Sözlük Düzeltme Sistemi — Detaylı Teknik Sunum ve Uygulama Rehberi

> **Amaç:** Konuşmadan/serbest metinden gelen komutları normalize edip, kanonik terimlere dönüştürmek; düzenli ifadeler ve kurallarla biçim hatalarını düzeltmek; genişletilebilir bir sözlük (lexicon) altyapısı ile varyantları kanonik forma eşlemek.

---

1) Üst Düzey Mimari ve Veri Akışı

Girdi: Kullanıcının konuşması veya serbest metin (ör: “pe te ze üçü gece moduna al, a blok koridoru 5 e 5 yap”).

Aşamalar:

1. **trKucuk**: Türkçe küçük harf normalizasyonu (İ→i, I→ı) + `lower()`.
2. **normalizeBosluk**: Boşlukların tekilleştirilmesi, baş/son kırpma.
3. **harfAdlariniHarfeCevir**: “pe te ze üç” → “p t z üç”.
4. **turkceSayilariRakamaCevir**: “on iki” → “12”, “üç” → “3”.
5. **gridKural**: “5 e 5”, “5’e 5”, “5x5”, “5 × 5” → “5x5”.
6. **dakikaBirimKural**: “10 dakikalık”, “10 dakika” → “10 dk”.
7. **blokYazimKural**: “a block”, “A blok” → “A blok”; çıplak “block” → “blok”.
8. **Tokenizasyon**: Boşluklara göre parçalama (token listesi).
9. **Eşleme Döngüsü**: 3-gram → 2-gram → 1-gram sırayla sözlük eşlemesi.
10. **fiilKumesi** içindekiler olduğu gibi geçirilir.
11. Doğrudan eşleşen n-gram varsa kanonik terime dönüştürülür.
12. **Son İşlem**:
13. “ptz 3” veya “ptz3” → “PTZ3” (bitişik ve büyük).
14. “ch 13”, “channel 13”, “kanal 13” → “kanal13”.
15. **normalizeBosluk** ile son bir temizlik.

Çıktı: Normalize ve kanonikleştirilmiş komut dizgesi (ör: “PTZ3 geceModu al A blok koridor 5x5”).

> **Not:** Bu boru hattında sözlüğün kritik rolü: çok kelimeli varyantlar önce normalizasyonla basitleştirilir (sayılar, harf adları, kılavuz kurallar), sonra n-gram taramasıyla kanonize edilir.

---

2) Bileşenlerin Derinlemesine İncelemesi

2.1 `terimSozluk`: Kanonik → Varyantlar

- **Yapı:** `Dict[str, List[str]]`. Sol: kanonik form, sağ: konuşmada/serbest metinde duyulabilecek yazım/telaffuz varyantları.
- **Örnekler:**
- `"PTZ": ["peteze", "pe te ze", "p t z", "pteze", "pte z"]`
- `"kanal": ["ch", "channel", "kanal", "çeyç"]`
- Çok kelimeli varyantlar desteklenir: `"arkaKoridor": ["arka koridor", "arka corridor", "arka korıdor"]`

Tasarım İlkeleri:

- Kanonik anahtarlar **programatikte kullanılabilir**, boşluk içeriyorsa bile; ancak çoğunlukla camelCase kullanmak yönetilebilirliği artırır (ör: `arkaKoridor`).
- Varyantlar **küçük harf** ve **imla hatası** içerebilir; normalizasyon pipeline'ı sonrası eşleşir.

2.2 `fiilKumesi`: Dokunma

- Bu küme içindeki fiiller/yardımcılar eşleme döngüsünde **hiç değiştirilmeden** geçirilir. Böylece "kapat", "duraklat" gibi eylemler sözlükten dolayı yanlışlıkla başka bir terime dönüştürülmez.

2.3 `harfAdlari`: Harf Adı → Harf

- Türkçe alfabenin harf adları konuşmada çok çıkar ("pe", "te", "ze").
- `harfAdlariniHarfeCevir` bunları tek harfe indirger: "pe te ze üç" → "p t z 3".

2.4 `sayiSozluk`: 0–25 Türkçe Sayılar

- "on iki" gibi **iki kelimelik** sayılar önce değiştirilir; sonra tek kelimelikler.
- Bu sıra, "on iki" içindeki "on" ve "iki"nin ayrı ayrı iki kez dönüştürülmesi gibi istenmeyen durumları engeller.

2.5 Kurallar (Regex ile)

- **gridKural**: "5 e 5"/"5'e 5"/"5x5"/"5 × 5" → **"5x5"**
- **dakikaBirimKural**: "10 dakika(lık)" → **"10 dk"**
- **blokYazimKural**:
- "a block"/"A block"/"a blok" → **"A blok"** (harf + blok → harf büyük)
- çıplak **"block" → "blok"**

2.6 `varyantHarita` (v2k): Varyant → Kanonik

- Sözlük ters çevrilir: her varyant küçük harfe normalize edilip kanonik anahtara map edilir.
- Eşleşme **O(1)** hash lookup ile yapılır.

2.7 `sozlukDuzelt2`: Ana Düzeltici

- **Ön-işleme** adımlarını sırasıyla uygular.
- **n-gram tarama:** 3 → 2 → 1 kelime uzunluğunda pencerelerle arar; **en uzun eşleşme** öncelikli.
- **Fiillerin korunması:** `fiilKumesi` içinde ise atlanır.

- **Son formatlama:** `PTZ` + rakam bitiştirme, `kanal` birleştirme.

Önemli: Eşleşme **doğrudan** (exact) yapılır; şu an fuzzy/dizimsel benzerlik (Levenshtein vb.) yoktur. İhtiyaç olursa eklenebilir (aşağıdaki Geliştime Başlıkları'na bakın).

---

3) Akışın Örnekle Açıklanması

Girdi: `"pe te ze üçü gece modu a blok koridoru 5 e 5 yap"`

1. `trKucuk` → aynı (zaten küçük).
2. `normalizeBosluk` → aynı.
3. `harfAdlariniHarfeCevir` → "p t z üçü gece modu a blok koridoru 5 e 5 yap"
4. `turkceSayilariRakamaCevir` → "p t z 3ü gece modu a blok koridoru 5 e 5 yap"
5. `gridKural` → "... 5x5 ..."
6. `dakikaBirimKural` → (değişmez)
7. `blokYazimKural` → "A blok ..."
8. **Tokenlar**: `["p","t","z","3ü","gece","modu","A","blok","koridoru","5x5","yap"]`
9. **Eşleme:**
10. `p t z` → `PTZ` (3-gram yakalar)
11. `3ü` içinde "3ü" ayrı tokendir; `PTZ` sonrası `ptz 3` formatlama kuralı rakamla bitiştirir.
12. "gece modu" → `geceModu`
13. "A blok" → `A blok` (kural sonucu)
14. "koridor" varyantları → `koridor`
15. "5x5" → kanonik "5x5"
16. "yap" fiil; `fiilKumesi`'nde yoksa olduğu gibi kalır.
17. **Son işlem**: "PTZ3 geceModu A blok koridor 5x5 yap"

---

4) Kenar Durumlar ve Dikkat Noktaları

- **Unicode sınırları (\b)**: Python `re` Unicode uyumlu; Türkçe karakterlerde kelime sınırı çoğu durumda doğru çalışır. Ancak kesme işareti (’), özel boşluklar (NBSP) gibi durumlarda dikkatli olun.
- **Türkçe küçük harf**: `trKucuk` basit bir normalize edici (İ→i, I→ı). Bu, pratikte yeterli; ancak dil-özgü çözümler (ICU) daha sağlamdır.
- **N-gram önceliği**: En uzun eşleşme stratejisi aynı bölgeyi iki kez eşlemeyi engeller.
- **Fiillerin kapsamı**: `fiilKumesi` genişletilmek istenirse JSON'a taşınabilir (bk. Bölüm 6.2).
- **Performans**: Her token pozisyonu için en fazla 3 hash lookup; pratikte **O(N)**.

---

5) Çıktı Stili Kuralları

- **PTZ + sayı**: `"ptz 3"`, `"PTZ 3"`, `"ptz3"` → **"PTZ3"** (bitişik, büyük)
- **Kanal**: `"ch 13"`, `"channel 13"`, `"kanal 13"` → **"kanal13"**
- **Blok**: `"a block"` → **"A blok"**; yalnız `"block"` → **"blok"**

Bu kurallar çıktıyı hem **tutarlı** hem de **işlenebilir** hale getirir.

---

6) Sözlüğü Kalıcılaştırma (JSON) — Nasıl Hazırlanır, Nereye Kaydedilir?

6.1 Önerilen Proje Yapısı

```
project_root/
  src/
    normalizer.py         # ana kod (mevcut dosya)
  config/
    terimler.json         # kanonik → varyantlar (en sık güncellenecek)
    fiiller.json          # fiilKumesi (isteğe bağlı dışa alım)
    harf_adlari.json      # harf adları eşlemesi (genelde sabit)
    sayilar.json          # 0–25 sayı haritası (genelde sabit)
  tests/
    test_normalizer.py
```

> **Nereye kaydedilmeli?** Değişiklik yapacağınız ana dosya ``. Diğerleri ihtiyaç olursa dışarı alınabilir; yoksa Python içinde sabit kalabilir.

6.2 JSON Biçimi

`` — *sadece terim sözlüğü için*:

```
{
  "PTZ": ["peteze", "pe te ze", "p t z", "pteze", "pte z"],
  "PTZ1": ["ptz 1", "ptz bir", "peteze bir"],
  "PTZ2": ["ptz 2", "ptz iki", "peteze iki", "ptz iki"],
  "PTZ3": ["ptz 3", "ptz üç", "peteze üç", "pe te ze üç"],
  "NVR": ["en ve ar", "enver", "n v r"],
  "IR": ["ay ar", "ir", "infrared"],
  "geceModu": ["gece modu", "ir modu", "gece görüşü"],
  "hareketli": ["hareketliye", "hareketli moda", "devriye", "patrol",
"motion"],
  "preset": ["prizet", "pirezset", "önayar"],
  "5x5": ["beşe beşlik", "beşe beş", "beş e beş", "5 e 5", "5'e 5", "5e5"],
  "3x3": ["üçe üçlük", "üçe üç", "üç e üç", "3 e 3", "3'e 3", "3e3"],
  "blok": ["block", "blok"],
  "koridor": ["corridor", "koridor"],
  "arkaKoridor": ["arka koridor", "arka corridor", "arka korıdor"],
  "kanal": ["ch", "channel", "kanal", "çeyç"]
}
```

İsteğe bağlı diğer dosyalar (eğer dışa almak isterseniz):

- `config/fiiller.json` → `{"aç": true, "kapat": true, ... }` veya dizi `{"fiiller": ["aç", "kapat", ...]}`
- `config/harf_adlari.json` → `{ "pe": "p", "te": "t", "ze": "z", ... }`
- `config/sayilar.json` → `{ "sıfır": 0, "bir": 1, ..., "yirmi beş": 25 }`

Neden ayrı dosyalar? Sürümleme ve yetkilendirme. Operasyonel ekip sadece `terimler.json` ile oynar; dilbilimsel sabitler değişmez.

6.3 JSON'ı Yükleme (Startup)

```
import json
from pathlib import Path

CONFIG_DIR = Path(__file__).resolve().parent.parent / "config"
TERIMLER_JSON = CONFIG_DIR / "terimler.json"

def load_terimler() -> dict:
    with open(TERIMLER_JSON, "r", encoding="utf-8") as f:
        return json.load(f)

# Uygulama başlangıcında
terimSozluk = load_terimler()
v2k = varyantHarita(terimSozluk)
```

Not: `ensure_ascii=False` ile kaydetmek; Türkçe karakterler korunur.

6.4 JSON'a Kalıcı Ekleme (Runtime → Dosyaya Yazma)

```
import json
from pathlib import Path

def sozlugeEkleKalici(kanonik: str, *varyantlar: str, dosya_yolu: Path =
TERIMLER_JSON):
    # 1) JSON'u oku
    with open(dosya_yolu, "r", encoding="utf-8") as f:
        sozluk = json.load(f)

    # 2) Sözlükte yoksa ekle
    if kanonik not in sozluk:
        sozluk[kanonik] = []

    # 3) Varyantları ekle (tekilleştir)
    for v in varyantlar:
        if v not in sozluk[kanonik]:
            sozluk[kanonik].append(v)

    # 4) Güvenli yedek + yaz
    yedek = dosya_yolu.with_suffix(".bak.json")
    with open(yedek, "w", encoding="utf-8") as f:
        json.dump(sozluk, f, ensure_ascii=False, indent=2)

    with open(dosya_yolu, "w", encoding="utf-8") as f:
        json.dump(sozluk, f, ensure_ascii=False, indent=2)
```

```
    # 5) Uygulama içi haritaları güncelle
    global terimSozluk, v2k
    terimSozluk = sozluk
    v2k = varyantHarita(terimSozluk)
```

İpucu: Dosyayı *sık* yazıyorsanız, lock mekanizması (ör. `filelock`) ekleyin.

6.5 İyi Uygulamalar

- **Sürüm kontrolü (Git)**: `config/terimler.json` değişiklikleri PR ile gözden geçirilsin.
- **Yedekleme**: Yazmadan önce `.bak.json` oluşturun (yukarıda var).
- **Validasyon**: JSON yapısını şemaya göre doğrulayın (bk. 6.6).
- **Test**: Yeni eklemeyi `tests/test_normalizer.py` ile kapsayın.

6.6 Basit JSON Şeması (Mantıksal)

- **Tür**: Nesne
- **Anahtar**: `string` (kanonik)
- **Değer**: `string[]` (varyantlar, boş olamaz)
- **Kısıtlar**:
- Varyantlar küçük harf ile tutulabilir; ancak zorunlu değil. Pipeline zaten küçültüyor.
- Aynı varyant iki kanonik anahtara **gidemez** (çakışma testi önerilir).

Basit doğrulama örneği:

```
def validate_terimler(sozluk: dict):
    assert isinstance(sozluk, dict)
    all_variants = {}
    for canon, vars_ in sozluk.items():
        assert isinstance(canon, str)
        assert isinstance(vars_, list) and vars_
        for v in vars_:
            assert isinstance(v, str)
            key = v.strip().lower()
            if key in all_variants and all_variants[key] != canon:
                raise ValueError(f"Varyant çakışması: '{v}' hem {a
```