DATA SCIENCE REPORT

STAGE 3

ENTITY MATCHING BETWEEN FILMCRAVE AND IMDB TUPLES USING MAGELLAN

Team Members

1. Saranya Baskaran

2. Shivanee Nagarajan

3. Varun Ramesh

i. Data Description

We have chosen IMDB and FilmCrave websites for movie reviews. In this stage, we will be matching movie tuples using Magellan's end to end EM-Workflow.

a) IMDB:

IMDB is one of the most popular and reliable movie review sites.

Sample URL: imdb-link

A total of 3750 tuples are present in this table. (Named as imdb1.csv in the data directory)

b) Filmcrave:

Filmcrave is an online movie social network that allows users to write movie reviews, share movie lists, watch trailers and interact with other members

Sample URL: filmcrave

A total of 3220 tuples are present in this table. (Named as filmcrave1.csv in the data directory)

Attributes present in both these tables are provided below -

Column	Description	Type
Title	Title of the movie	String
Overall Rating	Overall rating of the movie	Float
Year	Year of release	Integer
Genre	Genres of the film	String
MPAA_Rating	MPAA rating obtained by the movie	String
Directors	Directors of the movie	String
Actors	Stars of the movie	String
Plot	Description of the movie plot	String

ii. Blocker Stage

The Blocker we use is a combination of overlap and attribute equivalence blockers. We run these blockers as a cascaded model (i.e.) we run each blocker on the previous blocker's output.

1. **Overlap blocker** with overlap size = 1 on the attribute 'Title'. (word-level and no stop words). – 11,39,838 tuples.
2. **Overlap blocker** with overlap size = 1 on attribute 'Director'. (word-level and no stop words) – 9459 tuples.
3. **Attribute Equivalence blocker** on the attribute 'year'. – 2024 tuples.

The above blocker stage did not miss any of the positive matches as the cascaded blocker model we ran only matched movie Titles and Director names up to one word and then matched the year. As our data sources were clean and reliable, the output of this blocker stage had all the positive matches. The following file has been saved as BlockingSurvivedDataset.csv in the data directory.

iii. Matching Stage

We sampled 500 tuples and labelled these from the above 2024 tuples and split it into set I and set J. Set I has 250 tuples and set J has 250 tuples. Both of these files are stored as Trainset.csv and Testset.csv in the data directory. The labelled data set was named as labelled.csv in the data directory.

After running the six classifiers, the following output was observed on set I. Both Naïve Bayes and Linear Regression have an f1-score of 1.000

	Matcher	Average precision	Average recall	Average f1
0	DecisionTree	0.989717	0.995122	0.992322
1	RF	1.000000	0.995122	0.997531
2	SVM	0.895191	1.000000	0.943919
3	LinReg	1.000000	1.000000	1.000000
4	LogReg	0.994595	0.995122	0.994791
5	NaiveBayes	1.000000	1.000000	1.000000

Observing these scores, there was no requirement for debugging across the cross-validation iterations. (The drilled down stats for f-1 has been provided in the ipynb file).

iv. Results on test set J

The following result was observed on set J using the Linear Regression classifier which was the best-performing classifier.

Matcher	Average Precision	Average Recall	Average F1
Decision Tree	97.33	98.38	97.85
Random Forest	97.87	99.46	98.66
Linear Regression	98.4	100.00	99.2
Logistic Regression	98.4	100.00	99.2
Naïve Bayes	98.4	100.00	99.2

The final best matcher was Linear Regression matcher which was picked based on the above table. [Based on precision, recall and F-1].

v. Results on test set J

The following result was observed on set J using the Linear Regression classifier which was the best-performing classifier.

Matcher	Average Precision	Average Recall	Average F1
Decision Tree	97.33	98.38	97.85
Random Forest	97.87	99.46	98.66
Linear Regression	98.4	100.00	99.2
Logistic Regression	98.4	100.00	99.2
Naïve Bayes	98.4	100.00	99.2

The final best matcher was Linear Regression matcher which was picked based on the above table. [Based on precision, recall and F-1].

vi. Time Estimates

The time provided below is collective time spent by all the three of us -

Time spent in understanding software: **3 hours.**
Blocking: **2 hours. (Tried different combinations).**
Labelling Data: **1 hour.**
Finding best matcher: **1 hour.**

vii. Recall Comment

The recall results that we extracted our high and hence, we did not debug or improve the learner anymore.

viii. Bonus Comments

1. Documentation

The Magellan notebook has very good support for a number of method calls making it very easy for a new user to call API's available in Magellan. The notebooks provide a running documentation for all the API calls available making it very user friendly.

2. Easy API Calls / ML Support

Magellan provides excellent support for ML classifiers which helps both an ML and a non-ML user to make use of this support. Furthermore, the drilled down stats provide an easy way to debug our classifier stage.

3. Installation

The current anaconda package on MacOS Sierra "10.13.4" faces errors when installing through pip. The error is related to the clang libstdc++ being linked to the cython compilation rather than libc++. This can be reset by setting LDFLAGS and CPPFLAGS to the right qt/lib and qt/include package. Even then we faced errors with ld:map error linker to x86_64 architecture. The conda installation went through fortunately enough. We are still unsure of how the ld:map error can be fixed as it seems to be related to the llvm shipped by Apple. One suggestion is to have a "requirements.txt" file with all the versions. The user can directly run "pip install requirements.txt" rather than individual installation of all the packages.

4. Information – Gain/ Active Learning based labelling

Currently, we are randomly sampling tuples for labelling from the blocker survived dataset. But, it might be very helpful to build a good model by picking those training tuples which provide the best information gain. This can also be combined with the concept of active learning.

5. Loading labels from a table rather than UI labelling

It would be incredibly helpful if we can load a pre-calculated labels file directly into Magellan rather than depending on UI labelling. We saved the sampled set as a csv file and gave the labels in, but it would be very helpful if a file can be directly loaded.

6. Rate of progress using UI on large datasets

We initially tried to run only one overlap blockers on about 200,000 tuples and this took a very long time! It could be very helpful if there is a progress % or progress bar that displays how much more eta this command would take. This way the user can work on other parts of the project or take a break ☺