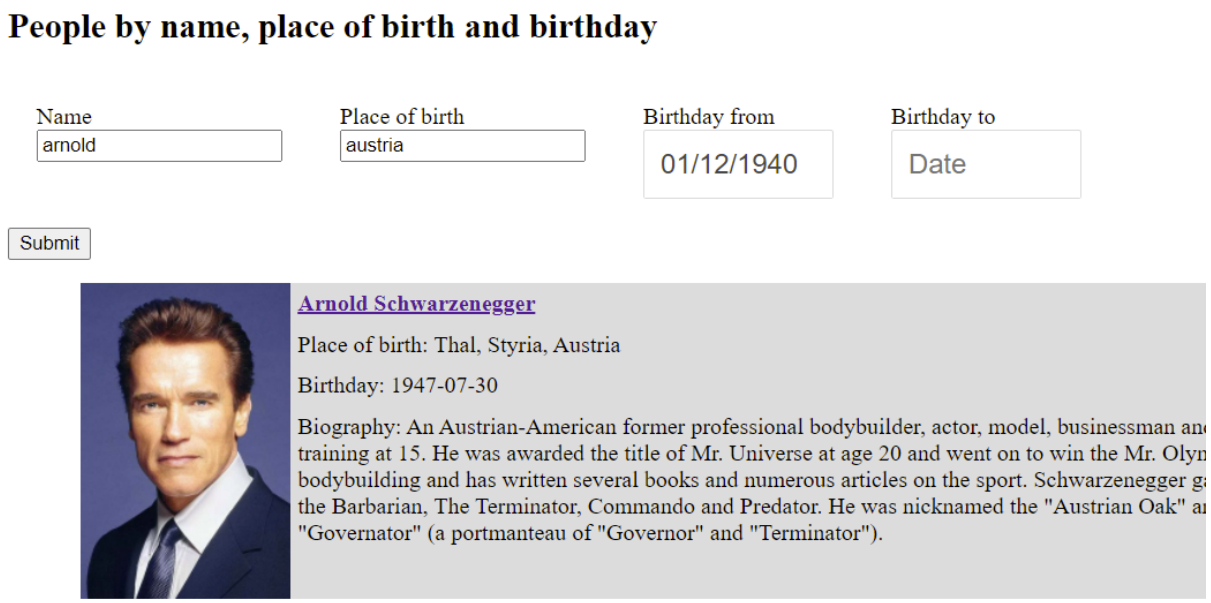
Rys. 16 - movies by actor and date



Rys. 17 - movies by genre, limit określa ile filmów się wyświetla

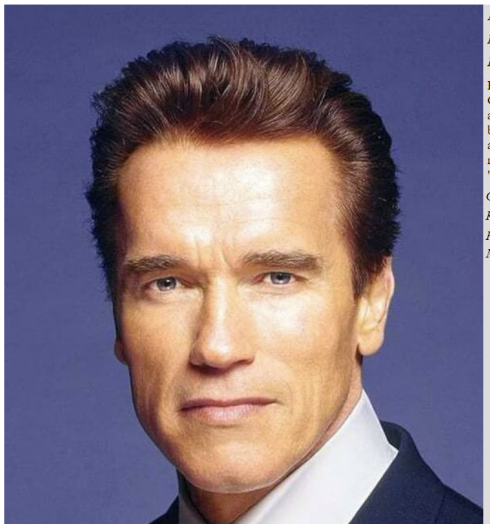


Rys. 18 - movies by named entities



Rys. 19 - people by name, place of birth and birthday

Root Movies by title Movies by actor Movies by genres Movies by named entities People by name Actors from movies Crew from movies



Arnold Schwarzenegger

Birthday: 1947-07-30

Place of birth: Thal, Styria, Austria

Biography: An Austrian-American former professional bodybuilder, actor, model, businessman and politician who served as the 38th Governor of California (2003–2011). Schwarzenegger began weight training at 15. He was awarded the title of Mr. Universe at age 20 and went on to win the Mr. Olympia contest a total of seven times. Schwarzenegger has remained a prominent presence in the sport of bodybuilding and has written several books and numerous articles on the sport. Schwarzenegger gained worldwide fame as a Hollywood action film icon, noted for his lead roles in such films as Conan the Barbarian, The Terminator, Commando and Predator. He was nicknamed the "Austrian Oak" and the "Styrian Oak" in his bodybuilding days, "Arnie" during his acting career and more recently the "Governator" (a portmanteau of "Governor" and "Terminator").

Gender: man

Known for department: Acting

Popularity: 21.291

Movies:

- *The Terminator (1984)*
- *Terminator 2: Judgment Day (1991)*
- *Terminator Genisys (2015)*
- *The Expendables (2010)*
- *Predator (1987)*
- *The Expendables 2 (2012)*
- *Terminator 3: Rise of the Machines (2003)*
- *Total Recall (1990)*
- *Escape Plan (2013)*
- *Batman & Robin (1997)*
- *The Expendables 3 (2014)*
- *Terminator: Dark Fate (2019)*
- *True Lies (1994)*
- *The Last Stand (2013)*
- *Commando (1985)*

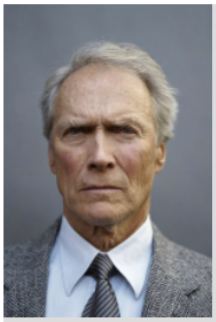
Rys. 20 - person details

Actors from movie

Title

dirty harry

Submit



Clint Eastwood

Played in: Dirty Harry

Place of birth: San Francisco, California, USA

Birthday: 1930-05-31

Biography: Clinton "Clint" Eastwood Jr. (born May 31, 1930) is an American film actor, director, producer, composer and for Name in Sergio Leone's Dollars Trilogy of spaghetti westerns ("A Fistful of Dollars," "For a Few Dollars More," and "The Go films ("Dirty Harry," "Magnum Force," "The Enforcer," "Sudden Impact," and "The Dead Pool") during the 1970s and 1980s. cultural icon of masculinity. Eastwood won Academy Awards for Best Director and Producer of the Best Picture, as well as re films in particular, as well as others including "Play Misty for Me" (1971), "The Outlaw Josey Wales" (1976), "Pale Rider" (1 commercial success and/or critical acclaim. Eastwood's only comedies have been "Every Which Way but Loose" (1978) and i his career after adjusting for inflation. Eastwood has directed most of his own star vehicles, but he has also directed films in w Award nominations and "Changeling" (2008), which received Golden Globe Award nominations. He has received considerabl awarded two of France's highest honors: in 1994 he received the Ordre des Arts et des Lettres medal and in 2007 was awarded Since 1967 Eastwood has run his own production company, Malpaso, which has produced the vast majority of his films. He al women, although he has only married twice. An audiophile, Eastwood is also associated with jazz and has composed and perf Clint Eastwood, licensed under CC-BY-SA, full list of contributors on Wikipedia.

Andrew Robinson

Played in: Dirty Harry

Place of birth: New York City, New York, USA

Birthday: 1942-02-14

Biography: Andrew Robinson (born February 14, 1942) is an American film, stage, and television actor. Robinson is known to television and in low-budget films. He is best known for his roles in Dirty Harry and Hellraiser as well as on Star Trek: Deep

Rys. 21 - actors from movie

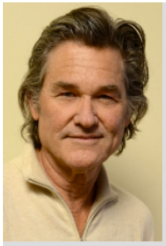
Crew from movie

Title
django

Jobs (separator ", ")

Department

Submit



Kurt Russell

Crew in: Django Unchained

Department: Crew

Job: Thanks

Place of birth: Springfield, Massachusetts, USA

Birthday: 1951-03-17

Biography: An American television and film actor. His first acting roles were as a child in television series, including a lead role in the Western series The Travels of Jaimie McP
Walt Disney Company, where he became, according to Robert Osborne, the "studio's top star of the '70s". In 1979, Russell was nominated for an Emmy Award for the made-for-
for Best Performance by an Actor in a Supporting Role in a Motion Picture for his performance opposite Meryl Streep in the 1984 film, Silkwood. During the 1980s, Russell wa
such as former air force hero-turned robber Snake Plissken in the futuristic action film Escape from New York, Antarctic helicopter pilot R.J. MacReady in the horror film The T
film Big Trouble in Little China (1986). Both Escape from New York and Big Trouble in Little China have since become cult films. In 1994, he had a starring role in the military
Olympic hockey coach Herb Brooks in Miracle (2004) won the praise of critics. In 2006, he appeared in the disaster-thriller Poseidon, and in 2007 Quentin Tarantino's Death Pr
article Kurt Russell, licensed under CC-BY-SA, full list of contributors on Wikipedia.

Quentin Tarantino

Crew in: Django Unchained

Department: Writing

Job: Writer

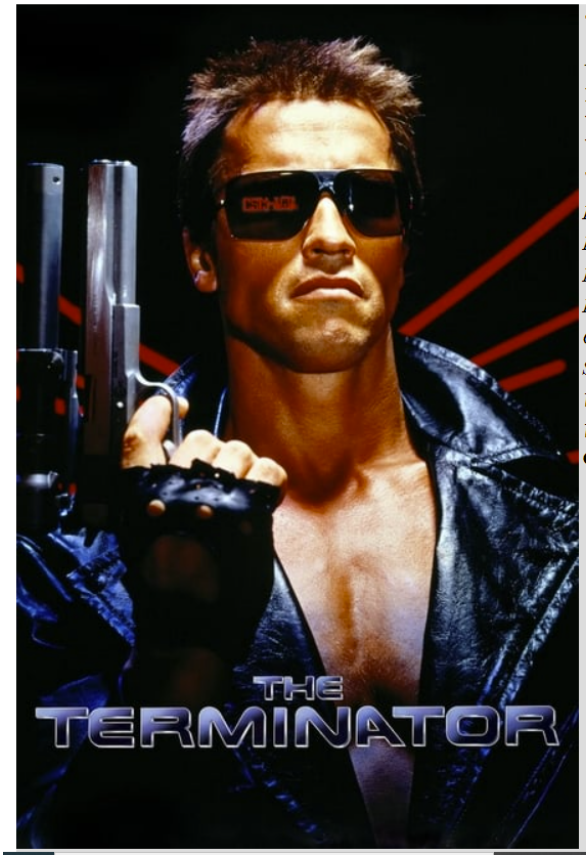
Place of birth: Knoxville, Tennessee, USA

Birthday: 1963-03-27

Biography: Quentin Jerome Tarantino (born March 27, 1963) is an American film director, screenwriter, producer, cinematographer and actor. In the early 1990s he was an indep
of violence. His films have earned him a variety of Academy Award, Golden Globe, BAFTA and Palme d'Or Awards and he has been nominated for Emmy and Grammy Awards
Tarantino was born in Knoxville, Tennessee, the son of Connie McHugh Tarantino Zastoupil, a health care executive and nurse born in Knoxville, and Tony Tarantino, an actor a
him to quit school at age 17, to attend an acting class full time. Tarantino gave up acting while attending the acting school, saying that he admired directors more than actors. Tar
close attention to the types of films people liked to rent, and has cited that experience as inspiration for his directorial career. Description above from the Wikipedia article Quent
Wikipedia.

Rys. 22 - crew from movie

Root | Movies by title | Movies by actor | Movies by genres | Movies by named entities | People by name | Actors from movies | Crew f



The Terminator

Release date: 1984-10-26

In the post-apocalyptic future, reigning tyrannical supercomputers teleport a
whose unborn son is destined to lead insurgents against 21st century mechar
warrior to safeguard Sarah. Can he stop the virtually indestructible killing m

Tagline: Your future is in its hands.

Homepage: https://www.20thcenturystudios.com/movies/theterminator

Runtime: 108

Budget: 6400000

Revenue: 78371200

Original language: en

Status: Released

Vote average: 7.6

Vote count: 10464

Cast:

- *Arnold Schwarzenegger*
- *Lance Henriksen*
- *Michael Biehn*
- *Linda Hamilton*
- *Brian Thompson*
- *Rick Rossovich*
- *Bill Paxton*
- *Earl Boen*
- *Bess Motta*
- *Shawn Schepps*
- *Dick Miller*
- *Marianne Muellerleile*
- *Paul Winfield*
- *Harriet Medin*

Rys. 23 - movie details