CONTENT-BASED

MOVIE RECOMMENDATION

USING AN INTERACTIVE CONVERSATIONAL AGENT (CHATBOT)

BARATSAS
SOTIRIS
f2821803

SPANOS
NIKOS
f2821826

ASSIGNMENT
BIG DATA CONTENT ANALYTICS

CONTENT-BASED

MOVIE RECOMMENDATION

USING AN INTERACTIVE CONVERSATIONAL AGENT (CHATBOT)

BARATSAS
SOTIRIS
f2821803

SPANOS
NIKOS
f2821826

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS
MSc in Business Analytics (FT18-19)

Who we are

Sotiris Baratsas

Nikos Spanos

Presentation Outline

1 The Challenge

2 Trends & Existing Solutions

3 Our Solution

4 Potential Applications

The Challenge

Build an effective movie recommendation system

It might be difficult for a person to find a new movie to watch, however they usually know what they are in the mood for (a specific genre usually) and they also know a few other movies they already like.

Movie Recommendation Approaches

Popularity-based recommenders

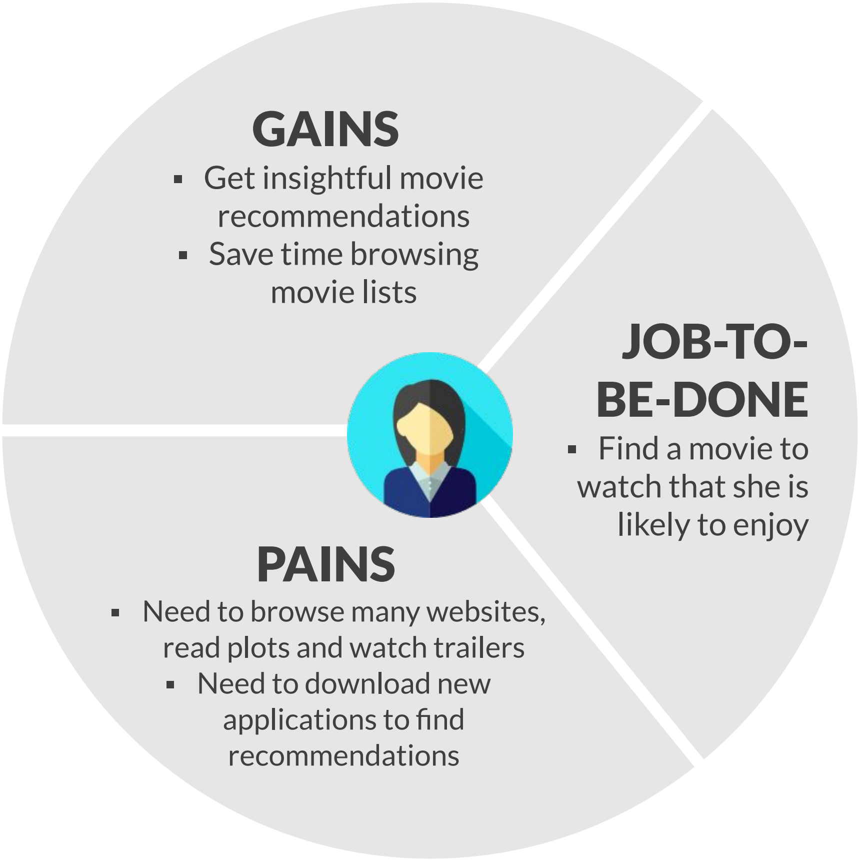
Content-based recommenders

Collaborative Filtering

Chatbots trend

“Chatbots” is one of the fastest growing digital trends.
In certain countries, consumers declared they actually prefer interacting with text & voice assistants instead of humans.

Understanding our user



GAINS

- Get insightful movie recommendations
- Save time browsing movie lists

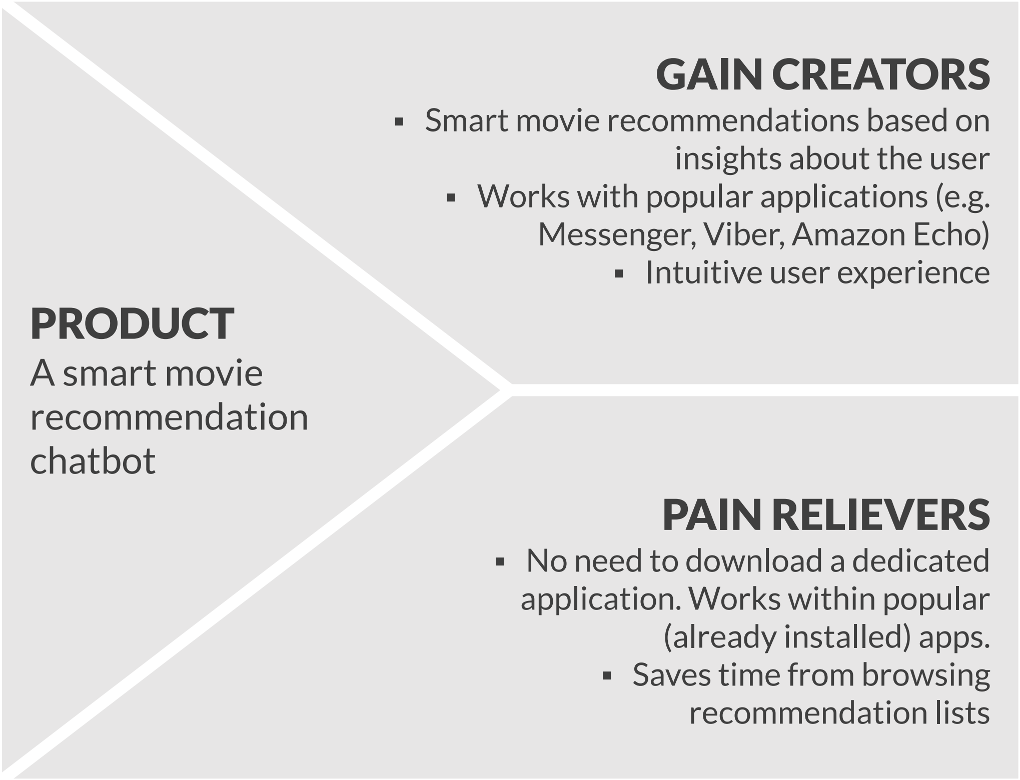
JOB-TO-BE-DONE

- Find a movie to watch that she is likely to enjoy

PAINS

- Need to browse many websites, read plots and watch trailers
- Need to download new applications to find recommendations

USER PROFILE



PRODUCT

A smart movie recommendation chatbot

GAIN CREATORS

- Smart movie recommendations based on insights about the user
- Works with popular applications (e.g. Messenger, Viber, Amazon Echo)
- Intuitive user experience

PAIN RELIEVERS

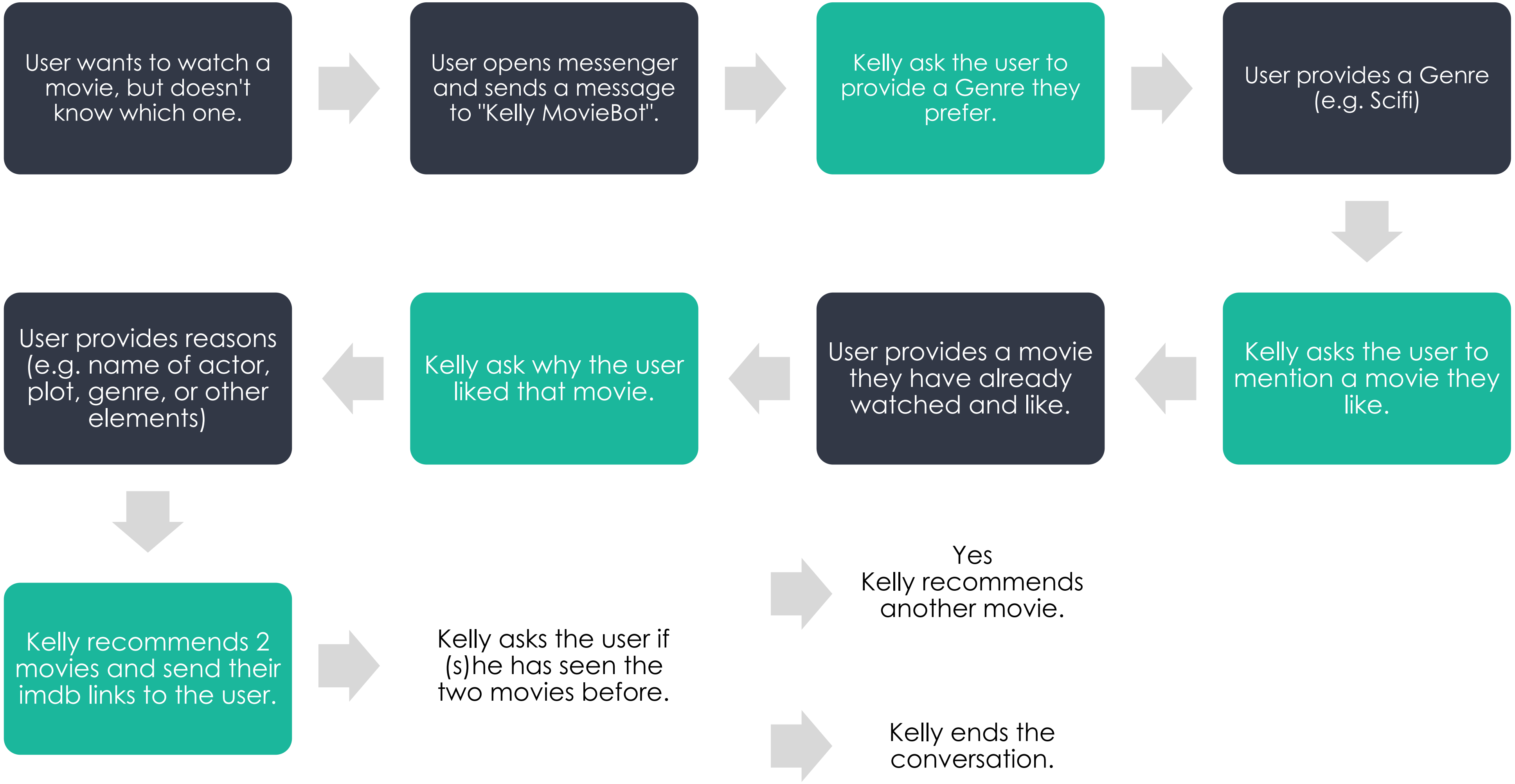
- No need to download a dedicated application. Works within popular (already installed) apps.
- Saves time from browsing recommendation lists

VALUE PROPOSITION

“

As a user, I want to get recommendations
based on a movie I already like,
so that I can find a new movie to watch.

Optimal user journey



User wants to watch a movie, but doesn't know which one.

User opens messenger and sends a message to "Kelly MovieBot".

Kelly ask the user to provide a Genre they prefer.

User provides a Genre (e.g. Scifi)

Kelly asks the user to mention a movie they like.

User provides a movie they have already watched and like.

Kelly ask why the user liked that movie.

User provides reasons (e.g. name of actor, plot, genre, or other elements)

Kelly recommends 2 movies and send their imdb links to the user.

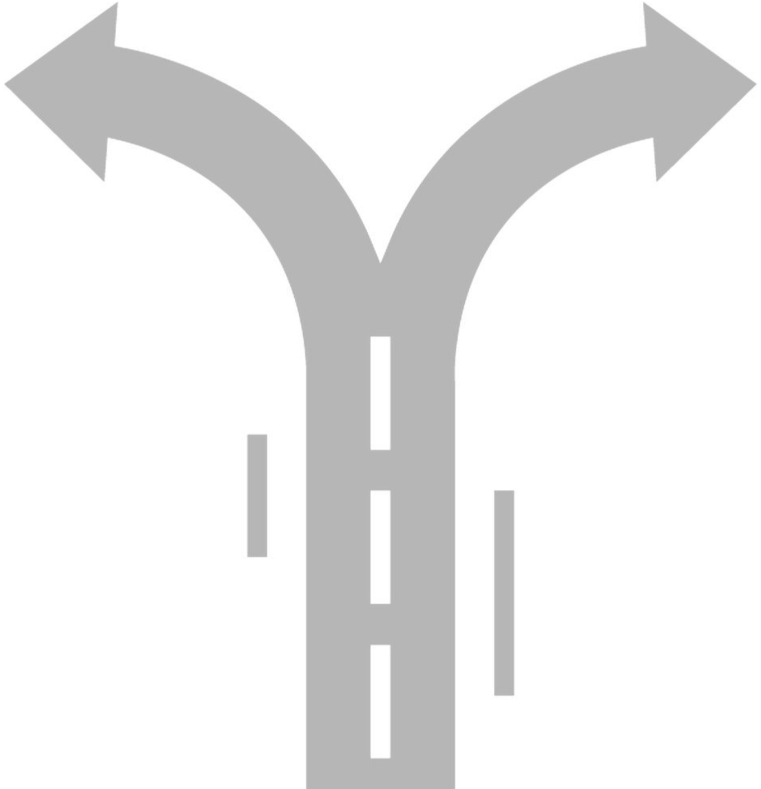
Kelly asks the user if (s)he has seen the two movies before.

Yes
Kelly recommends another movie.

Kelly ends the conversation.

OUR SOLUTION

Initial Approaches



Train embeddings using Wikipedia links

Train FastText embeddings using movie characteristics
(e.g. actors, plot, genre, director, etc)

Using Wikipedia Links

MAIN IDEA

Parse Wikipedia, create a dataset of articles about movies and their links to internal or external URLs. Train embeddings using pairs of {movies, links}. Based on a user input (=a movie they already like), recommend the movies mapped closer to that one.

Using Wikipedia Links

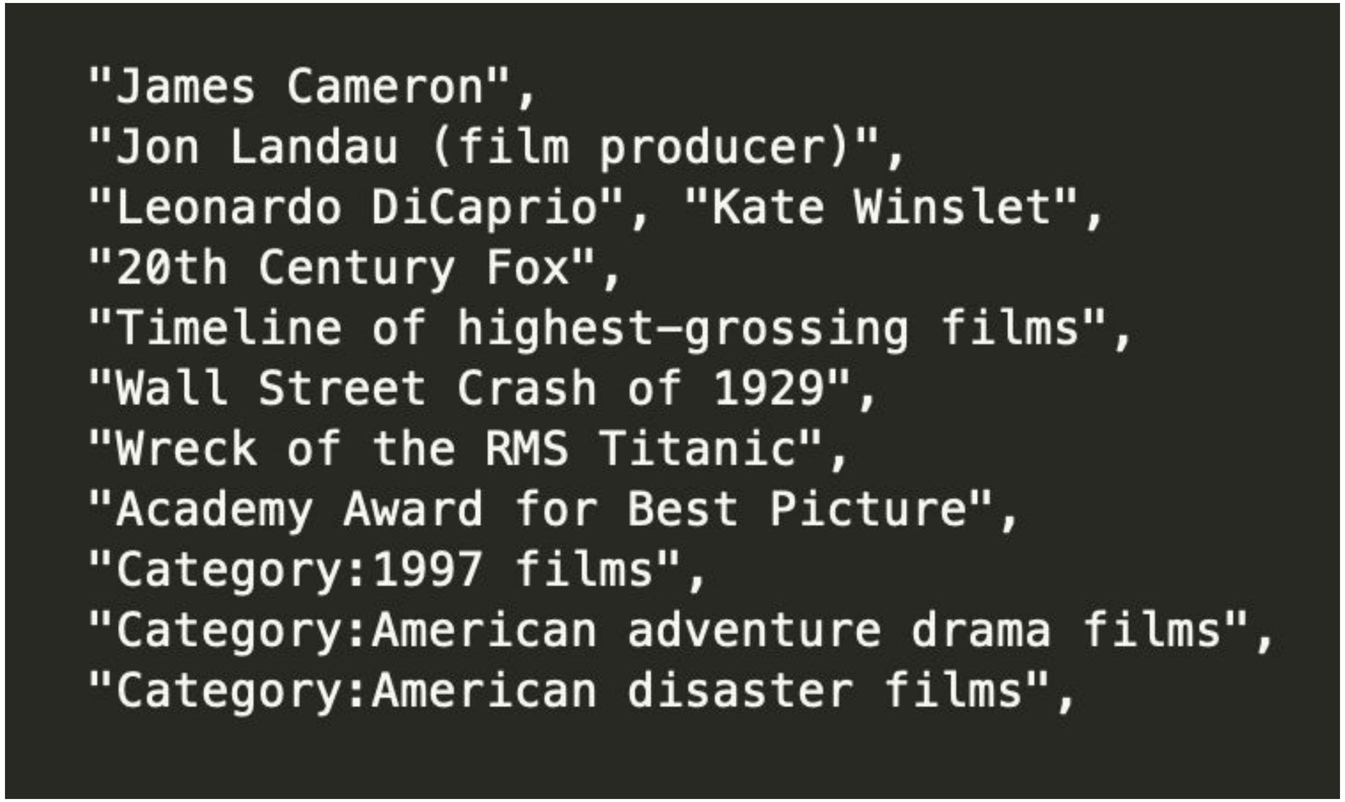
MAIN IDEA

Parse Wikipedia, create a dataset of articles about movies and their links to internal or external URLs. Train embeddings using pairs of {movies, links}. Based on a user input (=a movie they already like), recommend the movies mapped closer to that one.

WHY IT WORKS

✔ Nature and structure of the links

Using Wikipedia Links



```
"James Cameron",
"Jon Landau (film producer)",
"Leonardo DiCaprio", "Kate Winslet",
"20th Century Fox",
"Timeline of highest-grossing films",
"Wall Street Crash of 1929",
"Wreck of the RMS Titanic",
"Academy Award for Best Picture",
"Category:1997 films",
"Category:American adventure drama films",
"Category:American disaster films",
```

Example Links for
Titanic (1997 film)

Using Wikipedia Links

MAIN IDEA

Parse Wikipedia, create a dataset of articles about movies and their links to internal or external URLs. Train embeddings using pairs of {movies, links}. Based on a user input (=a movie they already like), recommend the movies mapped closer to that one.

WHY IT WORKS

- ✔ Nature and structure of the links
- ✔ Mapping of movies, as well as links

Using Movie Characteristics & FastText

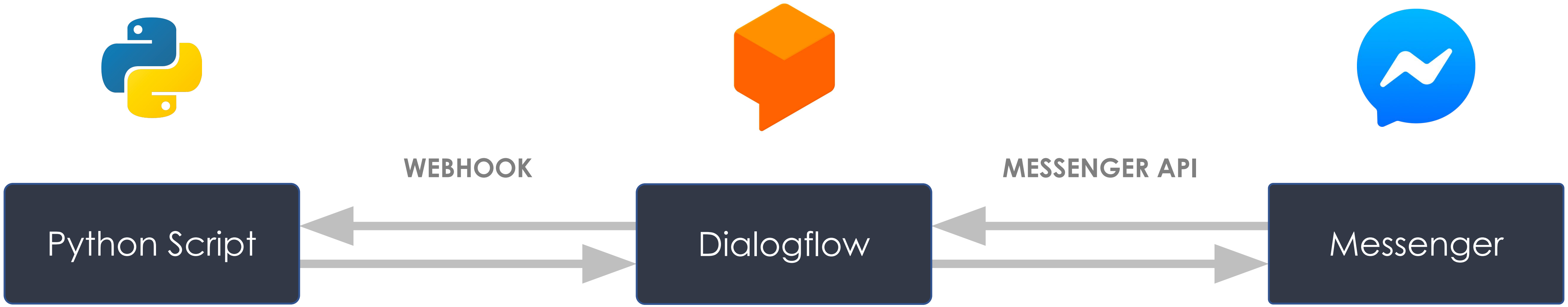
MAIN IDEA

- Start with an iMDb dataset with 5000 movies.
- Parse using BeautifulSoup to extract information (i.e. plot summary & imdb rating) + Data Cleaning
- Create word embeddings based on cast, plot, genre, director
- Create a scoring and matching algorithm based on cosine distance (1-cosine similarity)

WHY IT WORKS

✔ Takes into account a combination of different movie features
✔ Maps words using the FastText algorithm
✔ Takes into account user inputs to penalize or reward

Components



WEBHOOK

MESSENGER API

Python Script

Dialogflow

Messenger

Access to dataset
Embeddings
Scoring algorithm
Matching algorithm

Manages conversation
Records user inputs and passes them to Python
Mediator between UI and back-end

User interaction
Matching output

Cleaning & Transformations

- ✗ Duplicate links on the dataset.
- ✗ Clean every string (object variable) from extra spaces and special characters.
- ✗ Clean every string (object variable) from extra spaces and special characters.
- ✗ Drop columns that we didn’t want.
- ✗ Drop the rows were the duration was less than 70 minutes. Since we observed that in the dataset we had also Series episodes.

- ✓ Separated in different column each of the 7 genres a movie may had.
- ✓ We built two functions to correct the genre and the movie title given as input by the user.
- ✓ We combined the value from each column to a unified text per movie.
- ✓ Update the column IMDB Rating and scrapping the summary plot of each movie
- ✓ Create a naive approach of comparing the movies to each other.

Embeddings

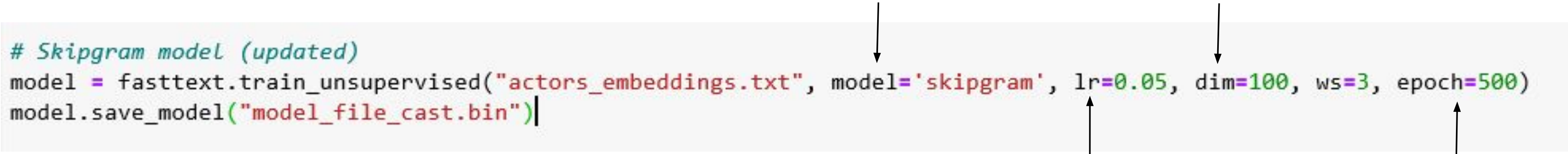
Cast Embeddings

Plot Embeddings

Features Embeddings

Skipgram vs CBOW

Controls the size of the vectors



```
# Skipgram model (updated)
model = fasttext.train_unsupervised("actors_embeddings.txt", model='skipgram', lr=0.05, dim=100, ws=3, epoch=500)
model.save_model("model_file_cast.bin")
```

Learning Rate
(Too high can cause the model to overfit)

Window Size (stochastic)

Recommendation algorithm

Phase 1

- ✓ Input Genre
- ✓ Input Movie Title
- ✓ Input reasons you like the above movie
- ✓ Clean the above inputs if appropriate

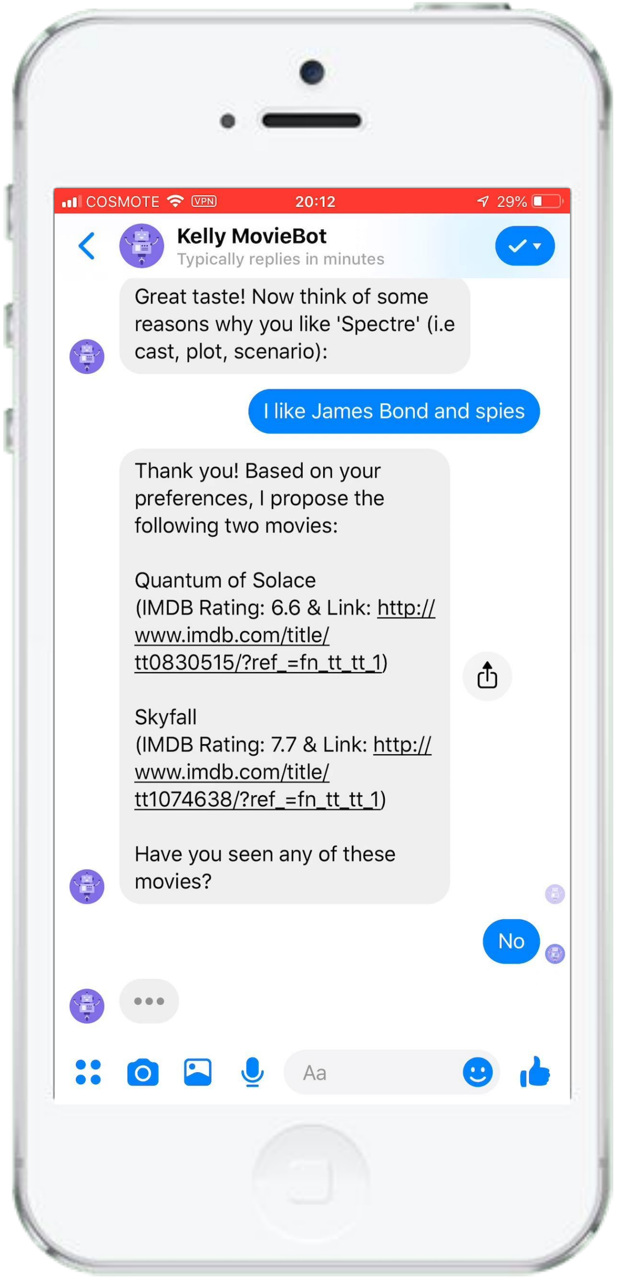
Phase 2

- ✓ Slice the dataset based on the genre given as input.
- ✓ Check with an If/Else statement whether the movie is in the movies_list.
- ✓ If TRUE then we follow the approach of cosine distance and word embeddings.
  - ● After this step we filter the 5 most similar movies to the one chosen from the user.
- ✓ If FALSE we skip the cosine distance approach since we miss the important columns.

Phase 3

- ✓ Calculate the movie scoring based on three scoring parameters:
  - ● Primary Genre (x0.2)
  - ● IMDB Rating (x0.3)
  - ● Number of Words (x0.5)
- ✓ Out of the 5 movies selected from cosine distance in Phase 2 we recommend the 3 movies with the highest scoring.

LET’S SEE A DEMO



Kelly MovieBot

Typically replies in minutes

Great taste! Now think of some reasons why you like 'Spectre' (i.e cast, plot, scenario):

I like James Bond and spies

Thank you! Based on your preferences, I propose the following two movies:

Quantum of Solace
(IMDB Rating: 6.6 & Link: http://www.imdb.com/title/tt0830515/?ref_=fn_tt_tt_1)

Skyfall
(IMDB Rating: 7.7 & Link: http://www.imdb.com/title/tt1074638/?ref_=fn_tt_tt_1)

Have you seen any of these movies?

No

•••

NEXT STEPS

Improvements

Better configuration of dialogue engine

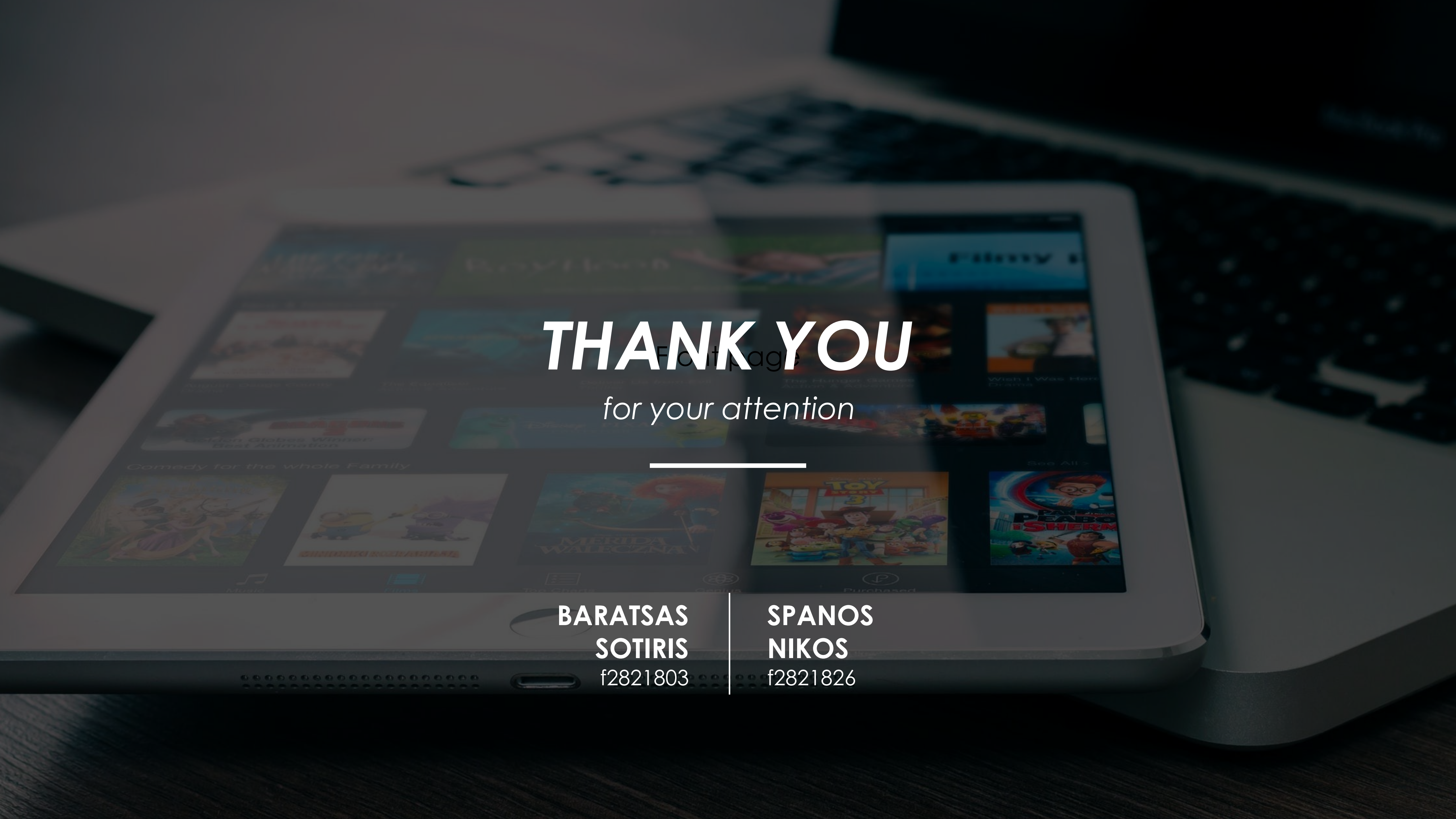
Larger dataset

Optimize scoring & matching algorithms

Potential Business Applications

Deploy this to a cinema to boost sales

Create an audience-facing entertainment app, sponsored by streaming companies



THANK YOU

for your attention

BARATSAS SOTIRIS
f2821803

SPANOS NIKOS
f2821826