732A92/TDDE16 Text Mining

Introduction to Lab L1: Information Retrieval

Huanyu Li

LINKÖPING UNIVERSITY

Agenda

- Aims of this lab
- Lab task
- Instruction
- Notes
- Work on jupyter notebook

Aims

- To capture basic web crawling technique
- Get start to work with basic NLP techniques
- To implement vector model for storing and querying web information

Lab task --- To implement an Android app search engine

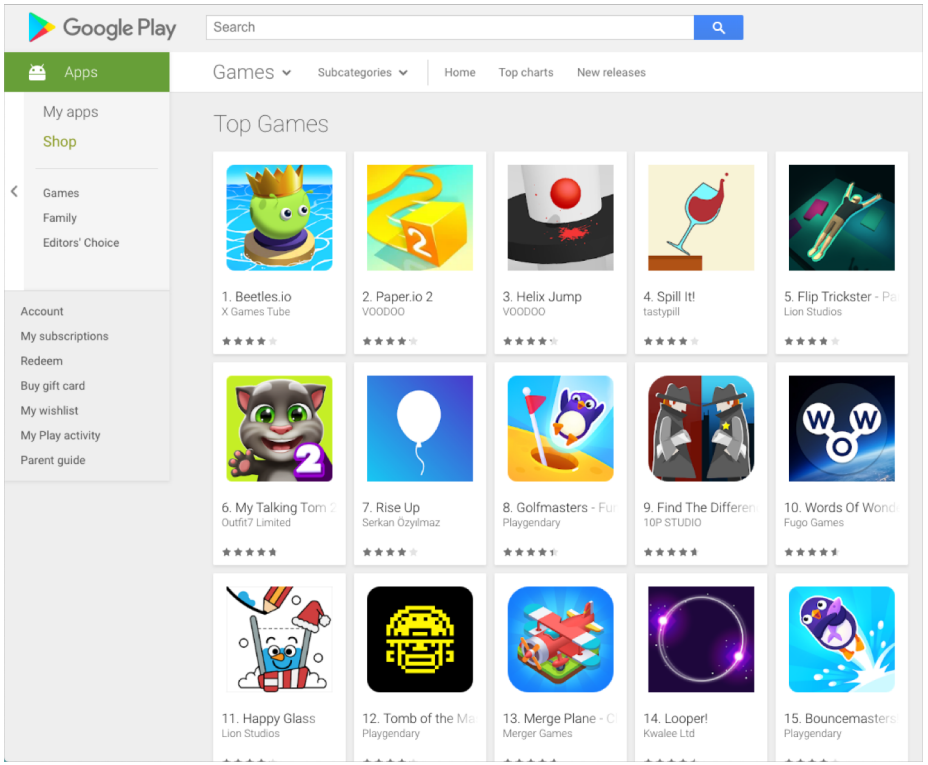
- Build a storage for data that describe a number of apps
  - Crawl the app store for Android app
  - Obtain the descriptive text for each app
  - Store the data locally
- Build a Information Retrieval model
  - Vector Model
  - Tf-idf
- Ranked query processor
  - Cosine similarity

Instruction

- Step 1
  - Access web site such as Google Play and extract at least 1000 app descriptions (No constraint on what kind of app)
- Step 2
  - Preprocess app descriptions: tokenization, normalization, etc.
  - Build and store a vector model using tf-idf to define weights
- Step 3
  - Preprocess and model given query keywords or sentence
  - Write a ranked query processor

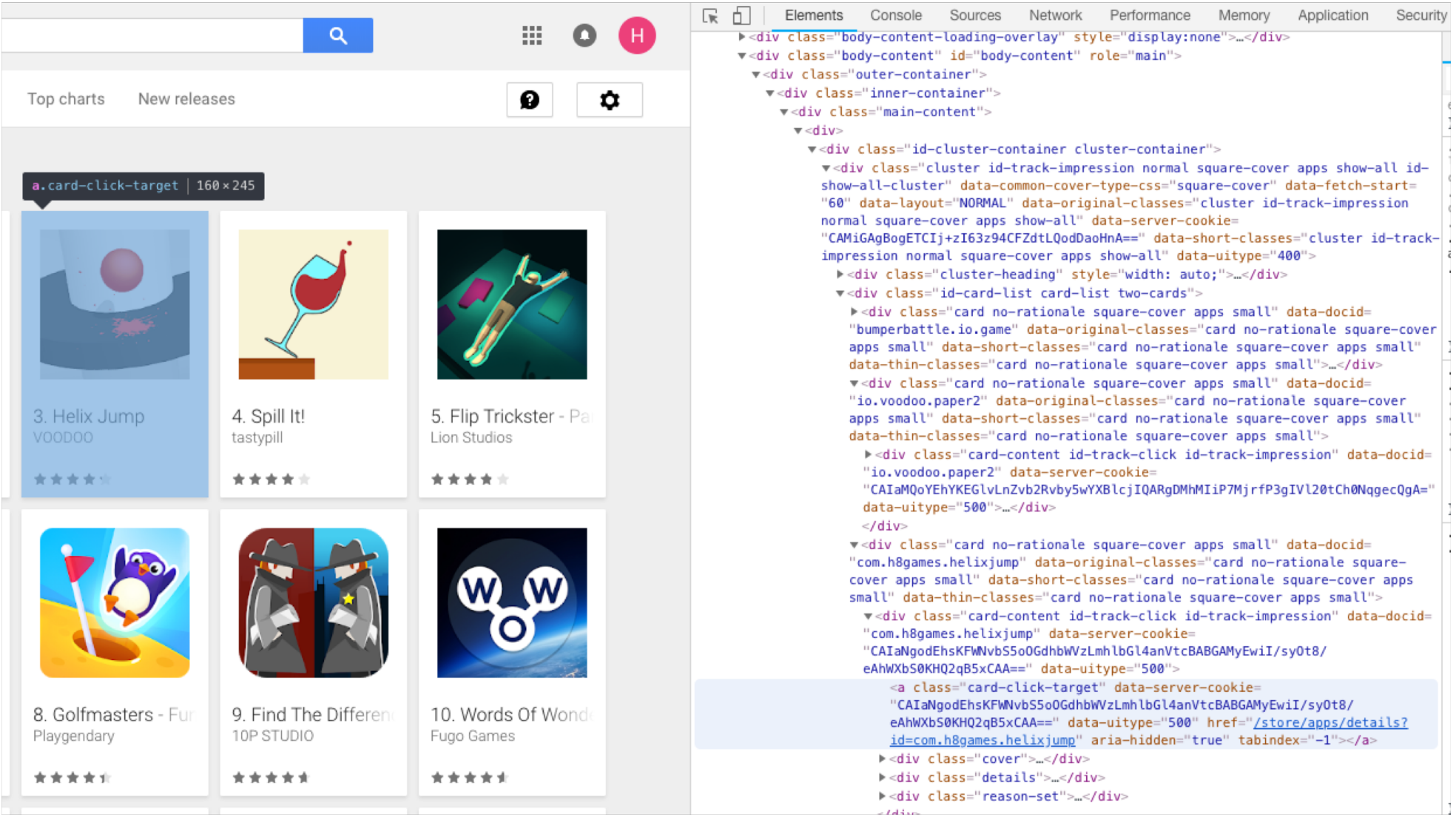
Instruction - an example

- Step 1
  - Access web site such as Google Play and extract at least 1000 app descriptions (No constraint on what kind of app)
  - *https://play.google.com/store/apps/category/GAME/collection/topselling_free*



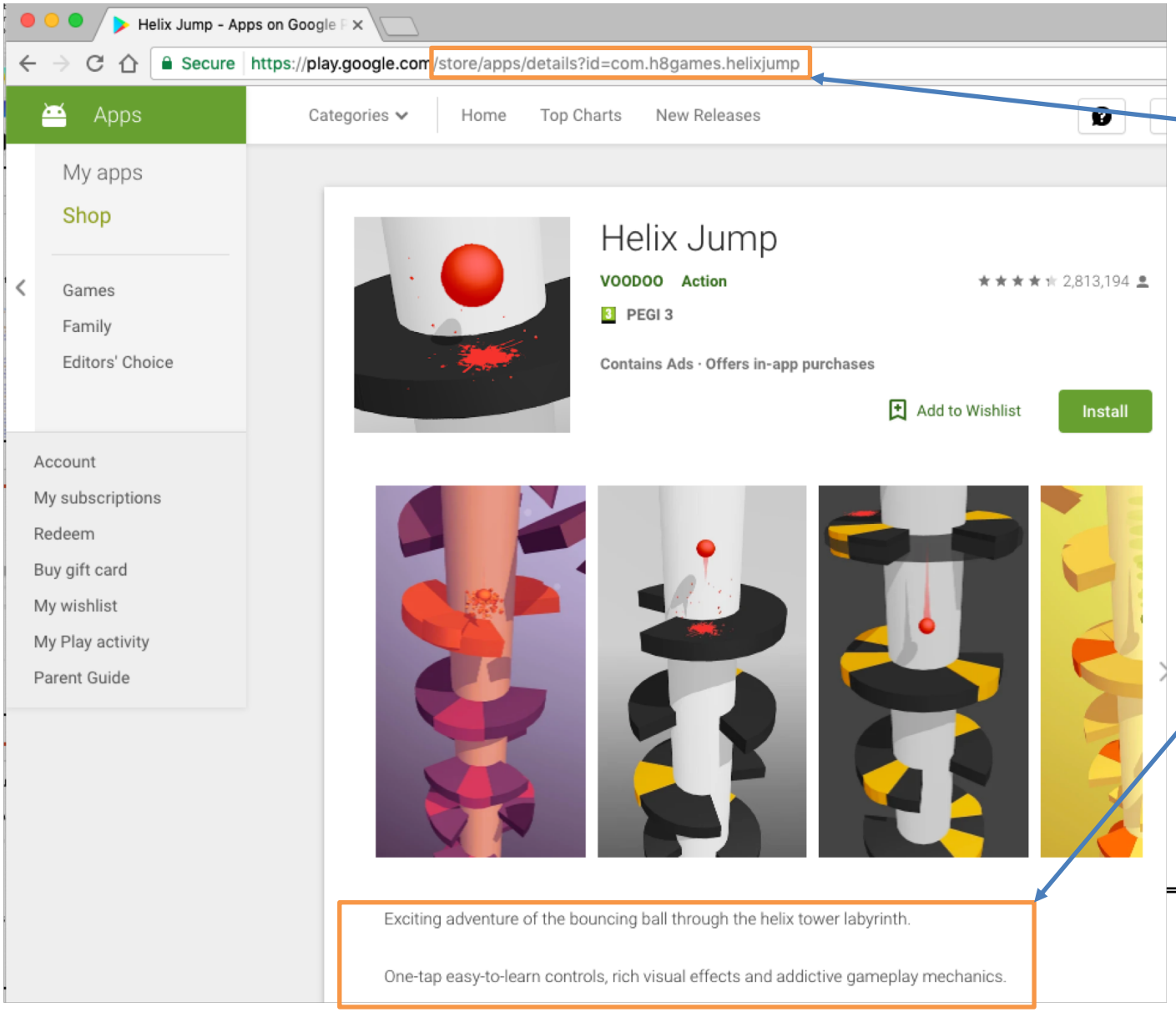
Instruction - an example

- Step 1
  - Inspect the webpage with Chrome or save it locally and open it with an editor to check the content



Instruction - an example

- Step 1
  - Each app has an unique identifier and a piece of descriptive text



Identifier

Descriptive text

Instruction - an example

- Step 1
  - Get webpage html text by using functions in *urllib* module or other module such as *beautifulsoup*
  - Extract app identifier by using regular expression (*re* module)
    - (e.g. using regular expression "href=\"/store/apps/details.*?\"")
    - re.findall(pattern, string)
  - Access the specific webpage of each app and extract descriptive text by using regular expression
    - Add &hl=en to acquire only descriptions (some apps in Swedish will be returned)
    - Extract description
    - (e.g. using regular expression "itemprop=\"description.*?\">.*?<div jsname=\".*?\">.*? ")
  - Store the data locally

Instruction - an example

- Step 2
  - Preprocess the data (*nltk* or *spaCy*)
    - Remove non-alpha-numeric characters
    - Tokenize
    - Lowercase
    - Remove stop words
    - Stemming
  - Construct Vector Model
    - Compute tf-idf using *numpy* module or other modules such as *scikit-learn*

Instruction - an example

- Step 3
  - Preprocess search keywords or sentence
    - Same approach as before
  - Compute tf-idf for search keywords or sentence
    - Be clear about how you calculate the weights tf-idf or how the functions from a library calculates the weights
  - Compute similarities and return top k results

Notes and Work on jupyter notebook

- Test your code fragments and functions as soon as you achieve
  - Make sure you have correctly obtained at least 1000 apps' urls first
  - Then access the app webpage
- Avoid call crawling function after you have saved app description in local files
- You might need to add parameter after the url
  - *https://play.google.com/store/apps/collection/topselling_free?start=480&num=60*
- Run
  - *source /home/TDDE16/labs/environment/bin/activate*
  - *source /home/732A92/labs/environment/bin/activate*

Agenda

- Aims of this lab
- Lab task
- Instruction
- Notes
- Work on jupyter notebook

www.liu.se

LINKÖPING UNIVERSITY