**EGE ÜNİVERSİTESİ**
**MÜHENDİSLİK FAKÜLTESİ**
**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**


# İŞLEMSEL ZEKA

## 2021-2022 BAHAR YARIYILI

## DÖNEM PROJESİ RAPORU

**PROJE KONUSU:** Doğal dil işleme yöntemleri ile duygu analizi

**HAZIRLAYAN**

**05190000023 - Sinan Burak Filiz**

**TESLİM TARİHİ**

**20.06.2022**

## 1. Problemin Tanımı

Bir konu hakkında yapılan yorumların olumlu ya da olumsuz olduğunu tanımlayan bir program geliştirmek. Konu seçimim film yorumları oldu. Bu tanımlamayı yapmak için doğal dil işleme yöntemleri kullanmak gerekiyor. Programım öğrenme yöntemlerinden birini kullanarak, verisetinden beslenecek ve tanımlamayı yapacak.

## 2. Araştırma

Öncelikle bireysel olarak araştırma yaparak başladım. Popüler bir konu olan duygu analizi alanında kaynak bulmak çok kolay olsa da, Türkçe dili konusunda daha dar bir çerçeve mevcut. Bu bağlamda öncelikle İngilizce temelli bir literatür taraması ve araştırma yaptım. Hazır kütüphaneler kullanmanın en uygun yöntem olduğuna karar verdikten sonra, programımın öğrenme yöntemine karar vermem gerekiyordu. Bu alanda da araştırmalar yaptım. Ardından benzer çalışmaları inceledim. Tüm bu araştırmalardan edindiğim bilgileri kıyaslamalar yaptım. Sonuç olarak yapmış olduğum taramalarla araştırma sürecini bitirdim.

Bu süreçte özellikle doğal dil işleme ve duygu analizinin kullanım alanının genişliği ilgi çekiciydi. Ayrıca yazılım alanın en zorlu konularından biri olduğunu fark ettim. Bilgisayarlar her ne kadar analitik konularda çok başarılı ve hızlı olsalar da, insanın doğal olarak çok basitçe yaptığı konuşma ve konuşulanı anlama konuları gibi konularda çok daha zorlanabiliyorlar. Bu durum gerçekten çok ilgi çekici bir durum.

## 3. Kullanılan Ortam, Yöntem ve Kütüphaneler

Araştırmalarımın sonucunda en uygun ortamın python olduğuna karar verdim. Bu yönde yoğunlaştım ve ilerledim. Bu alanda kullanabileceğim en gelişmiş ve yaygın kütüphanelerin "NLTK (Natural Language Toolkit), SpaCy, TextBlob, Stanford CoreNLP, Gensim" olduğunu öğrendim. Açık kaynaklı kodlar üzerindeki çalışmalarım sonucunda bu problem için "SpaCy" kütüphanesini kullanmaya karar verdim. Kullanmış olduğum SpaCy sürümü "2.3.5" dir.

Ayrıca "en_core_web_sm==2.3.1" sürümünü ekledim.

Programın öğrenme yöntemi olarak, derin öğrenmenin bir dalı olan "Convolutional neural network" (CNN) kullandım. Doğal dil işleme alanında sıklıkla kullanılan bu yöntem ayrıca görüntü analizinde de çok tercih ediliyor.

Ayrıca çeviri yapma gibi ekstra fonksiyonlar ekledim.
Bunun için Google translate'i kullandım. Versiyon "googletrans 3.1.0a0".

## 4. Geliştirilen Yöntem

Geliştirim kısmında, öncelikle belli başlıklara ayırmanın faydalı olduğunu düşünüyorum, bunlar:

0) Doğal dil işleme kullanarak, metin verilerini temizleme ve kullanıma uygun hale getirme.
1) Makine öğrenimi kullanarak duygu analizi konusunda sınıflandırma yapmak.
2) Veri setini yüklemek ve programı eğitmek.
3) Başarım değerlendirmeleri yapmak.
4) İyileştirme çalışmaları.
5) Ekstra çalışmalar.

## 5. Gerçekleştirilen Adımlar ve İlgili Kaynak Kodlar

### 0. Doğal dil işleme kullanarak, metin verilerini temizleme ve kullanıma uygun hale getirme.

Bu adımla ilgili hemen her şeyi “SpaCy” hazır bir şekilde yapıyor.
Birkaç örnek kod göstermek gerekirse;

**Tokenizing**

```
>>> import spacy
>>> text = """
Dave watched as the forest burned up on the hill,
only a few miles from his house. The car had
been hastily packed and Marta was inside trying to round
up the last of the pets. "Where could she be?" he wondered
as he continued to wait for Marta to appear with the pets.
"""
>>> nlp = spacy.load("en_core_web_sm")
>>> doc = nlp(text)
>>> token_list = [token for token in doc]
>>> token_list
[
, Dave, watched, as, the, forest, burned, up, on, the, hill, ,,
, only, a, few, miles, from, his, house, ., The, car, had,
, been, hastily, packed, and, Marta, was, inside, trying, to, round,
, up, the, last, of, the, pets, ., ", Where, could, she, be, ?, ", he, wondered,
, as, he, continued, to, wait, for, Marta, to, appear, with, the, pets, .,
]
```

**Removing Stop Words**

```
>>> filtered_tokens = [token for token in doc if not token.is_stop]
>>> filtered_tokens
[
, Dave, watched, forest, burned, hill, ,,
, miles, house, ., car,
, hastily, packed, Marta, inside, trying, round,
, pets, ., ", ?, ", wondered,
, continued, wait, Marta, appear, pets, .,
]
```

**Normalizing Words**

```
>>> lemmas = [
...     f"Token: {token}, lemma: {token.lemma_}"
...     for token in filtered_tokens
... ]
>>> lemmas
['Token: \n, lemma: \n', 'Token: Dave, lemma: Dave',
 'Token: watched, lemma: watch', 'Token: forest, lemma: forest',
 # ...
]
```

1. **Makine öğrenimi kullanarak duygu analizi konusunda sınıflandırma yapmak.**

"SpaCy" "-textcat" olarak adlandırılan bir sınıflandırıcıya sahip.

**Bir pipeline oluşturmak gerekiyor.**

```
nlp = spacy.load("en_core_web_sm")
if "textcat" not in nlp.pipe_names:
    textcat = nlp.create_pipe(
        "textcat", config={"architecture": "simple_cnn"}
    )
    nlp.add_pipe(textcat, last=True)
else:
    textcat = nlp.get_pipe("textcat")

textcat.add_label("pos")
textcat.add_label("neg")
```

**Ardından eğitim döngüsü kurmak gerekiyor**

Bu kısım en önemli kısımlardan birisi. Bu kısımda sınıflandırma yaparak verileri optimize etmek, döngüler arası temiz bir öğrenme sözlüğü oluşturabilmek ve doğru şekilde besleme yapmak gerekiyor. Bu kısımda basit bir CNN yapısı oluşturdum ve bunu veri setiyle besledim.

Buradaki yapılması gerekenler üç basamaktan oluşuyor:

1. Base yani "ana" spacy pipeline'ındaki textcat component'ını eklemek.
2. Textcat component'ını eğitmek için bir loop'a yani bir döngüye sokmak.
3. Son olarakta belli bir sayıda döngüden sonra ilerlemeyi değerlendirmek.

```python
def train_model(                                                    # ⚠ 5
        training_data: list,
        test_data: list,
        iterations: int = 20
) -> None:
    nlp = spacy.load("en_core_web_sm")
    if "textcat" not in nlp.pipe_names:
        textcat = nlp.create_pipe(
            "textcat", config={"architecture": "simple_cnn"}
        )
        nlp.add_pipe(textcat, last=True)
    else:
        textcat = nlp.get_pipe("textcat")

    textcat.add_label("pos")
    textcat.add_label("neg")

    training_excluded_pipes = [
        pipe for pipe in nlp.pipe_names if pipe != "textcat"
    ]
    with nlp.disable_pipes(training_excluded_pipes):
        optimizer = nlp.begin_training()
        print("Beginning training")
        print("Loss\tPrecision\tRecall\tF-score")
        batch_sizes = compounding(
            4.0, 32.0, 1.001
        )
        for i in range(iterations):
            print(f"Training iteration {i}")
            loss = {}
            random.shuffle(training_data)                           # ⚠ 5
            batches = minibatch(training_data, size=batch_sizes)
            for batch in batches:
                text, labels = zip(*batch)
                nlp.update(text, labels, drop=0.2, sgd=optimizer, losses=loss)
            with textcat.model.use_params(optimizer.averages):
                evaluation_results = evaluate_model(
                    tokenizer=nlp.tokenizer,
                    textcat=textcat,
                    test_data=test_data
                )
                print(
                    f"{loss['textcat']}\t{evaluation_results['precision']}"
                    f"\t{evaluation_results['recall']}"
                    f"\t{evaluation_results['f-score']}"
                )

    with nlp.use_params(optimizer.averages):
        nlp.to_disk("model_artifacts")
```

## 2. Başarım değerlendirmeleri yapmak.

Değerlendirme yapmak için f-score yöntemini kullandım. Bunun için bir “precision” ve bir de “recall” değeri lazım. Veri seti kullanarak programımın tahminlerini 4 kategoriye ayıracak bir kod yazdım. Bunlar “ doğru pozitif ” “ yanlış pozitif ” “ doğru negatif ” ve “ yanlış negatif ”. Ardında gerekli matematiksel işlemler yapılıyor. Her ne kadar “precision” ve “recall” değerleri tek başlarına bir başarım verisi olsalarda, en güvenilir yöntemin bu iki verinin harmonik ortalaması olan f-score olduğunu öğrendim.

“precision” ve “recall” değerlerinin hesaplanması için kullandığım kısım

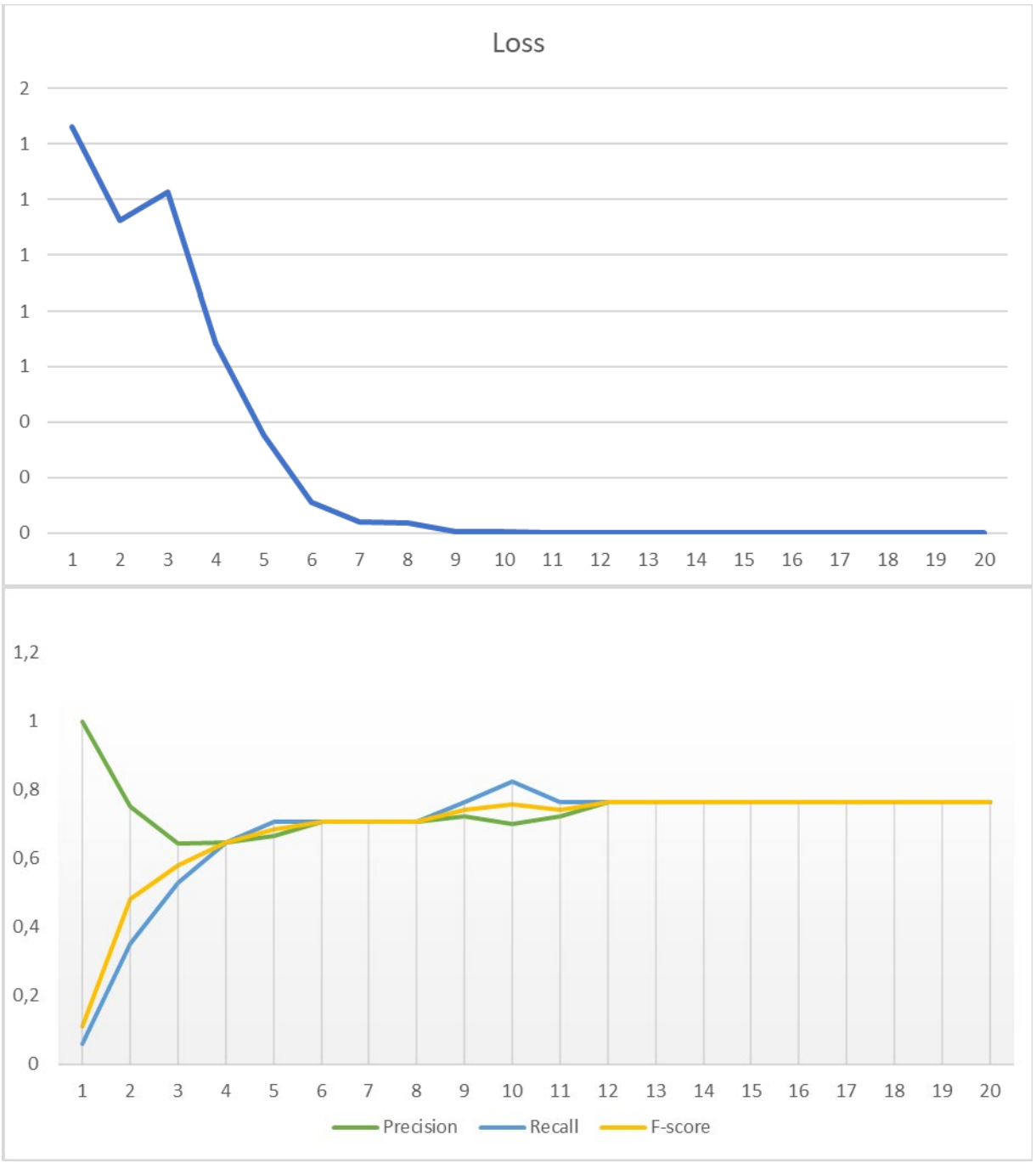
```python
def evaluate_model(
        tokenizer, textcat, test_data: list
) -> dict:
    reviews, labels = zip(*test_data)
    reviews = (tokenizer(review) for review in reviews)
    true_positives = 0
    false_positives = 1e-8 #payda sıfır olamaz
    true_negatives = 0
    false_negatives = 1e-8
    for i, review in enumerate(textcat.pipe(reviews)):
        true_label = labels[i]["cats"]
        for predicted_label, score in review.cats.items():

            if (
                    predicted_label == "neg"
            ):
                continue
            if score >= 0.5 and true_label["pos"]:
                true_positives += 1
            elif score >= 0.5 and true_label["neg"]:
                false_positives += 1
            elif score < 0.5 and true_label["neg"]:
                true_negatives += 1
            elif score < 0.5 and true_label["pos"]:
                false_negatives += 1
    precision = true_positives / (true_positives + false_positives) #kesinlik
    recall = true_positives / (true_positives + false_negatives)    #duyarlılık
```

f-score hesabı için kullandığım kısım

```python
    if precision + recall == 0:
        f_score = 0
    else:
        f_score = 2 * (precision * recall) / (precision + recall) #harmonik ortalama
    return {"precision": precision, "recall": recall, "f-score": f_score}
```

## 3. Deneysel çalışmalar.

Veriseti proje içerisinde mevcut. Ayrıca kaynakçada da belirtiliyor. Deneysel çalışmalarım sırasında karşıma çıkan değerleri grafik olarak aldığımda şu tablolar oluşuyor.




Bu tablolar ışığında ve eğitim verilerini arttırdıkça programın çok başarılı sonuçlar verdiği anlaşılıyor.

## 4. Sonuç

Projenin gerçekleştirme testlerinde yorumladığı sonuçlar oldukça başarılı. Küçük bir veri seti kullanıldığında bazı sapmalar olabiliyor. Bu sapmaların altında yatan sebep bazı yorumlar ve puanlarının çok da uyumlu olmaması ve CNN'in yeterli miktarda eğitilmemiş olması olur. Eğitim verisini arttırdığımda ise doğal olarak bu sorun giderilir.

Bu projenin günlük hayatta birçok alanda kullanılabileceğini düşünüyorum. Tabi ki ilk akla gelen alan e-ticaret alanı. Bu alanda şu anki durumuyla kullanılabileceği gibi; bazı geliştirmelerle çok kullanışlı bir araç olabilir. Örneğin sadece duygu analizi yerine;

yapılan yorumun ne hakkında (örneğin kargo firması hakkında, satıcı hakkında, ürün hakkında) olduğunu analiz eden bir program, müşteri memnuniyetini arttırmada etkili olacaktır.

Programı geliştirirken Doğal dil analizinin kullanım alanlarının genişliği hakkında bilgi sahibi oldum. Bu alanda kullanılan öğrenme yöntemlerini araştırdım. Derin öğrenme konusunda kendimi daha fazla geliştirmiş oldum. Doğal dil işleme alanındaki açık kaynaklı kütüphanelerin çokluğunu ve kullanışlılığını keşfederek bu alanda çalışmak isteyen insanların büyük bir literatüre ulaşabileceğini gördüm. Bu durum hem yapay zeka alanına hem de doğal dil işleme alanına olan ilgimi büyük oranda arttırdı.

## 5. Kaynakça


- https://realpython.com/python-nltk-sentiment-analysis/
- https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/
- Veri Seti: https://ai.stanford.edu/~amaas/data/sentiment/
- https://realpython.com/sentiment-analysis-python/
- https://spacy.io/usage/projects#textcat


## 6. Ek-7

Başarımı arttırmak için eğitim aşamasında rastgele veri kullanımının önemini fark ettim. Bu durum belli kelimeler özelinde yaşadığım hatalı sonuçların düzelmesini sağladı. Ayrıca veri setini genişletmek ve iterasyon sayısını arttırmak doğal olarak sonuçları iyileştirdi. Kullandığım kütüphane ve Unicode değişim biçimini güncel tutmanın da sonuçlarda iyileşmeler sağladığını gördüm.

Not:

Kaynakça kısmında belirttiğim sitelerdeki hazır kodların bir kısmından faydalandım. Çeviri kısmına ekledim. Gerçek zamanlı test kalitesini arttırmak için bir fonksiyon yazdım. Derin öğrenme kısmında güncellemeler yaparak efektifliği arttırdım.

**Özdeğerlendirme Tablosu**

| | İstenen Madde | Var | Açıklama | Not |
|---|---|---|---|---|
| 1 | **Kapak Sayfası, Problemin Tanımı, Kullanılan Ortam, Yöntem ve Kütüphaneler (10)** | ☒ | Detaylı bir şekilde işlendi | 10 |
| 2 | **Araştırma (10)** | ☒ | Proje zamanının büyük kısmı bu aşamada kullanıldı. | 10 |
| 3 | **Önerilen Yöntem (10)** | ☒ | Video ve raporda detaylı biçimde açıklandı. | 10 |
| 4 | **Deneysel Çalışmalar (15)** | ☒ | Farklı fonksiyonlar eklemek üzerine durdum. | 15 |
| 5 | **Proje Rapor Biçimi, Organizasyonu, Boyutu, Kalitesi, Kaynakça ve atıflar (10)** | ☒ | Rapor formatı ve düzeninden memnunum. | 10 |

| | | | | |
|---|---|---|---|---|
| **6** | **Sonuç (20)** | ☒ | Sonuç kısmında gereken yorumlar yapıldı. Detaylı bir proje açıklaması da eklendi. | 20 |
| **7** | **Ek 7: Başarım İyileştirme (15)** | ☒ | Sürekli nasıl daha iyi yapılabilir diye düşündüm. | 15 |
| **10** | **Özdeğerlendirme Tablosu (10)** | ☒ | Özdeğerlendirme tablosunu özenle doldurdum. | 10 |
| | | | **100 üzerinden Toplam Not:** | 100 |

**Harcanan Zaman.**

**Sinan Burak Filiz** - **Araştırma 15 saat**
**Kod yazımı 12 saat**
**Deney ve iyileştirme 6 saat**
**Yorumlama ve raporlama 4 saat**