Система прогнозирования рейтинга фильмов

Добро пожаловать в презентацию, посвященную нашей системе прогнозирования рейтинга фильмов.

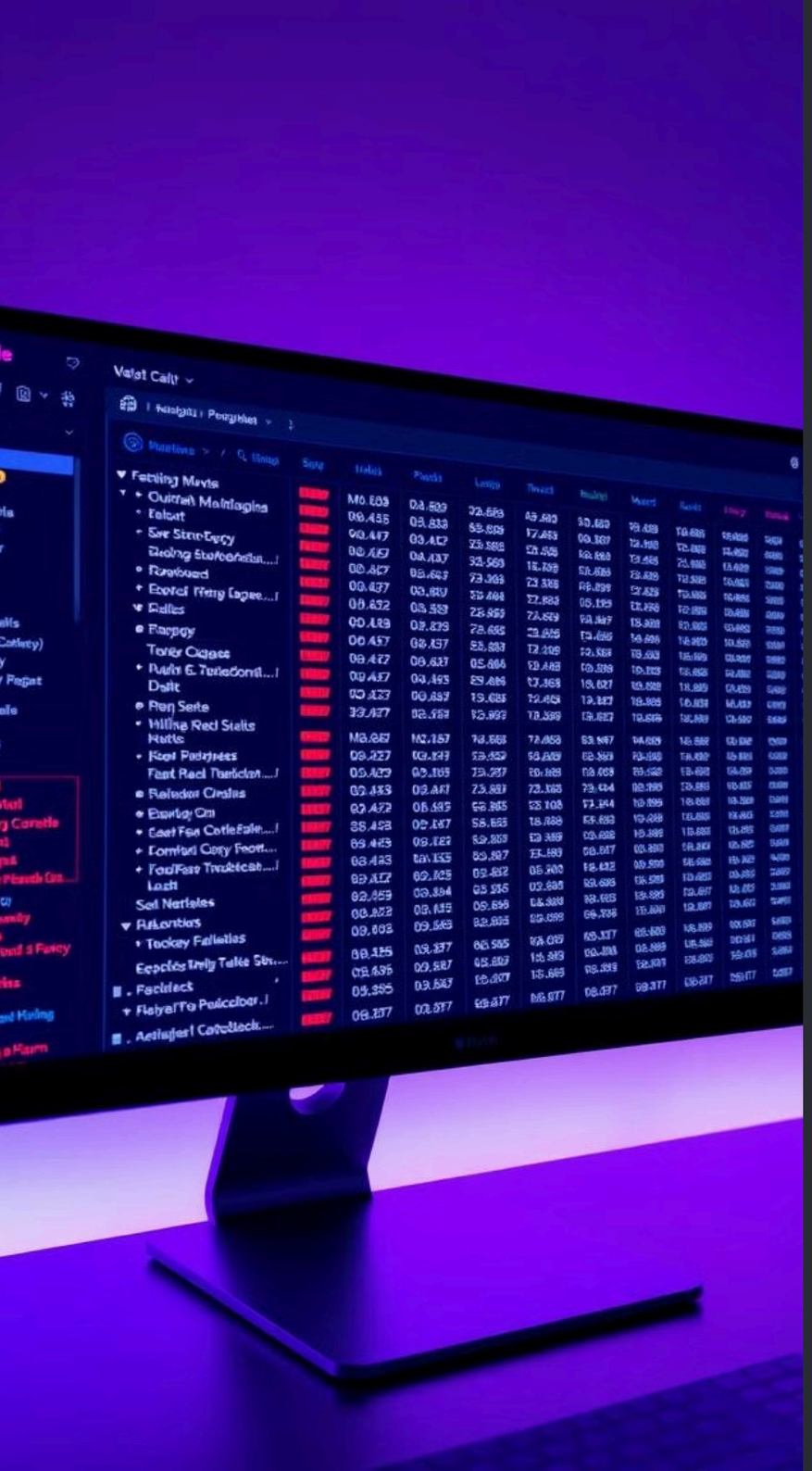
Проблема, которую решает наша система

Непредсказуемость

Трудности в прогнозировании успеха фильма, даже с учетом имеющихся данных.

Отсутствие точности

Существующие системы прогнозирования часто оказываются неточными и нерелевантными.



Источник данных: набор данных о лучших фильмах с сайта Kaggle

Kaggle

www.kaggle.com

IMDb Dataset Top-Rated fi...

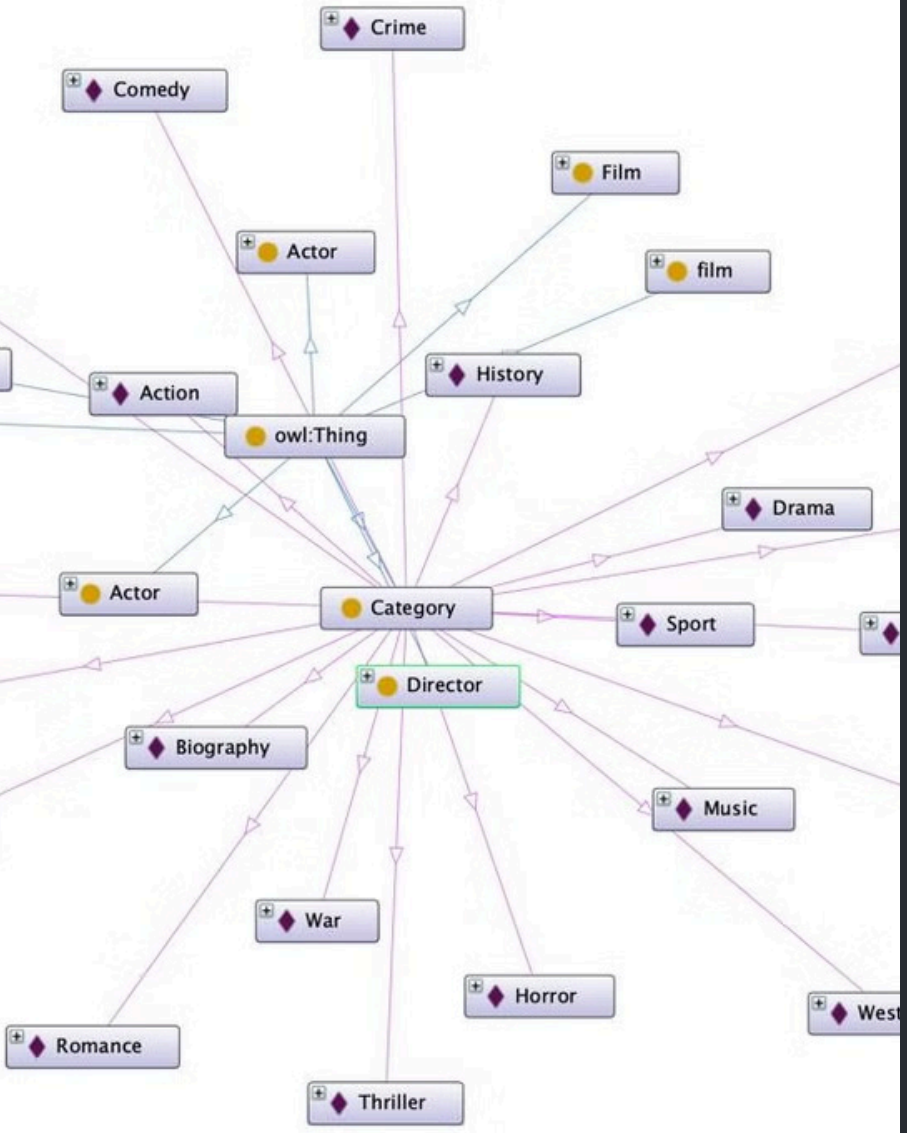
Contains Films, Short Films, Movies, Documentaries.

IMDb

Набор данных включает информацию о самых высоко оцененных фильмах с сайта IMDb.

Период

Данные охватывают широкий временной диапазон, начиная с 1898 года.



Онтология: построение графа знаний

Триплеты

Более 66 тысяч триплетов в нашем графе

Свойства

1 428 798 object properties

SparQL

Мы написали скрипт на python, который выполняет наши запросы на SparQL к нашему графу

```
PREFIX ns: <{ns}> SELECT ?film WHERE {{ ?film ns:hasActor <{ns}Actor/RyanGosling> . ?film ns:rating ?rating . <{ns}film/Searching> ns:rating ?ratingY . FILTER(?rating >= ?ratingY) }}
```

(rdflib.term.URIRef('http://www.semanticweb.org/martin/ontologies/2024/9/untitled-ontology-6/film/Drive'),)
(rdflib.term.URIRef('http://www.semanticweb.org/martin/ontologies/2024/9/untitled-ontology-6/film/La_La_Land'),)
(rdflib.term.URIRef('http://www.semanticweb.org/martin/ontologies/2024/9/untitled-ontology-6/film/The_Big_Short'),)
(rdflib.term.URIRef('http://www.semanticweb.org/martin/ontologies/2024/9/untitled-ontology-6/film/Blade_Runner_2049'),)
(rdflib.term.URIRef('http://www.semanticweb.org/martin/ontologies/2024/9/untitled-ontology-6/film/The_Notebook'),)

```
PREFIX ns: <{ns}>
SELECT ?film
WHERE {{
  ?film ns:hasDirector <{ns}Director/SatoshiKon> .
  ?film ns:duration ?duration .
  ?film ns:rating ?rating .
```

(rdflib.term.URIRef('http://www.semanticweb.org/martin/ontologies/2024/9/untitled-ontology-6/film/Tôkyô_goddofâzâzu'),)
(rdflib.term.URIRef('http://www.semanticweb.org/martin/ontologies/2024/9/untitled-ontology-6/film/Pâfekuto_burû'),)
(rdflib.term.URIRef('http://www.semanticweb.org/martin/ontologies/2024/9/untitled-ontology-6/film/Sennen_joyû'),)
(rdflib.term.URIRef('http://www.semanticweb.org/martin/ontologies/2024/9/untitled-ontology-6/film/Papurika'),)

```
[9]  # Преобразование DataFrame в массив NumPy
     triples = triples_df.values

     # Проверка, что все элементы в triples являются строками
     assert all(isinstance(item, str) for row in triples for
```

```
[10] from ampligraph.evaluation import train_test_split_no_un

     X_train, X_valid = train_test_split_no_unseen(np.array(t

     print('Train set size: ', X_train.shape)
     print('Test set size: ', X_valid.shape)
```

```
Train set size:  (53696, 3)
Test set size:  (13000, 3)
```

```
     from ampligraph.latent_features import ScoringBasedEmbed
     from ampligraph.latent_features.loss_functions import ge
     from ampligraph.latent_features.regularizers import get

     model = ScoringBasedEmbeddingModel(k=50,
                                        eta=5,
                                        scoring_type='ComplEx
                                        seed=0)

     # Optimizer, loss and regularizer definition
     optimizer = tf.keras.optimizers.Adam(learning_rate=1e-4)
     loss = get_loss('multiclass_nll')
     regularizer = get_regularizer('LP', {'p': 3, 'lambda': 1

     # Compilation of the model
     model.compile(optimizer=optimizer, loss=loss, entity_rel
```

```
[12] model.fit(X_train,
               batch_size=int(X_train.shape[0] / 10),
               epochs=50, # Number of training epochs
               verbose=True # Displays a progress bar.
               )
```

Embedding

```
from scipy.special import expit
probs = expit(scores)

pd.DataFrame(list(zip([' '.join(x) for x in statements],
                      ranks,
                      np.squeeze(scores),
                      np.squeeze(probs))),
             columns=['statement', 'rank', 'score', 'prob']).sort_values("prob")
```

	statement	rank	score	prob
0	FilmDeadpool2 rating 9.1	[21373, 34591]	-0.584321	0.397863
2	FilmDeadpool2 rating 8.7	[23251, 32317]	-0.457857	0.430378
5	FilmDeadpool2 rating 8.3	[18341, 33653]	-0.452671	0.441219
9	FilmDeadpool2 rating 8.4	[32198, 31962]	-0.397395	0.401938
1	FilmDeadpool2 rating 8.1	[21298, 11828]	0.060685	0.540115
8	FilmDeadpool2 rating 8.0	[23304, 73]	1.270129	0.767104
3	FilmDeadpool2 rating 7.9	[22711, 67]	1.528384	0.818905
7	FilmDeadpool2 rating 7.8	[17468, 12]	2.011032	0.8628170
6	FilmDeadpool2 rating 7.7	[12518, 6]	2.586570	0.929992
4	FilmDeadpool2 rating 7.6	[16263, 1]	5.183377	0.996673

```
from ampligraph.evaluation import mr_score, mrr_score, hits_at_n_score

# Преобразование массива рангов в одномерный массив
flat_ranks = ranks.flatten()
# Фильтрация одномерного массива, чтобы оставить только положительные значения (начиная с 1)
res_ranks = flat_ranks[flat_ranks > 0]

mr = mr_score(res_ranks)
mrr = mrr_score(res_ranks)

print("MRR: %.2f" % (mrr))
print("MR: %.2f" % (mr))

hits_10 = hits_at_n_score(res_ranks, n=10)
print("Hits@10: %.2f" % (hits_10))
hits_3 = hits_at_n_score(res_ranks, n=3)
print("Hits@3: %.2f" % (hits_3))
hits_1 = hits_at_n_score(res_ranks, n=1)
print("Hits@1: %.2f" % (hits_1))
```

```
MRR: 0.24
MR: 126.70
Hits@10: 0.40
Hits@3: 0.24
Hits@1: 0.12
```

Выводы и заключение

Наша система прогнозирования рейтинга фильмов представляет собой мощный инструмент для киноиндустрии. Она позволяет повысить точность прогнозирования, оптимизировать маркетинговые кампании и улучшить процесс принятия решений.