

Abstract geometric lines in the top left corner, consisting of several overlapping, irregular polygons and lines in a light beige color.

CAR RATINGS AND REVIEWS

SEARCH ENGINE

Alessandro Franceschini

Raffaele Andrei

Filippo Gelosini

COLLEZIONE TESTUALE

Contiene le recensioni relative a diversi modelli di automobile.

La collezione è stata presa da Kaggle, una piattaforma che mette a disposizione gratuitamente dataset di ogni tipo.



ANKURJAIN · UPDATED 5 YEARS AGO

▲ 48

New Notebook

Download (53 MB)



Edmunds-Consumer Car Ratings and Reviews

Which car-brand/model/type got the most ratings?



COLLEZIONE TESTUALE

- 282.055 text items suddivisi in 50 file CSV, uno per ogni marca principale di automobile
- Dimensioni medie di ciascun text item: 0.5 kB
- Struttura di ciascun text item:

	Review_Date	Author_Name	Vehicle_Title	Review_Title	Review	Rating
0	on 04/02/18 16:47 PM (PDT)	Coop	2006 Subaru Baja Crew Cab Sport 4dr Crew Cab AWD SB (2.5L 4cyl 4A)	Great Car for the Mission	As a student I don't need a big truck this the versatility of the Baja is optimal. It hauls everyth...	5



STRUMENTI UTILIZZATI

Linguaggio utilizzato

Python

Librerie utilizzate:

- Whoosh – indicizzazione e ricerche full-text
- Word2Vec – sviluppo del modello word2vec
- Pandas – manipolazione dei file CSV per addestramento e testing del modello word2Vec
- Torch – addestramento e testing del modello di sentiment analysis
- Transformers – creazione del modello di sentiment analysis
- Nltk – operazioni testuali
- Sklearn – scelta dei file utilizzati per l'addestramento e per il testing del modello word2vec e di sentiment analysis
- Datasets e huggingface_hub – gestione, condivisione e recupero del modello di sentiment creato dal sito «huggingface.co»

SEARCH ENGINE - BASE

Schema Woosh:

```
schema = Schema(  
    reviewID = NUMERIC(stored=True),  
    reviewDate = DATETIME(stored=True),  
    authorName = TEXT(stored=True),  
    rawVehicleName = TEXT(stored=True),  
    reviewTitle = TEXT(stored=True),  
    reviewText = TEXT(stored=True),  
    reviewRating = NUMERIC(stored=True),  
    vehicleName = TEXT(stored=True)
```

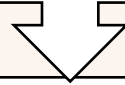
QUERY LANGUAGE

Tutte le versioni del search engine consentono:

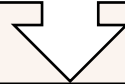
- Ricerche libere
- Ricerche per campo
- Ricerche a AND e a OR su qualunque campo
- Per ogni ricerca, una funzione di parsing «decide» se cercare le recensioni oppure le macchine
- La versione del search engine con sentiment analysis consente anche ricerche effettuate sui campi «sentimentLabel» e «sentimenScore»

SEARCH ENGINE – WORD2VEC: CREAZIONE MODELLO

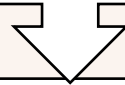
Divisione random del dataset (70% addestramento e 30% test)



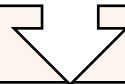
Preprocessing (Tokenizzazione e rimozione di caratteri non alfanumerici)



Estrazione delle frasi per l'addestramento



Creazione del modello Word2Vec (tramite Gensim)



Salvataggio del modello su file testuale

SEARCH ENGINE – WORD2VEC: IMPLEMENTAZIONE

Schema Woosh: stesso schema utilizzato per le ricerche full-text

```
schema = Schema(  
    reviewID = NUMERIC(stored=True),  
    reviewDate = DATETIME(stored=True),  
    authorName = TEXT(stored=True),  
    rawVehicleName = TEXT(stored=True),  
    reviewTitle = TEXT(stored=True),  
    reviewText = TEXT(stored=True),  
    reviewRating = NUMERIC(stored=True),  
    vehicleName = TEXT(stored=True)
```

Recupero dal file testuale il modello creato

```
word2VecModel = KeyedVectors.load_word2vec_format(word2VecModelPath)
```

Per ogni documento, calcola la lista di vettori di embeddings

```
review_tokens = preprocessing(pd.Series(reviewText))[0].split()  
review_embeddings = [model[token] for token in review_tokens if token in model]  
review_vector = np.mean([embedding for embedding in review_embeddings], axis=0)  
vectors[reviewID] = review_vector
```

Salva su file binario i vettori di embeddings medi

```
with open("word2vec_vectors.bin", "wb") as file:  
    pickle.dump(all_vectors, file)
```

SEARCH ENGINE – WORD2VEC: IMPLEMENTAZIONE

Preprocessing della query e calcolo i vettori di embeddings

```
query_tokens = preprocessing(pd.Series(query_str))[0].split()  
query_embeddings = [word2VecModel[token] for token in query_tokens if token in word2VecModel]
```

Calcola il vettore medio dei vettori di embeddings relativi alla query

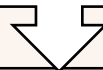
```
query_vector = np.mean([embedding for embedding in query_embeddings], axis=0)
```

La funzione di ranking è la cosine similarity tra i due vettori

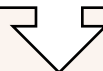
```
similarity = np.dot(query_vector, doc_vector) / (np.linalg.norm(query_vector) * np.linalg.norm(doc_vector))
```


SEARCH ENGINE – SENTIMENT ANALYSIS: CREAZIONE MODELLO

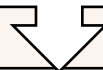
Divisione random del dataset (90% addestramento e 10% test)



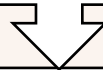
Preelaborazione dei dati (tokenizzazione e stemming)



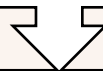
Creazione del modello preaddestrato (distilbert-base-uncased)



Addestramento (campo reviewTitle)



Valutazione delle prestazioni



Salvataggio del modello su HuggingFace

SEARCH ENGINE – SENTIMENT ANALYSIS: CREAZIONE MODELLO

Valutazione delle prestazioni

car_sentiment


This model is a fine-tuned version of [distilbert-base-uncased](#) on the None dataset. It achieves the following results on the evaluation set:

- Loss: 0.3386
- Accuracy: 0.8943
- F1: 0.7962

Parametri di addestramento

```
training_args = TrainingArguments(  
    output_dir=repo_name,  
    learning_rate=2e-5,  
    per_device_train_batch_size=8,  
    per_device_eval_batch_size=8,  
    num_train_epochs=2,  
    weight_decay=0.01,  
    save_strategy="epoch",  
    push_to_hub=True,  
)
```

Classificazione delle sequenze

 **Inference API** ⓘ

🔮 Text Classification Examples ▾

this car sucks

Compute

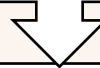
Computation time on cpu: 0.242 s

<div><div></div><div>LABEL_0</div></div>	0.979
<div><div></div><div>LABEL_1</div></div>	0.021

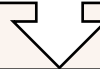
</> JSON Output Maximize

SEARCH ENGINE – SENTIMENT ANALYSIS: INDICIZZAZIONE

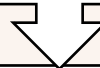
Caricamento del modello recuperato da HuggingFace



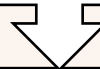
Definizione dello schema Whoosh



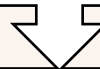
Parsing e preelaborazione dei text item



Calcolo del punteggio di sentiment (relativo al titolo della recensione)



Calcolo del label di sentiment (positivo, negativo, neutrale)



Aggiunta dei documenti all'indice

SEARCH ENGINE – SENTIMENT ANALYSIS: IMPLEMENTAZIONE

Schema Whoosh modificato

```
schema = Schema(  
    reviewID = NUMERIC(stored=True),  
    reviewDate = DATETIME(stored=True),  
    authorName = TEXT(stored=True),  
    rawVehicleName = TEXT(stored=True),  
    reviewTitle = TEXT(stored=True),  
    reviewText = TEXT(stored=True),  
    reviewRating = NUMERIC(stored=True),  
    vehicleName = TEXT(stored=True),  
    sentimentScore = NUMERIC(stored=True),  
    sentimentLabel = TEXT(stored=True)  
)
```

Caricamento del modello di sentiment

```
sentiment_model = pipeline(model="af41/car_sentiment")
```

Calcolo del punteggio di sentiment e dei label

```
sentiment = sentimentModel(reviewTitle)[0]  
label = sentiment['label']  
sentiment_score = sentiment['score']  
  
if sentiment_score > 0.6 and label == "LABEL_1": #positive label  
    sentimentLabel = "positive"  
elif sentiment_score > 0.6 and label == "LABEL_0": #negative label  
    sentimentLabel = "negative"  
else:  
    sentimentLabel = "neutral"
```

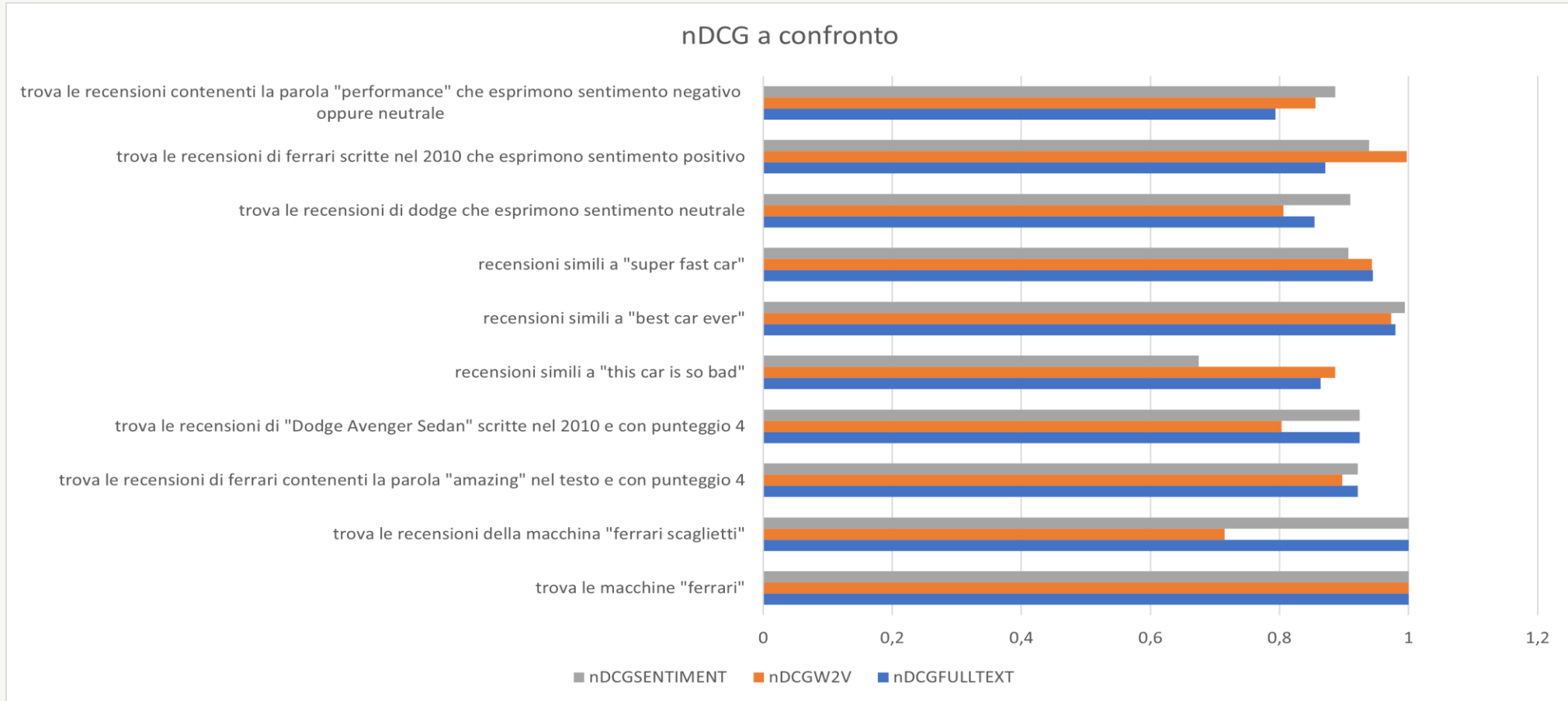
Modifica del ranking aggiungendo un fattore di sentiment

```
class MyWeighting(BM25F):  
    use_final = True  
    def final(self, searcher, docnum, score):  
        sentiment_score = searcher.stored_fields(docnum).get("sentimentScore", 0.5)  
        sentiment_factor = sentiment_score * 2 - 1  
        adjusted_score = score * sentiment_factor  
        return adjusted_score
```

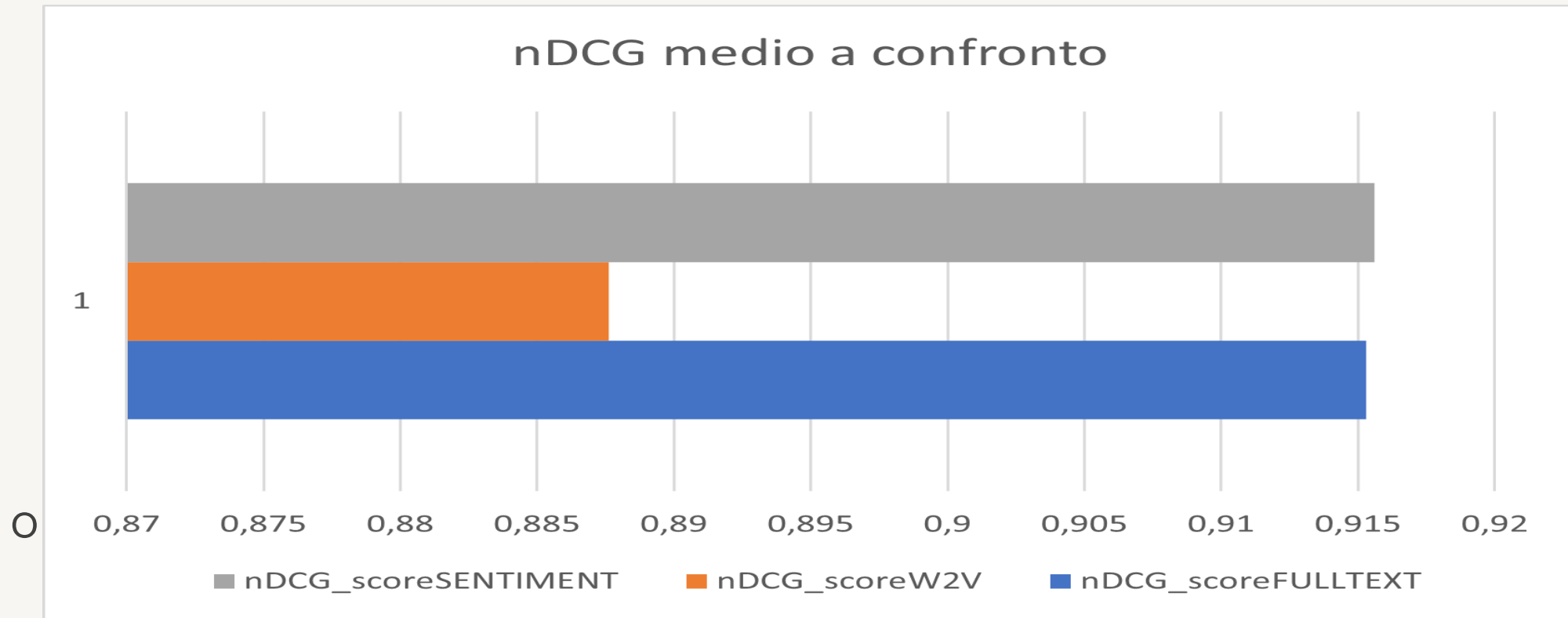
BENCHMARK – UIN E QUERY

UIN	Query
Trova le macchine «Ferrari»	Ferrari
Trova le recensioni della macchina «Ferrari Scaglietti»	Ferrari Scaglietti reviews
Trova le recensioni di macchine Ferrari contenenti la parola «amazing» nel testo e con punteggio 4	vehicleName:ferrari reviewText:amazing reviewRating:4
trova le recensioni di "Dodge Avenger Sedan" scritte nel 2010 e con punteggio 4	vehicleName:dodge AND vehicleName:avenger vehicleName:sedan reviewDate:2010 reviewRating:4
Trova recensioni simili a "this car is so bad"	this car is so bad
Trova recensioni simili a "best car ever"	best car ever
Trova recensioni simili a "super fast car"	super fast car
Trova le recensioni di dodge che esprimono sentimento neutrale	vehicleName:dodge sentimentLabel:neutral
Trova le recensioni di ferrari scritte nel 2010 che esprimono sentimento positivo	vehicleName:ferrari sentimentLabel:positive reviewDate:2010
Trova le recensioni contenenti la parola "performance" che esprimono sentimento negativo oppure neutrale	reviewText:performance sentimentLabel:neutral OR sentimentLabel:negative

BENCHMARK



BENCHMARK



BENCHMARK – TEMPO DI INDICIZZAZIONE



FULL TEXT

Nessun modello da addestrare



WORD2VEC

+10 minuti per l'addestramento del modello



SENTIMENT ANALYSIS

+40 minuti per l'addestramento del modello

BENCHMARK - CONCLUSIONI



- SEMPLICE IMPLEMENTAZIONE
- VELOCE DA INDICIZZARE
- SCARSA EFFICACIA SE LA QUERY NECESSITA DEL CONTESTO



- VELOCE DA INDICIZZARE
- RESTITUISCE BUONI RISULTATI ANCHE SE LE PAROLE CERCATE NON SONO PRESENTI NEL TESTO
- SCARSA EFFICACIA IN QUERY LUNGHE E TROPPO SPECIFICHE



- PERFORMANCE LEGGERMENTE MIGLIORI ALLA VERSIONE FULL-TEXT
- AMPLIA LE POSSIBILITÀ DI RICERCA
- LENTO DA INDICIZZARE

A series of thin, light brown lines forming an abstract geometric pattern in the top left corner of the slide. The lines intersect to create various triangular and polygonal shapes.

GRAZIE

Alessandro Franceschini

Raffaele Andrei

Filippo Gelosini