NLP Kütüphaneleri ve Terimler Rehberi

🔧 Kütüphaneler

TikToken

- **Tanım**: OpenAI tarafından geliştirilen hızlı tokenizer kütüphanesi
- **Kullanım Alanı**: GPT modelleri için tokenization işlemleri
- **Avantajları**: Yüksek performans, OpenAI modelleriyle tam uyumluluk
- **Desteklenen Encoding'ler**:
  - `gpt2`: GPT-2 modeli için (50,257 vocab)
  - `cl100k_base`: GPT-3.5/4 için (100,277 vocab)
  - `p50k_base`: Code modelleri için

Transformers (Hugging Face)

- **Tanım**: En popüler açık kaynak NLP kütüphanesi
- **Kullanım Alanı**: Hazır eğitilmiş modelleri kullanma ve fine-tuning
- **Desteklenen Modeller**: BERT, GPT, T5, Gemma, LLaMA, Claude vs.
- **Ana Sınıflar**:
  - `AutoModel`: Model otomatik yükleme
  - `AutoTokenizer`: Tokenizer otomatik yükleme
  - `AutoProcessor`: Çoklu modalite (metin+görsel) işleme

PyTorch (Torch)

- **Tanım**: Facebook'un derin öğrenme framework'ü
- **Kullanım Alanı**: Neural network oluşturma, tensor işlemleri
- **Özellikler**: Dinamik computation graph, GPU desteği
- **Neden Gerekli**: Transformers kütüphanesi PyTorch backend'ini kullanır

Datasets (Hugging Face)

- **Tanım**: NLP veri setlerini yönetme kütüphanesi
- **Özellikler**: Otomatik caching, lazy loading, streaming
- **Popüler Dataset'ler**: IMDB, GLUE, Squad, Common Crawl

Matplotlib

- **Tanım**: Python'un temel veri görselleştirme kütüphanesi

- **Kullanım Alanı**: Grafik çizme, model performans analizi
- **Alternatifler**: Seaborn, Plotly, Bokeh

Hugging Face Hub

- **Tanım**: Hugging Face platformu ile etkileşim kütüphanesi
- **Özellikler**: Model upload/download, authentication, repository yönetimi
- **API Token**: Private modellere erişim için gerekli

📖 Terimler ve Kavramlar

Tokenization (Token'laştırma)

- **Tanım**: Metni daha küçük parçalara (token) ayırma işlemi
- **Amaç**: Bilgisayarların metni işleyebilir hale getirme
- **Türleri**:
  - **Word-level**: Kelimelere ayırma (`"Hello world"` → `["Hello", "world"]`)
  - **Subword-level**: Alt kelimelere ayırma (`"running"` → `["run", "ning"]`)
  - **Character-level**: Karakterlere ayırma (`"Hi"` → `["H", "i"]`)

Vocabulary (Sözcük Dağarcığı)

- **Tanım**: Model tarafından bilinen tüm token'ların listesi
- **Vocabulary Size**: Token sayısı (örn: GPT-2'de 50,257)
- **Trade-off**:
  - Büyük vocab = Daha detaylı anlama + Daha fazla memory
  - Küçük vocab = Daha az memory + Daha az detaylı anlama

Token ID

- **Tanım**: Her token'a atanan benzersiz sayısal kimlik
- **Örnek**: `"The"` → `464`, `"cat"` → `2574`
- **Amaç**: Bilgisayarların metinle matematiksel işlem yapabilmesi

Encoding (Kodlama)

- **Tanım**: Metni token ID'lere çevirme işlemi
- **Örnek**: `"Hello"` → `[15496]`
- **Metod**: `tokenizer.encode(text)`

Decoding (Kod Çözme)

- **Tanım**: Token ID'leri tekrar metne çevirme işlemi

- **Örnek**: `[15496]` → `"Hello"`
- **Metod**: `tokenizer.decode(token_ids)`

OOV (Out-of-Vocabulary)

- **Tanım**: Vocabulary'de bulunmayan kelimeler
- **Çözüm**: `<unk>` (unknown) token kullanımı
- **Modern Yaklaşım**: Subword tokenization ile OOV problemini minimize etme

Special Tokens (Özel Token'lar)

- **`<unk>`**: Bilinmeyen kelimeler için
- **`<pad>`**: Padding (doldurma) için
- **`<bos>`**: Begin of Sentence (cümle başı)
- **`<eos>`**: End of Sentence (cümle sonu)
- **`<sep>`**: Separator (ayırıcı)
- **`<cls>`**: Classification token

BPE (Byte Pair Encoding)

- **Tanım**: Subword tokenization algoritması
- **Çalışma Prensibi**: En sık tekrarlanan karakter çiftlerini birleştirme
- **Kullanım**: GPT modelleri, birçok modern tokenizer
- **Avantaj**: OOV problemini çözer, optimal vocabulary boyutu

Subword Tokenization

- **Tanım**: Kelimeleri anlamlı alt parçalara ayırma
- **Örnek**: `"unbelievable"` → `["un", "believ", "able"]`
- **Avantajlar**:
  - Nadir kelimeler için daha iyi genellme
  - Dil bağımsız çalışabilme
  - Optimal vocabulary boyutu

Model Prefix

- **Tanım**: Eğitilmiş model dosyalarının adlandırma ön eki
- **Örnek**: `model_prefix="spm_tokenizer"` → `smp_tokenizer.model`, `smp_tokenizer.vocab`

Character Coverage

- **Tanım**: Tokenizer'ın kapsayacağı karakter oranı

- **Varsayılan**: 0.9995 (önerilen)
- **Amaç**: Nadir karakterleri göz ardı ederek vocabulary boyutunu optimize etme

AutoProcessor vs AutoTokenizer

- **AutoTokenizer**: Sadece metin işleme
- **AutoProcessor**: Çoklu modalite (metin + görsel + ses) işleme
- **Kullanım**: Vision-Language modelleri için AutoProcessor gerekli

Pre-trained Model (Önceden Eğitilmiş Model)

- **Tanım**: Büyük veri setlerinde önceden eğitilmiş model
- **Avantaj**: Sıfırdan eğitim yerine transfer learning
- **Örnekler**: GPT-4, BERT, Gemma, LLaMA

Fine-tuning

- **Tanım**: Önceden eğitilmiş modeli belirli görev için uyarlama
- **Amaç**: Domain-specific performansı artırma
- **Yöntem**: Model ağırlıklarını küçük öğrenme oranıyla güncelleme

Reverse Vocabulary

- **Tanım**: Token ID'den token'a dönüşüm sözlüğü
- **Kodda**: `reverse_vocab = {v: k for k, v in vocab.items()}`
- **Amaç**: Decoding işlemini hızlandırma

JSON Vocabulary File

- **Tanım**: Vocabulary'yi JSON formatında saklama
- **Format**: `{"token": id, "the": 1, "cat": 2}`
- **Avantaj**: İnsan tarafından okunabilir, platformlar arası uyumluluk

🔍 Pratik Örnekler

Token Sayısı Karşılaştırması

```
Metin: "The capital of France is Paris"

Word-level: ["The", "capital", "of", "France", "is", "Paris"] → 6 token
GPT-2: [464, 3139, 286, 4881, 318, 6342] → 6 token
Gemma: [651, 6272, 576, 4843, 603, 7362] → 6 token
Character-level: ["T","h","e"," ","c","a","p","i","t","a","l",...] → 26 token
```

Vocabulary Boyutu Etkisi

- **Küçük Vocab (1K)**: Basit görevler, hızlı işleme, düşük memory
- **Orta Vocab (50K)**: Genel amaçlı, dengeli performans
- **Büyük Vocab (200K+)**: Yüksek kalite, çok dil desteği, yüksek memory

🎯 Hangi Tokenizer Ne Zaman Kullanılır?

TikToken

- OpenAI API'leri ile çalışırken
- Token sayısı hesaplama (maliyet tahmini)
- GPT modelleriyle uyumluluk gerektiğinde

Transformers AutoTokenizer

- Hugging Face modellerini kullanırken
- Fine-tuning yapacağınızda
- Çoklu dil desteği gerektiğinde

SentencePiece

- Kendi tokenizer'ınızı eğiteceksiniz
- Çok dilli projeler için
- Google modelleriyle (T5, PaLM) çalışırken

Bu rehber, modern NLP'de tokenization'ın temellerini ve pratik uygulamalarını kapsamaktadır.