**Final Proje Raporu: Metin Tabanlı Veri Setleri Üzerinde Benzerlik Tespiti**

**1. Giriş**

Bu projede, dini metin yorumlarından oluşan bir veri seti kullanılarak farklı vektörleştirme yöntemleriyle metinler arası benzerlik tespiti yapılmıştır. Reddit platformundan elde edilen yorumlar, ön işlemden geçirilerek hem TF-IDF hem de Word2Vec modelleriyle analiz edilmiştir. Projenin amacı, giriş olarak verilen bir yorum ile en benzer 5 yorumu farklı modeller aracılığıyla önermektir.

**2. Veri Seti ve Ön İşleme**

**Veri Kaynağı**

Reddit üzerindeki r/Christianity gibi dini temalı subreddit'lerden Python kullanılarak praw kütüphanesi yardımıyla yorumlar çekildi. Toplamda yaklaşık 15.000'den fazla yorum toplanmıştır. Ham veri dosyası: religious\_texts.csv

**Ön İşleme Aşamaları**

Ön işleme için lemmastem.py dosyası kullanıldı. Adımlar:

* Noktalama işaretleri ve sayılar temizlendi.
* Metinler küçük harfe çevrildi.
* NLTK kullanılarak tokenizasyon yapıldı.
* Stopword'ler çıkarıldı (nltk.corpus.stopwords)
* İki farklı yol izlendi:
  + Lemmatizasyon (WordNetLemmatizer)
  + Stemming (PorterStemmer)

Çıktı dosyaları:

* clean\_lemmatized.csv
* clean\_stemmed.csv

**3. Modelleme**

**3.1 TF-IDF**

tfidf\_vectorizer.py dosyası ile iki TF-IDF matrisi üretildi:

* tfidf\_lemmatized.csv
* tfidf\_stemmed.csv

Bu dosyalar, giriş yorumu ile tüm yorumlar arasındaki cosine benzerliğini hesaplayarak en benzer 5 yorumu bulmak için kullanıldı.

**3.2 Word2Vec**

word2vec.py dosyası ile 16 farklı Word2Vec modeli eğitildi. Model parametreleri:

* 2 yöntem: CBOW, Skip-Gram
* 2 pencere boyutu: 2, 4
* 2 vektör boyutu: 100, 300
* 2 ön işleme yöntemi: lemmatize, stem

Model isimleri örneği:

word2vec\_lemmatized\_cbow\_window2\_dim100.model

word2vec\_stemmed\_skipgram\_window4\_dim300.model

Eğitim için gensim.models.Word2Vec kullanıldı. Eğitim dosyası: word2vec.py

**4. Benzerlik Ölçümü**

generate\_model\_outputs.py dosyasında giriş metni olarak "observe lying vanity forsake mercy" ifadesi girildi. Tüm modeller bu metinle ilgili en benzer 5 yorumu önerdi. Eğer kelimeler modelde yoksa, " Giriş kelimeleri modelde yok." uyarısı verildi.

Bu öneriler model\_outputs.csv dosyasına kaydedildi.

**5. Değerlendirme**

**5.1 Ortalama Cosine Benzerlik**

model\_average\_similarity\_scores.csv dosyasındaki cosine benzerlikler, her model için ortalama skor hesaplamak için kullanıldı. Kod: compute\_average\_similarity.py

**5.2 Manuel Skorlama**

Model önerileri manuel olarak 0–1 arasında puanlandı. Puanlar manual\_scoring\_template.csv üzerinde girildi. Doldurulmuş hali: manual\_scoring\_filled.csv

**5.3 Model Karşılaştırma**

compare\_models\_scores.py dosyası, hem otomatik hem manuel skorlara göre karşılaştırma tablosu (model\_comparison\_scores.csv) oluşturdu.

**5.4 Jaccard Benzerlik Matrisi**

Aynı giriş metnine verilen öneri kümeleri jaccard\_matrix.csv dosyasında yer aldığı gibi, jaccard.py scripti ile değerlendirildi. Model kümeleri arasındaki Jaccard skorları hesaplandı.

**6. Yorum: Hangi Model Ne Zaman Uygun?**

* Lemmatize edilmiş Skip-Gram modeller yüksek anlam benzerliği için ideal.
* CBOW modeller daha hızlı ama yüzeysel öneriler için uygun.
* TF-IDF, kelime tabanlı eşleştirme gereken durumlarda daha açıklayıcı.

**7. Sonuç**

* 16 modelin çıktıları başarıyla üretildi ve değerlendirildi.
* Gerekli tüm tablolar ve dosyalar GitHub üzerinde paylaşıldı.
* Kodlar ayrı dosyalarda modüler olarak yazıldı.

**8. Kaynaklar**

* Reddit API (PRAW)
* Gensim
* NLTK
* Scikit-learn
* Pandas, NumPy, Matplotlib

GitHub Projesi: <https://github.com/Mahsumx/NLPodev>