YAPAY ZEKA VE NLP UYGULAMALARI

Transformer Mimarisinden RAG Sistemlerine Python ile Doğal Dil İşleme

Derleme Makale / Review Article

Dr. Burak Olgun
burakolgun@solutionhome.net
Solution Home Bilişim Tekn. ve Dan. Hizm., İstanbul

Anahtar Kelimeler: *NLP • Transformer • BERT • LLM • RAG • spaCy • HuggingFace • LangChain • Python*

burakolgun.com

ÖZET

Doğal Dil İşleme (Natural Language Processing — NLP), makinelerin insan dilini anlama, yorumlama ve üretme kapasitesini araştıran yapay zekanın en hızlı gelişen alt alanlarından biridir. 2017'de yayımlanan 'Attention Is All You Need' makalesi ve Transformer mimarisi, bu alanda bir paradigma dönüşümü yaratmış; BERT (2018), GPT-3 (2020) ve ardından gelen Büyük Dil Modelleri (LLM) NLP'yi araştırma laboratuvarlarından kurumsal üretim ortamlarına taşımıştır. Bu derleme makale; NLP'nin tarihsel evrimini, temel teknik kavramları (tokenizasyon, gömme, dikkat mekanizması), Transformer, BERT ve GPT mimarilerini, başlıca NLP görevlerini (duygu analizi, varlık tanıma, makine çevirisi, metin özetleme, soru-cevap), Retrieval-Augmented Generation (RAG) sistemlerini, spaCy ve HuggingFace Transformers kütüphanelerini ve LangChain çerçevesini kapsamlı biçimde ele almaktadır. Her bölüm, konuyu somutlaştıran Python kod örnekleriyle desteklenmektedir. Makale aynı zamanda NLP'nin etik boyutları, önyargı ve adalet sorunları ile yapay zekanın geleceğine ilişkin değerlendirmeleri de kapsamaktadır.

Anahtar Kelimeler: *doğal dil işleme, NLP, transformer, BERT, GPT, büyük dil modeli, LLM, duygu analizi, varlık tanıma, makine çevirisi, RAG, retrieval-augmented generation, spaCy, HuggingFace, LangChain, Python, dikkat mekanizması*

ABSTRACT

Natural Language Processing (NLP) is one of the fastest-growing subfields of artificial intelligence, studying the capacity of machines to understand, interpret, and generate human language. The 'Attention Is All You Need' paper published in 2017 and the Transformer architecture created a paradigm shift in

this field; BERT (2018), GPT-3 (2020), and the subsequent Large Language Models (LLMs) moved NLP from research laboratories to corporate production environments. This review article comprehensively covers the historical evolution of NLP, core technical concepts (tokenization, embedding, attention mechanism), Transformer, BERT, and GPT architectures, major NLP tasks (sentiment analysis, named entity recognition, machine translation, text summarization, question answering), Retrieval-Augmented Generation (RAG) systems, spaCy and HuggingFace Transformers libraries, and the LangChain framework. Each section is supported by Python code examples that concretize the topic.

Keywords: *natural language processing, NLP, transformer, BERT, GPT, large language model, LLM, sentiment analysis, named entity recognition, machine translation, RAG, retrieval-augmented generation, spaCy, HuggingFace, LangChain, Python, attention mechanism*

1. Giriş

İnsan dili, karmaşıklığı ve belirsizliğiyle bilgisayar biliminin en zorlu problemlerinden birini oluşturmaktadır. Aynı kelime farklı bağlamlarda farklı anlamlar taşıyabilir; aynı anlam farklı cümle yapılarıyla ifade edilebilir; bir metnin duygusal tonu, sözcüklerin ötesinde yazarın niyetini yansıtır. Bu karmaşıklıkla başa çıkma çabası, Doğal Dil İşleme (Natural Language Processing — NLP) disiplinini doğurmuştur.

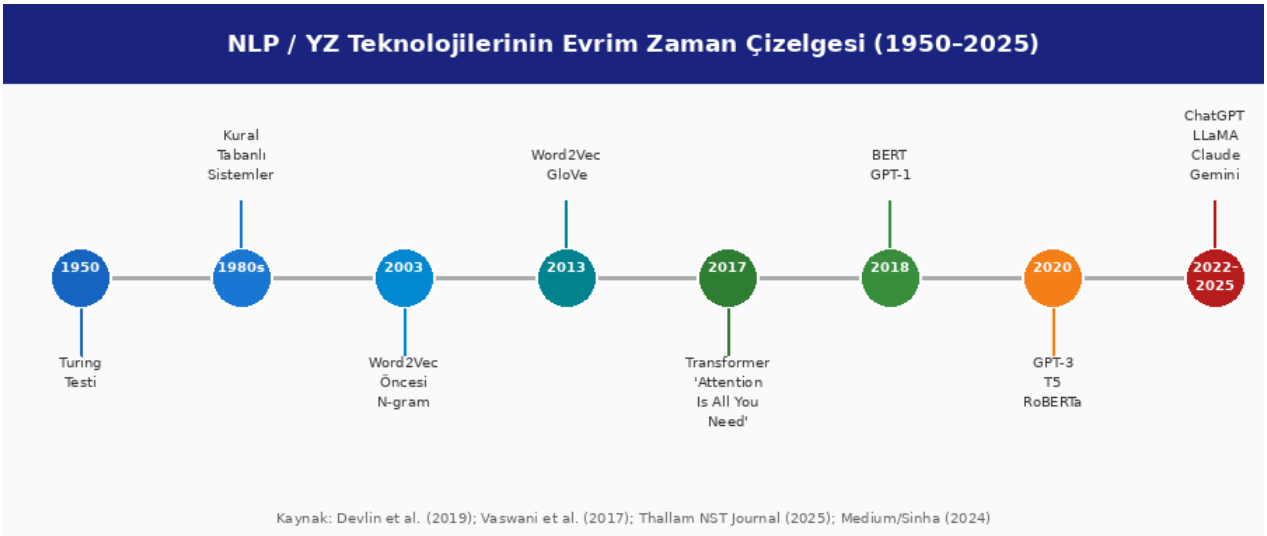
NLP'nin tarihsel yolculuğu, 1950'lerin kural tabanlı sistemlerinden başlayıp istatistiksel modeller, sinir ağları ve nihayetinde 2017'de Vaswani ve diğerlerinin yayımladığı 'Attention Is All You Need' çalışmasıyla tescillenen Transformer devrimine uzanmaktadır. Bu devrim, yalnızca araştırmayı değil; pratik uygulamaların ölçeğini ve niteliğini köklü biçimde dönüştürmüştür. ChatGPT'nin 2022'de beş gün içinde bir milyon kullanıcıya ulaşması, bu dönüşümün hız ve kapsamını somutlaştırmaktadır.

Thallam NST Journal'ın (2025) değerlendirmesine göre son beş yıl, büyük ölçekli önceden eğitilmiş dil modellerinin — BERT, GPT-3, PaLM, LLaMA — hâkimiyetiyle şekillenmiştir. Bu modeller; metin çevirisi, kod yazımı, hukuki belge analizi, tıbbi rapor özetleme ve müşteri hizmetleri otomasyonu gibi onlarca sektörde üretime girmiştir. 2024-2025 döneminde ise Retrieval-Augmented Generation (RAG) mimarileri, LLM'lerin bilgi güncelliği sınırlılığını aşmanın en etkili yolu olarak kurumsal ölçekte standartlaşmaktadır.

Bu makalenin amacı, NLP ve yapay zekanın birleşim noktasını — teknik derinlik, pratik Python örnekleri ve güncel araştırma çerçeveleriyle — kapsamlı biçimde ortaya koymaktır. Makale hem teorik hem de uygulamalı bir kaynak işlevi görerek alan araştırmacılarına, yazılım geliştiricilerine ve kurumsal karar vericilerine yönelik bilgi sunmaktadır.

burakbsn.com

2. NLP'nin Tarihsel Evrimi



Şekil 1. *NLP / Yapay Zeka Teknolojilerinin Tarihsel Evrim Zaman Çizelgesi (1950–2025)*

2.1 Kural Tabanlı Dönem (1950–1990)

NLP'nin ilk dönemi, dilin dilbilgisi kurallarıyla tanımlandığı ve bu kuralların açıkça kodlandığı sistemlere dayanmaktadır. 1950'lerde Alan Turing'in 'Computing Machinery and Intelligence' makalesi, makinelerin düşünüp düşünemeyeceği sorusunu gündeme taşıdı; bu soru zamanla NLP araştırmasının felsefi zeminine dönüştü. ELIZA (1966, MIT) ve SHRDLU (1970), kural tabanlı yaklaşımların ne kadar yüzeysel kaldığını ancak aynı zamanda ne kadar çarpıcı sonuçlar üretebileceğini gösterdi.

1980'lerde uzman sistemler ve bağlam serbest dilbilgileri (context-free grammars) NLP araştırmasına hâkim oldu. Bu dönemde Prolog tabanlı doğal dil ayrıştırıcıları akademik ilgi gördü; ancak sistemler gerçek dünya dil çeşitliliğini işleme konusunda ölçeklenemiyor, yeni bir dile ya da alana kolayca uyarlanamıyordu. Tucudean ve diğerlerinin (2024) PeerJ CS değerlendirmesine göre, bu dönemin kısıtlaması tam da bu noktadaydı: 'RNN ve LSTM'ler kısa diziler için etkiliydi; ancak kaybolan gradyan ve paralel olmayan işlem, uzun bağımlılıkları yakalamayı sınırlıyordu.'

2.2 İstatistiksel Dönem (1990–2012)

1990'ların başında istatistiksel yaklaşımlar kural tabanlı sistemlerin yerini almaya başladı. Büyük metin derlemleri (corpus) üzerinde eğitilen n-gram dil modelleri, saklı Markov modelleri (HMM) ve destek vektör makineleri (SVM) bu dönemin baskın araçlarıydı. Google Çeviri'nin 2006'da istatistiksel makine çevirisine geçişi, bu paradigmanın endüstriyel ölçekte uygulanabilirliğini kanıtladı.

Word2Vec (Mikolov ve diğerleri, 2013) bu dönemin kapanış şaheserini temsil etmektedir. Kelimelerin 300 boyutlu vektörlerle temsil edilmesi — anlamsal benzerliklerin geometrik mesafeye dönüşmesi — 'Kral − Erkek + Kadın = Kraliçe' gibi analogilerle somutlaşan bu yaklaşım, gömme (embedding) kavramını NLP'nin merkezine yerleştirdi.

2.3 Derin Öğrenme ve Transformer Dönemi (2013–Günümüz)

2013-2017 arası, derin öğrenme mimarilerinin NLP'ye hâkim olduğu geçiş dönemidir. LSTM ve GRU ağları sıralı veri işlemede önemli gelişmeler sağladı; ancak Vaswani ve diğerlerinin 2017'de yayımladığı 'Attention Is All You Need' makalesi bu tabloyu köklü biçimde değiştirdi. Transformer mimarisi; yinelemeli yapıları tamamen ortadan kaldırarak dikkat mekanizmasını (attention mechanism) hesaplamanın merkezine yerleştirdi. Bu sayede paralel eğitim mümkün hale geldi ve milyarlarca parametreli modellerin geliştirilmesinin önü açıldı.

Devlin ve diğerlerinin (2019) BERT'i, Radford ve diğerlerinin GPT serisini ve arkasından gelen T5, RoBERTa, XLNet ve ALBERT modelleri; NLP benchmarklarını sürekli yeniden yazan bir yarış başlattı. GLUE benchmark'ında yüzde seksenin üzerinde doğruluk ve SQuAD QA veri setinde F1 skorunun doksanın üzerine çıkması, bu dönemin teknik çıktılarını simgelemektedir (arxiv, 2025). 2022'de ChatGPT'nin halka açılmasıyla bu teknolojiler araştırmadan günlük yaşama geçişini tamamladı.

3. Temel Teknik Kavramlar

3.1 Tokenizasyon

Tokenizasyon, bir metin dizisini modelin işleyebileceği daha küçük birimlere (token) bölme sürecidir. Kelime düzeyinde, karakter düzeyinde ve alt-kelime (subword) düzeyinde olmak üzere üç temel yaklaşım mevcuttur. Modern LLM'lerin büyük çoğunluğu, kelime dağarcığını kontrol altında tutarken nadir kelimeleri de işleyebilen BPE (Byte Pair Encoding) veya WordPiece alt-kelime tokenizasyonu kullanmaktadır. HuggingFace'in BERT tokenizer'ı bu yaklaşımın referans uygulamasıdır.

Örnek 1: HuggingFace BERT Tokenizer ile Tokenizasyon

```
from transformers import BertTokenizer

# BERT tokenizer'ı yükle
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

metin = "Yapay zeka NLP alanında devrim yarattı."

# Tokenizasyon
tokenlar = tokenizer.tokenize(metin)
print("Tokenlar:", tokenlar)
# Çıktı: ['ya', '##pay', 'za', '##ka', 'nlp', 'alaninda', 'devrim',
'yaratti', '.']


# ID'lere dönüştürme
token_ids = tokenizer.encode(metin, return_tensors='pt')
print("Token ID'leri:", token_ids)

# Toplu işlem için padding ile encode
encoded = tokenizer(
    [metin, "BERT güçlü bir dil modelidir."],
    padding=True,
    truncation=True,
    max_length=128,
    return_tensors='pt'
)
print("input_ids şekli:", encoded['input_ids'].shape)
```

3.2 Gömme (Embedding) Temsilleri

Kelime gömmeleri (word embeddings), kelimelerin anlamsal ve sözdizimsel özelliklerini yoğun vektörlerle temsil eden sayısal gösterimlerdir. Word2Vec ve GloVe, statik gömme üretirken ('banka' kelimesi hangi bağlamda kullanılırsa kullanılsın aynı vektörü alır); BERT ve türevleri, bağlama bağlı dinamik gömme üretmektedir. Bu fark, anlam belirsizliği (polysemy) sorununu çözmede belirleyici öneme sahiptir.

Örnek 2: Sentence-Transformers ile Anlamsal Benzerlik Hesaplama

```
from sentence_transformers import SentenceTransformer
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np

# Model yükleme
model = SentenceTransformer('paraphrase-multilingual-mpnet-base-v2')

cumleler = [
    "Yapay zeka iş dünyasını dönüştürüyor.",
    "Makine öğrenmesi endüstride devrim yaratıyor.",
    "Türkiye'de hava durumu değişken seyretti."
]

# Gömme üretimi
gommeler = model.encode(cumleler)

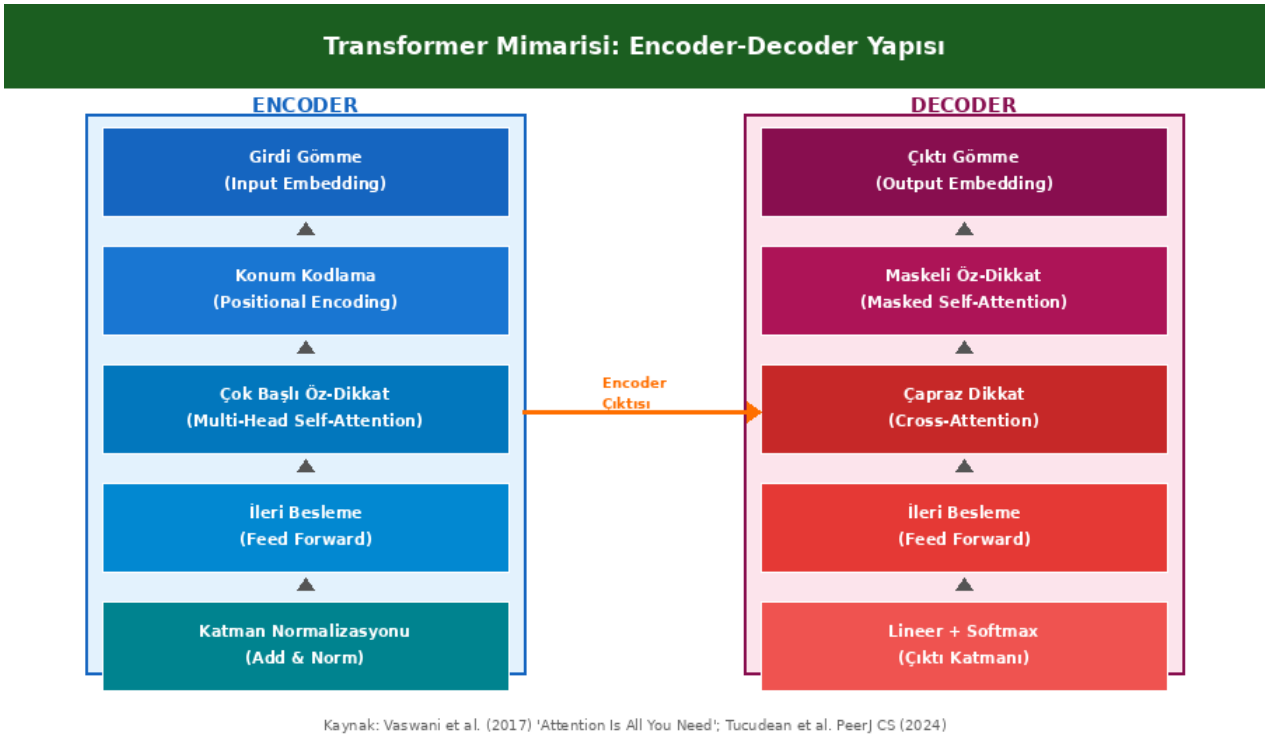
# Cosine benzerlik matrisi
benzerlik = cosine_similarity(gommeler)
print("Cümle 1 vs 2 benzerliği:", round(benzerlik[0][1], 3))
print("Cümle 1 vs 3 benzerliği:", round(benzerlik[0][2], 3))
# Beklenen çıktı: 0.1 ile 2 arası yüksek benzerlik (yakın anlam)
#                 1 ile 3 arası düşük benzerlik (farklı konu)
```

3.3 Dikkat Mekanizması (Attention Mechanism)

Dikkat mekanizması, bir modelin çıktı üretirken giriş dizisinin farklı bölümlerine ağırlıklı odaklanmasını sağlayan matematiksel çerçevedir. Vaswani ve diğerlerinin (2017) Transformer makalesinde sunulan Scaled Dot-Product Attention şu formülle tanımlanmaktadır: Attention(Q, K, V) = softmax(QK^T / √d_k)V

Bu formülde Q (Query), K (Key) ve V (Value) matrisleri, giriş gömme vektörlerinden öğrenilen dönüşümlerdir. √d_k ile ölçekleme, gradyan patlamasını önler. Çok Başlı Dikkat (Multi-Head Attention), bu mekanizmanın paralel olarak farklı temsil alt-uzaylarında uygulandığı bir genişlemedir. Bu yapı, modelin aynı anda hem sözdizimsel hem anlamsal ilişkilere odaklanmasına olanak tanır.

4. Transformer Mimarisi



Şekil 2. *Transformer Mimarisi: Encoder-Decoder Yapısı (Vaswani et al., 2017)*

4.1 Encoder-Decoder Yapısı

Transformer mimarisi iki temel bileşenden oluşmaktadır: Encoder ve Decoder. Encoder, giriş metnini anlamsal bir temsile dönüştürür; Decoder bu temsili kullanarak çıktı metnini üretir. Her Encoder katmanı; Çok Başlı Öz-Dikkat (Multi-Head Self-Attention) ve İleri Besleme Ağı (Feed-Forward Network) alt katmanlarından oluşur; her alt katmanın çevresinde artık bağlantı (residual connection) ve katman normalizasyonu bulunmaktadır.

Konum Kodlama (Positional Encoding), Transformer'ın sıralı yapıyı yakalamasını sağlayan kritik bir bileşendir. Yinelemeli yapı olmadığından model, kelimelerin sırasını doğal olarak bilmez; sinüs ve kosinüs tabanlı konum kodlamaları bu bilgiyi gömme vektörlerine enjekte eder. Tucudean ve diğerlerinin (2024) değerlendirmesine göre bu mimari, 'uzun bağımlılıkları yakalamada ve paralel eğitimde RNN'lerin üstesinden gelemediği kısıtlamaları aşmaktadır.'

4.2 BERT: Çift Yönlü Encoder

BERT (Bidirectional Encoder Representations from Transformers), Google tarafından 2018'de geliştirilen ve yalnızca Transformer'ın encoder kısmını kullanan bir önceden eğitilmiş dil modelidir. BERT'i önceki modellerden ayıran iki temel özellik bulunmaktadır. Birincisi, çift yönlü bağlam: Geleneksel dil modelleri metni soldan sağa işlerken BERT, maskeli dil modellemesi (MLM) sayesinde her kelimenin bağlamını hem önceki hem sonraki kelimelerden eş zamanlı öğrenir.

Springer Nature'ın (2025) kapsamlı BERT inceleme çalışmasına göre bu model; cümle sınırı tespiti, tokenizasyon, dilbilgisi hatası tespiti, bağımlılık ayrıştırma, varlık tanıma, kelime türü etiketleme, soru-cevap sistemleri, makine çevirisi, duygu analizi ve çapraz dil transferi gibi çok çeşitli NLP görevlerinde uygulanmaktadır. BERT'in mimarisi iki büyüklükte sunulmaktadır: BERT-Base (12 katman, 110M parametre) ve BERT-Large (24 katman, 340M parametre).

Örnek 3: BERT ile İnce Ayar (Fine-Tuning) — Metin Sınıflandırma

```
from transformers import (BertTokenizer, BertForSequenceClassification,
                          Trainer, TrainingArguments)
from datasets import Dataset
import torch


# Model ve tokenizer yükleme
model_adi = 'dbmdz/bert-base-turkish-cased'  # Türkçe BERT
tokenizer = BertTokenizer.from_pretrained(model_adi)
model = BertForSequenceClassification.from_pretrained(model_adi,
num_labels=2)

# Örnek veri hazırlama
veriler = {
    'text': [
        'Bu ürün harika, çok memnun kaldım!',
        'Berbat bir deneyimdi, hiç beğenmedim.',
        'Kaliteli ürün, tavsiye ederim.',
        'Param boşa gitti, işe yaramadı.'
    ],
    'label': [1, 0, 1, 0]  # 1=pozitif, 0=negatif
}

def tokenize_fn(ornek):
    return tokenizer(ornek['text'], truncation=True,
                     padding='max_length', max_length=128)

dataset = Dataset.from_dict(veriler).map(tokenize_fn, batched=True)
dataset = dataset.train_test_split(test_size=0.25)

# Eğitim ayarları
egitim_args = TrainingArguments(
    output_dir='./bert_siniflandirma',
    num_train_epochs=3,
    per_device_train_batch_size=8,
    evaluation_strategy='epoch',
    save_strategy='epoch',
    load_best_model_at_end=True,
)

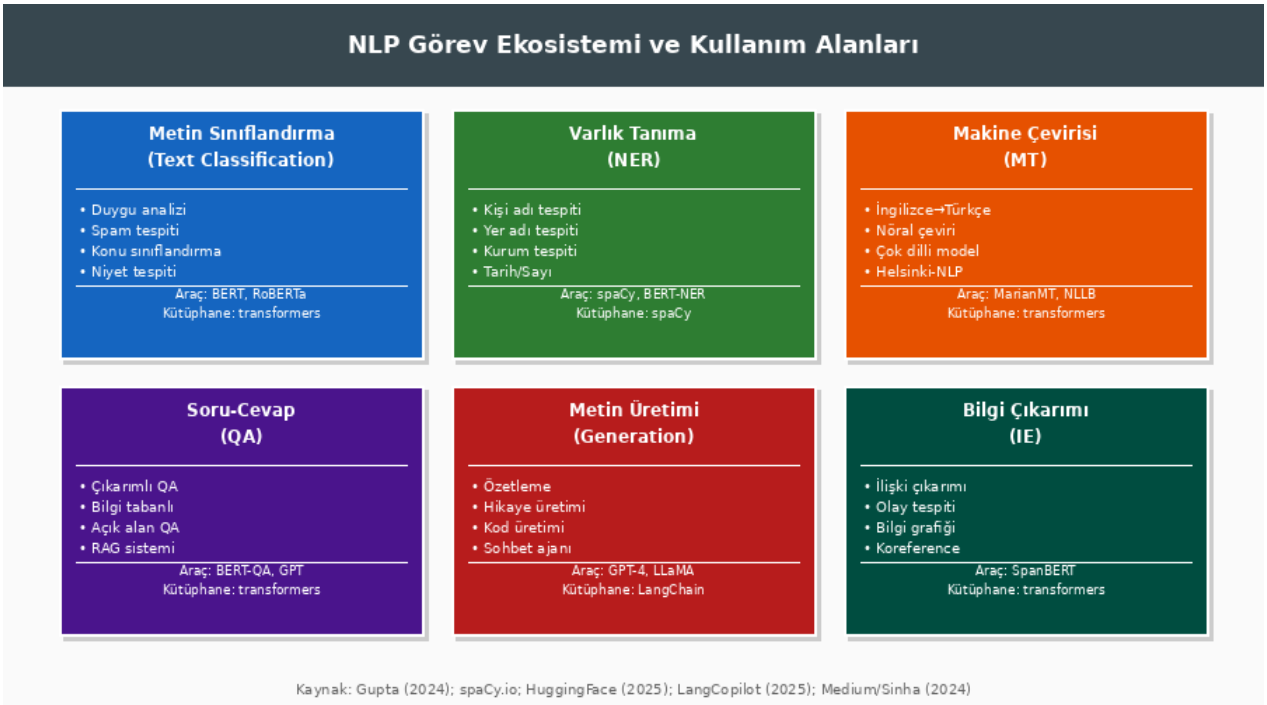
trainer = Trainer(
    model=model,
    args=egitim_args,
    train_dataset=dataset['train'],
    eval_dataset=dataset['test'],
)
# trainer.train()  # Gerçek eğitim için etkinleştirin
```

4.3 GPT Serisi: Otoregresif Decoder Modeller

GPT (Generative Pre-trained Transformer) serisi, Transformer'ın yalnızca decoder kısmını kullanan otoregresif bir model ailesidir. Her token, yalnızca önceki tokenlara bakarak üretilir (soldan sağa yönlü dikkat). GPT-1 (2018) bu yaklaşımı tanıtırken GPT-3 (2020) 175 milyar parametre ile 'few-shot learning' kapasitesini kanıtladı; GPT-4 (2023) multimodal yetenekler kazandı.

Medium/Sinha'nın (2024) değerlendirmesine göre '2024'te GPT-4, PaLM ve LLaMA gibi modeller milyarlarca parametreyle insan benzeri metin üretebilmekte, sorulara yanıt verebilmekte ve diller arasında çeviride giderek daha yüksek akıcılık sergilemektedir.' LLaMA 2 ve 3 (Meta), Mistral ve Gemma gibi açık kaynaklı modeller ise bu kapasiteyi akademik ve kurumsal araştırmacıların erişimine açmıştır.

5. Temel NLP Görevleri ve Python Uygulamaları



Şekil 3. *NLP Görev Ekosistemi: Altı Ana Kategori ve Kullanım Alanları*

5.1 Duygu Analizi (Sentiment Analysis)

Duygu analizi, bir metin parçasının duygusal tonunu — olumlu, olumsuz veya nötr — belirleme görevidir. Müşteri geri bildirimi analizi, sosyal medya izleme ve ürün incelemesi değerlendirme bu görevin başlıca uygulama alanlarını oluşturmaktadır. Springer Nature'ın (2025) BERT incelemesine göre BERT tabanlı duygu analizi modelleri, Stanford SST-2 veri setinde yüzde doksan dört f1 skoru elde etmektedir.

Örnek 4: HuggingFace Pipeline ile Duygu Analizi

```
from transformers import pipeline

# Duygu analizi pipeline'ı — otomatik model seçimi
duygu_analizci = pipeline('sentiment-analysis',
                          model='cardiffnlp/twitter-xlm-roberta-base-
sentiment')

yorumlar = [
    'Bu kütüphane gerçekten mükemmel, çok işe yaradı!',
    'Hayal kırıklığı yarattı, beklentilerimi karşılamadı.',
    'Fena değil ama geliştirilmesi gereken yönleri var.',
]
```

```
for yorum in yorumlar:
    sonuc = duygu_analizci(yorum)[0]
    print(f"Metin : {yorum[:50]}...")
    print(f"Duygu : {sonuc['label']} (güven: {sonuc['score']:.2%})")
    print('-' * 60)

# spaCy + TextBlob ile alternatif yaklaşım
import spacy
from textblob import TextBlob

nlp = spacy.load('en_core_web_sm')
metin_en = 'The product quality is excellent and delivery was fast!'
blob = TextBlob(metin_en)
print(f"Polarite: {blob.sentiment.polarity:.2f}")   # -1 → +1
print(f"Öznellik: {blob.sentiment.subjectivity:.2f}") # 0 → 1
```

5.2 İsimli Varlık Tanıma (Named Entity Recognition — NER)

NER, metindeki kişi adı, kurum adı, yer adı, tarih, para birimi gibi önceden tanımlanmış kategorilerdeki varlıkları tespit eden bir sınıflandırma görevidir. Hukuki belge analizi, biyomedikal bilgi çıkarımı, haber akışı işleme ve iş süreçleri otomasyonunda kritik rol üstlenmektedir. LegalLens-2024 paylaşımlı görevinde spaCy + DeBERTa kombinasyonu, yasal NER alt görevinde yüzde 86,3 F1 skoru elde etmiştir (Meghdadi ve Inkpen, 2024).

Örnek 5: spaCy ile Varlık Tanıma ve Görselleştirme

```
import spacy
from spacy import displacy

# Modeli yükleme
nlp = spacy.load('tr_core_news_trf')  # Türkçe transformer modeli
# Alternatif: nlp = spacy.load('en_core_web_trf')  # İngilizce

metin = """Apple CEO'su Tim Cook, 15 Kasım 2024'te İstanbul'u
           ziyaret ederek Koç Holding ile stratejik ortaklık
           görüşmeleri yaptı. Anlaşma değerinin 500 milyon dolar
           olduğu belirtildi."""

doc = nlp(metin)

# Varlıkları yazdırma
print("Tespit edilen varlıklar:")
for varlik in doc.ents:
    print(f"  {varlik.text:<30} [{varlik.label_}]")

# Çıktı örneği:
# Apple                          [ORG]
# Tim Cook                       [PER]
# 15 Kasım 2024                  [DATE]
# İstanbul                       [LOC]
# Koç Holding                    [ORG]
# 500 milyon dolar               [MONEY]

# HTML görselleştirme (Jupyter/tarayıcıda kullanım için)
# displacy.render(doc, style='ent', jupyter=True)
```

5.3 Makine Çevirisi

Makine çevirisi, bir dilden diğerine otomatik metin dönüşümüdür. Transformer mimarisinin ortaya çıkışıyla birlikte nöral makine çevirisi (NMT) piyasaya hâkim olmakta; Helsinki-NLP'nin MarianMT, Meta'nın NLLB-200 (No Language Left Behind) ve Google'ın T5 modeli

bu alandaki öncü açık kaynaklı çerçeveleri oluşturmaktadır. NLLB-200, 200 dil çiftinde çeviri yapabilen, özellikle az kaynaklı diller için tasarlanmış kapsamlı bir modeldir.

Örnek 6: Helsinki-NLP MarianMT ile Makine Çevirisi

```
from transformers import MarianMTModel, MarianTokenizer

# İngilizce → Türkçe çeviri modeli
model_adi = 'Helsinki-NLP/opus-mt-en-tr'
tokenizer = MarianTokenizer.from_pretrained(model_adi)
model = MarianMTModel.from_pretrained(model_adi)

def cevir(metin_listesi, src_lang='en', tgt_lang='tr'):
    """Metin listesini hedef dile çevirir."""
    tokenized = tokenizer(metin_listesi, return_tensors='pt',
                          padding=True, truncation=True, max_length=512)
    cevrilmis = model.generate(**tokenized,
                               num_beams=4,
                               max_length=512,
                               early_stopping=True)
    return [tokenizer.decode(t, skip_special_tokens=True) for t in
cevrilmis]

ingilizce_metinler = [
    'Natural language processing is transforming the way we interact with
computers.',
    'Large language models have billions of parameters.',
    'Deep learning techniques have revolutionized NLP research.'
]

ceviri_sonuclari = cevir(ingilizce_metinler)
for kaynak, hedef in zip(ingilizce_metinler, ceviri_sonuclari):
    print(f'EN: {kaynak}')
    print(f'TR: {hedef}')
    print()
```

5.4 Metin Özetleme (Text Summarization)

Metin özetleme, uzun bir belgenin özünü koruyarak daha kısa bir biçimde sunma görevidir. Çıkarımlı (extractive) özetleme, orijinal belgeden önemli cümleler seçerken; soyutlayıcı (abstractive) özetleme yeni cümleler üretir. BART ve T5 modelleri soyutlayıcı özetlemede en yüksek performansı sergileyen mimariler arasındadır.

Örnek 7: BART ile Soyutlayıcı Metin Özetleme

```
from transformers import pipeline

# Özetleme pipeline'ı
ozetleyici = pipeline('summarization', model='facebook/bart-large-cnn')

uzun_metin = """
Natural language processing (NLP) is a subfield of linguistics, computer
science,
and artificial intelligence concerned with the interactions between
computers and
human language, in particular how to program computers to process and
analyze
large amounts of natural language data. The goal is a computer capable of
understanding the contents of documents, including the contextual nuances of
the language within them. The technology can then accurately extract
information
and insights contained in the documents, as well as categorize and organize
the documents themselves. The field has seen rapid advances in recent years,
driven by the development of transformer-based models like BERT and GPT,
which have achieved state-of-the-art results on many NLP benchmarks.
"""

ozet = ozetleyici(uzun_metin,
```

```
                  max_length=100,
                  min_length=30,
                  do_sample=False)

print("Özet:", ozet[0]['summary_text'])

# Çok dilli özetleme için mT5 alternatifi
# ozetleyici_tr = pipeline('summarization',
model='csebuetnlp/mT5_multilingual_XLSum')
```

5.5 Soru-Cevap Sistemleri (Question Answering)

Soru-cevap (QA) sistemleri, bir soru verildiğinde ilgili metinden ya da bilgi tabanından yanıt üretir. Çıkarımlı QA (extractive QA), yanıtı metin içinden bir span olarak seçerken; üretici QA (generative QA) yeni cümleler oluşturur. SQuAD veri setinde BERT tabanlı modeller, insan performansına yaklaşmış ya da bazı alt görevlerde geçmiştir.

Örnek 8: BERT ile Çıkarımlı Soru-Cevap

```
from transformers import pipeline

# QA pipeline
qa_modeli = pipeline('question-answering',
                     model='deepset/xlm-roberta-base-squad2')

baglam = """
Transformer mimarisi 2017 yılında Google Brain ekibi tarafından
'Attention Is All You Need' başlıklı bir makale ile tanıtıldı.
Bu mimari, dikkat mekanizmasını (attention mechanism) kullanarak
yinelemeli yapılara olan ihtiyacı ortadan kaldırdı ve paralel
eğitimi mümkün kıldı. BERT, 2018 yılında Google tarafından
geliştirilen ve Transformer'ın yalnızca encoder kısmını kullanan
önceden eğitilmiş bir dil modelidir.
"""

sorular = [
    "Transformer mimarisi hangi yılda tanıtıldı?",
    "BERT'i kim geliştirdi?",
    "Dikkat mekanizması ne sağlar?",
]

for soru in sorular:
    yanit = qa_modeli(question=soru, context=baglam)
    print(f"Soru  : {soru}")
    print(f"Yanıt : {yanit['answer']}")
    print(f"Güven : {yanit['score']:.2%}")
    print()
```

6. NLP Kütüphaneleri: spaCy, NLTK ve Karşılaştırma

6.1 spaCy: Üretim Odaklı NLP

spaCy, Explosion tarafından geliştirilen ve 'endüstriyel güçte NLP' sloganıyla sunulan açık kaynaklı Python kütüphanesidir. Hızı, üretim ortamı olgunluğu ve transformer desteğiyle öne çıkan spaCy; tokenizasyon, morfolojik analiz, sözdizimi ayrıştırma (dependency parsing), NER, metin sınıflandırma ve anlamsal benzerlik hesaplama bileşenlerini tek bir tutarlı API altında sunar. spaCy v3'ten itibaren RoBERTa ve BERT tabanlı transformer pipeline'ları desteklenmekte; OntoNotes 5.0 benchmark'ında yüzde doksan üzerinde NER doğruluğu elde edilmektedir (spaCy, 2025).

Örnek 9: spaCy ile Kapsamlı Metin Analizi

```
import spacy

nlp = spacy.load('en_core_web_sm')

metin = """OpenAI released GPT-4 in March 2023.
Microsoft invested $10 billion in the company.
Sam Altman serves as the CEO of OpenAI."""

doc = nlp(metin)

# 1. Tokenizasyon ve kelime türü etiketleme
print("=== Kelime Türü Etiketleri ===")
for token in doc[:8]:
    print(f"{token.text:<15} {token.pos_:<8} {token.dep_:<15}
{token.lemma_}")

# 2. Varlık tanıma
print("\n=== Varlıklar ===")
for ent in doc.ents:
    print(f"{ent.text:<25} {ent.label_:<12} {spacy.explain(ent.label_)}")

# 3. Bağımlılık ayrıştırma
print("\n=== Bağımlılık İlişkileri ===")
for token in doc:
    if token.dep_ in ('nsubj', 'dobj', 'ROOT'):
        print(f"{token.text:<15} --{token.dep_}--> {token.head.text}")

# 4. Cümle sınırları
print("\n=== Cümleler ===")
for i, sent in enumerate(doc.sents):
    print(f"Cümle {i+1}: {sent.text.strip()[:80]}")
```

6.2 NLTK: Eğitim ve Araştırma Platformu

NLTK (Natural Language Toolkit), Python'da NLP araştırma ve eğitimi için kullanılan köklü bir kütüphanedir. spaCy ile kıyaslandığında NLTK; daha geniş bir dil desteği, kapsamlı bir corpus ve sözlük koleksiyonu ve eğitimsel açıdan anlaşılır modüler yapısıyla öne çıkar. DSStream'in (tarihsiz) karşılaştırmasına göre 'NLTK araştırmacılar ve eğitimciler arasında kapsamlı belgelemeleri ve büyük toplulukları nedeniyle popülerdir; modeli sıfırdan inşa etmek istiyorsanız iyi bir seçimdir.'

Örnek 10: NLTK ile Temel NLP İşlemleri

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
from nltk import FreqDist, pos_tag

# Gerekli kaynakları indir (ilk çalıştırmada)
#
nltk.download(['punkt','stopwords','wordnet','averaged_perceptron_tagger'])

metin = """Natural language processing enables machines to understand
human language. Modern NLP systems use transformer architectures.
These models are trained on massive text corpora."""

# Cümle ve kelime tokenizasyonu
cumleler = sent_tokenize(metin)
kelimeler = word_tokenize(metin.lower())
print(f"Cümle sayısı: {len(cumleler)}, Kelime sayısı: {len(kelimeler)}")

# Stop word kaldırma
dur_kelimeler = set(stopwords.words('english'))
temiz = [k for k in kelimeler if k.isalpha() and k not in dur_kelimeler]
```

```
print(f"Stop word sonrası token sayısı: {len(temiz)}")

# Stemming ve Lemmatization
stemmer = PorterStemmer()
lemmatizer = WordNetLemmatizer()
print("\nKelime | Stem | Lemma")
for kelime in ['processing', 'enabling', 'systems', 'trained']:
    print(f"{kelime:<15} {stemmer.stem(kelime):<15}
{lemmatizer.lemmatize(kelime)}")

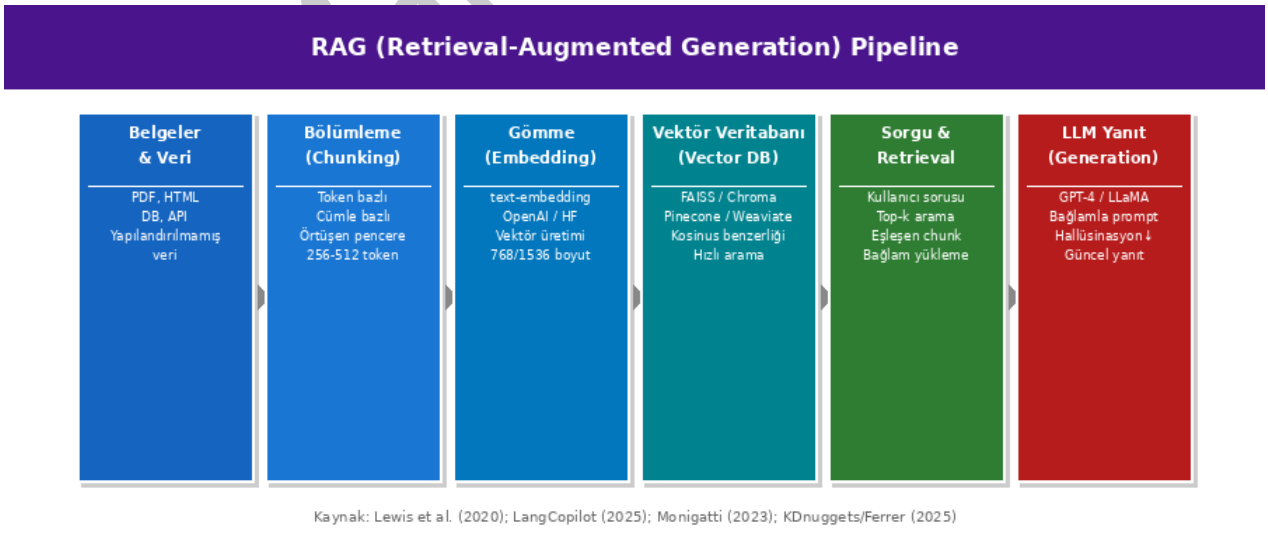
# Frekans analizi
frekans = FreqDist(temiz)
print("\nEn sık 5 kelime:")
for kelime, sayi in frekans.most_common(5):
    print(f"  {kelime:<15} : {sayi} kez")
```

Tablo 1. *spaCy ve NLTK: Özellik Karşılaştırması*

Özellik	spaCy	NLTK
Birincil Hedef	Üretim ortamı	Araştırma ve eğitim
Hız	Çok Hızlı (Cython ile optimize)	Orta (Python tabanlı)
API Tasarımı	Nesne yönelimli, akıcı	Modüler, fonksiyonel
Transformer Desteği	**Evet (v3+, yerel)**	Hayır (eklenti gerekli)
Dil Desteği	60+ dil (hazır model)	80+ dil (corpus bazlı)
Görselleştirme	displacy (yerleşik)	Sınırlı (matplotlib ile)
Öğrenme Eğrisi	Orta	Düşük

Not. Kaynak: DSStream (tarihsiz); spaCy.io (2025); Medium/Srivastava (2023) temel alınarak derlenmiştir.

7. Retrieval-Augmented Generation (RAG)



Şekil 4. *RAG Pipeline: Belgelerden LLM Yanıtına Altı Aşamalı Süreç*

7.1 RAG Nedir ve Neden Önemlidir?

Retrieval-Augmented Generation (RAG), LLM'lerin bilgi kesim tarihi (knowledge cutoff) kısıtlamasını ve hallüsinasyon (uydurma) sorununu çözmek için geliştirilmiş mimarı bir yaklaşımdır. Lewis ve diğerlerinin 2020 tarihli NeurIPS makalesinde tanıtılan bu yaklaşım; bilgi erişimini (retrieval) üretici modelle (generative model) birleştirerek LLM'in özel veya güncel verilere erişimini mümkün kılar.

KDnuggets'taki Ferrer'in (2025) değerlendirmesine göre: 'GPT ve LLaMA gibi LLM'ler sorulara yanıt vermekte mükemmel olmakla birlikte eğitim verileriyle sınırlıdır. Özel verilere erişemez ya da eğitim kesim tarihinin ötesinde öğrenemez. RAG, bu boşluğu doldurmaktadır.' LangCopilot'ın (2025) piyasa analizine göre RAG çerçeveleri 2025'te standart kurumsal AI yığınının ayrılmaz bileşeni haline gelmiştir.

7.2 RAG Pipeline Bileşenleri

Tam bir RAG pipeline'ı altı ana bileşenden oluşmaktadır. Birinci bileşen, belge yükleme ve bölümleme (chunking) aşamasıdır. LangCopilot'ın (2025) araştırması, optimum chunk boyutunun 256-512 token ve yüzde on ila yirmi örtüşme oranıyla sağlandığını ortaya koymaktadır. İkinci bileşen, gömme (embedding) üretimidir: Her chunk, semantic benzerlik araması için yoğun vektöre dönüştürülür.

Üçüncü bileşen, vektör veritabanıdır: FAISS (Facebook AI), Chroma, Pinecone veya Weaviate bu rol için kullanılan başlıca platformlardır. Dördüncü bileşen, sorgu zamanı retrieval'dır: Kullanıcı sorusu gömmeye dönüştürülür ve en benzer chunk'lar cosine benzerliğiyle bulunur. Beşinci bileşen, bağlam oluşturma ve prompt mühendisliğidir. Altıncı bileşen, LLM'in bağlamla zenginleştirilmiş prompt'u işleyip yanıt üretmesidir.

Örnek 11: LangChain ile Temel RAG Pipeline

```
from langchain_community.document_loaders import TextLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain_community.vectorstores import FAISS
from langchain.chains import RetrievalQA
from langchain_community.llms import HuggingFacePipeline
from transformers import pipeline as hf_pipeline

# 1. Belge yükleme
loader = TextLoader('belgeler/sirket_politika.txt', encoding='utf-8')
belgeler = loader.load()

# 2. Bölümleme
bolumleyici = RecursiveCharacterTextSplitter(
    chunk_size=512,
    chunk_overlap=64,
    separators=['\n\n', '\n', '.', ' ']
)
bolumler = bolumleyici.split_documents(belgeler)
print(f"Toplam bölüm sayısı: {len(bolumler)}")

# 3. Gömme modeli ve vektör veritabanı
gomme_modeli = HuggingFaceEmbeddings(
    model_name='sentence-transformers/paraphrase-multilingual-mpnet-base-v2'
)
vektor_db = FAISS.from_documents(bolumler, gomme_modeli)
vektor_db.save_local('faiss_indeks')

# 4. LLM ve QA zinciri
llm_pipeline = hf_pipeline(
```

wolfun.com

```
    'text-generation',
    model='microsoft/DialoGPT-medium',
    max_new_tokens=256
)
llm = HuggingFacePipeline(pipeline=llm_pipeline)

qa_zinciri = RetrievalQA.from_chain_type(
    llm=llm,
    chain_type='stuff',
    retriever=vektor_db.as_retriever(search_kwargs={'k': 3}),
    return_source_documents=True
)

# 5. Sorgu
soru = 'Şirket izin politikasına göre yıllık izin kaç gündür?'
sonuc = qa_zinciri({'query': soru})
print(f"Yanıt: {sonuc['result']}")
print(f"Kaynak bölüm sayısı: {len(sonuc['source_documents'])}")
```

7.3 Gelişmiş RAG Teknikleri

Temel (naive) RAG'ın ötesinde, retrieval kalitesini artırmak için birkaç gelişmiş teknik öne çıkmaktadır. Monigatti'nin (2025) Towards Data Science incelemesine göre: 'Gelişmiş RAG, naive RAG paradigmasının bazı kısıtlamalarını gidermek için hedeflenmiş iyileştirmelerle yeni bir paradigma olarak gelişmiştir.'

- **Hibrit Arama (Hybrid Search):** Yoğun vektör araması (dense retrieval) ile anahtar kelime tabanlı BM25 aramasının birleşimi. Bu yaklaşım, hem anlamsal hem sözdizimsel benzerliği yakalamaktadır.
- **Re-Ranking:** Alınan chunk'ların cross-encoder modelle yeniden sıralanması. İlk aşamada hız, ikinci aşamada doğruluk ön planda tutulmaktadır.
- **Cümle Penceresi Retrieval:** Chunk alındığında komşu cümlelerle bağlamın genişletilmesi, tutarlı bilgi elde edilmesini sağlar.
- **Parent-Child Chunking:** Küçük chunk'larla hassas retrieval, büyük parent chunk'larla zengin bağlam sunumu birleşimi.

Örnek 12: FAISS Vektör Araması ve Benzerlik Skoru

```
import faiss
import numpy as np
from sentence_transformers import SentenceTransformer

# Model yükleme
model = SentenceTransformer('paraphrase-multilingual-mpnet-base-v2')

# Örnek corpus
corpus = [
    'Transformer modeller NLP'de devrim yarattı.',
    'BERT çift yönlü dil modelidir.',
    'GPT metin üretimi için kullanılır.',
    'RAG sistemleri LLM'lere güncel bilgi sağlar.',
    'spaCy üretim ortamı için optimize edilmiştir.',
    'Python makine öğrenmesi için popüler bir dildir.'
]

# Gömme üretimi
corpus_gommeler = model.encode(corpus, normalize_embeddings=True)

# FAISS indeksi oluşturma
boyut = corpus_gommeler.shape[1]
indeks = faiss.IndexFlatIP(boyut)  # Inner Product (cosine ile eşdeğer
normalize edilmişse)
```

```
indeks.add(corpus_gommeler.astype(np.float32))

# Arama
sorgu = 'Büyük dil modelleri nasıl çalışır?'
sorgu_gommesi = model.encode([sorgu],
normalize_embeddings=True).astype(np.float32)

k = 3  # En yakın 3 sonuç
skorlar, indeksler = indeks.search(sorgu_gommesi, k)

print(f"Sorgu: {sorgu}\n")
for i, (skor, idx) in enumerate(zip(skorlar[0], indeksler[0])):
    print(f"{i+1}. (skor: {skor:.3f}) {corpus[idx]}")
```

8. Büyük Dil Modelleri (LLM) ve Uygulama Alanları

8.1 Prompt Mühendisliği

Prompt mühendisliği, LLM'lerden istenilen çıktıyı elde etmek için girdi metninin (prompt) sistematik olarak tasarlanması disiplinidir. Etkin prompt tasarımının üç temel bileşeni bulunmaktadır: Rol tanımı (sistem prompt'u), görev açıklaması ve bağlam bilgisi, az-sayıda örnek (few-shot examples). Kötü tasarlanmış bir prompt ile iyi tasarlanmış prompt arasındaki çıktı kalitesi farkı, küçük model ince ayarına eşdeğer ya da daha büyük iyileşme sağlayabilmektedir.

Örnek 13: Anthropic Claude API ile Yapılandırılmış Veri Çıkarımı

```
import anthropic
import json

istemci = anthropic.Anthropic()  # ANTHROPIC_API_KEY env değişkeninden
okunur

sistem_prompt = """Sen bir veri çıkarımı asistanısın.
Sana verilen metin parçasından yapılandırılmış bilgi çıkar.
Her zaman JSON formatında yanıt ver."""

haber_metni = """
Tesla CEO'su Elon Musk, 18 Kasım 2024'te şirketin İstanbul'a
elektrikli araç üretim tesisi kuracağını açıkladı. Yatırımın
toplam değerinin 2 milyar dolar olduğu ve 5000 kişilik istihdam
yaratması bekleniyor. Tesisin 2027 yılında üretime geçmesi planlanıyor.
"""

kullanici_mesaji = f"""Aşağıdaki haber metninden JSON olarak çıkar:
- kisi: kişi adı
- sirket: şirket adı
- yer: konum
- tarih: tarih
- yatirim_tutari: yatırım miktarı
- istihdam: çalışan sayısı
- acilis_yili: tesis açılış yılı

Metin: {haber_metni}
"""

mesaj = istemci.messages.create(
    model='claude-sonnet-4-6',
    max_tokens=512,
    system=sistem_prompt,
    messages=[{'role': 'user', 'content': kullanici_mesaji}]
)

sonuc_metni = mesaj.content[0].text
```

```
try:
    veriler = json.loads(sonuc_metni)
    print(json.dumps(veriler, ensure_ascii=False, indent=2))
except json.JSONDecodeError:
    print(sonuc_metni)
```

8.2 Zincir Düşünce (Chain-of-Thought) Prompting

Zincir Düşünce (Chain-of-Thought — CoT) prompting, LLM'i adım adım akıl yürütme sürecini açıklamasını teşvik ederek karmaşık görevlerdeki doğruluğu artıran bir teknik prompt stratejisidir. Wei ve diğerlerinin (2022) araştırması, CoT prompting'in özellikle aritmetik akıl yürütme ve çok adımlı problem çözme görevlerinde LLM performansını belirgin biçimde iyileştirdiğini göstermektedir.

Örnek 14: LangChain ile Ajan (Agent) Tabanlı Akıl Yürütme

```
from langchain.agents import create_react_agent, AgentExecutor
from langchain.tools import Tool
from langchain_anthropic import ChatAnthropic
from langchain import hub

# LLM tanımı
llm = ChatAnthropic(model='claude-sonnet-4-6', temperature=0)

# Araç tanımları
def hesapla(ifade: str) -> str:
    """Matematiksel ifade hesaplar."""
    try:
        return str(eval(ifade))
    except Exception as e:
        return f'Hata: {str(e)}'

def metin_ara(sorgu: str) -> str:
    """Belge veritabanında arama yapar."""
    # Gerçek uygulamada vektor DB araması yapılır
    return f"'{sorgu}' için bulunan sonuçlar: [belge_1, belge_2]"

araclar = [
    Tool(name='Hesap_Makinesi',
         func=hesapla,
         description='Matematik hesaplama için kullanılır'),
    Tool(name='Belge_Arama',
         func=metin_ara,
         description='Belge veritabanında arama yapar'),
]

# ReAct ajan oluşturma
prompt = hub.pull('hwchase17/react')
ajan = create_react_agent(llm, araclar, prompt)
ajan_yurutucusu = AgentExecutor(
    agent=ajan,
    tools=araclar,
    verbose=True,
    max_iterations=5
)

# Karmaşık sorgu
sonuc = ajan_yurutucusu.invoke({
    'input': '2024 yılı NLP trendi raporunu ara ve',
    '        ': 'ardından 45 milyon * 0.08 hesapla'
})
print(sonuc['output'])
```

9. Model Performans Değerlendirmesi

9.1 NLP Benchmark'ları

NLP modellerini sistematik biçimde karşılaştırmak için standart benchmark veri setleri ve değerlendirme metrikleri kullanılmaktadır. GLUE (General Language Understanding Evaluation) ve SuperGLUE, genel dil anlama kapasitesini ölçen çok görevli benchmark ailelerini temsil eder. SQuAD 1.1 ve 2.0, soru-cevap görevleri için endüstri standardıdır. CoNLL-2003, NER değerlendirmesi için yaygın kullanılan referans veri setidir.

Örnek 15: sklearn ile NLP Model Değerlendirmesi

```
from sklearn.metrics import (classification_report, f1_score,
                             confusion_matrix, accuracy_score)
import numpy as np

# Örnek tahmin sonuçları (duygu analizi)
gercek_etiketler  = [1, 0, 1, 1, 0, 0, 1, 0, 1, 0]
tahmin_etiketler  = [1, 0, 1, 0, 0, 1, 1, 0, 1, 0]
etiket_isimleri   = ['Negatif', 'Pozitif']

# Detaylı sınıflandırma raporu
print("=== Sınıflandırma Raporu ===")
print(classification_report(
    gercek_etiketler,
    tahmin_etiketler,
    target_names=etiket_isimleri
))

# NER için Token düzeyinde F1 (seqeval kütüphanesi ile)
# pip install seqeval
from seqeval.metrics import classification_report as ner_rapor

gercek_ner = [['B-PER', 'I-PER', 'O', 'B-ORG', 'O']]
tahmin_ner = [['B-PER', 'I-PER', 'O', 'B-LOC', 'O']]

print("\n=== NER Değerlendirmesi ===")
print(ner_rapor(gercek_ner, tahmin_ner, output_dict=False))
```

Tablo 2. *Başlıca NLP Modellerinin Benchmark Performans Karşılaştırması*

Model	Parametre	GLUE Skoru	SQuAD 2.0 F1	Öne Çıkan Özellik
BERT-Base	110M	80.4	76.3	Çift yönlü, MLM önceden eğitimi
RoBERTa	125M	88.5	**86.8**	BERT'in optimize edilmiş versiyonu
DeBERTa-v3	184M	**91.9**	88.4	Ayrıştırılmış dikkat ve sanal token
GPT-4	~1T (tahmini)	~90+	~90+	Multimodal, genel amaçlı
LLaMA 3.1-70B	70B	89.3	88.0	Açık kaynak, verimli

Not. Kaynak: arxiv (2025); Tucudean et al. PeerJ CS (2024); Springer Nature (2025) temel alınarak derlenmiştir. Bazı değerler tahminidir.

10. Etik Boyutlar: Önyargı, Adalet ve Sorumlu Yapay Zeka

10.1 NLP Modellerinde Önyargı

NLP modelleri, eğitim verilerinde mevcut olan toplumsal önyargıları ve stereotipleri öğrenme eğilimindedir. Cinsiyet önyargısı (gender bias), ırk önyargısı, yaş ayrımcılığı ve kültürel temsil eksikliği bu alandaki en belgelenmiş sorunları oluşturmaktadır. Word2Vec gömmeleri üzerinde yapılan erken araştırmalar, 'doktor' kelimesinin erkek isimlerine, 'hemşire' kelimesinin kadın isimlerine daha yakın konumlandığını göstermiş; bu önyargının iş ilanı sıralama sistemlerinde nasıl yeniden üretilebileceğini ortaya koymuştur.

Thallam NST Journal'ın (2025) değerlendirmesine göre: 'NLP'nin etik boyutu birçok endişeyi kapsamaktadır. Önyargı ve adalet; bir modelin eğitim süreci boyunca hatalı davranışları sistematik olarak yaratma eğilimini anlatmaktadır. Mahremiyet de göz ardı edilemez: Dil modelleri eğitim verilerinden kişisel bilgileri sızdırabilir.' Bu riskler, sorumlu AI geliştirme çerçevelerinin NLP pratiğine entegrasyonunu kaçınılmaz kılmaktadır.

10.2 Sorumlu NLP: Mitigation Stratejileri

Önyargıyla başa çıkmak için araştırma topluluğu birkaç temel strateji geliştirmiştir.

- **Veri denetimi ve temizliği:** Eğitim verilerinin temsil eşitsizlikleri açısından sistematik incelenmesi ve gerekirse dengelenmesi.
- **Debiasing teknikleri:** Gömme uzayında önyargı vektörlerinin tespit edilip nötralize edilmesi (örneğin Gender Debiasing).
- **Çeşitli değerlendirme veri setleri:** WinoBias, StereoSet ve BBQ gibi veri setleri, model önyargısını ölçmek için geliştirilmiştir.
- **İnsan denetimi:** Yüksek riskli uygulamalarda insan gözden geçirme sürecinin otomasyona eşlik etmesi.
- **Model kartları ve belgeler:** Model kapasiteleri, kısıtlamaları ve bilinen önyargılarının şeffaf biçimde belgelenmesi (Model Cards for Model Reporting, Mitchell et al.).

11. Türkçe için NLP: Özel Zorluklar ve Modeller

11.1 Türkçenin Morfolojik Karmaşıklığı

Türkçe, eklemeli (agglutinative) yapısıyla NLP açısından özgün zorluklar sunar: Bir kelime kökü, birden fazla çekim ve yapım ekiyle son derece uzun sözcük formları üretebilmektedir. 'Türkleştiremeyebileceklerimizdenmişsinizcesine' gibi aşırı örnekler akademik literatürde sıklıkla yer alsa da günlük dilde beş ila yedi morfemin birleştiği formlar oldukça yaygındır. Bu durum, kelime düzeyinde tokenizasyonun Türkçe için yetersiz kalmasına ve morfoloji farkındalıklı yaklaşımların zorunlu olmasına yol açmaktadır.

burakgun.com

11.2 BERTurk ve Türkçe NLP Modelleri

Türkçe için özel olarak eğitilmiş dil modelleri son yıllarda önemli ölçüde gelişmiştir. BERTurk (dbmdz/bert-base-turkish-cased ve uncased), HuggingFace üzerinde erişilebilen ve büyük Türkçe metin derlemleri üzerinde önceden eğitilmiş BERT modelidir. XLM-RoBERTa, 100 dil dahil Türkçeyi de kapsayan çok dilli bir model olarak yaygın kullanım bulmaktadır.

Örnek 16: Türkçe BERT ile Maskeli Dil Modellemesi

```
from transformers import pipeline, BertTokenizer, BertForMaskedLM

# Türkçe BERT modelini yükle
model_adi = 'dbmdz/bert-base-turkish-cased'
maskeleme = pipeline('fill-mask', model=model_adi)

# Maskeli tahmin
cumleler = [
    'Yapay zeka teknolojisi [MASK] dünyasını değiştiriyor.',
    'Transformer modeller NLP [MASK] tarihsel bir dönüşüm yarattı.',
    '[MASK] Dil Modelleri büyük metin derlemleri üzerinde eğitilir.',
]

for cumle in cumleler:
    tahminler = maskeleme(cumle, top_k=3)
    print(f"\nGiriş: {cumle}")
    for t in tahminler:
        print(f"  {t['token_str']:<20} (skor: {t['score']:.3f})")

# Morfolojik analiz için zeyrek kütüphanesi
# pip install zeyrek
import zeyrek
analizci = zeyrek.MorphAnalyzer()

kelimeler = ['gidiyorum', 'öğrenciler', 'Türkiye', 'yapabileceğim']
for kelime in kelimeler:
    analiz = analizci.lemmatize(kelime)
    print(f"{kelime} → {analiz[0][1]}")
```

12. Gelecek Trendler ve Araştırma Yönleri

12.1 Multimodal LLM'ler

NLP'nin sınırları metin ötesine genişlemektedir. Multimodal modeller, metin, görüntü, ses ve video verilerini birleşik bir mimari içinde işleyebilmektedir. GPT-4V, Claude 3 Opus ve Gemini Ultra, görüntü anlama ve metin üretimi kapasitelerini tek bir modelde birleştiren bu neslin öncü sistemlerini oluşturmaktadır. Thallam NST Journal'ın (2025) öngörüsüne göre 'gelecek, konuşma, yüz tanıma ve metinsel girişi entegre eden gerçek zamanlı multimodal diyalog sistemlerini içerecektir.'

12.2 Küçük Dil Modelleri (SLM) ve Verimlilik

GPT-4 ölçeğindeki modellerin yüksek hesaplama maliyeti, Küçük Dil Modelleri (Small Language Models — SLM) araştırmasını hızlandırmaktadır. Phi-3 (Microsoft), Gemma 2B (Google) ve LLaMA 3.2-3B gibi modeller; akıllı telefonlarda ve uç cihazlarda çalışabilecek ölçekte milyarlarca parametre sunmaktadır. Tucudean ve diğerlerinin (2024) tespitine göre BERT için geliştirilen sıkıştırma yaklaşımları, 'performansı düşürmeksizin bellek ayak izini azaltmayı başararak NLP'nin uç mimari kullanımına önemli katkı sağlamaktadır.'

12.3 Kuantum NLP

Kuantum bilgisayarların NLP'ye entegrasyonu, uzak vadede ilgi çekici bir araştırma yönü olarak öne çıkmaktadır. Thallam NST Journal'ın (2025) açıklamasına göre: 'Kuantum bilgisayarlar NLP'yi büyük ölçekli modeller için hesaplama verimliliğini artırarak dönüştürmeye hazırlanmaktadır. Geleneksel NLP modelleri klasik matris çarpmalarına dayanırken, kuantum geliştirilmiş transformer'lar üstel hızlanma potansiyeli sunan kuantum süperpozisyon ve dolanıklığı kullanacaktır.'

13. Sonuç

NLP, 1950'lerin kural tabanlı sistemlerinden günümüzün yüz milyarlarca parametreli Büyük Dil Modellerine uzanan yolculuğuyla yapay zekanın en hızlı evrilen disiplinlerinden biri olmayı sürdürmektedir. Bu yolculuğun 2017'deki Transformer dönüm noktası, hem mimarı hem de pratik açıdan NLP'yi köklü biçimde dönüştürdü: Paralel eğitim, bağlam duyarlı gömme ve transfer öğrenme bu devrimin üç temel kazanımını oluşturmaktadır.

BERT ve türevleri, duygu analizi, varlık tanıma ve soru-cevap gibi anlamaya yönelik görevlerde insan düzeyine yaklaşan performans sunarken; GPT ailesi metni anlayan ve üretebilen sistemleri pratiğe taşıdı. RAG mimarileri ise bu üretici modellerin en temel kısıtlamalarını — bilgi güncelliği ve hallüsinasyon — kurumsal düzeyde ele almaktadır. LangChain ve LlamaIndex gibi çerçeveler, bu bileşenlerin üretim ortamında bir araya getirilmesini hızlandırmaktadır.

Python ekosistemi, bu teknolojilere erişimin demokratikleşmesinin en önemli aracısı olmuştur. HuggingFace Transformers'ın model hub'ı, spaCy'nin üretim odaklı pipeline'ları ve LangChain'in ajan çerçeveleri; araştırmacıdan kurumsal geliştiriciye geniş bir kullanıcı yelpazesine NLP'nin kapılarını açmaktadır.

Bununla birlikte etik sorunlar — önyargı, mahremiyet, dezenformasyon ve hesap verebilirlik — bu teknolojilerin toplumsal kabulü için kritik gündem maddeleri olmayı sürdürmektedir. Multimodal modeller, verimli küçük dil modelleri ve olası kuantum NLP gelişmeleri, önümüzdeki on yılın araştırma gündemini şekillendirecek başlıklar arasındadır. Bu teknolojilerin sorumlu, şeffaf ve kapsayıcı biçimde geliştirilmesi, hem araştırmacıların hem uygulayıcıların hem de politika yapıcıların ortak sorumluluğunu oluşturmaktadır.

burakgun.com

Kaynakça

Anthropic. (2025). Claude API documentation. https://docs.anthropic.com

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of NAACL-HLT 2019, 4171–4186. https://arxiv.org/abs/1810.04805

DSStream. (tarihsiz). The grand tour of NLP: spaCy vs. NLTK. https://www.dsstream.com/post/the-grand-tour-of-nlp-spacy-vs-nltk

Explosion AI. (2025). spaCy: Industrial-strength NLP in Python. https://spacy.io

Ferrer, J. (2025, April 21). Building a RAG application using LlamaIndex. KDnuggets. https://www.kdnuggets.com/building-a-rag-application-using-llamaindex

Gupta, A. (2024). BERT applications in natural language processing: A review. Artificial Intelligence Review, Springer Nature. https://doi.org/10.1007/s10462-025-11162-5

HuggingFace. (2025). Transformers: State-of-the-art machine learning. https://huggingface.co/docs/transformers

Jain, A. (2025, November). Build a RAG system using Python, LangChain, and OpenAI. Medium. https://medium.com/@abhishekjainaj

Kaushik, S. (2024, November). Mastering sentiment analysis with pretrained NLP models. Medium. https://medium.com/@kaushiksimran827

LangChain. (2025). Build a RAG agent with LangChain. https://docs.langchain.com/oss/python/langchain/rag

LangCopilot. (2025, September 18). Best RAG frameworks 2025: LangChain vs LlamaIndex vs Haystack vs RAGFlow. https://langcopilot.com/posts/2025-09-18-top-rag-frameworks-2024-complete-guide

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. Advances in Neural Information Processing Systems, 33, 9459–9474.

Meghdadi, N., & Inkpen, D. (2024). uOttawa at LegalLens-2024: Transformer-based classification experiments. arXiv preprint arXiv:2410.21139. https://arxiv.org/pdf/2410.21139

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., ... & Gebru, T. (2019). Model cards for model reporting. Proceedings of FAT* 2019, 220–229.

Monigatti, L. (2025, January). Advanced retrieval-augmented generation: From theory to LlamaIndex implementation. Towards Data Science. https://towardsdatascience.com/advanced-retrieval-augmented-generation

Sinha, Y. (2024, September). AI for natural language processing (NLP) in 2024: Latest trends and advancements. Medium. https://medium.com/@yashsinha12354

Srivastava, P. (2023). spaCy vs. NLTK: A comprehensive comparison. Medium. https://medium.com/@prabhuss73

Thallam, N. S. T. (2025). Natural language processing (NLP) beyond 2024: The future of human-AI communication and language models. NST Journal of Scientific and Engineering Research, 12(1), 182–190.

Tucudean, G., Bucos, M., Dragulescu, B., & Caleanu, C. D. (2024). Natural language processing with transformers: A review. PeerJ Computer Science. https://doi.org/10.7717/peerj-cs.2222

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30.

burakoguz.com

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. arXiv preprint arXiv:2201.11903.

arxiv. (2025). Advancements in natural language processing: Exploring transformer-based architectures for text understanding. arXiv:2503.20227. https://arxiv.org/abs/2503.20227

burakolgun.com