İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

HECE TABANLI İSTATİSTİKSEL YÖNTEMLER İLE YAZIM HATASI BULMA VE DÜZELTME

YÜKSEK LİSANS TEZİ

Müh. Özlem Sema EROĞLU

504021547

Anabilim Dalı: Bilgisayar Mühendisliği
Programı: Bilgisayar Mühendisliği

Tez Danışmanı: Prof.Dr. Eşref ADALI

MAYIS 2005

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

HECE TABANLI İSTATİSTİKSEL YÖNTEMLER İLE YAZIM HATASI BULMA VE DÜZELTME

YÜKSEK LİSANS TEZİ
Müh. Özlem Sema EROĞLU
504021547

Tezin Enstitüye Verildiği Tarih : 9 Mayıs 2005
Tezin Savunulduğu Tarih : 30 Mayıs 2005

Tez Danışmanı : Prof. Dr. Eşref ADALI
Diğer Jüri Üyeleri Doç. Dr. Sabih ATADAN (İ.T.Ü)
Prof. Dr. Oya KALIPSIZ (Y.T.Ü.)

MAYIS 2005

ÖNSÖZ

Yüksek Lisans eğitimim süresince danışmanlığımı üstlenen, bilgi birikimini paylaşan, tez çalışmam konusunda desteğini eksik etmeyen ve doğal dil işleme konusunda çalışmak isteyen bir grup insanı biraraya getirerek ortak bir bilgi dağarcığı oluşmasını sağlayan hocam Prof. Dr. Eşref Adalı'ya teşekkürlerimi sunarım.

Teknik bilgi konusunda her zaman destek olarak tez çalışmam sırasında yol gösterici fikirleriyle büyük katkılar sağlayan ve bu amaçla değerli zamanlarını esirgemeyen Yüksek Müh. Gülşen Eryiğit ve Yüksek Müh. A. Cüneyd Tantuğ'a teşekkürü bir borç bilirim.

Tüm eğitim hayatım boyunca her ne başarı elde ettiysem bunun asıl temellerini atmış olan ilkokul öğretmenim Gülten Ayhan'a ve eğitim hayatımın her döneminde bana en büyük desteği veren sevgili aileme herşey için çok teşekkür ederim.

Ayrıca Yüksek Lisans eğitimim süresince hep yanımda olan, birlikte ortaya koyduğumuz çalışmalardan gurur duyduğum Simla Dilsiz'e ve desteğini hep yanımda hissettiğim, bana güç veren Hikmet Dinçer'e teşekkür ederim.

Mayıs, 2005 Özlem Sema Eroğlu

İÇİNDEKİLER

TABLO LİSTESİ

ŞEKİL LİSTESİ

SEMBOL LİSTESİ

Ü	:	Ünlü harf
S	:	Ünsüz harf
SS	:	Hece ve sözcük sonlarında bulunabilecek kurallı ünsüz harf çifti
P(A|B)	:	A'nın B'ye göre koşullu olasılığı
C(.)	:	Parantez içindeki varlığın sayısı
$\mathbf{w_{ij}}$	:	$w_i, w_{i+1},...,w_j$ biçiminde sözcük dizisi
$\mathbf{\log_2 a}$	:	a'nın 2 tabanında logaritması
<s>	:	Hece tabanlı n-gram modelde sözcük başı, sözcük tabanlı n-gram modelde cümle başı ifadesi
<b>	:	Boşluk
$\mathbf{P_{GT}(\alpha)}$	:	α n-gramının Goog-Turing yöntemi ile hesaplanmış olasılığı
$\mathbf{P}\binom{x}{y}$	:	x'in y'li permütasyonları

ÖZET

HECE TABANLI N-GRAM MODELLER İLE YAZIM HATASI BULMA VE DÜZELTME

Bu çalışmada Türkçe için hece tabanlı n-gram bir model oluşturulmuş, bu modelin yazım hatası bulma ve düzeltme uygulamalarında kullanılırlığı araştırılmıştır. N-gram modeller doğal dil işleme uygulamalarında sıklıkla kullanılan istatistiksel bir yöntemdir. Sözcük tabanlı n-gram modeller daha önce çeşitli diller için oluşturulmuş ve farklı doğal dil işleme amaçları için kullanılmıştır. Ancak Türkçe biçimbirimsel açıdan bitişken yapılı bir dildir; köklere ekler getirilerek sınırsız sayıda sözcük türetilebilir. Oysa istatistiksel yöntemler her bir farklı birimin eğitim kümesinde kaç kez geçtiğini saymaya dayanır. Sözcük tabanlı n-gram modellerin bu nedenle Türkçe için yeterli olmayışı hece tabanlı bir modelin çıkış noktası olmuştur. Bu çalışma kapsamında Türkçe'de olabilecek tüm heceler üretilmiş, n-gram model oluşturulurken ilgilenilecek farklı hece sayısının türetilebilecek farklı sözcük sayısına göre çok daha yönetilebilir düzeyde olduğu görülmüştür. Sadece harf sayısı kuralı konularak toplam 152,048 hece ve ses kuralları eklendiğinde yalnızca 6,160 farklı hece üretilmiştir.

Bir kısım eğitim kümesi üzerinden istatistiksel bilgi toplayarak oluşturulan bigram ve trigram modeller, daha sonra test kümeleri üzerinde yazım hatalarının bulunması ve düzeltilmesi için denenmiş, sonuçları tartışılmıştır.

Geliştirilen hece tabanlı modelin başarımının ölçülmesi amacıyla aynı eğitim kümeleri kullanılarak harf tabanlı n-gram model de geliştirilmiş ve aynı test kümeleri üzerinde denenerek başarımları kıyaslanmaya çalışılmıştır. Harf tabanlı yöntemde farklı birim sayısı alfabedeki harf sayısıyla sınırlı olduğu için bu bakımdan hece tabanlı yönteme göre daha avantajlı gibi görünse de, bu çalışmada bigram ve trigram modellerde birim olarak heceleri kullanmanın daha iyi sonuçlar verdiği gözlenmiştir. Bunun nedeni hecelerin art arda dizilme istatistiklerinin, sözcüklerin geneli ve ses kuralları hakkında daha çok bilgi sağlamasıdır.

Bu tez kapsamında heceleme işlemi gerçekleştirildikten sonra n-gram modellerin oluşturulma aşaması için geliştirilmiş olan uygulama dilin özelliklerinden bağımsız olduğu için başka dillerde de kullanılabilecek şekilde genelleştirilebilir.

Anahtar Sözcükler: Doğal dil işleme, n-gram, hece, yazım hatası bulma, yazım hatası düzeltme

SPELLING CHECK AND CORRECTION BY USING SYLLABLE BASED N-GRAM MODELS

SUMMARY

N-gram models are widely used in Natural Language Processing. But languages like Turkish, which are agglutinative in morphological structure, and hence have a huge vocabulary size, are not suitable for word-based n-gram models. In this study, syllable based n-gram models are examined for Turkish.

In Turkish, the rules for segmenting words into syllables are very clear and the size of regular syllable vocabulary is considerably low to be compared with word vocabulary size. As the statistical methods are based on counting, this low vocabulary size enables syllables to be favourable for use in n-gram models. With only the rules of vowel and consonant counts of a syllable, 152,048 possible syllables are generated syntactically. Adding the phonological rules resulted 6,160 syllable types for Turkish.

To compare the performance of letter based and syllable based n-grams, letter based bigram and trigram models are also taken into the scope of this work.

Syllable based bigram and trigram models are developed and tested for spelling checking and correction in Turkish text. Developing the n-gram models are based on counting the distinct syllables and the consecution of syllables in a training corpus according to the degree of the model. Once the model is constructed it can be used by various Natural Language Processing applications. Here, we use the model mainly for spelling checking and correction, and also we concisely show the ability of the model for word generation in Turkish.

Keywords: Natural Language Processing, n-gram, syllable, spelling checking, spelling correction.

1. BÖLÜM
GİRİŞ VE ÇALIŞMANIN AMACI

Yapay Zeka uygulamaları insanların kavramsal yeteneklerini bilgisayar ortamında modellemeye dayanır. Yapay Zekanın bir alt alanı olan Doğal Dil İşleme de insanların birbirleriyle iletişim kurmasını sağlayan doğal dillerin modellenmesini temel alır. Mevcut bilgisayar sistemleri ile kullanıcılar arasındaki en önemli engel iletişim problemidir. Bu sistemlerin kullandığı biçimsel diller genellikle kullanıcılar için öğrenilmesi ve kullanımı zor dillerdir. Her ne kadar günümüzde henüz insanların bilgisayarlar ile bir doğal dil kullanarak anlaşabileceği aşamaya gelinmemişse de bu hedefe yönelik çalışmalar gün geçtikçe hız kazanmakta, doğal dil işleme konusunda geliştirilen uygulamalar çeşitlenmektedir.

Doğal dillerin modellenmesi yalnızca insan-bilgisayar arası iletişimi kolaylaştırmakla kalmayacak, farklı doğal diller arasında otomatik çeviri yapılabildiği takdirde farklı ana dilleri konuşan insanlar arası iletişimi de sağlayacaktır. İki farklı dil arasında iletişim sağlanabilmesi için kullanılan bu dillerin kendilerine özgü dilbilgisi kurallarının iyi tanımlanmış olması gerekir. Ancak doğal diller doğaları gereği son derece karmaşık ve esnektirler. Bu nedenle sözlü ya da yazılı her tür iletişimin belli bir disiplin içinde yapılabilmesi için iletilen verinin sözkonusu dile ait kurallar kapsamında ve karşı tarafın doğru biçimde yorumlayabileceği şekilde hatasız sağlanması gerekmektedir. Dolayısıyla doğal dil işleme uygulamaları içinde bir metin içindeki yazım hatalarının bulunması ve düzeltilmesi ile ilgili uygulamalar hem insan-bilgisayar arası hem de ana dilleri farklı olan insanlar arası iletişimi kolaylaştırmak adına önemli bir yer tutmaktadır.

Biçimsel dillerin aksine doğal dillerin esnek ve karmaşık yapısı dilbilgisinin iyi bir biçimde öğrenilmesini güçleştirdiği için yazım hataları sıkça rastlanan bir durumdur. Diğer bir neden de daha hızlı biçimde yazı yazmayı sağlayan, ama yazım hatası yüzdesinin artmasına neden olan klavye kullanımının artmasıdır.

Günümüzde çoğu kelime işleme uygulaması yazım hatalarının bulunması ve düzeltilmesini sağlayan işlevleri içermektedir. Ancak mevcut bu işlevler genellikle ilgili dile ait sözlük kullanımına dayanır. Oysaki daha önce de belirttiğimiz gibi doğal dillerin son derece esnek bir yapıları vardır. Özellikle Türkçe gibi bitişken dillerde türetilebilecek sözcük sayısı çok fazla olduğu için her sözcüğün sözlüğe eklenmesi ve bu şekilde yazım hatası olup olmadığının sözlükten bakılarak kontrol edilmesi verimli bir çözüm olmamaktadır. Dilin üretkenliğinin dışında, zaman içinde diğer dillerden etkileşim ile dilin zenginleşmesi de sözlük kullanımını yetersiz kılan diğer bir etkendir.

Türkçe kurallı bir dil olduğu için kuralların tanımlanması yöntemiyle modellenerek yazım hatası bulma dahil farklı doğal dil işleme alanlarında bu tür modeller denenmiştir. Ancak her ne kadar kural tabanlı olsa da Türkçe de diğer tüm doğal diller gibi çok sayıda kural dışı durum içermektedir. Kural tabanlı yöntemlerde karşılaşılan diğer bir problem de doğal dillerin devamlı gelişmeye devam eden yapıları nedeniyle tanımlanan kuralların da sürekli güncellenmesi gerekliliğidir.

Doğal dillerin devingen yapıları gözönünde bulundurularak doğal dil işleme uygulamalarında özellikle son yıllarda istatistiksel yöntemler öne çıkmaya başlamıştır. İstatistiksel yöntemlerde öncelikle eğitim kümesi adı verilen geniş bir veri kümesi içinden bilgi toplanarak istatistiki bir model oluşturulur, daha sonra test kümesi olan başka bir kümenin yorumlanması için bu model kullanılır. Büyük eğitim kümelerinin kolaylıkla elde edilebilirliği arttıkça istatistiksel yöntemler daha da popüler hale gelmektedir.

Bu çalışmada Türkçe için hece tabanlı istatistiksel n-gram bir model oluşturulmuş, bu modelin dili ne kadar iyi tanımladığı, dolayısıyla yazım hatası bulma ve düzeltme uygulamalarında kullanılabilirliği araştırılmıştır. N-gram modeller doğal dil işleme uygulamalarında sıklıkla kullanılan istatistiksel bir yöntemdir. Sözcük tabanlı n-gram modeller daha önce çeşitli diller için oluşturulmuş ve farklı doğal dil işleme amaçları için kullanılmıştır. Ancak Türkçe biçimbirimsel açıdan bitişken yapılı bir dil olduğu için sözcük tabanlı n-gram modellerin bu dil için yeterli olmayışı hece tabanlı bir modelin çıkış noktası olmuştur.

Bu tez toplam altı bölümden oluşmaktadır. Birinci bölümde harfler, heceler ve ses kuralları açısından Türkçe'nin yapısına değinilmiş, yazım hatalarından ve yazım hatası bulma yöntemlerinden bahsedilmiştir.

İkinci bölüm, doğal dil işleme alanında istatistiksel yöntemler arasında önemli bir yere sahip olan n-gram modellerin tanıtılmasına ayrılmıştır. Bigram ve trigram modellerden detaylı olarak bahsedilmiştir. Sözcük tabanlı n-gram modeller ile metin içinde bir sonraki sözcüğün ne olduğu tahmin edilebilir. Sözcük tabanlı ve harf tabanlı n-gram modellerin nasıl kullanıldığına değinilmiştir. Bu bölümde son olarak veri seyrekliği probleminden ve bu problemi çözmek için kullanılan yöntemlerden bahsedilmiştir.

Üçüncü bölüm bu tezde yapılmış olan çalışmaların anlatıldığı bölümdür. Bu çalışmalar şunlardır; Türkçe sözcüklerin hecelerine ayrılması, hece tabanlı bigram ve trigam modellerin oluşturulması, harf tabanlı bigram ve trigram modellerin oluşturulması, bu modeller kullanılarak bir metindeki hataların bulunması ve düzeltilmesi işlemleri, modellerde ortaya çıkan veri seyrekliği probleminin ortadan kaldırılması için Good-Turing yöntemi ile modelin olasılık yoğunluğunu paylaştırma işlemleri, veri seyrekliğinin analizi amacıyla geliştirilen ve Türkçedeki tüm hecelerin üretilmesini sağlayan hece üretici, oluşturulan modellerin Türkçe hakkında ne kadar bilgi sağladığını göstermek amacıyla geliştirilmiş olan sözcük üretici.

Dördüncü bölümde geliştirilmiş olan yazılım, arayüz ve veri tabanı olmak üzere iki ana başlık altında tanıtılmaktadır. Arayüz bölümünde, çalışma kapsamında geliştirilmiş olan üç uygulamadan – Yazım Denetleyici, Hece Üretici ve Sözcük Üretici – bahsedilmiştir.

Son bölüm sonuç ve tartışmalar bölümüdür.

1.1 Türkçe'nin Yapısı

Yeryüzünde konuşulan diller kök yakınlığı bakımından dil ailelerine ayrılmaktadır. Başlıca dil aileleri; Hind-Avrupa, Hami-Sami, Ural-Altay, Çin-Tibet, Bantu dilleridir. Türkçe Ural-Altay dil ailesinin Altay koluna dahildir. Ural-Altay dilleri yapı bakımından bitişken dillerdir. Türkçe kural tabanlıdır ve kök sözcüklerin sonuna ekler getirilerek yeni sözcükler türetilir. Dil yapısı bakımından başlıca dil grupları

şunlardır; Bitişken Diller, (Ural-Altay dilleri, Japonca, Kore Dili), Kaynaşmalı Diller (Almanca, Latin), Eklemesiz Diller (Arapça, İbranice).

1.1.1 Harfler

Bir dil bilgisi terimi olarak ses; dilin parçalanamayan en küçük birimidir, temel taşıdır [1]. Sesin yazıdaki işareti, harftir. Türkçe'de ses ile harf arasında birebir ilgi vardır. Bir ses yazıda bir harfle gösterilirken, bir harfin okunuşunda da bir ses çıkarılır. Yani a, b, c, d gibi sesler yazıda birer harfle gösterilir.

Tablo 1.1'de görülebileceği gibi Türkçe'de 8 ünlü, 21 ünsüz harf vardır.

Tablo 1.1: Türkçe'de Harfler

Ünlü Harfler	a, e, ı, i, o, ö, u, ü
Ünsüz Harfler	b, c, ç, d, f, g, ğ, h, j, k, l, m, n, p, r, s, ş, t, v, y, z

1.1.2 Heceler

Ağızdan bir solukta çıkan ses ya da ses birliğine hece denir. [2] Türkçe'de hecenin temelini oluşturan sesler ünlülerdir. [1] Heceler de sözcüklerin ses yapısını oluştururlar. Ünlüler tek başlarına hece özelliği gösterdikleri halde ünsüzler yanlarına ünlü almadan bir ses bütünlüğü, bir hece oluşturamazlar. Dolayısıyla Türkçe bir sözcükte kaç tane ünlü varsa, o kadar da hece var demektir. Çünkü, Türkçe bir hecede, birden fazla ünlünün bulunması mümkün değildir.

Ünsüzler, kendilerini takip eden ünlülerle birleşerek hece oluştururlar. Bu sebeple bir sözcük hecelerine ayrılırken -yan yana iki ünsüz gelmemişse- ünlü+ünsüz şeklinde değil, ünsüz+ünlü şeklinde hecelenir:

ev - in - iz - de değil, *e - vi - niz – de*

güz - el - ler - in değil, *gü - zel - le - rin* vb.

Sözcük içinde iki ünsüzün yan yana gelmesi durumunda ünsüzlerden birincisi önceki heceye, ikincisi sonraki heceye ait olacak şekilde heceleme yapılır:

bil - gin, öğ - ret - men - lik

Yazıda, sözcüğün hecelerine doğru yerden ayrılıp ayrılmadığı çok basit bir uygulamayla kontrol edilebilir: Sözcük, hecelerine ayrıldığı şekliyle çok kolay ve akıcı bir şekilde söylenebiliyorsa heceleme doğru yapılmıştır. Tutukluk veya zorlanma oluyorsa sözcük, yanlış yerden bölünmüş demektir.

Ünlü sesle biten heceler açık, ünsüz sesle biten heceler kapalı hecelerdir. Türkçe'de heceler en az bir en çok dört sesten oluşur ama kural dışı durumlar yaratan yabancı kaynaklı sözcükler vardır (örnek:*bronz*). Tablo 1.2'de Türkçe bazı sözcüklerin nasıl hecelerine ayrıldığı gösterilmiştir.

Tablo 2.1: Türkçe'de Sözcüklerin Hecelerine Ayrılmasına İlişkin Örnekler

ela	e-la
elma	el-ma
türkü	tür-kü
türkler	türk-ler
saat	sa-at
bronz	bronz

Türkçe'de hece çeşitleri

Türkçe bir hecede en fazla dört ses bulunabilir. Türkçe'de, heceyi oluşturan seslerin sayısına ve bu seslerin hecedeki yerine göre altı çeşit hece vardır: (Aşağıdaki kısaltmalarda Ü ünlü, sesli yerine; S ünsüz, sessiz yerine kullanılmıştır.)

1. Bir ünlüden oluşan heceler (Ü): *e - rik, a-rı, u - yan.*

2. Bir ünlü,bir ünsüzden oluşan heceler (Ü+S): *el - ma, or - du, ül - ke.*

3. Bir ünlü, iki ünsüzden oluşan heceler (Ü+S+S): *ilk, üst, art.*

4. Bir ünsüz, bir ünlüden oluşan heceler (S+Ü): *el - ma, ar - ka - daş, gör- gü.*

5. Bir ünsüz, bir ünlü, bir ünsüzden oluşan heceler (S+Ü+S): *bil-dik, yal – nız – lık.*

6. Bir ünsüz, bir ünlü ,iki ünsüzden oluşan heceler (S+Ü+S+S):

Türk, kurt, sarp, se-vinç-ten.

Bunlardan ilk üçü sözcüğün sadece ilk hecesi olabilir. Diğerleri sözcüğün başında, ortasında veya sonunda bulunabilir.

Türkçe heceler ve sözcükler iki ünsüzle başlamaz; *blok, bravo, grup, klasik, kral, kontrat, spor, stop, stres, plaj, program, tren* gibi sözcükler, başka dillerden alınmadır. Ağızda bu iki ünsüz arasında bir ünlü türetilir; *kıral, sipor, tiren.*

Hece ve sözcük sonunda, aşağıdaki ünsüz çiftleri dışında ünsüz grupları bulunmaz:

- -lç, -lk, -lp, -lt

Örnek : *ölç; ilk, kalk; alp, kulp; alt, bunalt, salt.*

- -nç, -nk, -nt:

Örnek : *dinç, genç, gülünç, sevinç; denk; ant, kunt.*

- -rç, -rk, -rp, -rs, -rt

Örnek : *sürç, burç; bark, görk, Türk; sarp, serp; sars, pars, ters;art, kart, kurt, ört, yırt, yurt,yoğurt*.

- -st

Örnek : *ast, üst*

aşk, arş, çift, disk, felç, film, fötr, harf, lüks, misk, modernizm, popülizm, risk, şevk, tolerans gibi sözcükler, Türkçe'nin bu ses özelliğine uymayan alınma sözcüklerdir.

Arapçadan ve batı dillerinden alınan sözcüklerden bu ses özelliğine uymayanlar, araya bir ünlü getirilmek suretiyle Türkçeye uydurulmuştur. Bunlara ünlüyle başlayan bir ek veya sözcük gelirse türetilen ünlüler düşer: *akıl (< akl) - aklı, fikir (<fikr) - fikre, ömür (<ömr) - ömrü, seyir (<seyr) - seyret-, şükür (< şükr) - şükretmek; film (< film), lüküs (< lüks), moderin (< modern).*

1.1.3 Türkçe'nin Ses Kuralları

Türkçe'de bir sözcük içerisinde bulunan ünlüler ve ünsüzler birbirlerini etkileyerek benzeşir. [2] Bu benzeşme Türkçe'nin konuşurken en az çaba harcamaya yönelik yapısından kaynaklanır.

1.1.3.1 Büyük Ünlü Uyumu

Türkçe bir sözcükte kalın ünlülerden sonra kalın, ince ünlülerden sonra ince ünlüler gelir. Kalın ve ince ünlüler Tablo 1.3'de gösterilmiştir.

Tablo 1.3: Türkçe'de Kalın Ünlüler ve İnce Ünlüler

Kalın Ünlüler	a, ı, o, u
İnce Ünlüler	e, i, ö, ü

Bileşik sözcüklerde büyük ünlü uyumu aranmaz.

-yor, -ken, -ki, -leyin, -(i)mtrak ekleri büyük ünlü uyumuna uymaz.

1.1.3.2 Küçük Ünlü Uyumu

Türkçe bir sözcükte düz ünlülerden düz ünlüler, yuvarlak ünlülerden sonra dar yuvarlak ya da geniş düz ünlüler gelir. Düz ve yuvarlak ünlüler Tablo 1.4'de gösterilmiştir.

Tablo 1.4: Türkçe'de Düz Ünlüler ve Yuvarlak Ünlüler

Düz Ünlüler	a, e, ı, i
Yuvarlak Ünlüler	o, ö, u, ü

1.1.3.3 Ünlü Düşmesi

Sözcüklere çekim ve yapım eklerinin gelmesi ve bileşik sözcüklerin yapımı sırasında bir ünlünün kaybolmasıdır. Ünlü düşmesinin görüldüğü durumlar şunlardır;

- Organ adlarında. Örnek : *ağız* → *ağzım*
- Yabancı sözcüklerde. Örnek : *ömür* → *ömrüm*
- Bileşik sözcüklerde. Örnek : *sütlü + aş* → *sütlaç*
- Eklerde. Örnek : *baba + im* → *babam*

1.1.3.4 İki Ünlünün Yan Yana Gelmesi

Türkçe sözcüklerde iki ünlü yan yana gelmez. İki ünlünün yan yana geldiği sözcükler yabancı dillerden Türkçeye geçmiş sözcüklerdir.

Örnek : fiil, saadet, şuur, kanaat

1.1.3.5 Koruma Ünsüzleri

Türkçe sözcüklerde iki ünlü yan yana gelmediği için ünlü ile biten sözcüğe, ünlü ile başlayan bir ek gelirse araya n,s,ş,y koruma ünsüzleri girer.

Örnek :

pencere + in → *pencere**n**in*

komşı + u → *komşu**s**u*

1.1.3.6 Ünsüz Benzeşmesi

Türkçe sözcüklerde sert ünsüzlerden sonra sert, yumuşak ünsüzlerden sonra yumuşak ünsüzler gelir. Ünsüz benzeşmesinin görüldüğü durumlar şunlardır;

- Sözcük içinde: *eski, başka, Tanrı, yıldız* vb.

- Sözcük sonunda: *Türk-çe, diş-çi, bil-gi, av-cı, gör-gü* vb.
- Çekim eklerinde: *aç-tım*
- Ad durum eklerinde ve ek eylemde: *ağaç-ta, sınıf-ta, çocukta, kitapta* vb.

1.1.3.7 Ünsüz Düşmesi

Sözcüklerde bazı ünsüzlerin kaybolmasıdır. Ünsüz düşmesi genellikle yapım ve çekim eklerinin sözcüklerle birleşmesi sırasında görülür.

Örnek : *sıcak-cık* → *sıcacık* *çabuk-cak* → *çabucak*

1.1.3.8 Sert Ünsüzlerin Yumuşaması

Türkçe sözcüklerin sonunda bulunan ç, k, p, t ünsüzleri iki ünlü arasında yumuşayarak c, ğ, b, d'ye dönüşür.

Örnek : *ağaç* → *ağacı* *çocuk* → *çocuğu*

kitap → *kitabı* *yurt* → *yurdu*

1.2 Yazım Hataları

Bir doğal dil ile yazılmış herhangi bir metin içinde çok sayıda yazım hatasına rastlanabilir. Doğal dillerin biçimsel diller kadar kurallı bir yapıları olmadığı ve daha karmaşık bir yapıya sahip oldukları için bu hataların bulunup düzeltilmesi daha çok zahmet gerektiren bir işlemdir.

Yazım hatalarının başlıca nedenlerini şu şekilde sıralayabiliriz;

- Klavye kullanım hataları: *oğlum-oplum*
- Sesbilgisi hataları: *tehdit - tehtit*
- Dilbilgisi hataları: *oğlum – oğulum*
- Yazı tanıma yapılırken oluşan hatalar: *koşmak - kosmak*

Yazım hatalarının 80%'i yazım sırasında tek harfte hata yapılmasından kaynaklanan hata tipleridir.[14] Bu hata tipleri;

- Harf eklenmesi: *tehhdit*
- Harf silinmesi: *tedit*
- Bir harfin yerine başka harf gelmesi: *tehtit*

- Harf sıralarının karışması: *tedhit*

Hata bulma ve düzeltme uygulamalarında iki temel hedef vardır;

- Hata Bulma

  Varolan yazım hatalarının saptanması

- Hata Düzeltme

  Bulunan hatalı sözcüklerin yerine gelebilecek doğru sözcük seçenekleri üretilmesi

Hata bulma yöntemleri iki ana grupta toplanabilir; bağlamdan bağımsız yöntemlerde her sözcük kendi başına incelenir, bağlama duyarlı yöntemlerde ise sözcüğün tek başına doğru yazılmış olmasının dışında ilgili metin içinde bulunduğu konumda olma durumunun hatalı olup olmadığı da araştırılır. Bazı durumlarda yazım hatası nedeniyle ifade edilmek istenen sözcüğün yerini kendisinden farklı ama tek başına ele alındığında dilbilgisi kuralları içinde doğru olan bir sözcük alabilir. Örneğin "*Obasındaki kitapları topluyordu*." Cümlesinde *obasındaki* sözcüğü diğer sözcüklerden bağımsız olarak ele alınırsa doğru bir sözcüktür, ancak içinde geçtiği konu bağlamında muhtemelen *odasındaki* sözcüğünün hatalı yazılması nedeniyle ortaya çıkmış bir sözcük olduğu düşünülür. Yazım hatalarının düzeltilmesi amacıyla geliştirilmiş olan bir uygulamanın bu tip hataları da göz önünde bulundurması gerekmektedir.

1.3 Yazım Hatası Bulma Yöntemleri

Yazım hatalarının bulunması için üç temel yöntem kullanılmaktadır; sözlük kullanımı, kural tabanlı yöntemler, istatistiksel yöntemler.

1.3.1 Sözlük Kullanan Yöntemler

Yazım hatalarının bulunması için sözlük kullanmak ilk akla gelen ve mantığı en basit olan yöntemdir. İki temel aşama içerir;

Sözlük oluşturma: İyi sonuçlar alınabilmesi için ilgili dile ait tüm sözcükleri içeren bir sözlük oluşturulmalıdır.

Sözlükteki sözcüklerle eşleştirme: Hata ayıklaması yapılacak metin içindeki sözcükler tek tek sözlükte aranarak kontrol edilmelidir

Sözlükteki sözcüklerle eşleştirme aşamasının doğru sonuç verebilmesi için tüm sözcüklerin sözlükte yer alması gerektiğinden bahsedildi.. Ancak genellikle doğal dillerde bunun belli bir sınırı yoktur. Ek almış sözcüklerin de sözlükte yer alması gerekir ve özellikle Türkçe gibi sözcüklerin çoğunun köklere ekler eklenmesiyle türetildiği dillerde sözlük boyutu çok büyümektedir. Bu da hem doğru ve eksiksiz bir sözlük oluşturmayı güçleştirmekte hem de sözlük içinden eşleştirme yapılması aşamasının verimini düşürmektedir. Ayrıca özel isimlerin de hatalı bulunmaması için sözlüğe eklenmeleri gerekir. Tablo 1.5 arka kökünden türetilebilen sözcüklerden bazılarını göstermektedir.

Tablo 1.5: "arka" Kökünden Türetilebilen Sözcüklere Örnekler

arka
arkadaş
arkadaşım
arkadaşın
arkadaşı
arkadaşlar
arkadaşlarım
arkadaşlarımızca
arkası
arkasında
...

Bu durum akla sadece ilgilenilen konu kapsamındaki sözcüklerin sözlükte yer alması fikrini getirir; örneğin arkeoloji ile ilgili metinlerde hata ayıklama yapılacaksa sadece arkeoloji terimlerini içeren bir sözlük oluşturulması gibi. Ancak en özel kapsamlı metinlerde bile çok genel sözcükler de mutlaka yer alır.

Bunların dışında sözlük kullanımını olumsuz etkileyen bir diğer etken de dilin devingen yapısıdır. Doğal diller zaman içinde sürekli gelişir, yeni buluşlar ya da ortaya çıkan yeni kavramların karşılanması için dile yeni sözcükler eklenir, ya da diğer dillerden etkileşim ile yabancı sözcükler kullanılmaya başlanarak dilin bir parçası haline gelirler. Dil geliştikçe kullanılan sözlüğün de güncellenmesi gerekir.

1.3.2 Kural Tabanlı Yöntemler

Bitişken dillerde kural tabanlı yazım hatası bulma yöntemleri sözcüklerin biçimbirimsel analiz ile ek ve köklerine ayrılmasına ve dil için tanımlı olan kurallar çerçevesinde sözcüğün doğruluğunun araştırılmasına dayanır. Ancak sözcüklerin hatalı olup olmadığını araştırmak üzere kök ve eklerine ayrıştırılmaları için güçlü

biçimbirimsel analiz tekniklerine ihtiyaç duyulur. [3] Çünkü özellikle Türkçe gibi dillerde sözcüklerin art arda gelen eklerle türetilmesi ve kök ve eklerin ses uyumu gibi kurallar nedeni ile biçim değiştirebilir olması karmaşık sözcük biçimlerinin oluşmasına neden olur. Kök ve eklerine ayrılan sözcüklerin doğruluğu kökler için sözlük kullanımıyla, ekler için de tanımlı olan kurallara göre araştırılır.

Kural tabanlı yöntemlerde bazı durumlarda biçimbirimsel analiz tek başına yeterli olmayabilir, anlam da etkili olabilir. Örneğin fiilden fiil türeten "ala" eki bazı fiil kökleri için dilde geçerli olan sözcükler türetirken (√ *it-ele-mek,* √ *şaş-ala-mak*) bazı fiil kökleri için kullanılmaz (× *koş-ala-mak,* × *sev-ele-mek*). Başka bir örnek de yapım eklerinin tekrarlı kullanılabilmesi durumudur. Biçimbirimsel açıdan doğru analiz edilebilen ancak aslında dilde kullanımı olmayan sözcükler türetilebilir. Örneğin *göz* kökünden *göz-lük, gözlük-çü, gözlükçü-lük* gibi doğru (dilde kullanımı olan) sözcükler türeyebilirken *gözlükçülük-çü, gözlükçülükçü-lük* gibi hatalı (dilde kullanılmayan) sözcükler de türeyebilir.

Kök ve ek ilişkilerini belirleyen kurallar dışında ayrıca belirlenmesi gereken ses kuralları da vardır. Bunlar sesli uyumu (√ *yapmayacaktınız,* × *yapmayacektiniz),* sessiz benzeşmesi (-da ekinin –ta olması; *yolda, uçakta), k*aynaştırma (*bahçesi, bahçeyi), k*ök bozulması (*ben→bana, ara→arıyor, burun→burnum)* gibi kurallardır.

Bunların dışında bir de azımsanamayacak sayıda kural dışı durumlar vardır.

Kural tabanlı yöntemle geliştirilen bir yazım hatası bulma uygulamasının tüm bu durumları tek tek ele alması gerekmektedir. [3] Ancak doğal dillerin devingen yapısı nedeniyle zaman içinde ihtiyaçlar doğrultusunda kurallar değişebilir, esneyebilir.

Türkçe için kural tabanlı hata bulma konusunda şu çalışmalar yapılmıştır; Oflazer ve Solak'ın "Design and Implementation of a Spelling Checker for Turkish" [3], Oflazer ve Güney'in "Spelling Correction in Agglutinative Languages" [4], Güngör'ün "Computer Processing of Turkish: Morphological and Lexical Investigation" [5], Oflazer'in "Error-Tolerant Finite State Recognition with Applications to Morphological Analysis and Spelling Correction" [6] adlı çalışmaları.

1.3.3 İstatistiksel Yöntemler

Doğal dil işleme alanında son yıllarda istatistiksel yöntemler öne çıkmaya başlamıştır. İstatistiksel yöntemler ile yazım hatası bulma ve düzeltme işlemi bir

sözü dilbilimsel açıdan kesinlikle doğru ya da hatalı diye ayırmak yerine dilde sıklıkla kullanılan sözcüklerin neler olduğu sorusuna cevap arar. Bu nedenle doğal dillerin esnek ve gelişmeye açık yapısına en uygun olan yöntemdir.

İstatistiksel yöntemlerde öncelikle eğitim kümesi adı verilen geniş bir veri kümesi içinden bilgi toplanarak istatistiki bir model oluşturulur, daha sonra test kümesi olan başka bir kümenin yorumlanması için bu model kullanılır. Büyük eğitim kümelerinin kolaylıkla elde edilebilirliği arttıkça istatistiksel yöntemler daha da popüler hale gelmektedir.

Bir sözcük tek başına ele alındığında hatalı olmayabilir ancak içinde bulunduğu bağlamda hatalı olabilir. Düşük olasılıklı bir sözcük dizisinin yazım hatası bulma amacıyla, yüksek olasılıklı sözcük dizilerinin ise hata düzeltme amacıyla kullanıldığı n-gram modeller, bağlama duyarlı yazım hatası bulma amacıyla ortaya çıkmıştır. [15-18]

Bu tür hataların bulunması için geliştirilmiş olan bir başka yaklaşım da sözcükler arasındaki belirsizliğin karışıklık kümeleri* kullanılarak modellenmesidir. Bu durumda yazım hatası düzeltme işlemi bir çeşit belirsizlik çözümü işlemine dönüşmektedir. Bu yaklaşım tüm yazım hatalarını çözmeye çalışmaktan çok sıklıkla karıştırılan sözcükler arasında karar vermeye çalışır. Bayes sınıflayıcıları** ve karar listeleri*** bu tür yazım denetleyiciler sınıfına dahildir. [19-22]

Yazım hatası denetleme konusunda bir diğer yaklaşım da "Winnow-based" yöntemidir. Bu yöntem ağırlıkların güncellenmesi algoritması ile çok sayıda özelliği içinde barındırabilmektedir. [9]

Türkçe için istatistiksel yöntemlerle yapılmış olan doğal dil ile ilgili çalışmalar; Hakkani-Tür'ün "Statistical Language Modeling for Turkish" [7], Tür'ün "A Statistical Information Extraction System for Turkish" [8] adlı çalışmalarıdır.

---

* İngilizce "confusion sets" terimine karşılık kullanılmıştır.

** İngilizce "Bayes Classifiers" terimine karşılık kullanılmıştır.

*** İngilizce "decision lists" terimine karşılık kullanılmıştır.

2. BÖLÜM
N-GRAM MODELLER

Klasik yaklaşımda dil modelleme önceki sözcüklere bakarak sonraki sözcüğün tahmin edilmesini ifade eder. [10] Bir metin içinde bir sonraki harfi tahmin etme üzerine olan Shannon Oyunu'na atıfta bulunan bu yaklaşım, konuşma tanıma ve optik karakter tanıma uygulamalarının temelini oluşturmakta, aynı zamanda el yazısı tanıma, yazım hatası bulma ve istatistiksel çeviri gibi uygulamalarda kullanılmaktadır. Doğal dil işleme uygulamalarında n-gram modeller n adet birimin art arda sıralanma olasılıklarını kullanarak dilin modellenmesini sağlayan istatistiksel bir yöntemdir. Burada birim sözü ile kastedilen oluşturulacak modelin kullanım amacına yönelik olarak seçilen, ilgili dile ait sözcükler, harfler gibi yapı taşlarıdır.

N-gram modellerin en genel kullanım amacı (n-1) birim kullanılarak n. birimin tahmin edilmesidir. Birimlerin olasılıkları kullanılarak meydana getirdikleri sözcük ya da cümlelerin olasılıkları hesaplanabilir. Harflerin temel birim olarak kabul edildiği aşağıdaki örnekte çeşitli n-gram modellere göre sözcüğün nasıl ayrıştırıldığı görülmektedir.

Örnek : *okul*

1-gram : *o – k – u – l*

2-gram : *ok – ku – ul*

3-gram : *oku – kul*

Benzer şekilde sözcükler de temel birim olarak kabul edilebilir.

Örnek : *Bugün okula gittim*.

1-gram : *Bugün – okula – gittim*

2-gram : *Bugün okula – okula gittim*

3-gram : *Bugün okula gittim*

Olasılık hesabı yapılabilmesi için önce hangi birimlerin art arda gelme olasılıklarıyla ilgileniliyorsa o birimlerin her bir örneğinin kaç adet olduğu sayılır. Örneğin sözcüklerin art arda gelme olasılıklarıyla ilgileniliyorsa eğitim kümesindeki tüm sözcüklerin kaçar defa geçtiği sayılır. Daha sonra örneğin eğer bigram model kullanılıyorsa iki sözcüğün art arda gelme durumlarının sayısı bulunur. Bu sonuçlar kullanılarak iki sözcüğün art arda gelme olasılıkları hesaplanır.

Basit n_Gram modeline göre her birimin diğer bir birimin peşinden gelme olasılığı eşittir. [11] Örneğin sözcük tabanındaki bir modelde, eğitim kümesi 100,000 sözcük içeriyorsa bir sözcüğün herhangi başka bir sözcükten sonra gelme olasılığı 1/100,000 olacaktır. Daha gerçekçi bir yaklaşımla, bir sözcüğün başka bir sözcüğün ardından gelme olasılığı sözcüğün rastlanma sıklığına bağlıdır. Örneğin 1,000,000 sözcüklük örnek bir eğitim kümesi içerisinde "bu" sözcüğü 69,971 defa, "tavşan" sözcüğü ise 11 defa geçmektedir. Buna göre bir sonraki sözcüğün "bu" olma olasılığı 0.07, "tavşan" olma olasılığı 0.00001 diyebiliriz. Ancak tek başına sözcüğün rastlanma sıklığına bakmak yeterli değildir. Örneğin "Hemen sonra bu" cümlesinin sonuna "tavşan" gelme olasılığı "bu" gelme olasılığından daha yüksektir. Bu durumda bir sözcüğün gelme olasılığı kendisinden önce gelen sözcüklere göre koşullu olasılığı ile hesaplanabilir. Örneğin "beyaz"dan sonra "tavşan" gelme olasılığı olan P(tavşan|beyaz) diğer durumlarda "tavşan" gelme olasılığına göre daha yüksektir.

Sözcük tabanlı n-gram bir modelde bir cümlenin olasılığı şu şekilde verilir:

$$P(w_1, w_2, \ldots, w_{n-1}, w_n) \tag{2.1}$$

Zincir kuralı uygulanarak n. sözcüğün olasılığı:

$$\begin{aligned} P(w^n{}_1) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w^2{}_1)\ldots P(w_n \mid w^{n-1}{}_1) \\ P(w^n{}_1) &= \prod P(w_k \mid w^{k-1}{}_1) \end{aligned} \tag{2.2}$$

olarak bulunur.

Ancak $P(w_n|w^{n-1}{}_1)$ olasılığını hesaplamak için kendisinden n sözcük önce gelen sözcüğe göre olasılığını hesaplamak gerekir. Bu nedenle $P(w_n|w^{n-1}{}_1)$ olasılığı için bir yaklaştırma* yapılır; örneğin bigram bir modelde bir sözcüğün olasılığını sadece bir

* "yaklaştırma" İngilizce "approximation" terimine karşılık kullanılmıştır.

önceki sözcük belirler şeklinde bir kabul yapılır. Buna göre n-gram modeller için genel olarak bir sonraki sözcüğün kendinden önceki n sözcüğe göre koşullu olasılığı aşağıdaki şekilde bulunur:

$$P(w_n \mid w^{n-1}{}_1) \approx P(w_n \mid w^{n-1}{}_{n-N+1}) \tag{2.3}$$

Denklem 2.3, w_n sözcüğünün olasılığının sadece kendinden önce gelen n sözcüğün olasılığına yaklaştırılabileceğini gösterir.

Sözcük tabanlı bigram bir modelde, bir cümlenin olasılığı, denklem 2.3'ün 2.2'de yerine konmasıyla elde edilir, sonuçta:

$$P(w^n{}_1) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1}) \tag{2.4}$$

Örneğin P(tavşan|Arabanın önünden hızla geçen beyaz bir) olasılığını hesaplamak yerine P(tavşan|bir) olasılığı hesaplanır. Bu şekilde bir sözcüğün olasılığının yalnızca kendinden önce gelen sözcüğe bağlı olması varsayımına Markov varsayımı denir. Markov modellerinde bir birimin olasılığı kendinden n önceki birimlerin olasılıkları kullanılarak bulunur. Markov zinciri son durumu kendinden önceki sonlu sayıda duruma bağlı olan bir çeşit ağırlıklı sonlu durum makinesidir. Sözcük tabanlı bigram model her bir sözcüğe bir durum atanmış olan bir çeşit basit Markov zinciri olarak düşünülebilir. Bu şekilde n-gram modeller için genelleştirecek olursak n-gram modeller için

Bigram model : yalnızca bir önceki sözcüğe bakılır

Trigram model: önceki iki sözcüğe bakılır

....

n-gram model : önceki n-1 sözcüğe bakılır.

Böylece bigram modele 1. derece Markov modeli, trigram modele 2. derece Markov modeli ve n-gram modele de (n-1). derece Markov modeli denir.

2.1 En Büyük Olabilirlik Kestirimi*

Sözcük tabanlı bir n-gram modelde bir n-gramın kendinden önce gelen sözcük dizisine göre koşullu olasılığı, bu n-gramın geçme sayısı, aynı sözcük dizisiyle başlayan tüm n-gramların sayısına bölünerek bulunabilir. Buna En Büyük Olabilirlik Kestirimi denir. Bu tür bir kestirim eğitim kümesinde geçen n-gramlara en büyük olasılığı veren parametrelerin seçilmesini ifade eder.

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{\sum_w C(w_{n-1}w)} \tag{2.5}$$

w_{n-1} sözcüğü ile başlayan tüm bigramların sayısı w_{n-1} sözcüğünün unigram sayısına eşit olacağı için denklem 2.5 şu şekilde sadeleştirilebilir:

$$P(w_n \mid w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})} \tag{2.6}$$

N-gramlar genelinde parametre kestirimi için denklem şu şekildedir:

$$P(w_n \mid w^{n-1}_{n-N+1}) = \frac{C(w^{n-1}_{n-N+1}w_n)}{C(w^{n-1}_{n-N+1})} \tag{2.7}$$

Burada $C(w^{n-1}_{n-N+1}w_n)$ seçilmiş bir n-gramın rastlanma sıklığını,
$C(w^{n-1}_{n-N+1})$ ön dizgisinin rastlanma sıklığını ifade eder.

2.2 Bigram Model

Bir dil modelinde seçilen bir birimin olasılığı yalnızca kendinden önce gelen birime bağlıysa bu modele bigram (2-gram) model denir.

Bigram modeli ile k sözcükten oluşan bir cümlenin olasılığı şu şekilde hesaplanır:

$$P(W) = P(w_2 \mid w_1)P(w_3 \mid w_2)P(w_4 \mid w_3)...P(w_k \mid w_{k-1}) \tag{2.8}$$

* İngilizce "Maximum Likelihood Estimation" terimine karşılık kullanılmıştır.

Toplam 1616 farklı sözcük içeren yaklaşık 10,000 cümlelik bir küme kullanılarak oluşturulmuş olan bigram model kullanılarak aşağıdaki cümlenin olasılığı şu şekilde bulunmuştur. [11] (Berkeley Restaurant Projesi)

P(I want to eat Chinese food)

= P(I|<s>) P(want|I) P(to|want) P(eat|to)P(Chinese|eat) P(food|Chinese)

= 0.25 * 0.32 * 0.65 * 0.26 * 0.002 * 0.60

= 0.000016

Burada <s> cümle başını ifade etmektedir.

Görüldüğü gibi olasılık değerlerinin her zaman 1'den küçük olmaları nedeniyle her bir olasılık çarpanı, sonucu gittikçe küçültür ve eğer çok sayıda n-gram içeren bir cümlenin olasılığı hesaplanıyorsa sayı çok küçüldüğü için hata oluşur. Bu durumun engellenmesi için olasılıkları doğrudan çarpmak yerine olasılık değerlerinin logaritması alınır ve hepsi toplanır. Logaritma alarak toplamak doğrusal olarak çarpma işlemine denk düşer. En son bulunan sayının ters logaritması alınarak cümlenin olasılığı bulunur.

Bigram modelde her bir birim ikilisinin olasılığı Denklem 2.5'de verildiği şekilde bulunur.

Burada $C(w_{n-1}w_n)$ seçilmiş bir bigramın sayısını, $\sum wC(w_{n-1}w)$ aynı sözcükle başlayan tüm bigramların sayılarının toplamını ifade eder.

Örneğin tablodaki her bigram sayısı uygun olan unigram sayısı ile bölünerek her birinin olasılıkları hesaplanır.

2.3 Trigram Model

Bir dil modelinde seçilen bir birimin olasılığı yalnızca kendinden önce gelen iki adet birime bağlıysa bu modele trigram (3-gram) model denir.

Trigram modeli ile k sözcükten oluşan bir cümlenin olasılığı şu şekilde hesaplanır:

$$P(W) = P(w_3 \mid w_1 w_2) P(w_4 \mid w_2 w_3) P(w_5 \mid w_3 w_4) ... P(w_k \mid w_{k-2} w_{k-1}) \quad (2.9)$$

Trigram modelde her bir birim üçlüsünün olasılığı yine Denklem 2.7 kullanılarak hesaplanır.

2.4 N-gram Yönteminde Kullanılan Birimler

N-gramlar seçilen bir birim bazında, birimlerin peşpeşe gelme sayıları bulunarak dilin modellenmesini sağlar. Kullanılan birimler sözcük, harf ya da hece olabilir. Bu çalışmada temel olarak heceler seçilmiştir. Ayrıca karşılaştırma yapabilmek amacıyla harfler için de aynı işlemler yapılmıştır. 3. bölümde oluşturulan bu modeller tanıtılacaktır. Bu bölümde ise sözcük kullanımı ve harf kullanımı genel olarak anlatılacaktır.

2.4.1 Sözcükler

Eğitim kümesindeki her bir farklı sözcük sayılır. Daha sonra n-gram modelin derecesine göre bir önceki bölümde anlattığımız formüller kullanılarak her bir n-gramın olasılığı hesaplanır.

Örneğin sözcük tabanlı bigram bir modelde bir cümlenin olasılığı şu şekilde hesaplanır

P(I want to eat Chinese food) =

P(I|<s>) P(want|I) P(to|want) P(eat|to) P(Chinese|eat) P(food|Chinese)

Aynı cümle için Türkçe'de benzer şekilde

P(Çin yemeği yemek istiyorum) =

P(Çin|<s>) P(yemeği|Çin) P(yemek|yemeği) P(yemek|istiyorum)

yazılabilir.

Bu örnekte de görülebileceği gibi Türkçe bitişken yapılı bir dil olduğu için bir cümle içinde sözcükler genellikle basit biçimleriyle değil türetilmiş sözcükler olarak yer alır. Daha önce de belirtildiği gibi sözcük tabanlı n-gram modeller bir eğitim kümesindeki farklı sözcüklerin sayılmasına dayanır. Türkçe'de eklerin art arda eklenmesiyle türetilebilecek çok sayıda sözcük olduğu için sayılması gereken farklı sözcük adedi İngilizce gibi sözcüklerin sıklıkla yalın biçimde kullanıldığı dillere göre çok daha fazladır. Tablo 2.1 İngilizce ve Türkçe iki eğitim kümesi üzerinde toplanan bilgiler sonucu elde edilen farklı sözcük sayısı adetlerini göstermektedir.

Tablo 2.1: Türkçe ve İngilizce Onar Milyonluk Eğitim Kümelerinden Elde Edilmiş Farklı Sözcük Sayısı Değerleri [7]

Dil	**Farklı Sözcük Sayısı**
İngilizce	97,734
Türkçe	474,957

Bu durum Türkçe gibi bitişken dillerde sözcük tabanlı n-gram model kullanıldığı takdirde veri seyrekliği* problemine neden olur. Bunu aşmak için sözcük tabanlı modeller yerine Türkçe'de kurallı bir biçimde ek ve köklerin art arda sıralanması özelliğinden faydalanılan çalışmalar yapılmıştır. Hakkani-Tür [7] Türkçe için istatistiksel modeller önerdiği çalışmasında sözcükleri biçimbirimsel analiz ile daha küçük birimlere ayırarak bir dil modeli geliştirmiş ve bu modeli biçimbirimsel analiz sırasında karşılaşılan belirsizlik problemini gidermek için kullanmıştır. Benzer şekilde Gökhan Tür [8] bu tür modelleri istatistiksel bilgi çıkarımı** konusunda yaptığı çalışmasında incelemiştir.

2.4.2 Harfler

Eğitim kümesindeki sözcüklerin içerdikleri harfler sayılır. Daha sonra n-gram modelin derecesine göre her bir n-gramın olasılığı hesaplanır.

Örneğin harf tabanlı trigram bir modelde "okullar" sözcüğünden oku-kul-ull-lla-lar

trigramları elde edilebilir. Elde edilen bu trigramlar "okullar" sözcüğünün olasılığını bulmak için kullanılır.

"okyullar" sözcüğünden elde edilen oky-kyu-yul-ull-lla-lar trigramları içinde "kyu" trigramının olasılığı Türkçe için çok düşük olacağından bu sözcüğün olasılığı da çok düşük olacaktır.

Ancak örneğin *yazilmiş* sözcüğünü ele alınacak olursa hata düzeltmede harf tabanlı yöntemin yetersiz kalacağı görülür. *yaz-azi-zil-ilm-lmi-mis* biçiminde trigramlar içeren bu sözcük için modelde örneğin *zil* trigramının sıklığı düşük ama *miş* trigramının sıklığı yüksek olabilir. Bu durumda hata düzeltme işleminde *zil* trigramı kendisine benzer olan ve sıklığı daha yüksek olan *zıl* ile yerdeğiştirirken *miş* trigramı

---

* "Veri seyrekliği" İngilizce "data sparseness" terimine karşılık kullanılmıştır.

** "Bilgi çıkarımı" İngilizce "information extraction" terimine karşılık kullanılmıştır.

aynen bırakılır ve ortaya *yazılmiş* sözcüğü çıkar. Bulunan bu sözcüğün olasılığı her bir trigramın olasılıklarının çarpımı olarak verileceğinden doğru bir sözcük olarak bulunur.

Bu nedenle harf tabanlı yöntem, bigram ve trigram modeller kullanıldığında yetersiz kalmaktadır.

2.5 Sözcük Tahmini

N-gram modellerde temel birim olarak sözcük seçildiğinde sözcüklerin art arda gelme olasılıkları modellenir, bu nedenle n-gramlar bir cümlede herhangi bir sözcükten sonra gelebilecek sözcüklerin tahmin edilmesi amacıyla da kullanılır. Sözcük tahmini, konuşma tanıma, el yazısı tanıma ve optik karakter tanıma gibi uygulamalarda önemli bir yer tutmaktadır.

2.6 Veri Seyrekliği Problemi ve Olasılık Yoğunluğunu Paylaştırma* İşlemleri

George Kingsley Zipf tarafından ortaya atılan ve Zipf Yasası olarak bilinen kurala göre bir dildeki farklı sözcük çeşitlerinin çoğunluğunun rastlanma sıklığı çok düşük, küçük bir kısmının rastlanma sıklığı yüksektir.

Bölüm 2.1'de anlatılan En Büyük Olabilirlik Kestirimi yöntemiyle yapılan olasılık hesabında, eğitim kümesi içinde rastlanmayan çok sayıda farklı sözcüğe sıfır olasılık değeri atanır. Bu nedenle Zipf yasasına göre bu yöntem ihtiyacı tam karşılayamamaktadır.

Geoffrey Sampson [12] dil modellerinde n-gramların olasılıklarının kestirilmesi konusunu bir örnekle şu şekilde açıklar; bir bahçede görülen çeşitli kuşların görülme sıklıkları ele alınmış ve örneğin ilk görülen 1000 adet kuşun ne olduğu not edildiğine 212 serçe, 109 güvercin, 58 karga ve daha az sayılarda diğer çeşitlerden görüldüğü kaydedilmiş olsun. Toplam 30 farklı çeşit kuş görülmüş olduğunu varsayalım. Bir sonraki görülecek kuşun karga olma olasılığını hesaplamak için bu sayılar nasıl kullanılabilir sorusu sorulduğunda ilk akla gelen cevap en iyi tahminin 58/1000=0.058 şeklinde olacağıdır. Peki ilk görülen 1000 kuş içinde yer almayan

---

* "Olasılık yoğunluğunu paylaştırma" İngilizce "Distributing probability density" terimine karşılık kullanılmıştır.

ancak ara sıra da olsa bahçede görülebilen bir kuş çeşidi, diyelim ki bülbül görülme olasılığı nedir? Elimizdeki verilere göre eğer karga görme olasılığı 58/1000 ise bülbül görme olasılığı 0/1000 yani 0 olmalıdır. Bu durumda bülbül görme olasılığı bu şekilde bir hesaplamayla gerçek değerinin altında, karga görme olasılığı ise gerçek değerinin üstünde hesaplanmaktadır. Anlatılan bu örnek, dil modellerinde veri seyrekliği problemini de açıklamaktadır.

Bölüm 1.4'de belirtildiği gibi sözcük tabanlı bigram bir modelde bir cümlenin olasılığı şu şekilde bulunabilir.

P(Çin yemeği yemek istiyorum) =

P(Çin|<s>) P(yemeği|Çin) P(yemek|yemeği) P(yemek|istiyorum)

Örnek cümle şu şekilde de olabilirdi; "Hint yemeği yemek istiyorum." Ancak eğer etiğim kümesi içinde "Hint" ve "yemeği" sözcükleri hiç art arda gelmemişlerse modelde P(yemeği|Hint) = 0 olacaktır. Bu durumda cümlenin olasılığı da 0 olarak bulunacaktır.

P(Hint yemeği yemek istiyorum) =

P(Hint|<s>) P(yemeği|Hint) P(yemek|yemeği) P(yemek|istiyorum) = 0

Eğitim kümesi içinde hiç rastlanmayan ama gerçekte olası bir n-gram öğesi içeren bir sözcük dizisinin olasılığı sıfır olur. Bu probleme veri seyrekliği problemi denir. Gerçek hayatta her eğitim kümesi sonlu sayıda sözcük içereceği için hiç kullanılmamış olan sözcükler nedeniyle veri seyrekliği ciddi bir problem haline gelir.

2.6.1 Olasılık Yoğunluğunu Paylaştırma

Eğitim kümesinde hiç geçmediği için gerçek olasılık değerinin altında olasılık atanan sözcükler olması nedeniyle oluşan veri seyrekliği problemi sıfır olasılıklı n-gramlara sıfır olmayan bir olasılık değeri atamaya yönelik tekniklerle çözülebilir. Buna olasılık yoğunluğunu paylaştırma ya da düzleştirme* denir. Bu işlem veri seyrekliği problemini çözer ancak modelin bulanıklaşmasına neden olur; tüm olasılık değerleri biraz daha ortalamaya yaklaşır. N-gram modellerin olasılık yoğunluğunu paylaştırma işlemi için aşağıda kısaca açıklanan yöntemler kullanılmıştır.

---

* "Düzleştirme" İngilizce "Smoothing" terimine karşılık kullanılmıştır.

2.3.3.1 Bir Ekleme Yöntemi

(Lidstone 1920; Johnson 1932; Jeffreys 1948) N-gram modellerde her bir n-gramın olasılığının hesaplanması için öncelikle n-gramların eğitim kümesi içinde kaç kez geçtikleri sayılır. Bir ekleyerek olasılık yoğunluğunu dağıtma yöntemi her bir n-gram adedine 1 eklenmesidir. Örneğin unigram (1-gram) bir model için bir ekleyerek yeni olasılık değerlerleri şu şekilde bulunur:

$$P(w_x) = \frac{C(w_x)}{\sum_i C(w_i)} = \frac{C(w_x)}{N} \tag{2.10}$$

Burada c* = Ayarlanmış sayı (adjusted count)'dır.

$$c_i^* = (c_i + 1)\frac{N}{N+V} \tag{2.11}$$

N (eğitim kümesindeki toplam n-gram sayısı) ile normalize ederek sayılardan p^*_i olasılığı elde edilebilir. Burada V dildeki toplam n-gram sayısıdır.

Diğer bir yoldan p^*_i doğrudan sayılardan şu şekilde bulunabilir.

$$p_i^* = \frac{(c_i + 1)}{N+V} \tag{2.12}$$

Bir ekleme yöntemi n-gram modellerin olasılık yoğunluklarını paylaştırma amacı için ilk akla gelen ve diğer yöntemlere temel olan bir yaklaşımdır. Diğer yöntemlere göre daha kolay uygulanabilir. Ancak n-gram sayılarına bir ekleme sıfır olasılıklı n-gramların olasılığını çok arttırırken yüksek olasılıklı olanları çok azaltır, gerçek olasılık değerlerinin çok fazla değişmesine neden olur. 1 yerine 0,5 gibi daha küçük değerler kullanılarak bu sorun çözülmeye çalışılabilir ama her farklı durum için bu değerin ne olacağına karar vermeyi gerektirir. Bu nedenle zayıf bir yöntemdir. Diğer yöntemlere temel oluşturması bakımından önemlidir.

2.3.3.2 Witten-Bell Yöntemi

Bir ekleme yöntemine göre daha gelişmiş olan bu yöntem şu yaklaşımı temel alır; bir n-gram'a bir kez rastlanmış olması olasılığı kullanılarak hiç rastlanmamış n-gram'ların olasılıkları modellenebilir. Bu yöntem bir kez rastlanan n-gram'ların sayısını, hiç rastlanmayan n-gram'ların sayısını bulmak için kullanır.

Eğer P(wi | wi-1,…,wi-m) = 0 ve wi-1,…,wi-m dizisi birçok farklı wi için geçiyorsa düzleştirilmiş olasılık P^*(wi | wi-1,…,wi-m) daha yüksek olur. Ancak wi-1 ve wi eğitim kümesinde yoksa düzleştirilmiş olasılık halen sıfır olarak kalır.

Witten-Bell yöntemi yüksek olasılıklı n-gramların olasılığını bir ekleme yöntemindekine göre daha az düşürür ve daha doğru sonuçlar verir.

2.3.3.3 Good-Turing Yöntemi

(Good, 1953) Witten-Bell yöntemine göre daha karmaşık bir yapıya sahip olan Good-Turing yönteminde sıfır ya da düşük olasılıklı n-gram'ların olasılık tahmini için daha yüksek olasılıklı n-gram'lar kullanılır.

Bir n-gramın düzleştirilmiş olasılığını bulmak için önce düzleştirilmiş sayı denilen ve düzleştirme işleminden sonra o n-gramın eğitim kümesinde kaç kez geçmiş olduğunu göstereceği varsayılan değer hesaplanır.

$$c^* = (c+1)\frac{(N_{c+1})}{N_c} \tag{2.13}$$

Burada N_c eğitim kümesinde c kez geçen n-gramların sayısını, yani C(α)=c olan α elemanların sayısını, ve N_{c+1} c+1 kez geçen n-gramların sayısını ifade eder. Bu yüzden N_c değerine c sıklığının sıklığı adı verilir.

Bu değer N ile gösterilen tüm n-gramların sayısına bölünerek c kez geçen bir α n-gramının olasılığı bulunabilir. [13]

$$P_{GT}(\alpha) = \frac{c^*}{N} \tag{2.14}$$

Gale ve Sampson tarafından Good-Turing yöntemini temel alan "Good Turing Frequency Estimation Without Tears" [12] adlı çalışmada bu amaçla kullanılabilecek bir algortima geliştirilmiştir (1995). Good –Turing yöntemi diğer pek çok düzleştirme tekniğinin temelini oluşturur.

2.3.3.4 Kneser-Ney Yöntemi

Witten-Bell yöntemine benzer ancak bir sözcükten önce gelen farklı dizi sayısını sayısını (eğer bigram modelse farklı sözcük sayısını) ele alır.

2.3.3.5 Jelinek-Mercer Yöntemi

(Jelinek-Mercer, 1980) Yüksek dereceden bir n-gram modelin bir n-gramın olasılığını tahmin etmek için yeterli bilgiyi sağlayamadığı durumlarda daha düşük derecelerden n-gram modeller kullanılarak bir ara değer bulunmasını temel alan yöntemdir.

2.3.3.6 Katz Yöntemi

(Katz,1987) Eğer bir n-gramın olasılığı belli bir eşik seviyesinden düşükse daha düşük dereceden n-gramlara bakılarak yeni bir olasılık değerlendirmesi yapılır.

2.3.3.7 Church-Gale Yöntemi

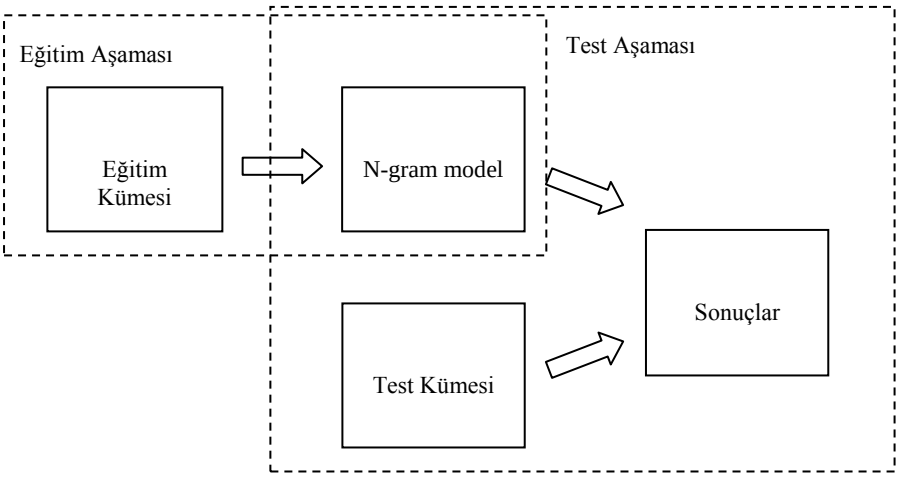
(Church-Gale, 1991) Katz yöntemine benzer, daha düşük dereceden ve daha yüksek dereceden n-gram modellerden elde edilmiş olan bilgi kullanılır.

3. BÖLÜM
HECE TABANLI İSTATİSTİKSEL YÖNTEM İLE YAZIM HATASI BULMA VE DÜZELTME

Bundan önceki bölümlerde yazım hataları, yazım hatası bulma yöntemleri, Türkçe'de hece yapısı gibi konular anlatılmıştır. N-gram modeller ile bir dilin nasıl modellenebileceği konusundan detaylı olarak bahsedilmiştir. Bu bölümde bu tez kapsamında Türkçe için geliştirilmiş olan hece tabanlı n-gram model anlatılacak ve daha sonra bu modelin Türkçe bir metin üzerinde yazım hatası bulma ve düzeltme amacıyla kullanılabileceği ispatlanmaya çalışılacaktır.

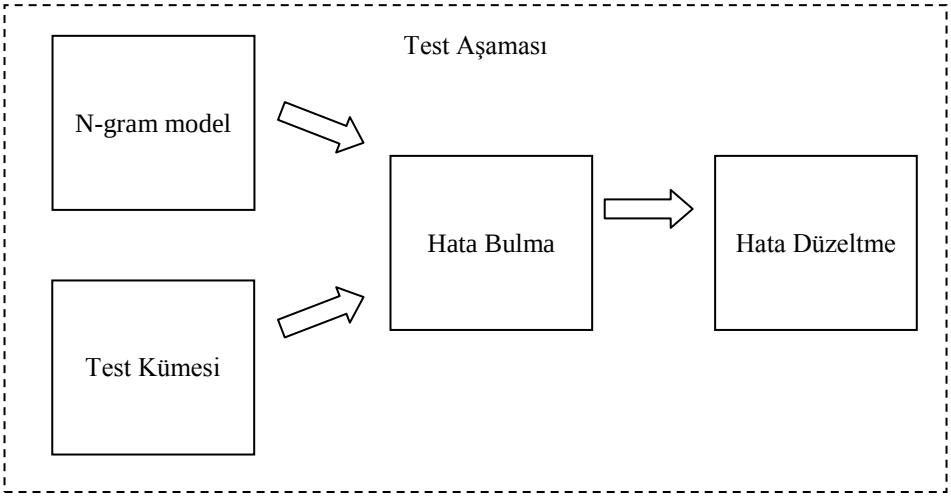
3.1 Giriş

Bu çalışmada temel olarak iki aşamadan oluşan bir sistem geliştirilmiştir. (Şekil 3.1) İlk aşama bir eğitim kümesi kullanarak n-gram bir dil modeli oluşturulması, ikinci aşama ise bu model kullanılarak bir test kümesi içindeki hatalı sözcüklerin bulunması ve düzeltilmesidir.



Şekil 3.1: Sistemin Genel Yapısı

İkinci aşama kendi içinde iki alt bölüme ayrılır; (Şekil 3.2) bunlardan birincisi hatalı olan sözcüklerin saptanması, ikinsici ise hatalı sözcüğün yerine sistem tarafından daha doğru olduğuna karar verilen başka bir sözcüğün kullanıcıya önerilmesidir.



Şekil 3.2: Test Aşaması

3.2 Türkçe için N-gram Modelde Kullanılabilecek Birimler

N-gram modellerde sözcük tabanlı ya da harf tabanlı yöntemlerden bölüm 2.4'de bahsedilmişti. Oysa ki doğal dillerde bir dilin özelliklerini içeren, harften daha büyük sözcükten daha küçük başka bir yapı taşı daha vardır; heceler.

Türkçe'de kullanılan farklı hece sayısı türetilebilecek farklı sözcüklerin sayısına göre daha sınırlıdır. Tablo 3.1'de bazı örnek eğitim kümelerinden elde edilmiş olan istatistiksel bilgiler verilmiştir. Görülebileceği gibi daha küçük kümelerde farklı sözcük adedi farklı hece adedinden sadece bir kaç kat fazla iken, eğitim kümesinin boyutu arttıkça farklı sözcük adedi çok hızlı biçimde artmakta, farklı hece adedinden onlarca kat fazla olmaktadır. Bu nedenle Türkçe için sözcük tabanlı n-gram modellerde yaşanan veri seyrekliği problemi açısından hece tabanlı bir yöntem daha avantajlıdır.

Tablo 3.1: Örnek Eğitim Kümelerinden Alınmış Olan Bazı İstatistikler

	1.Örnek (29KB)	2.Örnek (688KB)	3.Örnek (1.9 MB)	4.Örnek (1,86MB)	5.Örnek (42,3MB)
Sözcük Adedi	1,989	7,761	23,789	83,014	323,513
Hece Adedi	834	1,397	2,386	4,647	8,797
Toplam Sözcük Adedi	3,682	97,871	198,31	774,762	5,413,671
Toplam Hece Adedi	10,218	183,216	557,863	2,829,970	19,773,659
Bigram Adedi	2,982	5,987	15,745	41,08	110,559

Ayrıca art arda gelen hecelerin incelenmesi ve istatistiklerinin çıkarılması, kural tabanlı yöntemlerde tek tek ele alınması gereken sesli uyumu, sessiz benzeşmesi gibi kuralların da burada kendiliğinden ele alınmasını sağlamaktadır. Bölüm 2.4.2'de bahsedilmiş olan harf tabanlı yöntemlerin yetersiz kaldığı durumlarda hece tabanlı yöntemin daha iyi sonuç verdiği bu çalışmada görülmüştür. Örneğin Bölüm 2.4.2'de ele alınmış olan örnek "yazilmiş" sözcüğü için hece tabanlı bigram modelde "ya-zil" ve "zil-miş" bigramları bulunur, "ya-zil" bigramında ikinci hece daha yüksek olasılık sağlayan "zıl" hecesiyle değiştirilir, daha sonra "zıl-miş" bigramı ele alınarak "zıl-mış" olarak düzeltilir.

İlerleyen bölümlerde anlatıldığı gibi bu çalışma kapsamında Türkçe için olabilecek tüm heceler sentetik olarak üretildiğinde 152,048 farklı hece elde edilmiştir. Ancak bu işleme Türkçe hece kuralları eklendiğinde elde edilen hece sayısı 6,160 olmuştur. Tablo 3.1'de 4. örnek olan eğitim kümesinin bu sayıya yakın farklı hece içerdiği, 5. örnek olan kümenin ise 8,797 farklı hece adedine sahip olduğu görülmektedir. Bunun nedeni kuraldışı sözcüklere sıkça rastlanmasıdır. Tabloda görülen bigram adetleri hece tabanlı bigram model için verilmiştir.

Hece tabanlı bir model oluşturulabilmesi için öncelikle sözcüklerin hecelerine ayrılması gerekmektedir.

Bu çalışmada Türkçe'de hece tabanlı modellerin kullanılabilirliği araştırılmış, karşılaştırma yapabilmek amacıyla harf tabanlı modeller de oluşturulmuştur.

3.3 Türkçe için Sözcüklerin Hecelerine Ayrılması

Bölüm 1.1'de Türkçe'de hece yapısı anlatılmıştı. Ünsüz harfler kendilerinden hemen önce bir sesli harf yoksa ve en çok iki sağında bir sesli harf varsa sonraki sesli ile birleşerek bir hece oluşturur, aksi halde önceki sesli ile birleşerek hece oluşturur.

Bir sözcüğün hecelerine ayrılması için bu çalışmada uygulanan yöntem aşağıda açıklanmıştır.

Boş bir hece oluşturulur. Her bir sözcük için sondan başlanarak harf harf bakılır. Okunan harf heceye eklenir. Ünlü harf ise ve ilk harfteysek (sözcük başına gelindiyse) (Kural-1) mevcut hece heceler listesine eklenir. Ünlü harf ise ve önceki harf de ünlüyse (Kural-2) mevcut hece heceler listesine eklenir ve yeni boş bir hece oluşturulur. Ünsüz harf ise, son harfte değilsek ve mevcut hece bir ünlü içeriyorsa ve şu kuralları da sağlıyorsa hece heceler listesine eklenir ve yeni boş bir hece oluşturulur; ya sözcük başına gelinmiş olmalı (Kural-3), ya önceki harf ünlü olmalı (Kural-4), ya da eğer önceki harf ünsüz ise şimdiki harf baştan ikinci harf olmamalı (Kural-5).

Örnek: "ela" sözcüğünün hecelerine ayrılması

1. adım: mevcut hece = "", hece listesi = ""
2. adım: mevcut hece = "a", hece listesi = ""
3. adım: mevcut hece = "la", hece listesi = "la"
4. adım: mevcut hece = "e", hece listesi = "e-la"

Örnek: "elma" sözcüğünün hecelerine ayrılması

1. adım: mevcut hece = "", hece listesi = ""
2. adım: mevcut hece = "a", hece listesi = ""
3. adım: mevcut hece = "ma", hece listesi = "ma"
4. adım: mevcut hece = "l", hece listesi = "ma"
5. adım: mevcut hece = "el", hece listesi = "el-ma"

Örnek: "türkü" sözcüğünün hecelerine ayrılması

1. adım: mevcut hece = "", hece listesi = ""
2. adım: mevcut hece = "ü", hece listesi = ""

3. adım: mevcut hece = “kü”, hece listesi = “kü”
4. adım: mevcut hece = “r”, hece listesi = “kü”
5. adım: mevcut hece = “ür”, hece listesi = “kü”
6. adım: mevcut hece = “tür”, hece listesi = “tür-kü”

Örnek: “saat” sözcüğünün hecelerine ayrılması

1. adım: mevcut hece = “”, hece listesi = “”
2. adım: mevcut hece = “t”, hece listesi = “”
3. adım: mevcut hece = “at”, hece listesi = “at”
4. adım: mevcut hece = “a”, hece listesi = “at”
5. adım: mevcut hece = “sa”, hece listesi = “sa-at”

Örnek: “türkler” sözcüğünün hecelerine ayrılması

1. adım: mevcut hece = “”, hece listesi = “”
2. adım: mevcut hece = “r”, hece listesi = “”
3. adım: mevcut hece = “e”, hece listesi = “”
4. adım: mevcut hece = “l”, hece listesi = “ler”
5. adım: mevcut hece = “k”, hece listesi = “ler”
6. adım: mevcut hece = “rk”, hece listesi = “ler”
7. adım: mevcut hece = “ürk”, hece listesi = “ler”
8. adım: mevcut hece = “türk”, hece listesi = “türk-ler”

3.4 Hece Tabanlı Bigram Model

Türkçe için hece tabanlı bigram model oluşturulması için eğitim kümeleri üzerinde şu işlemler yapılmıştır;

- Her bir sözcüğün hecelerine ayrılarak her hecenin kaç kez geçtiği sayılır.
- Art arda gelen her iki hece, olasılığının daha sonra kullanılacağı bir bigram oluşturur. Bu bigramların eğtim kümesinde kaçar kez geçtiğinin sayılır.
- Her bir bigramın olasılığı hesaplanır.

3.4.1 Hecelerin Sayılması

Eğitim kümesindeki her bir sözcük hecelerine ayrılır ve her hecenin kaç kez geçtiği sayılır. Tablo 3.2'deki örnekler arkeoloji ile ilgili bir metinden elde edilmiş olan hece sayılarından alınmıştır.

Tablo 3.2: Örnek Kümeden Alınmış Hece Sayıları

Hece	Adet
a	155
aç	1
af	1
ak	5
al	25
alt	2
am	1
an	9
ap	2
ar	33

3.4.2 Bigramların Sayılması

Eğitim kümesindeki her bir bigramın kaç kez geçtiği sayılır. Tablo 3.3'deki örnek önceki bölümde bahsedilen arkeoloji ile ilgili metinden elde edilmiş olan bigram sayılarından alınmıştır. Bu aşamada henüz sadece sayma işlemi yapıldığı ve olasılık hesabı yapılmadığı için olasılıklar sıfır olarak görülmektedir.

Tablo 3.3: Örnek Kümeden Alınmış Bigram Sayıları

Hece1	Hece2	Adet	Olasılık
a	<b>	1	0
a	bu	1	0
a	çı	3	0
a	dam	1	0
a	dı	1	0
a	ğaç	6	0
a	ğır	1	0
a	it	2	0
a	ka	1	0
a	kar	2	0
a	kı	1	0

3.4.3 Olasılıkların Hesaplanması

Hecelerin geçme sayıları ve bigramların geçme sayıları kullanılarak bigramların olasılıkları hesaplanır (Tablo 3.4). Bölüm 2.1den

$P(h_n|h_{n-1}) = C(h_{n-1}h_n) / C(h_{n-1})$

Örneğin

P(abu) = C(abu)/C(a) = 1/155 = 0.00645

P(açı) = C(açı)/C(a) = 3/155 = 0.01935

Tablo 3.4: Örnek Kümeden Alınmış Bigram Olasılıkları

Hece1	Hece2	Adet	Olasılık
a	<b>	1	0,00645
a	bu	1	0,00645
a	çı	3	0,01935
a	dam	1	0,00645
a	dı	1	0,00645
a	ğaç	6	0,03871
a	ğır	1	0,00645
a	it	2	0,0129
a	ka	1	0,00645
a	kar	2	0,0129
a	kı	1	0,00645
a	la	9	0,05806

Bulunan olasılık değerleri bir sözcüğün olasılığını hesaplamak için kullanılabilir. Denklem 2.8'de belirtilmiş olduğu üzere bir bigram dizisinin olasılığı kendisini meydana getiren bigramların olasılıklarının çarpımıdır. Buna göre hece tabanlı modelde bir sözcüğün olasılığı kendisini meydana getiren bigramların olasılıklarının çarpımıdır.

Örneğin; P(arkeoloji) = P(ke|ar) P(o|ke) P(lo|o) P(ji|lo)

3.5 Hece Tabanlı Trigram Model

Bu çalışmada önceki bölümde anlatılan hece tabanlı bigram model ile birlikte başarımlarının kıyaslanabilmesi amacıyla yine hece tabanlı trigram bir model daha gerçekleştirilmiştir. Bu bağlamda yine eğitim kümesinde her bir hecenin kaç kez geçtiği kullanılmış, ayrıca her bir hece üçlüsünün kaç kez geçtiği sayılmıştır. Bigram modelden farkı sadece bir önceki heceye değil önceki iki heceye göre koşullu olasılık

hesabı yapılmasıdır. Bu olasılık bir trigramın eğitim kümesinde geçme sayısını kendinden önce gelen bigramı ile başlayan tüm trigramların sayısına bölerek elde edilir.

3.5.1 Trigramların Sayılması

Eğitim kümesindeki her bir trigramın kaç kez geçtiği sayılır. Tablo 3.5'deki örnek önceki bölümde bahsedilen arkeoloji ile ilgili metinden elde edilmiş olan trigram sayılarından alınmıştır. Bu aşamada henüz sadece sayma işlemi yapıldığı ve olasılık hesabı yapılmadığı için olasılıklar sıfır olarak görülmektedir.

Tablo 3.5: Örnek Kümeden Alınmış Trigram Sayıları

Hece 1	Hece 2	Hece 3	Adet	Olasılık
a	çı	lır	1	0
a	çı	sın	2	0
a	dam	la	1	0
a	dı	<b>	1	0
a	ğaç	<b>	5	0
a	ğaç	lar	1	0
a	ğır	lı	1	0
a	kar	lar	2	0
a	kı	şı	1	0
a	la	nı	2	0
a	la	nın	6	0
a	la	şım	1	0

3.5.2 Olasılıkların Hesaplanması

Bigramların ve trigramların geçme sayıları kullanılarak trigramların olasılıkları hesaplanır. Bölüm 2.1den

$P(h_n| h_{n-2} h_{n-1}) = C(h_{n-2} h_{n-1} h_n) / C(h_n\text{-}2 h_n\text{-}1)$

Örneğin

$P(\text{açılır}) = C(\text{açılır}) / C(\text{açı}) = 1/3 = 0{,}333333333$

Tablo 3.6 önceki bölümde bahsedilen arkeoloji ile ilgili metinden elde edilmiş olan trigram sayılarından alınan örnekleri göstermektedir.

Tablo 3.6: Örnek Kümeden Alınmış Trigram Olasılıkları

Hece 1	Hece 2	Hece 3	Adet	Olasılık
a	çı	lır	1	0,333333333
a	çı	sın	2	0,666666667
a	dam	la	1	1
a	dı	<b>	1	1
a	ğaç	<b>	5	0,833333333
a	ğaç	lar	1	0,166666667
a	ğır	lı	1	1
a	kar	lar	2	1
a	kı	şı	1	1
a	la	nı	2	0,222222222
a	la	nın	6	0,666666667
a	la	şım	1	0,111111111

Bulunan olasılık değerleri bir sözcüğün olasılığını hesaplamak için kullanılabilir. Denklem 2.9'da belirtilmiş olduğu üzere bir trigram dizisinin olasılığı kendisini meydana getiren trigramların olasılıklarının çarpımıdır. Buna göre hece tabanlı trigram modelde bir sözcüğün olasılığı kendisini meydana getiren trigramların olasılıklarının çarpımıdır.

Örneğin; P(arkeoloji) = P(o|arke) P(lo|keo) P(ji|olo)

3.6 Harf Tabanlı Bigram Model

Önceki iki bölümde anlatılan hece tabanlı bigram ve trigram modeller, benzer işlemler ile harf tabanlı olarak da gerçekleştirilmiştir. Harf tabanlı bigram modelde iki harfin art arda gelme sayıları kullanılarak bigram olasılıkları şu adımlar ile hesaplanır.

- Eğitim kümesinde her harfin kaç kez geçtiğinin sayılması.
- Art arda gelen her iki harf, olasılığının daha sonra kullanılacağı bir bigram oluşturur. Bu bigramların eğtim kümesinde kaçar kez geçtiğinin sayılması.
- Her bir bigramın olasılığı hesaplanır.

Tablo 3.7'de örnek bir eğitim kümesinden elde edilmiş olan harf tabanlı bigramların olasılıkları görülmektedir.

Tablo 3.7: Örnek Kümeden Alınmış Harf Tabanlı Bigram Olasılıkları

Harf1	Harf2	Adet	Olasılık
<s>	a	274	0,0759
a	n	380	0,149724
n	t	48	0,043636
t	i	126	0,161125
i	k	153	0,10046
<s>	y	324	0,089751
y	a	272	0,385269
a	k	243	0,095745
k	ı	72	0,070381
ı	n	332	0,316492
n	d	254	0,230909
d	o	61	0,057493
o	ğ	66	0,118919
ğ	u	73	0,231746

Harf tabanlı modelde bir sözcüğün olasılığı yine kendisini meydana getiren bigramların olasılıklarının çarpımıdır.

Örneğin; P(arkeoloji) = P(r|a) P(k|r) P(e|k) P(o|e) P(l|o) P(o|l) P(j|o) P(i|j)

3.7 Harf Tabanlı Trigram Model

Harf tabanlı bigram modelden farkı sadece bir önceki harfe değil önceki iki harfe göre koşullu olasılık hesabı yapılmasıdır. Eğitim kümesinde her bir bigramın kaç kez geçtiği sayıldıktan sonra, ayrıca her bir trigramın kaç kez geçtiği sayılmıştır. Bir trigramın eğitim kümesinde geçme sayısını kendinden önce gelen bigram ile başlayan tüm trigramların sayısına bölerek trigramın olasılığı elde edilir.

Harf tabanlı trigram modelde bir sözcüğün olasılığı kendisini meydana getiren trigramların olasılıklarının çarpımıdır.

Örneğin; P(arkeoloji) = P(k|ar) P(e|rk) P(o|ke) P(l|eo) P(o|ol) P(j|lo) P(i|oj)

Tablo 3.8'de örnek bir eğitim kümesinden elde edilmiş olan harf tabanlı trigramların olasılıkları görülmektedir.

Tablo 3.8: Örnek Kümeden Alınmış Harf Tabanlı Trigram Olasılıkları

Harf1	Harf2	Harf3	Adet	Olasılık
<s>	a	n	12	0,043796
a	n	t	7	0,018421
n	t	i	11	0,229167
t	i	k	32	0,253968
<s>	y	a	201	0,62037
y	a	k	32	0,117647
a	k	ı	38	0,156379
k	ı	n	32	0,444444
ı	n	d	118	0,355422
n	d	o	24	0,094488
d	o	ğ	45	0,737705
o	ğ	u	49	0,742424

3.8 Hata Bulma ve Düzeltme

Birinci aşamada eğitim kümesindeki hecelerin art arda gelme olasılığı kullanılarak oluşturulan bigram ve trigram modeller, ikinci aşamada bir test kümesi içindeki yazım hatalarının bulunması ve hatalı olduğu kestirilen bir sözcüğün yerine gelebilecek sözcüğün bulunması için kullanılmıştır. Bunun için öncelikle test kümesindeki her sözcüğün heceler tabanında oluşturulmuş olan n-gram'lar kullanılarak olasılığı belirlenmiş, sözcük için bulunan olasılık değeri belli bir değerin altındaysa kendisine en yakın daha yüksek olasılıklı sözcük yine n-gram'lar kullanılarak bulunmaya çalışılmıştır.

3.8.1 Hata Bulma

Hata bulma işlemi test aşamasındaki birinci temel işlemdir. Kullanılan n-gramın derecesine göre hesaplanan hatalı olma eşik sıklığı değeri temel alınarak test kümesindeki her bir sözcüğün hatalı olup olmadığına karar verilir. Bir sözcüğün içerdiği n-gramlardan herhangi birinin geçme sıklığı eşik sıklığından düşükse sözcük hatalı kabul edilir ve bu n-gram hata düzeltme aşamasında kullanılmak üzere sözcük içinde hatanın bulunduğu yer olarak işaretlenir.

3.8.1.1 Hatalı Olma Eşik Sıklığı Değeri

Yazım hatası bulma işleminde bir sözcüğün hatalı olup olmadığına karar verebilmek için içerdiği n-gramların geçiş sıklıkları kullanılır. Bu geçiş sıklıklarından en az bir tanesi belli bir eşik değerinin altında ise sözcüğün hatalı olduğuna karar verilir.

Eşik değeri sistem tarafından modeldeki toplam n-gram sayısının farklı n-gram sayısına bölünmesiyle elde edilir.

Eşik Değeri = n-gramların toplam sayısı / farklı n-gram adedi

Sistem tarafından bulunan eşik değeri istenirse kullanıcı tarafından değiştirilerek farklı değerler ile hata bulma sonuçları gözlenebilir.

3.8.2 Hata Düzeltme

Hata düzeltme işlemi hata bulmaya göre daha karmaşık bir yapıdadır. Hata düzeltme ancak hatalı olduğu saptanan bir sözcük için daha yüksek olasılıklı başka bir n-gramlar dizisi oluşturulabiliyorsa yapılabilir. Bunun için yine ilk aşamada oluşturulmuş olan n-gram modeller kullanılır.

Bölüm 1.2'de anlatılan hata tipleri, heceleme açısından bakıldığında aşağıda gösterildiği gibi hata çeşitlerinin oluşmasına neden olur;

1. Sözcüğün hece yapısını bozmayan, sadece bir tek hecenin bozulmasına neden olan hatalar.
    a. Fazladan bir ünsüz harf eklendiği için bir hecenin bozulması;

        örnek : *bi-lim-sel* → *bi-lim-sel**l**,* *ya-kın-do-ğu* → *ya**n**-kın-do-ğu*

    b. Bir ünsüz harf eksik olduğu için bir hecenin bozulması;

        örnek : *ço-rak* → *ço-ra,* *yıl-lar-ca* → *ıl-lar-ca*

    c. Bir ünsüz harf başka bir ünsüz harf ile karıştığı için bir hecenin bozulması;

        örnek : *yö-ne-ti-ci-nin* → *yö-ne-ti-**v**i-nin*

2. Sözcüğün hece yapısını bozmayan, ama iki hecenin bozulmasına neden olan hatalar.
    a. Bir ünsüz harf eksik olduğu için iki hecenin bozulması;

        örnek : *yak-la-şı-mı* → *ya-ka-şı-mı*

    b. İki ünsüz harfin sırası karıştığı için iki hecenin bozulması;

        örnek : *ar-dın-dan* → *a**d**-**r**ın-dan*

3. Sözcükte hece kaybına yol açan hatalar

a. Bir ünlü harf eksik olduğu için hecenin kaybolması;

örnek : *a-lı-nıp → lı-nı, a-lı-nıp → a –lınp, e-leş-ti-ri → e-leş-tri*

b. Bir ünlü harfin yerine ünsüz harf gelmesi nedeniyle hecenin kaybolması;

örnek : *sa-yı-la-bi-le-cek → sa-yı-labş-le-cek*

4. Sözcükte fazladan hece oluşmasına yol açan hatalar

a. Fazladan bir ünlü eklenmiş olması;

örnek : *mü-ze-le-rin-de → a-mü-ze-le-rin-de, so-rum-lu→ so-rum-lu-ı*

Bu hata çeşitlerinden ilk iki gruptakiler sözcüğün hece sayısını değiştirmediği için n-gram model kullanan uygulamada ele alınması daha kolay olan hatalardır. Bu çalışmada geliştirilen yazım denetleyicide, bu tür hatalar hatalı olduğu saptanan sözcükte hatalı hecenin yerine kendisine yakın başka bir hece konularak giderilmeye çalışılmıştır. Üçüncü ve dördüncü gruptaki hatalar sözcüğün hece sayısını değiştirdiği için basit ek kurallar tanımlanmasını gerektirmişlerdir. Bu kurallar, sözcüğün dilbilgisi kurallarına göre çözümlenmesi ile ilgili olmayıp sadece hatalı sözcüklerin hecelerine doğru ayrılmasını sağlamaya ve bu şekilde hece tabanlı modelin daha elverişli kullanılmasına yöneliktir. Bu çalışmada önce sözcüğün hece sayısının değişmediği varsayılarak yazılım test edilmiş, daha sonra bu ek kurallar da eklenerek aynı test kümesi üzerinde bir kez daha test edilmiştir. Bu test sonuçları beşinci bölümde verilecektir.

Bigram modelde düzeltme işlemi şu şekilde yapılır;

1. Adım:

Hataya neden olduğu saptanarak işaretlenmiş olan bigramın önce ilk hecesi sabit tutulup ikinci hecesi değiştirilerek, daha sonra ikinci hecesi sabit tutulup ilk hecesi değiştirilerek bulunan sözcüklerin olasılıkları hesaplanır ve hatalı sözcüğün olasılığından daha yüksek olasılık veren tüm sözcükler aday sözcük listesine eklenir. Bu işlem sırasında bir hecenin yerine konulabilecek heceler bir sonraki bölümde anlatılacak olan iki hecenin benzerlik derecesi ile kısıtlanmıştır.

Eğer hataya neden olan hece tek harften (bir tek ünlüden) oluşan bir hece ise, bir de bu hece sözcükten çıkarılarak sözcüğün olasılığı hesaplanır ve ilk sözcüğünkinden daha yüksek olasılıklı bir sözcük elde edilmişse bu da aday

sözcükler listesine eklenir. Bu kontrol yukarıdaki hata çeşitlerinden 4'dekinin ele alınmasını sağlar.

Eğer hataya neden olan hece bir ünsüz, bir ünlü ve iki ünsüzden oluşuyorsa (SÜSS) yukarıdaki hata çeşitlerinden 3'deki gibi bu hecenin hatalı bir ses düşmesi sonucu oluşmuş olma ihtimali vardır. Bu durumda ya son ünsüzün yerine bir ünlü harf getirilerek (SÜSS→SÜ-SÜ) ya da hecenin sonuna fazladan bir tane de ünlü harf eklenerek (SÜSS→SÜ-SSÜ) bu hece ikiye bölünür. Bu şekilde oluşan sözcüklerin her birinin olasılıkları hesaplanarak olasılık değerlerine göre aday sözcükler listesine eklenir. Bu şekilde eğer hata nedeniyle düşmüş olan bir hece varsa yeniden oluşturulmaya çalışılmaktadır.

Eğer hataya neden olan hece iki ünsüz bir ünlü (SSÜ) ya da iki ünsüz bir ünlü bir ünsüz (SSÜS) biçimindeyse, yukarıdaki hata çeşitlerinden 3'deki gibi bu hecenin hatalı bir ses düşmesi sonucu oluşmuş olma ihtimali vardır. Bu durumda ilk ünsüzden sonra bir ünlü harf eklenerek bu hece ikiye bölünür (SSÜ→SÜ-SÜ ve SSÜS→SÜ-SÜS). Bu şekilde oluşan sözcüklerin her birinin olasılıkları hesaplanarak olasılık değerlerine göre aday sözcükler listesine eklenir. Bu şekilde eğer hata nedeniyle düşmüş olan bir hece varsa yeniden oluşturulmaya çalışılmaktadır.

Ayrıca eğer SÜSS biçimindeki bir hecede son iki ünsüz harf aynı ise, ya da SSÜS biçimindeki bir hecede ilk iki ünsüz harf aynı ise Türkçe ses kurallarına göre bu olamayacağı için bu iki ünsüzden biri çıkarılarak (SÜSS→SÜS ve SSÜS→SÜS) bulunan sözcük, olasılığına göre aday sözcükler listesine eklenir.

2. Adım:

Aday sözcük listesindeki sözcükler olasılıkları asıl sözcükten daha yüksek olan, fakat bu aşamaya kadar doğruluk kontrolleri yapılmamış sözcüklerdir. Bu aşamada her bir aday sözcüğün doğru olup olmadığı önceki bölümde anlatılan hata bulma yöntemi kullanılarak araştırılır, doğru bir sözcük bulununcaya ya da daha fazla olasılık iyileştirilmesi yapılamaz durumuna kadar aday sözcüğün olasılığı 1. adımda ilk paragrafta anlatılan hece değiştirme yöntemiyle iyileştirilmeye çalışılır.

3. Adım:

Aday sözcük listesindeki sözcüklerden en yüksek olasılığa sahip olan ilk on aday sözcük, hatalı sözcüğün yerine önerilmek üzere kaydedilir.

Trigram modelde de yine aynı adımlar ile hata düzeltme işlemi yapılır. Ancak bigramdakinden farklı olarak iki hecenin değil üç hecenin peşpeşe gelme olasılıkları kullanılır. Hataya neden olduğu saptanarak işaretlenmiş olan trigramın 1. adımda önce ilk iki hecesi sabit tutulup üçüncü hecesi değiştirilerek, daha sonra üçüncü hecesi sabit tutulup ilk iki hecesi değiştirilerek bulunan sözcüklerin olasılıkları hesaplanır ve hatalı olan sözcüğün yerine önerilmek üzere aday sözcükler listesine eklenir.

3.8.2.1 İki Hece Arasındaki Benzerlik Derecesi

Burada benzerlikten kastedilen iki hecenin birbirleriyle harf ve uzunluk yakınlıklarıdır. Bu çalışma kapsamında benzerlik değeri şu şekilde bulunur;

$$Benzerlik(h_1, h_2) = \frac{C(h_1 \cap h_2)}{\max(C(h_1)C(h_2))} \quad (3.1)$$

Burada $C(h_1 \cap h_2)$ iki hecenin ortak harf sayılarını, $\max(C(h_1)C(h_2))$ uzun olan hecenin harf sayısını ifade eder.

benzerlik(hece1, hece2) = tutan harf sayısı / uzun olan hecenin toplam harf sayısı

Örneğin

Benzerlik(“bin”,”bil”) = 2 / 3 = 0,6666

Benzerlik(“bin”,”lir”) = 1 / 3 = 0,3333

Benzerlik(“bin”,”min”) = 2 / 3 = 0,6666

olarak bulunur.

3.8.2.2 Sözcük Olasılığının Hesaplanması

Yukarıdaki adımlarda bir sözcüğün olasılığı kendisini meydana getiren n-gramların olasılıklarının çarpımı olarak hesaplanır.

Örnek : P(arkeoloji) = P(o|arke) P(lo|keo) P(ji|olo)

Ancak Bölüm 2.2'de belirtildiği gibi n-gramların olasılık değerleri tanım gereği her zaman 1'den küçük olduğu için sözcük olasılığı bulunurken sayı çok küçülebilmekte ve bu durum hataya neden olmaktadır. Bu yüzden sözcük olasılıkları önce her bir n-gramın olasılığının logaritması alınıp, bu değerler toplanıp en son ters logaritması alınarak hesaplanmıştır.

$$\log_2 P(w_{1\ldots}w_n) = \log_2 \prod P(w_i \ldots w_{i-1}) \tag{3.2}$$

$$\log_2 P(w_{1\ldots}w_n) = \sum \log_2 P(w_i \mid w_{i-1}) \tag{3.3}$$

Bulunan sonucun ters logaritması şu şekilde bulunur;

$$2^{\log(X)} = X \tag{3.4}$$

Bu çalışmada "2 tabanında logaritma" kullanılmıştır. Örneğin *arkeoloji* sözcüğünün olasılığı şu şekilde hesaplanır;

P(arkeoloji) = P(o|arke) P(lo|keo) P(ji|olo)

$= p_1\, p_2\, p_3$

$= 2^{(\log_2(p1) + \log_2(p2) + \log_2(p3))}$

Eğer modelde n-gramların olasılıkları En Büyük Olabilirlik Kestirimi yöntemi kullanılarak bulunmuş ise eğitim kümesi içinde geçmemiş olan n-gramların olasılığı sıfır olduğu için test kümesinde bu n-gramları içeren sözcüklerin olasılıkları sıfır olarak bulunur.

Eğer n-gramların olasılıkları Good-Turing olasılık düzleştirme yöntemi ile tüm olası n-gramlar arasında paylaştırılmışsa, eğitim kümesinde hiç geçmeyen n-gramlara bile bir olasılık değeri atanmış demektir.

Sözcük olasılığı hesaplanırken, içerdiği her bir n-gramın olasılığı ilgili n-gram tablosundan okunur. Eğer tabloda bu n-gram bulunamadıysa (eğitim kümesinde hiç geçmemişse) yazım denetleme aşamasında En Büyük Olabilirlik Yöntemi'nin kullanıldığı durumda bu n-gramın olasılığı sıfırdır. Good Turing Yöntemi'nin kullanıldığı durumda ise eğer bu n-gramın içerdiği her bir hece, bölüm 3.7'de anlatılan Türkçe için üretilmiş olan tüm heceler listesinde mevcutsa, yani Türkçe

diline ait bir hece ise, Good-Turing yönteminden elde edilmiş olan en düşük olasılık değerini alır, aksi halde sıfır değerini alır.

3.8.3 Örnek Bigram ve Trigram Modeller Kullanılarak Hata Bulma ve Düzeltme

Buraya kadar anlatılan işlemler örnek modeller ile daha detaylı biçimde aşağıda gösterilmektedir.

Toplam 3682 sözcük içeren arkeoloji ile ilgili küçük bir eğitim kümesi ile örnek bigram ve trigram modeller oluşturulmuştur. Bu modeller ile ilgili sayısal bilgiler Tablo 3.9'da verilmiştir.

Tablo 3.9: Örneklerde Kullanılacak Modeller ile İlgili Özet Bilgi

	Sözcük Adedi	**Hece Adedi**	**Bigram Adedi**	**Trigram Adedi**
Farklı	1989	834	3321	4636
Toplam	3682	13899	13898	10217

Her bir model için eşik sıklığı değeri toplam n-gram adedi farklı n-gramların sayısına bölünerek şu şekilde bulunur;

Bigram model için; 13,898 / 3,321 = 4

Trigram model için; 10,217 / 4,636 = 2

Modellerden bir tanesi seçilerek bir test kümesi için hata bulma işlemi yapılır. Örneğin bigram model seçilmiş ise test kümesindeki her bir sözcüğün içerdiği bigramların sıklıkları modelden okunur ve herhangi bir bigramının sıklığı 4'den küçük olan sözcükler hatalı kabul edilir. Bu işlem sırasında hataya neden olan bigram işaretlenir.

Hata düzeltme aşamasında öncelikle, hatalı olduğu saptanan sözcük içinde işaretlenmiş olan bigramın ilk hecesi ile başlayan tüm hece çiftleri modelden okunur. Alınan hece çiftlerinde ikinci hecesi hatalı olduğu varsayılan hece ile benzer ise o hece yerine denenerek oluşan sözcüğün olasılığında asıl sözcüğün olasılığından daha yüksek bir değer elde ediliyorsa aday sözcükler listesine eklenir, sözcük olasılığını en yüksek yapan çift bulunmaya çalışılır. Örneğin "dönebinde" sözcüğü için "ne-bin" bigramı hataya neden olduğu için işaretlenir. Tablo 3.10 örnek model içinde "ne" hecesi ile başlayan tüm bigramları göstermektedir.

Tablo 3.10: Örnek Modelde "ne" Hecesi ile Başlayan Bigramlar

Hece1	Hece2	Adet	Olasılık
ne		39	0,410526316
ne	bil	1	1,05E-02
ne	cek	2	2,11E-02
ne	de	2	2,11E-02
ne	den	4	4,21E-02
ne	ği	2	2,11E-02
ne	ğin	1	1,05E-02
ne	lir	1	1,05E-02
ne	me	3	3,16E-02
ne	mi	2	2,11E-02
ne	min	8	8,42E-02
ne	o	21	2,21E-01
ne	re	1	1,05E-02
ne	si	1	1,05E-02
ne	ti	2	2,11E-02
ne	ye	1	1,05E-02
ne	yi	1	1,05E-02
ne	yin	3	3,16E-02

"dönebinde" sözcüğü yerine önerilebilecek daha yüksek olasılıklı bir sözcük aranırken Bölüm 1.2'de bahsedilmiş olan hata tipleri gözününde bulundurularak sadece hatalı olan heceye benzer olan heceler seçilir.

Buna göre "ne-bin" hece çifti yerine denenecek olan hece çiftleri Tablo 3.10'da gösterilen "ne" ile başlayan tüm hece çiftleri değil, "bin" hecesine 0,5 olarak seçilen bir benzerlik eşik değerinden daha yüksek değerde benzerlik gösteren heceler alınarak elde edilen hece çiftleridir. Tablo 3.11'de ikinci hecesi "bin" hecesine benzerlik gösterdiği çin seçilen hece çiftleri benzerlik oranlarıyla birlikte gösterilmiştir.

Tablo 3.11: Örnek Modelde "ne" Hecesi ile Başlayan, İkinci Hecesi "bin" Hecesine Benzer Olan Bigramlar

Hece 1	Hece 2	2. hecenin "bin" hecesi ile benzerliği
ne	bil	0,6666
ne	ğin	0,6666
ne	min	0,6666
ne	yin	0,6666

Seçilen bu bigramların her biri "dönebinde" sözcüğünde "ne-bin" bigramı yerine konularak oluşturulan sözcüklerin olasılıkları hesaplanır ve en yüksek olasılığı veren sözcük işaretlenir.

P(döne**bil**de) = P(ne|dö) P(bil|ne)P(de|bil)

= 0,307692307692308 * 1,05E-02 * 0 = 0

P(döne**ğin**de) = P(ne|dö)P(ğin|ne) P(de|ğin)

= 0,307692307692308 * 1,05E-02 * 0,416666666666667 = 0.001346153

P(döne**min**de) = P(ne|dö)P(min|ne) P(de|min)

= 0,307692307692308 * 8,42E-02 * 0,333333333333333 = 0.008635897

P(döne**yin**de) = P(ne|dö)P(yin|ne) P(de|yin)

= 0,307692307692308 * 3,16E-02 * 0,4 = 0.003889230

Bu örnekte P(döne**min**de) en yüksek olasılık değerini alarak aday sözcükler içinde bulunan en iyi sonuç olmuştur.

İkinci adımda aynı işlemler işaretlenmiş olan bigramın ikinci hecesi ile biten tüm hece çiftleri alınarak tekrarlanır.

Tablo 3.12 oluşturulan model içinde "bin" hecesi ile biten tüm bigramları göstermektedir. Bu bigramlar içinde ilk hecesi "ne" ile benzer olan bir bigram bulunamadığı için ilk aşamada bulunmuş olan aday sözcükler aynen kalır.

Tablo 3.12: Örnek Modelde İkinci Hecesi "bin" Olan Bigramlar

Hece1	Hece2	Adet	Olasılık
<s>	bin	6	1,63E-03
ki	bin	1	9,52E-03
yüz	bin	1	0,142857143

Sonuç olarak tüm benzer çiftler denendikten sonra sözcüğün olasılığını en yüksek yapan bigramlar seçilir ve hatalı sözcükte yerine konarak bulunan ilk on sözcük düzeltilmiş sözcük olarak önerilir. Yukarıdaki örnekte "dönebinde" sözcüğü yerine "döneminde", "döneğinde" ve "döneyinde" sözcükleri önerilir.

Bölüm 1.2'de anlatıldığı gibi bazı durumlarda fazladan bir harfin sözcüğe eklenmesi hataya neden olur. Eğer hatalı olarak eklenen harf sesli harf ise sözcükte fazladan hece oluşmasına neden olabilir. Örneğin "amüzelerinde" sözcüğünde başa eklenen

“a” harfi fazladan bir hece, dolayısıyla fazladan bir bigram daha oluşmasına neden olur. Yukarıda anlattığımız hatalı olduğu işaretlenen bigramın önce ilk hecesi sonra ikinci hecesi değiştirilerek üretilen aday sözcüklerin dışında hatalı bigramın fazladan hece içermesi durumunu ele alabilmek amacıyla üçüncü bir kontrol daha yapılmıştır; eğer hatalı olarak işaretlenen bigramın bir hecesi tek harften oluşuyorsa o hece sözcükten tamamen çıkarılarak bulunan aday sözcüğün olasılığına bakılır ve daha iyi bir sonuç elde ediliyorsa düzeltme olarak önerilir. Örneğin “amüzelerinde” sözcüğü için “müzelerinde” ve “amazelerinde” biçiminde iki sözcük üretilir.

Benzer şekilde bölüm 3.8.2’deki yazım hatası düzeltme kısmında bahsedilen diğer tüm kurallar için bu şekilde düzeltmeler yapılır.

Bazı durumlarda bir sözcük birden fazla hata içeriyor olabilir. Bu durumda yukarıda anlatıldığı şekilde tek bir bigram üzerinde düzeltme yapmak yeterli olmamaktadır. Bu nedenle bulunan her aday sözcük hatalı olduğuna karar verildiği takdirde üzerinde düzeltme yapıldığı ve halen hatalı olduğu sürece tekrar aynı işlemlere tabi tutulur.

Örneğin “yazilmiş” sözcüğü için ilk aşamada “ya-zil” hecelerinden oluşan bigram hatalı olarak işaretlenir ve hata düzeltme işlemine tabi tutularak birinci adımda “yazılmiş” sözcüğü elde edilir. “yazılmiş” sözcüğünün sözcük olasılığı “yazilmiş” sözcüğünden daha yüksektir ancak “yazılmiş” sözcüğü de hata bulma işlemine girdiğinde hatalı bulunur ve bu kez “zıl-miş” hecelerinden oluşan bigram hataya neden olarak işaretlenir. Tekrar hata düzeltme işlemi gerçekleştirilir. Ancak bu aşamada ilk hatalı sözcükten uzaklaşmamak için daha fazla aday üretilmez, sadece en yüksek olasılığı veren sözcük alınır. Bu örnek için “yazılmış” sözcüğü bulunur ve bu sözcüğün hatasız olduğuna karar verileceği için daha fazla düzeltme işlemi yapılmaz.

3.9 Olasılık Yoğunluğunu Dağıtma

Bölüm 3.4’de anlatılan bigram model için Tablo 3.4’de örnek bir eğitim kümesinden elde edilmiş olan bigram sayıları ve olasılıkları gösterilmişti. Gerçekte Tablo 3.4’de görünmeyen, eğitim kümesinde rastlanmadığı için olasılığı sıfır olan bigramlar vardır, bu haliyle önceki bölümde oluşturulan bigram modelden alınmış bir örnek Tablo 3.13’de gösterilmiştir.

Tablo 3.13: Örnek Modelde Tüm Olası Hece Çiftleri İçin Gerçek Olasılıklar

Hece1	Hece2	Adet	Olasılık
a		1	6,45161290322581E-03
a	ba	0	0
a	be	0	0
a	bı	0	0
a	bi	0	0
a	bo	0	0
a	bö	0	0
a	bu	1	6,45161290322581E-03
a	bü	0	0
a	ca	0	0
a	ce	0	0
a	cı	0	0
a	ci	0	0
a	co	0	0
a	cö	0	0
a	cu	0	0
a	cü	0	0
a	ça	0	0
a	çe	0	0
a	çı	3	1,93548387096774E-02
a	çi	0	0

Dilde olası olan tüm hece çiftleri açısından bakıldığında elde edilen bigram modelin veri seyrekliği problemi ortaya çıkmaktadır. Tablo 3.13'de görüldüğü gibi "a" hecesiyle başlayabilecek olası hece çiftlerinin bir çoğuna eğitim kümesinde rastlanmadığı için bunların olasılıkları sıfır olarak bulunmuştur. Oysa örneğin "a-ba" hece çifti Türkçe'de gerçek kullanım olasılığı yüksek olan bir hece çiftidir. Bu durumda diyebiliriz ki eğitim kümesinin sınırlı sayıda sözcük içermesi bazı bigramların olasılıklarına gerçek olasılık değerlerinin altında değer atanmasına neden olurken bazı bigramlara da gerçek olasılık değerlerinin üstünde değer atanmasına neden olur. Bunun sonucunda sıfır olasılıklı bir bigram barındıran bir sözcüğün olasılığı gerçekte sıfırdan fazla olduğu halde sıfır olarak bulunur.

Veri seyrekliği probleminin başlıca nedeni eğitim kümesinin yetersizliğidir. Yukarıdaki örnekler sadece 834 farklı hece içeren 3682 sözcüklük bir eğitim kümesi kullanılarak oluşturulmuş olan bigram modelden alınmıştır. Eğitim kümesinin boyutu arttıkça sıfır olasılıklı bigramların sayısı azalacaktır ancak her eğitim kümesi sonlu

sayıda sözcük içereceği için eğitim kümesinin büyütülmesi sorunu tam olarak çözememektedir.

Veri seyrekliği problemi ve bu problemin giderilmesine yönelik uygulanan yöntemlerden bölüm 2.6'de bahsedilmişti. Bu tez kapsamında bahsedilen yöntemlerin başarımlarının kıyaslanması ile ilgili bir çalışma yapılmamış, daha önce yapılmış olan benzer çalışmalar kaynak alınmıştır.

N-gram olasılık yoğunluklarını paylaştırma için Gale ve Sampson tarafından geliştirilmiş olan ve Good-Turing yöntemini temel alan algoritma [12] bu çalışmada uyarlanarak kullanılmıştır. Bu algoritma modeldeki bigramların sıklıklarını ele alarak toplam olasılığı eğitim kümesinde geçiş sıklığı sıfır olan olası bigramlara da pay düşecek şekilde paylaştırır. Böylelikle eğitim kümesinde hiç geçmediği halde dilde mevcut olan bigramlara sıfırdan farklı bir olasılık atanmış olur, aynı zamanda toplam olasılık 1 olacak şekilde eğitim kümesinde geçmiş olan ve modelde yer alan bigramların olasılıkları da güncellenir. Bunun için öncelikle modeldeki bigramların geçiş sıklıkları sayılır, yani "sıklıkların sıklıkları" bulunur. Tablo 3.14'de örnek model için bigramların geçiş adetlerine göre sıklıkların listesi görülmektedir. Burada sadece ilk 15 satır alınmıştır. İlk satır, sıklığı 1 olan 1626 adet bigram olduğunu, ikinci satır sıklığı 2 olan 584 adet bigram olduğunu vs. ifade eder. Olasılık sütunu ise Good-Turing yöntemi ile hesaplanan olasılıkları göstermektedir.

Tablo 3.14: Örnek Model İçin Bigramların Geçiş Adetlerine Göre Sıklıkları

Bigram Adedi	**Sıklık (N_c)**	**Olasılık**
1	626	4,88E-05
2	584	1,15E-04
3	328	1,35E-04
4	163	2,77E-04
5	120	3,45E-04
6	68	4,13E-04
7	64	4,81E-04
8	57	5,49E-04
9	34	6,17E-04
10	26	6,85E-04
11	25	7,53E-04
12	21	8,22E-04
13	14	8,90E-04
14	25	9,58E-04
15	8	1,03E-03

Gale ve Sampson'ın algortimasında, modelde yer alan bigramların adetlerine göre olasılıkları yeniden hesaplanırken sıfır olasılıklı bigramlara düşen toplam olasılık değeri de hesaplanır. Bu nedenle geçiş sıklığı sıfır olan toplam bigram sayısı (N_0) bulunmalı ve bu toplam olasılık bu bigramlar arasında paylaştırılmalıdır.

Hece tabanlı modelde dildeki tüm hecelerin sayısı bilinirse bunların ikili permütasyonları hesaplanarak toplam bigram sayısı bulunabilir. Bu çalışma kapsamında Türkçe için önce sadece bir hecede bulunabilecek ünlü ve ünsüz sayılarının kuralları çerçevesinde tüm heceler üretilmiş, 152,048 farklı hece bulunmuştur. Daha sonra Bölüm 3.9.1 de anlatılacak olan kurallara göre hece üretildiğinde ise 6,160 farklı hece bulunmuştur. İkinci küme bazı yabancı kaynaklı sözcüklerde geçen heceleri içermese de Türkçedeki kurallı tüm heceleri içermektedir. Bu yüzden Tükçenin tüm hecelerini içeren kaynak olarak ikinci küme kullanılmıştır. Bu kümeye göre Türkçe'de olabilecek tüm bigramların sayısı

$$V = P\begin{pmatrix} 6160 \\ 2 \end{pmatrix} = 37{,}939{,}440$$

olarak bulunur. Geçiş sıklığı sıfır olan toplam bigram sayısı, tüm bigramların sayısından geçiş sıklığı sıfırdan farklı olan bigramların sayısı çıkarılarak bulunur;

$$N_0 = V - N \tag{3.5}$$

Örneğin sadece 3,321 adet bigramın sıklığı sıfırdan farklı ise adet sıklığı sıfır olan bigram sayısı 37,936,119 bulunur

$N_0 = 37{,}939{,}440 - 3{,}321 = 37{,}936{,}119$

Gale ve Sampson'ın algoritması tarafından hesaplanan sıfır olasılıklı bigramların toplam olasılık değeri bu toplam sayıya bölünerek her birine düşen olasılık değeri bulunur. Tablo 3.15'de örnek model için sıfır sıklıklı bigramların hesaplanan olasılık değeri görülebilir.

Tablo 3.15: Örnek Modelde Sıfır Sıklığa Sahip Bigramların Good-Turing Yöntemi ile Bulunan Olasılık Değeri

Bigram Adedi	Sıklık	Olasılık
0	37,936,119	3,08E-09

3.9.1 Hece Üretme

Bir n-gram model oluşturulduğunda eğitim kümesinde hiç geçmeyen ama dilde mevcut olan n-gramların olasılıklarının sıfırdan farklı olması gerektiği ve bunun için yapılan çalışma daha önceki bölümlerde anlatılmıştı. Hece tabanlı bir modelde sıfır olasılıklı n-gramların sayısının bulunabilmesi için olabilecek tüm hecelerin sayısının bilinmesi gerekir. Bu amaçla bu tez çalışması kapsamında Türkçe için tüm hecelerin üretilmesine yönelik bir çalışma yapılmıştır.

İlk olarak sadece bir hecede bulunabilecek harf sayısı ve ünlü-ünsüz sayısı göz önünde bulundurularak tüm heceler oluşturulmuştur.

Buna göre oluşan hece tipleri Tablo 3.16'da verilmiştir.

Tablo 3.16: Sentetik Olarak Üretilen Türkçe Hece Tipleri

Hece Tipi	**Örnek**
(Ü)	a
(Ü-S)	ab
(S-Ü)	ba
(S-Ü-S)	bak
(Ü-S-S)	alt
(S-Ü-S-S)	kalk
(S-S-Ü-S)	spor

Bu şekilde 152,048 adet farklı hece üretilmiştir. Ancak birinci bölümde anlatıldığı gibi Türkçe'de hece yapılarıyla ilgili birtakım kurallar vardır. Tablo 3.17 bu kurallar gözönüne alındığında üretilen hece tiplerini göstermektedir.

Tablo 3.17: Türkçe Hece Kurallarına Uygun Olarak Sentetik Üretilen Hece Tipleri

Hece Tipi	**Örnek**
(Ü)	a
(Ü-S)	ab
(S-Ü)	ba
(S-Ü-S)	bak
(Ü-SS)	alt
(S-Ü-SS)	kalk

Burada SS, hece sonlarında bulunabilecek kurallı "lç", "lk", "lp", "lt", "nç", "nk", "nt", "rç", "rk", "rp", "rs", "rt", "st" ünsüz çiftlerini ifade etmektedir. Bu şekilde 6,161 farklı hece üretilmiştir.

3.10 Geliştirilen N-gram Modeller Kullanılarak Sözcük Üretme

Bu tezde, bu çalışmanın asıl hedefi olan yazım hatalarının bulunması ve düzeltilmesi amacıyla geliştirilmiş olan bigram ve trigram modellerin, Türkçe'nin farklı açılardan analiz edilmesinde de rol oynayabileceği iddia edilmektedir. Oluşturulan altyapı kullanılarak, yeterince büyük eğitim kümeleri ile n-gram modeller geliştirildiğinde dil için sağlam bir istatistiksel model ortaya çıktığı gözlenmiştir. Bu iddiayı somutlaştrmak üzere çalışmanın asıl kapsamı dışında Sözcük Üretici adında bir proje daha geliştirilerek elde edilmiş olan modellerin rastgele sözcük üretme başarımları incelenmiştir. Sözcük Üretici'de, mevcut bir n-gram modelden sözcük başı simgesiyle başlayan rastgele bir n-gram seçilir, daha sonra her n-gramın peşinden gelebilecek en yüksek olasılıklı n-gram seçilip sözcük sonuna eklenir. Kaç adet n-gramın peşpeşe ekleneceği de rastgele seçilir. Bu tip bir uygulama, konuşma tanıma ve el yazısı tanıma gibi sistemlerde bölüm 2.5'de bahsedilen sözcük tahmini işlemi için kullanılabilir. Bu çalışmanın sözcük tabanlı modeller kullanılarak yapılan bu tür çalışmalardan farkı hece tabanlı olduğu için sonraki sözcüğün değil bir sözcük içindeki sonraki hecenin tahmin edilmesidir. Bu nedenle sözcük üretici olarak kullanılabilir.

Çalışma kapsamında hece tabanlı modellerin dışında harf tabanlı modeller de gerçeklendiği için Sözcük Üretici aynı zamanda harflerin art arda gelme olasılıkları da kullanılarak sözcük üretebilmektedir.

Tablo 3.18'de harf tabanlı ve hece tabanlı örnek bigram ve trigram modeller kullanılarak rastgele üretilmiş örnek sözcükler listelenmiştir.

Tablo 3.18: N-gram Modeller Kullanılarak Rastgele Üretilmiş Örnek Sözcükler

Harf Tabanlı Bigram	**Harf Tabanlı Trigram**	**Hece Tabanlı Bigram**	**Hece Tabanlı Trigram**
olarınd	duğun	cephe	teknolojik
rın	eri	metre	neredeki
sınd	mülerinda	kır	katmanla
warındarı	cıların	iz	bi
ma	hıları	sert	bir
verı	afarından	proto	ban
jind	bölgel	mar	türden
sınd	hayapı	çukur	yeniden
har	kındanınd	mü	kalan
far	yönemekt	uçları	kolleksiyonu
cerındar	zeyind	köpek	kişisel
çindarınd	üzeyinda	antik	ya
bindarınd	a	ları	kamyon
cerın	humaklar	bız	yıllar
ür	rumakları	iç	akla
yarı	yön	keoları	sürüklediği
i	çokarında	tı	astro
arınd	uzeyi	tedir	luristanda
v	ündan	sonu	akdeniz
rın	cilirin	ceylan	teknolojik
yarı	bonrak	kur	bız
olarında	stedirin	çö	kururken
da	banında	yem	kültünden
larınd	f	se	kurt
y	oy	kralı	kurtarma

4. BÖLÜM
YAZILIMIN AÇIKLANMASI

Bu bölümde hece tabanlı ve harf tabanlı n-gram modeller oluşturmaya olanak sağlayan ve oluşturulan model kullanılarak yazım hatası bulma ve düzeltme amacıyla geliştirilmiş olan yazılım tanıtılacaktır.

Bu çalışmada tek bir çatı altında dört ayrı yazılım geliştirilmiştir:

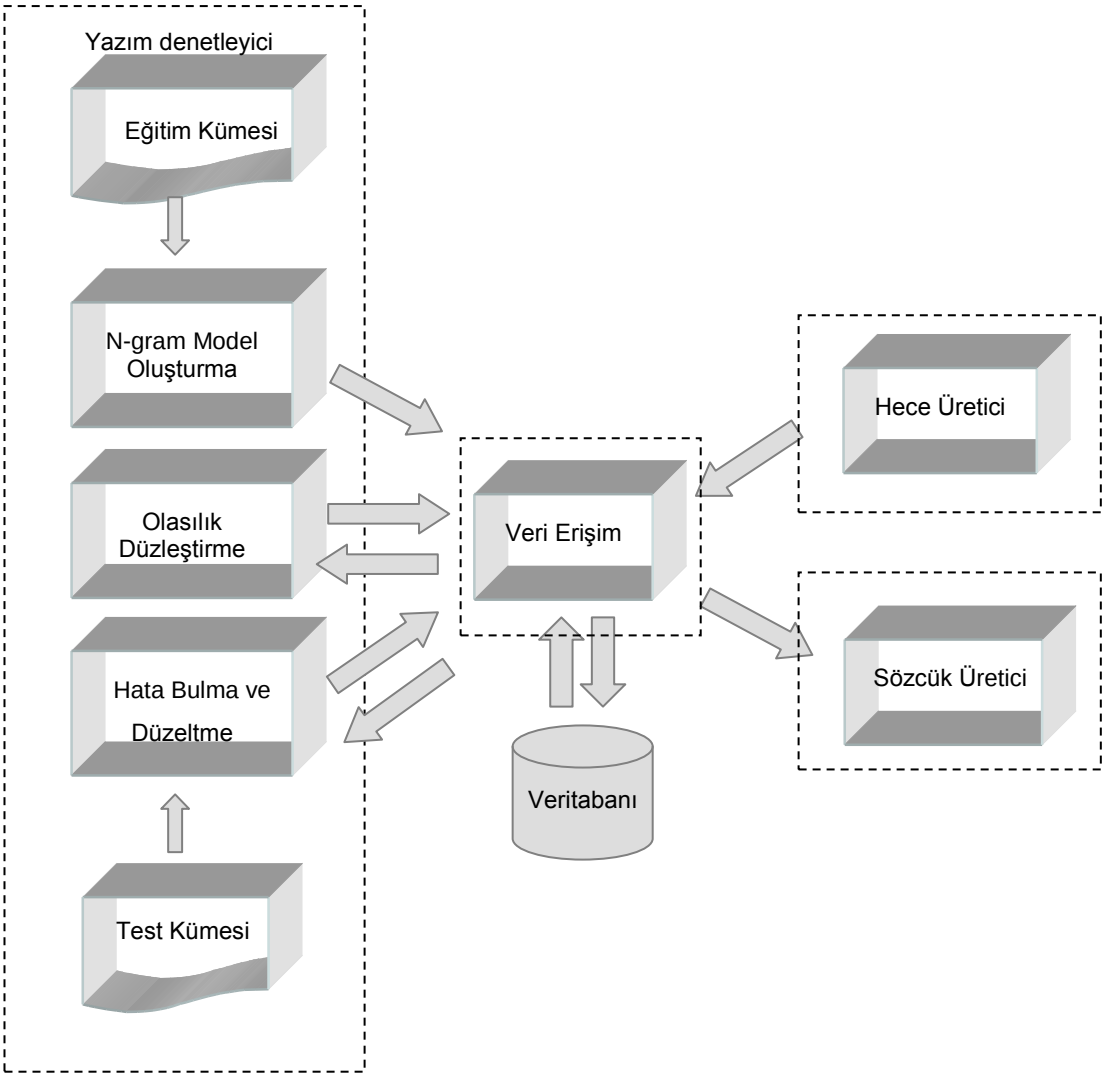
Yazım Denetleyici: N-gram modellerin oluşturulduğu, olasılık yoğunluk dağıtma işlemlerinin ve oluşturulan n-gram model kullanılarak yazım hatası denetleme işlemlerinin yapılabildiği uygulama yazılımıdır.

Hece Üretici: Türkçe için kurallı ve kuralsız olarak iki yöntem ile hecelerin üretilebildiği uygulama yazılımıdır. Burada üretilen kuralllı heceler daha sonra n-gram modelde sıfır olasılıklı n-gramların Good-Turing yöntemi ile gerçek olasılıklarının bulunması sırasında kullanılmak üzere veritabanına kaydedilir.

Sözcük Üretici: Oluşturulan n-gram modelin rastgele sözcük üretme konusunda başarımını test etmek amacıyla geliştirilmiş olan uygulama yazılımıdır. Veritabanından okunan ilgili modele ait istatistikler kullanılır.

Veri Erisim: Tüm diğer yazılımların veri tabanı erişimi için kullandıkları yöntemleri içeren kütüphanedir.

Şekil 4.1 sistemin genel yapısını göstermektedir.



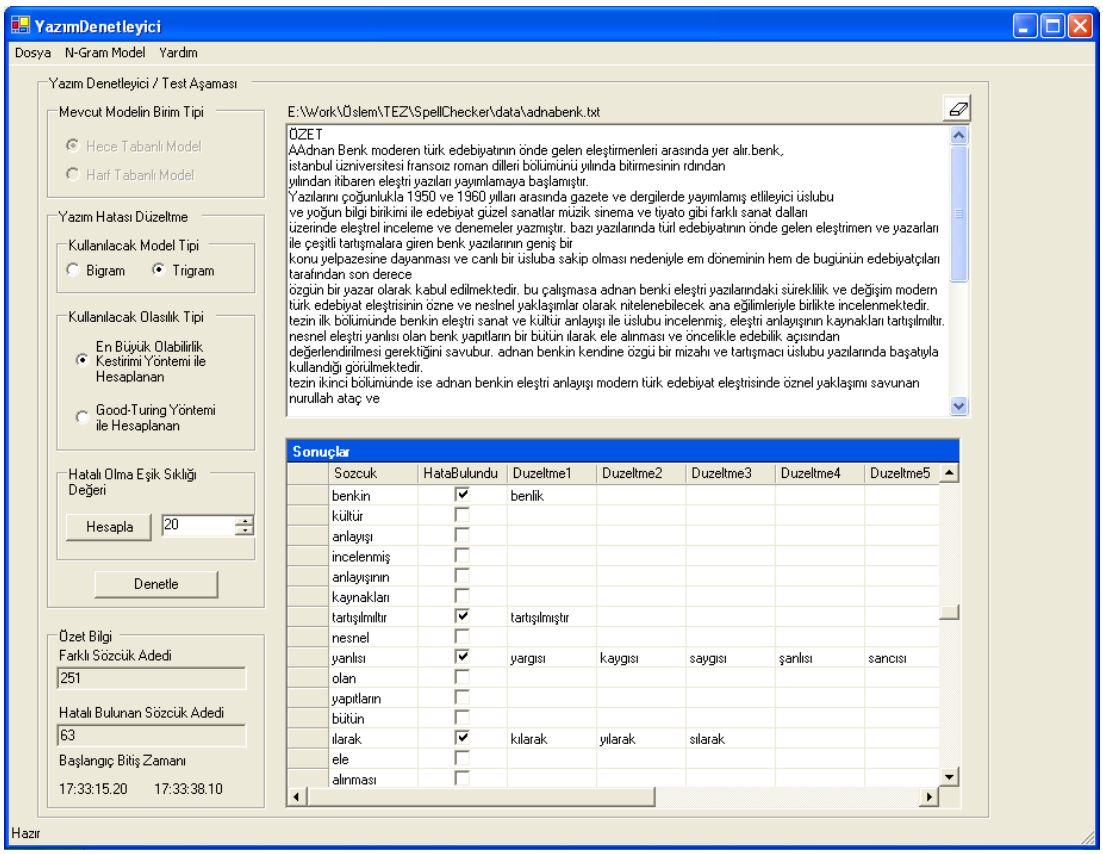
Şekil 4.1: Sistemin Genel Yapısı

4.1 Arayüz

Veri Erişim adlı proje sadece veritabanı ile diğer projeler arasında bir katman oluşturan kaynak kütüphanedir. Diğer üç proje ise birer arayüz aracılığı ile kullanıcı tarafından kullanılabilen uygulama yazılımlarıdır. Bu bölümde bu üç projenin kullanıcı arayüzleri tanıtılacaktır.

4.1.1 Yazım Denetleyici

N-gram modellerin oluşturulduğu, olasılık yoğunluk dağıtma işlemlerinin ve oluşturulan n-gram model kullanılarak yazım hatası denetleme işlemlerinin yapılabildiği, çalışmanın temel uygulama yazılımıdır. Şekil 4.2'de ana ekran görüntüsü yer almaktadır.

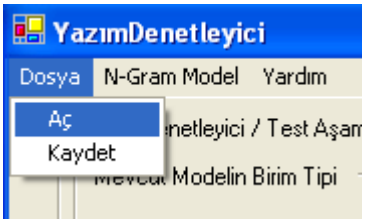


Şekil 4.2: Ana Ekran Görüntüsü

4.1.1.1 Menüler

Yazım Denetleyici ekranında menüden yapılan işlemler aşağıda açıklanmıştır.

Dosya menüsünün görüntüsü Şekil 4.3'de verilmiştir.



Şekil 4.3: Ana Ekran; Dosya Menüsü

Dosya Aç: Yazım hatası denetimi yapılacak olan giriş dosya açılır ve ekranda görüntülenir. Giriş dosya metin belgesi formatında olmalıdır.

Dosya Kaydet: Ekranda görülen metin değiştirilebilir bir alandır. Bir dosya olarak da saklanabilir.

N-Gram Model menüsünün görüntüsü Şekil 4.4'de verilmiştir.

Şekil 4.4: Ana Ekran; N-Gram Model Menüsü

N-Gram Model – Model Oluşturma: Şekil 4-10'da görülen N-gram model oluşturma ekranı açılır.

N-Gram Model – Düzleştirilmiş olasılıklar: Şekil 4.15'de görülen Düzleştirilmiş Olasılıklar ekranı açılır.

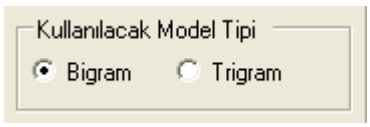
Yardım: Yardım ekranı açılır.

4.1.1.1 Alanlar

Şekil 4.5'de görülen "Mevcut Modelin Birim Tipi" alanı, veritabanında kayıtlı olan mevcut bir model varsa bu modelin harf tabanlı mı hece tabanlı mı olduğunu gösterir. Modelin bu özelliği yalnızca yeni model yaratılırken kullanıcı tarafından seçilebilir olduğu için bu ekranda seçilemez durumdadır.

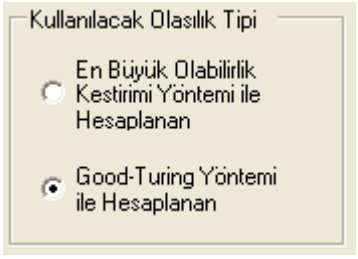
Şekil 4.5: Ana Ekran; Mevcut Modelin Birim Tipi Alanı

Şekil 4.6'da görülen "Kullanılacak Model Tipi" alanı, yazım denetleme işlemi için kullanıcının mevcut bigram ve trigram modeller arasında seçim yapmasına olanak verir.



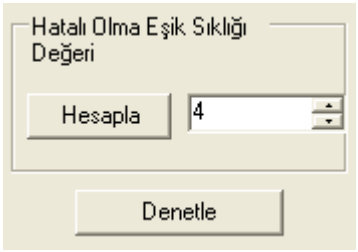
Şekil 4.6: Ana Ekran; Kullanılacak Model Tipi Alanı

Şekil 4.7'de görülen "Kullanılacak Olasılık Tipi" alanı, yazım denetleme işlemi için kullanılacak olasılık değerleri arasında seçim yapmaya olanak verir. N-gram model oluşturma aşamasında bigramların ve trigramların olasılıkları En Büyük Olabilirlik Kestirimi Yöntemi ile hesaplanır. Daha sonra Düzleştirilmiş Olasılıklar adımında Good-Turing yöntemi ile hesaplanan olasılıklar veritabanına kaydedilmişse yazım denetleme aşamasında kullanılabilir.



Şekil 4.7: Ana Ekran; Kullanılacak Olasılık Tipi Alanı

Şekil 4.8'de görülen "Hatalı Olma Eşik Sıklığı Değeri" alanı, bir sözcüğün doğru kabul edilmesi için her bir n-gramının modeldeki sıklığının en az ne olması gerektiğini gösteren değeri içerir. Hesapla düğmesine basıldığında seçili olan Model Tipine (bigram ya da trigram) göre ortalama sıklık değeri sistem tarafından hesaplanarak eşik değer alanına yazılır. Kullanıcı isterse bu değeri değiştirerek yazım denetiminde sözcüklerin doğru kabul edilmesini sağlayacak değeri ayarlayabilir.



Şekil 4.8: Ana Ekran; Hatalı Olma Eşik Sıklığı Değeri

Şekil 4.9'daki "Özet Bilgi" alanı, denetlenen metindeki farklı sözcük adedini ve bunlardan kaç tanesinin hatalı bulunduğunu gösterir.

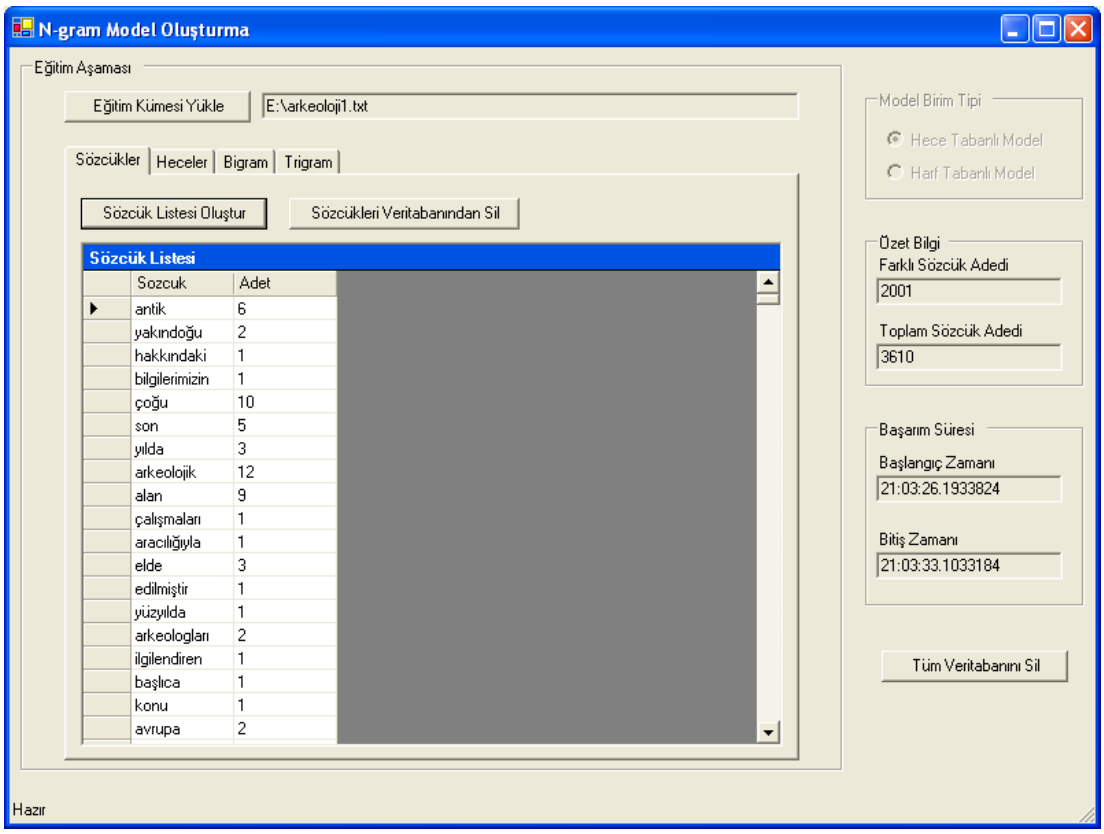
Şekil 4.9: Ana Ekran; Özet Bilgi Alanı

Denetle düğmesine basıldığında ekrandaki metin kutusu alanındaki metin alınarak yazım hatası denetleme işlemine tabi tutulur ve üretilen sonuçlar metnin alt kısmında görüntülenir.

4.1.1.3 Alt Ekranlar

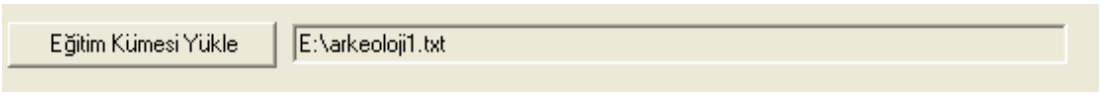
N-Gram Model Oluşturma:

Menüden N-Gram Model - Model Oluşturma seçildiğinde açılan ekran görüntüsü Şekil 4.10'da verilmiştir. N-gram Model Oluşturma ekranında eğitim aşamasının her bir adımı ayrı bir sekmede görülür. Bu adımlar sözcük listesinin, seçilen model tipine göre hece ya da harf listesinin, bigram listesinin ve trigram listesinin oluşturulmasıdır. Bu aşamalara ilişkin ekran görüntüleri EK-B'de verilmiştir.



Şekil 4.10: N-Gram Model Oluşturma Ekranı

N-gram model oluşturma aşamasında öncelikle şekil 4.11'de görülen alandan Eğitim Kümesi seçilmelidir.

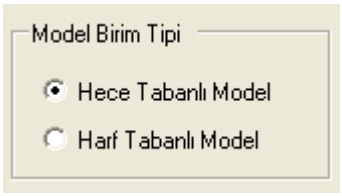


Şekil 4.11: N-Gram Model Oluşturma Ekranı; Eğitim Kümesi Seçimi

Sözcüklerin listelenmesi n-gram model oluşturma aşamasında önemsiz olmasına rağmen eğitim kümesindeki her bir farklı sözcüğün sıklığının kullanıcıya görüntülenmesi amacıyla eklenmiştir.

Heceler sekmesine geçildiğinde şekil 4.12'de görülen Model Birim Tipi alanı seçilebilir duruma gelir. Burada Harf Tabanlı Model Seçilirse bundan sonraki işlemler harf tabanlı yapılır, "Heceler" sekmesi "Harfler" olarak değişir. Harf

Tabanlı Model seçildikten sonra harf listesi oluşturulduğundaki ekran görüntüsü şekil Ekler-A, Şekil A.4'de verilmiştir.

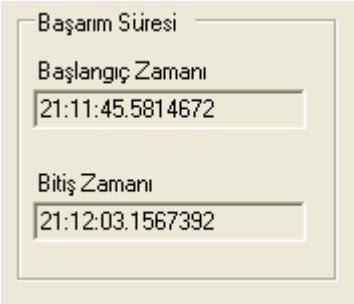


Şekil 4.12: N-Gram Model Oluşturma Ekranı; Model Birim Tipi Alanı

Şekil 4.13'de görülen "Özet Bilgi" alanı her bir sekmede oluşturulan liste ile ilgili özet bilgi verir.

Şekil 4.13: N-Gram Model Oluşturma Ekranı; Özet Bilgi Alanı

Şekil 4.14'de görülen "Başarım Süresi" alanı en son yapılan liste oluşturma işleminin balangıç ve bitiş zamanlarını gösterir.

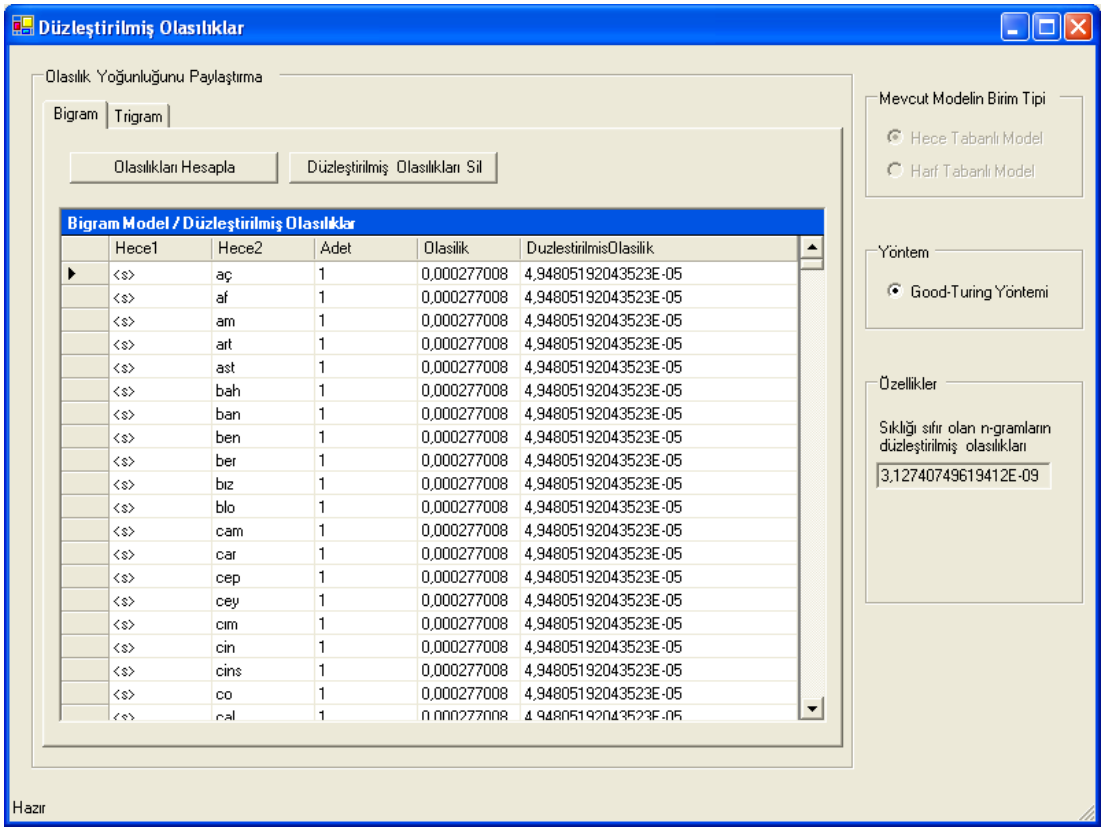


Şekil 4.14: N-Gram Model Oluşturma Ekranı; Başarım Süresi Alanı

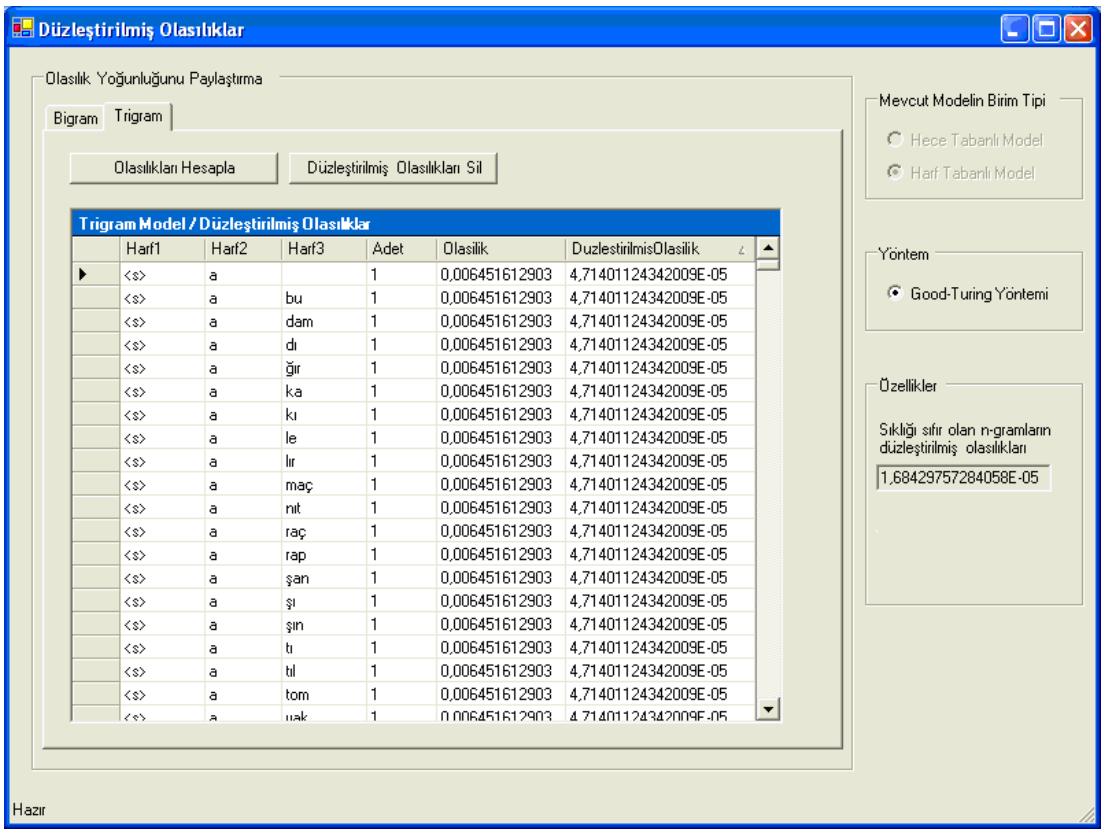
Her bir sekmede görülen "Listeyi Sil" düğmesi ile ilgili liste veritabanından silinir. Ekranın sağ alt köşesindeki "Tüm Veritabanını Sil" düğmesi mevcut modele ait tüm bilgilerin veritabanından silinmesini sağlar.

Düzleştirilmiş Olasılıklar:

Menüden N-Gram Model - Düzleştirilmiş Olasılıklar seçildiğinde açılan ekran görüntüsü Şekil 4.15'de verilmiştir. Bigram ve trigram modeller için ayrı sekmelerde olasılıklar hesaplanır. Şekil 4.15'de bigram model için, şekil 4.16'da da trigram model için hesaplanmış olan değerler görülmektedir.

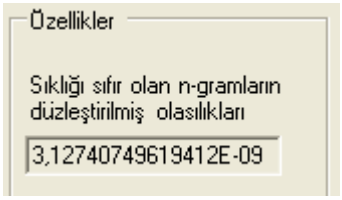


Şekil 4.15: Düzleştirilmiş Olasılıklar Ekranı, Bigram Sekmesi



Şekil 4.16: Düzleştirilmiş Olasılıklar Ekranı, Trigram Sekmesi

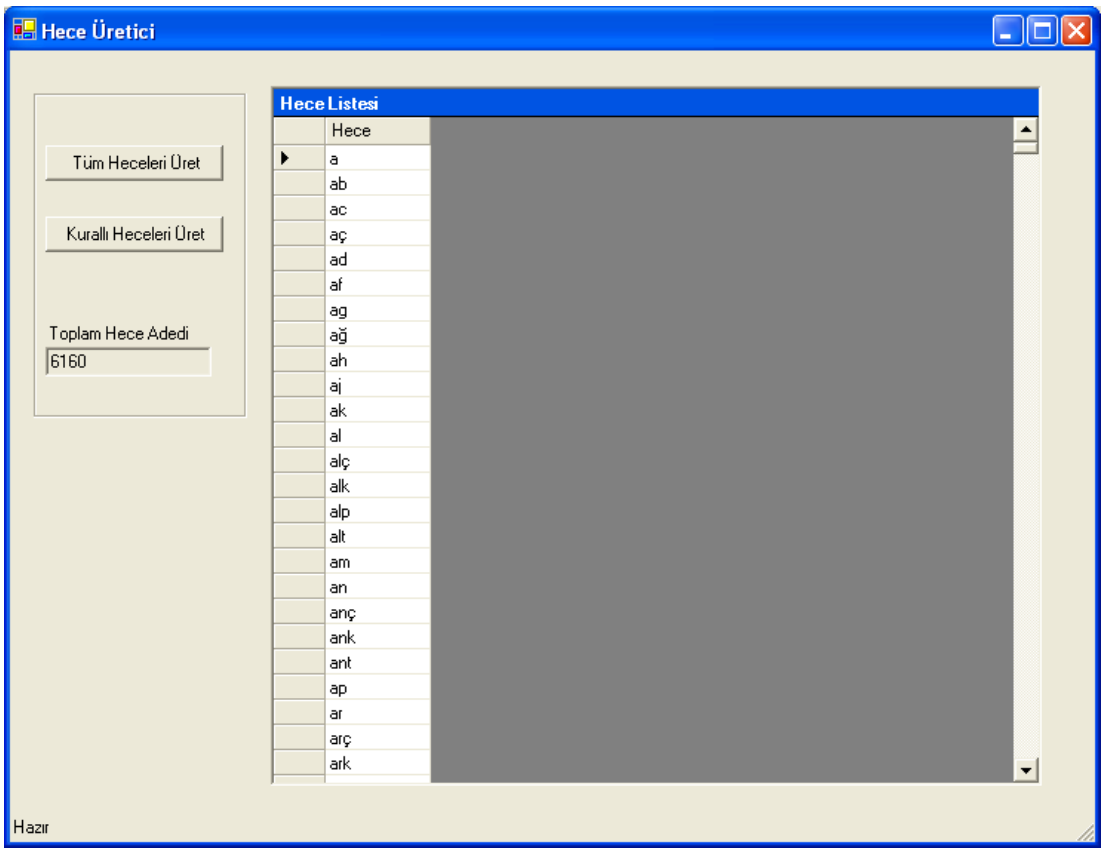
Düzleştirilmiş Olasılıklar ekranında sağ tarafta Mevcut Modelin Birim Tipi görülür. Sağ alt kısımda "Özellikler" alanında sıfır olasılıklı n-gramlara düzleştirme işlemi sonucunda atanan olasılık değeri görülür (Şekil 4.17)



Şekil 4.17: Düzleştirilmiş Olasılıklar Ekranı; Özellikler Alanı

4.1.2 Hece Üretici

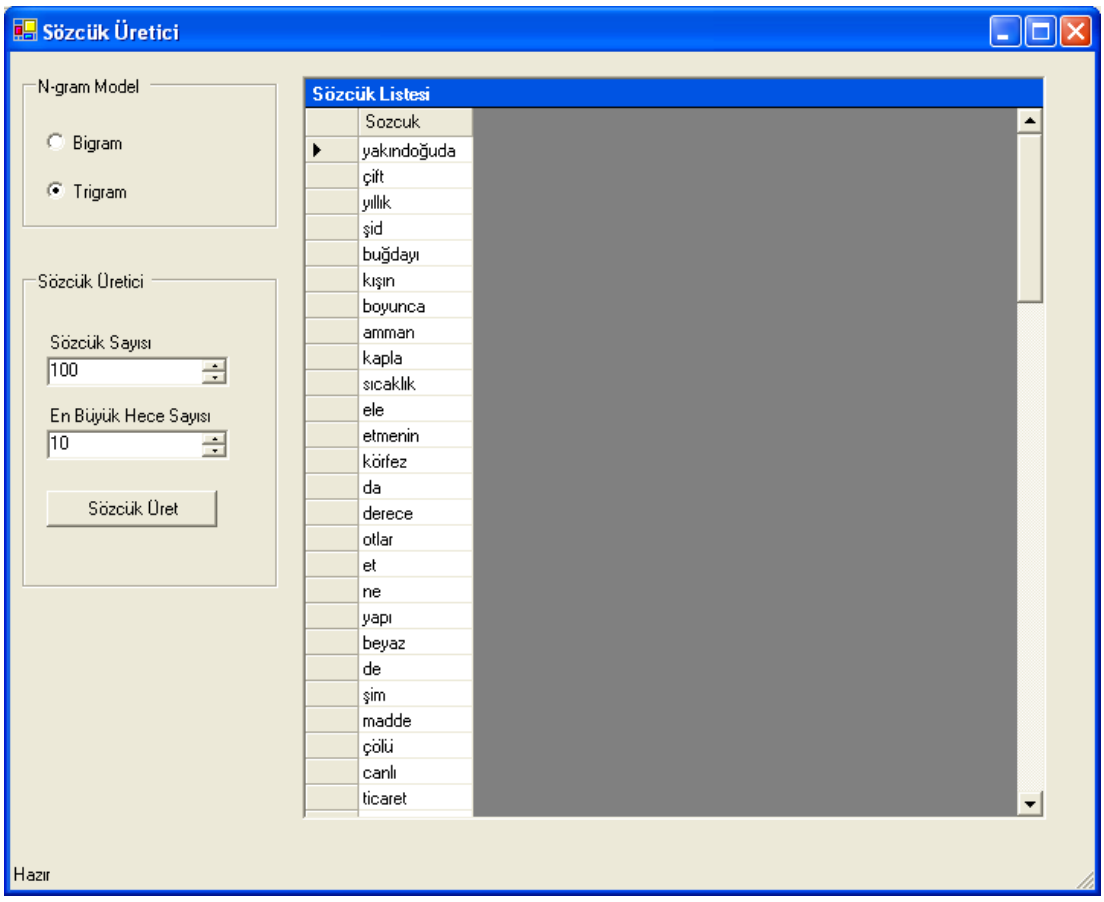
Şekil 4.18'de görülen Hece Üretici uygulamasında Türkçe için kurallı ve kuralsız olmak üzere iki ayrı biçimde tüm heceler üretilip listelenir. Burada kullanılan kuralların neler olduğu daha önce 3.9.1 nolu bölümde anlatılmıştır.



Şekil 4.18: Hece Üretici Ekranı

4.1.2 Sözcük Üretici

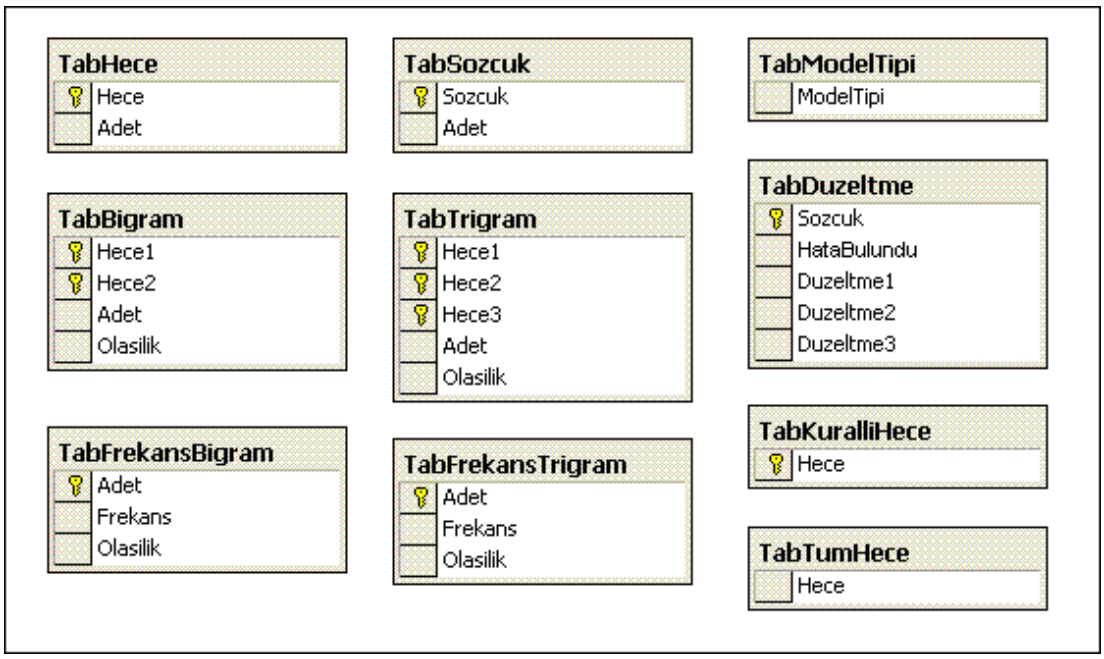
Şekil 4.19'da görülen Sözcük Üretici uygulamasında bigram ve trigram modeller kullanılarak rastgele sözcükler üretilir ve listelenir. Sözcük Üretme uygulamasından daha önce bölüm 3.10'da bahsedilmiştir. Burada kullanıcı kullanmak istediği model tipini, üretilmesini istediği sözcük sayısını, üretilecek sözcüklerin içereceği en büyük hece sayısını seçebilir.



Şekil 4.19: Sözcük Üretici Ekranı

4.2 Veri Tabanı

Bu çalışmada temel olarak iki aşamadan oluşan bir sistem geliştirilmiştir. İlk aşama bir eğitim kümesi kullanarak n-gram bir dil modeli oluşturulması, ikinci aşama ise bu model kullanılarak bir test kümesi içindeki hatalı sözcüklerin bulunması ve düzeltilmesidir. Bu nedenle birinci aşamada üretilen ve üretilme aşamasında veri tablosu yapılarında tutulan sonuçlar ikinci aşamada kullanılabilmek üzere bir veritabanında saklanır. Tabloların Yapısı Şekil 4.20'de görülmektedir.



Şekil 4.20: Veri Tabanı – Tabloların Yapısı

Tablolar oluşturuldukları ve kullanıldıkları aşamalara göre aşağıdaki şekilde gruplanabilir.

N-gram Model geliştirme aşamasında oluşturulan tablolar:

TabSozcuk: Eğitim kümesi içinde geçen her bir sözcük ve kaç adet geçtiği tutulur.

TabHece: Eğitim kümesi içinde geçen her bir hece ve kaç adet geçtiği tutulur. Eğer model harf tabanlı ise hece yerine harfler için aynı bilgiler tutulur.

TabBigram: Eğitim kümesi içinde art arda geçen her bir hece çifti, kaç adet geçtiği ve olasılık değeri tutulur. Eğer model harf tabanlı ise hece yerine harfler için aynı bilgiler tutulur.

TabTrigram: Eğitim kümesi içinde art arda geçen her bir hece üçlüsü, kaç adet geçtiği ve olasılık değeri tutulur. Eğer model harf tabanlı ise hece yerine harfler için aynı bilgiler tutulur.

TabModelTipi : Mevcut modelin hece tabanlı mı harf tabanlı mı olduğu bilgisi tutulur.

Olasılık Düzleştirme (Olasılık Yoğunluğunu Dağıtma) aşamasında oluşturulan tablolar:

TabFrekansBigram: Bigramlar için Good-Turing yöntemi ile hesaplanmış olan olasılık değerleri tutulur.

TabFrekansBigram: Bigramlar için Good-Turing yöntemi ile hesaplanmış olan olasılık değerleri tutulur.

Hece Üretme aşamasında oluşturulan tablolar:

TabTumHece: Türkçe için üretilebilen tüm heceler tutulur.

TabKuralliHece: Türkçe için hece kurallarına göre üretilebilen heceler tutulur.

Yazım Denetleme aşamasında oluşturulan tablolar:

TabDuzeltme: En son denetlenmiş olan test kümesi için denetlenen sözcük ve varsa düzeltmeler tutulur.

4.3 Nesnelerin Açıklanması

Şekil 4.1'de görülen işlemlerin gerçekleştirilmesi için nesneye dayalı bir programlama dili kullanılmış ve aşağıda açıklanan nesneler geliştirilmişir;

4.2.1 Hece Nesnesi

Eğitim kümesinde geçen hecelerin sayılması ve veritabanındaki TabHece tablosuna kaydedilmesine ilişkin işlemler bu nesne ile ele alınır. Nesnenin içerdiği yöntemler aşağıda listelenmiştir.

Yöntem No	Y 1.1
Yöntem Adı	ListeyiAl
İşlevsel Tanımı	Model tipi "hece" olarak seçilmiş ise eğitim kümesini içeren dosya okunarak her bir farklı hece sayılır ve hece ile adet bilgisini içeren bir veri tablosuna eklenir. Tüm liste oluştuğunda veri tabanına kaydedilir. Model tipi "harf" olarak seçilmiş ise aynı işlemler hece yerine harfler sayılarak yapılır. Model tipi veritabanına kaydedilir.
Yöntem Arayüzü	public DataTable ListeyiAl(StreamReader sr, int modelTipi)
Kullandığı Yöntemler	HeceYarat, HecelereEkle, SozcuktenHeceye, ModelTipiYaz
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Yaz
Veri yapıları	DataTable dtHece
Açıklama	Giriş dosyası sonuna kadar harf harf okunur, her bir sözcük okunduğunda hecelerine ayrılır ve her hece heceler listesine eklenir, hece listesi hece bazında tekildir (aynı heceye bir kez daha rastlanırsa listeye tekrar eklenmez, hece tablosunda adet alanı bir arttırılır). modelTipi alanı "harf" ise aynı işlemler hece yerine harfler ile yapılır

Yöntem No	Y 1.2
Yöntem Adı	HeceYarat
İşlevsel Tanımı	Veri Tablosu dtHecenin yaratılması
Yöntem Arayüzü	private void HeceYarat()
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtHece
Açıklama	Hece listesinin (ya da model tipine göre harf listesinin) tutulacağı veri tablosu dtHece yaratılır

Yöntem No	Y 1.3
Yöntem Adı	HecelereEkle
İşlevsel Tanımı	Veri Tablosu dtHece'ye hece eklenmesi
Yöntem Arayüzü	private void HecelereEkle(string hece)
Veri yapıları	DataTable dtHece
Açıklama	Verilen bir hece (ya da model tipine göre bir harf) dtHece listesinde mevcutsa adedi bir arttırılır, yoksa listeye eklenir.

Yöntem No	Y 1.4
Yöntem Adı	SozcuktenHeceye
İşlevsel Tanımı	modelTipine göre verilen sözcüğün harflerine ya da hecelerine ayrılması
Yöntem Arayüzü	private string SozcuktenHeceye(string sozcuk, int modelTipi)
Kullandığı Yöntemler	SozcuktenHarfe, SozcuktenHeceye(Y 1.5)
Açıklama	modelTipine göre Y 1.4 ya da Y 1.5 çağrılır

Yöntem No	Y 1.5
Yöntem Adı	SozcuktenHeceye
İşlevsel Tanımı	Verilen sözcüğün dilbilgisi kurallarına göre hecelerine ayrılması
Yöntem Arayüzü	private string SozcuktenHeceye(string sozcuk)
Kullandığı Yöntemler	-
Diğer Nesnelerden Kullandığı Yöntemler	BirHarf.UnluHarfmi
Açıklama	Verilen bir sözcük küçük harflere dönüştürülmüş ve ayraçlarla hecelerine ayrılmış olarak döndürülür.

Yöntem No	Y 1.6
Yöntem Adı	SozcuktenHarfe
İşlevsel Tanımı	Verilen sözcüğün harfleri alınır
Yöntem Arayüzü	private string SozcuktenHarfe(string sozcuk)
Açıklama	Verilen bir sözcük küçük harflere dönüştürülmüş ve ayraçlarla harflerine ayrılmış olarak döndürülür.

Yöntem No	Y 1.7
Yöntem Adı	ModelTipiYaz
İşlevsel Tanımı	"harf" ya da "hece" olarak seçilmiş olan model tipi veritabanına kaydedilir
Yöntem Arayüzü	private void ModelTipiYaz(int modelTipi)
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku, VeriErisim.Sil, VeriErisim.Yaz

4.2.2 BirHarf Nesnesi

Tek bir harfe ilişkin işlemlerin yapıldığı nesnedir, içerdiği tek yöntem olan UnluHarfmi yöntemi aşağıda gösterilmektedir.

Yöntem No	Y 2.1
Yöntem Adı	UnluHarfmi
İşlevsel Tanımı	Harfin ünlü harf olma kontrolü
Yöntem Arayüzü	private bool UnluHarfmi()
Açıklama	BirHarf nesnesinin o anki değerinin ünlü harf olup olmadığı bilgisi döndürülür.

4.2.3 BirHece Nesnesi

Tek bir heceye ilişkin işlemlerin yapıldığı nesnedir, içerdiği tek yöntem olan BenzerlikDerecesi yöntemi aşağıda gösterilmektedir.

Yöntem No	Y 3.1
Yöntem Adı	BenzerlikDerecesi
İşlevsel Tanımı	BirHece nesnesinin o anki değerinin verilen başka bir heceye olan benzerlik derecesi döndürülür.
Yöntem Arayüzü	private decimal BenzerlikDerecesi(string hece2)
Açıklama	Birinci hece ile ikinci hecenin ortak harf sayısının uzun olan hecenin toplam harf sayısınına oranlanmasıyla elde edilen benzerlik derecesi döndürülür.

4.2.4 Sözcük Nesnesi

Eğitim kümesinde geçen sözcüklerin sayılması ve veritabanındaki TabSozcuk tablosuna kaydedilmesine ilişkin işlemler bu nesne ile ele alınır. Nesnenin içerdiği yöntemler aşağıda listelenmiştir.

Yöntem No	Y 4.1
Yöntem Adı	ListeyiAl
İşlevsel Tanımı	Eğitim kümesini içeren dosya okunarak her bir farklı sözcük sayılır ve sözcük ile adet bilgisini içeren bir veri tablosuna eklenir. Tüm liste oluştuğunda veri tabanına kaydedilir.
Yöntem Arayüzü	public DataTable ListeyiAl(StreamReader sr)
Kullandığı Yöntemler	SozcukYarat, SozcuklereEkle
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Yaz
Veri yapıları	DataTable dtSozcuk
Açıklama	Giriş dosyası sonuna kadar okunur, her bir sözcük okunduğunda sözcükler listesine eklenir, sözcük listesi sözcük bazında tekildir (aynı sözcüğe bir kez daha rastlanırsa listeye tekrar eklenmez, sözcük tablosunda adet alanı bir arttırılır).

Yöntem No	Y 4.2
Yöntem Adı	SozcukYarat
İşlevsel Tanımı	Veri Tablosu dtSozcuk'ün yaratılması
Yöntem Arayüzü	private void SozcukYarat()
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtSozcuk
Açıklama	Sozcuk listesinin tutulacağı veri tablosu dtSozcuk yaratılır

Yöntem No	Y 4.3
Yöntem Adı	SozcuklereEkle
İşlevsel Tanımı	Veri Tablosu dtSozcuk'e sözcük eklenmesi
Yöntem Arayüzü	private void SozcuklereEkle(string sozcuk)
Veri yapıları	DataTable dtSozcuk
Açıklama	Verilen bir sozcuk dtSozcuk listesinde mevcutsa adedi bir arttırılır, yoksa listeye eklenir.

4.2.5 Bigram Nesnesi

Hece tabanlı model oluşturuluyorsa eğitim kümesinde art arda geçen her iki hecenin sayılması ve olasılıkları da hesaplanarak veritabanındaki TabBigram tablosuna kaydedilmesine ilişkin işlemler bu nesne ile ele alınır. Harf tabanlı model oluşturuluyorsa aynı işlemler harfler kullanılarak yapılır. Nesnenin içerdiği yöntemler aşağıda listelenmiştir.

Yöntem No	Y 5.1
Yöntem Adı	ListeyiAl
İşlevsel Tanımı	Eğitim kümesindeki bigramların sayılması ve olasılıklarının hesaplanması.
Yöntem Arayüzü	public DataTable ListeyiAl(StreamReader sr)
Kullandığı Yöntemler	BigramYarat, BigramEkle, OlasilikAta
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Yaz
Veri yapıları	DataTable dtBigram
Açıklama	Giriş dosyasındaki her bir hece çiftinin oluşturduğu bigramlar sayılır, bigramlar listesine eklenir. Bigram listesi hece1-hece2 bazında tekildir (aynı hece çiftine bir kez daha rastlanırsa listeye tekrar eklenmez, bigram tablosunda adet alanı bir arttırılır). Tüm liste oluştuktan sonra her bir bigramın olasılığı hesaplanarak veri tabanına kaydedilir.

Yöntem No	Y 5.2
Yöntem Adı	BigramYarat
İşlevsel Tanımı	Veri Tablosu dtBigram'ın yaratılması
Yöntem Arayüzü	private void BigramYarat()
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtBigram
Açıklama	Bigram listesinin tutulacağı veri tablosu dtBigram yaratılır

Yöntem No	Y 5.3
Yöntem Adı	BigramEkle
İşlevsel Tanımı	Veri Tablosu dtBigram'a bigram eklenmesi
Yöntem Arayüzü	private void BigramEkle(string hece1, string hece2)
Veri yapıları	DataTable dtBigram
Açıklama	Verilen bir hece çifti dtBigram listesinde mevcutsa adedi bir arttırılır, yoksa listeye eklenir. Eğer harf tabanlı bir modelse bu yönteme giriş değeri olarak hece yerine harf çifti gelir.

Yöntem No	Y 5.4
Yöntem Adı	OlasilikAta
İşlevsel Tanımı	dtBigram tablosundaki tüm bigramların olasılıklarının hesaplanması
Yöntem Arayüzü	private void OlasilikAta ()
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtBigram, DataTable dtHece
Açıklama	Veritabanından Hece tablosu da okunur, hece sıklıkları ve bigram sıklıkları kullanılarak bigram olaslıkları hesaplanır.

Yöntem No	Y 5.5
Yöntem Adı	OrtalamaFrekans
İşlevsel Tanımı	Bir modeldeki bigramların ortalama sıklığını bulur
Yöntem Arayüzü	private int OrtalamaFrekans ()
Veri yapıları	DataTable dtBigram
Açıklama	Toplam bigram sayısı farklı bigramların sayısına bölünerek ortalama sıklık degeri bulunur.

4.2.6 Trigram Nesnesi

Hece tabanlı model oluşturuluyorsa eğitim kümesinde art arda geçen her üç hecenin sayılması ve olasılıkları da hesaplanarak veritabanındaki TabTrigram tablosuna kaydedilmesine ilişkin işlemler bu nesne ile ele alınır. Harf tabanlı model oluşturuluyorsa aynı işlemler harfler kullanılarak yapılır. Nesnenin içerdiği yöntemler aşağıda listelenmiştir.

Yöntem No	Y 6.1
Yöntem Adı	ListeyiAl
İşlevsel Tanımı	Eğitim kümesindeki trigramların sayılması ve olasılıklarının hesaplanması.
Yöntem Arayüzü	public DataTable ListeyiAl(StreamReader sr)
Kullandığı Yöntemler	TrigramYarat, TrigramEkle, OlasilikAta
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Yaz
Veri yapıları	DataTable dtTrigram
Açıklama	Giriş dosyasındaki her bir hece üçlüsünün oluşturduğu trigramlar sayılır, trigram listesine eklenir. Trigram listesi hece1-hece2-hece3 bazında tekildir (aynı hece üçlüsüne bir kez daha rastlanırsa listeye tekrar eklenmez, trigram tablosunda adet alanı bir arttırılır). Tüm liste oluştuktan sonra her bir trigramın olasılığı hesaplanarak veri tabanına kaydedilir.

Yöntem No	Y 6.2
Yöntem Adı	TrigramYarat
İşlevsel Tanımı	Veri Tablosu dtTrigram'ın yaratılması
Yöntem Arayüzü	private void TrigramYarat()
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtTrigram
Açıklama	Trigram listesinin tutulacağı veri tablosu dtTrigram yaratılır

Yöntem No	Y 6.3
Yöntem Adı	TrigramEkle
İşlevsel Tanımı	Veri Tablosu dtTrigram'a trigram eklenmesi
Yöntem Arayüzü	private void TrigramEkle(string hece1, string hece2, string hece3)
Veri yapıları	DataTable dtTrigram
Açıklama	Verilen bir hece üçlüsü dtTrigram listesinde mevcutsa adedi bir arttırılır, yoksa listeye eklenir. Eğer harf tabanlı bir modelse bu yönteme giriş değeri olarak hece yerine harf üçlüsü gelir.

Yöntem No	Y 6.4
Yöntem Adı	OlasilikAta
İşlevsel Tanımı	dtTrigram tablosundaki tüm trigramların olasılıklarının hesaplanması
Yöntem Arayüzü	private void OlasilikAta ()
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtTrigram, DataTable dtBigram
Açıklama	Veritabanından Bigram tablosu da okunur, bigram sıklıkları ve trigram sıklıkları kullanılarak trigram olasılıkları hesaplanır.

Yöntem No	Y 6.5
Yöntem Adı	OrtalamaFrekans
İşlevsel Tanımı	Bir modeldeki trigramların ortalama sıklığını bulur
Yöntem Arayüzü	private int OrtalamaFrekans ()
Veri yapıları	DataTable dtTrigram
Açıklama	Toplam trigram sayısı farklı trigramların sayısına bölünerek ortalama sıklık degeri bulunur.

4.2.7 GirisCikisIslemleri Nesnesi

N-gram modelin oluşturulacağı eğitim kümesinin okunması, yazım denetimi yapılacak dosyanın okunması ve kaydedilmesi gibi giriş-çıkış işlemlerinin ele alındığı nesnedir. İçerdiği yöntemler aşağıda listelenmiştir.

Yöntem No	Y 7.1
Yöntem Adı	DosyaAdiAl
İşlevsel Tanımı	Dosya adı seçtirme.
Yöntem Arayüzü	public string DosyaAdiAl()
Açıklama	Bir dosya açılacağı zaman kullanıcının dosya adını seçmesi sağlanır.

Yöntem No	Y 7.2
Yöntem Adı	DosyaKaydet
İşlevsel Tanımı	Dosya adı seçilerek bir metnin kaydedilmesi.
Yöntem Arayüzü	private void DosyaKaydet(string str)
Açıklama	Hafızadaki metnin diske kaydedilmesi ve dosya adının kullanıcıdan alınması sağlanır.

4.2.8 OlasilikDuzleyici Nesnesi

Bu nesne Good-Turing Yöntemi kullanılarak bir modelin olasılık yoğunluğunun dağıtılmasına yönelik olarak Gale ve Sampson'un geliştirdikleri algoritmanın bu tez çalışması için uyarlanmış halini içerir. Nesnenin içerdiği yöntemler aşağıda listelenmiştir.

Yöntem No	Y 8.1
Yöntem Adı	DuzlestirilmisOlasilikAta
İşlevsel Tanımı	Oluşturulmuş olan bigram ya da trigram modelin olasılıklarının Good-Turing yöntemi ile paylaştırılması.
Yöntem Arayüzü	public double DuzlestirilmisOlasilikAta (int n, int modelTipi)
Kullandığı Yöntemler	FrekansTablosuYarat, FrekansEkle, GirisOku, GirisAnaliz
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku, VeriErisim.Yaz
Veri yapıları	DataTable dtFrekans, DataTable dtNgram
Açıklama	Giriş değerlerinden "n" n-gramın kaçıncı dereceden olduğunu ifade eder. n=2 ise bigram model, n=3 ise trigram model için olasılık yoğunluğu paylaştırma işlemi yapılır. "modelTipi" modelin hece tabanlı mı harf tabanlı mı olduğunu gösterir. Good-Turing yöntemi olasılığı sıfır olan n-gramlar için toplam bir olasılık değeri hesaplar. Bu değerin sıfır olasılıklı n-gramlar arasında paylaştırılabilmesi için dildeki tüm n-gramların sayısı bilinmelidir. Harf tabanlı yöntemde tüm harflerin ikili permütasyonları, hece tabanlı yöntemde tüm hecelerin ikili permütasyonları hesaplanır ve eğitim kümesinde geçmeyen n-gramlara da bir olasılık atanır. Eğitim kümesinde geçen n-gramların olasılıkları da güncellenir.

Yöntem No	Y 8.2
Yöntem Adı	FrekansTablosuYarat
İşlevsel Tanımı	Veri Tablosu dtFrekans'ın yaratılması
Yöntem Arayüzü	private void FrekansTablosuYarat()
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtFrekans
Açıklama	N-gramlarının sıklıklarının ve sonuç olasılıklarının listesinin tutulacağı veri tablosu dtFrekans yaratılır

Yöntem No	Y 8.3
Yöntem Adı	FrekansEkle
İşlevsel Tanımı	Veri Tablosu dtFrekans'a kayıt eklenmesi
Yöntem Arayüzü	private void FrekansEkle()
Veri yapıları	DataTable dtFrekans, DataTable dtNgram
Açıklama	N-gram modelde her bir sıklığın kaç adet n-garm tarafından paylaşıldığına bakılarak n-gram adetlerinin frekansları oluşturulur.

Yöntem No	Y 8.4
Yöntem Adı	EnBuyukAdet
İşlevsel Tanımı	En büyük n-gram adedini bulur.
Yöntem Arayüzü	private int EnBuyukAdet ()
Diğer Nesnelerden Kullanığı Yöntemler	VeriErisim.Oku
Açıklama	N-gram bir modelde en yüksek sıklığa sahip n-gramın sıklığı döndürülür.

Yöntem No	Y 8.5
Yöntem Adı	GirisOku
İşlevsel Tanımı	dtFrekans tablosundan okuma
Yöntem Arayüzü	private void GirisOku ()
Veri yapıları	DataTable dtFrekans
Açıklama	dtFrekans tablosundaki kayıtlar GirisAnalizi yönteminin kullanabileceği şekilde dizilere kopyalanır.

Yöntem No	Y 8.6
Yöntem Adı	GirisAnaliz
İşlevsel Tanımı	Good-Turing yöntemi ile olasılıkların hesaplanması
Yöntem Arayüzü	private void GirisAnaliz ()
Veri yapıları	ArrayList r, ArrayList n, ArrayList Z, ArrayList log_r, ArrayList log_Z, ArrayList rStar, ArrayList p
Açıklama	Good-Turing yöntemi ile n-gram model için tüm olasılıklar ve sıfır olasılıklı n-gramlar için de bir olasılık değeri hesaplanır.

4.2.9 Denetleyici Nesnesi

Bu nesne yazım hatası bulma ve düzeltme işlemleri için temel olarak kullanılan nesnedir. Veritabanındaki n-gram modele ilişkin tablolar okunarak bu işlemler gerçekleştirilir. Nesnenin içerdiği yöntemler aşağıda listelenmiştir.

Yöntem No	Y 9.1
Yöntem Adı	BigramOku
İşlevsel Tanımı	Oluşturulmuş olan bigram model veritabanından okunur.
Yöntem Arayüzü	private void BigramOku(int olasilikTipi)
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtBigram, DataTable dtKuralliHece
Açıklama	Veritabanından bigramlar olasılıklarıyla birlikte okunur. olasilikTipi alanı Good-Turing yöntemi ile bulunan düzleştirilmiş oalsılıkların kullanılıp kullanılmayacağını gösterir.

Yöntem No	Y 9.2
Yöntem Adı	TrigramOku
İşlevsel Tanımı	Oluşturulmuş olan trigram model veritabanından okunur.
Yöntem Arayüzü	private void TrigramOku(int olasilikTipi)
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtTrigram, DataTable dtKuralliHece
Açıklama	Veritabanından trigramlar olasılıklarıyla birlikte okunur. olasilikTipi alanı Good-Turing yöntemi ile bulunan düzleştirilmiş oalsılıkların kullanılıp kullanılmayacağını gösterir.

Yöntem No	Y 9.3
Yöntem Adı	DuzeltmeYarat
İşlevsel Tanımı	Veri Tablosu dtDuzeltme'nin yaratılması
Yöntem Arayüzü	private void DuzeltmeYarat()
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtDuzeltme
Açıklama	Denetleme sonuçlarının tutulacağı veri tablosu dtDuzeltme yaratılır

Yöntem No	Y 9.4
Yöntem Adı	DuzeltmeEkle
İşlevsel Tanımı	Veri Tablosu dtDuzeltme'ye kayıt eklenmesi
Yöntem Arayüzü	private void DuzeltmeEkle(string sozcuk)
Veri yapıları	DataTable dtDuzeltme
Açıklama	Test kümesindeki her farklı sözcük için yapılan denetleme işleminin sonucu dtDuzeltme tablosuna eklenir.

Yöntem No	Y 9.5
Yöntem Adı	Denetleme
İşlevsel Tanımı	Bir giriş metni sözcüklerine ayrılarak yazım denetimi yapılır, sonuçlar veritabanında saklanır.
Yöntem Arayüzü	public DataTable Denetle(string sr, int olasilikTipi, int esikDegeri, int n, int modelTipi)
Kullandığı Yöntemler	DuzetmeYarat, DuzeltmeEkle, BigramOku, TrigramOku, TrigramIleSozcukDuzelt, BigramIleSozcukDuzelt
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Sil, VeriErisim.Yaz
Veri yapıları	DataTable dtDuzeltme, DataTable dtBigram, DataTable dtTrigram
Açıklama	Veritabanında varsa mevcut denetleme sonuçları silinir. Yeni denetleme sonuçlarını tutmak üzere bir dtDuzeltme adlı veri tablosu yaratılır. "n" değerine göre bigram ya da trigram model veritabanından okunur ve giriş metnindeki her sözcük için ilgili modele göre sözcük düzeltme yöntemi tetiklenir. Her sözcük için en çok 10 adet düzeltme alınır ve sonuç tablosuna eklenir. Giriş metnin sonuna gelindiğinde sonuçlar veritabanına yazılır.

Yöntem No	Y 9.6
Yöntem Adı	BigramIleSozcukDuzelt
İşlevsel Tanımı	Bir sözcüğün hatalı olup olmadığının ve hatalı ise yerine gelebilecek sözcüklerin Bigram model kullanılarak bulunması
Yöntem Arayüzü	private bool BigramIleSozcukDuzelt(string sozcuk, ref DataTable dtDuzeltmeler, int olasilikTipi, int esikDegeri, int n, int modelTipi)
Kullandığı Yöntemler	SozcukDogrumu, BigramIleSozcukOlasiligi, SozcukAl, DuzeltilenlereEkle, HeceCikar, SozcukIyilestirBigram
Diğer Nesnelerden Kullandığı Yöntemler	Hece.SozcuktenHeceye, BirHece.BenzerlikDerecesi
Veri yapıları	DataTable dtDuzeltme, DataTable dtBigram, DataTable dtTrigram
Açıklama	Sözcük, içerdiği bigramlardan en az birinin sıklığı model için bulunan ya da kullanıcı tarafından verilebilen eşik sıklığı değerinden küçükse hatalı kabul edilir. Hatalı ise sözcüğün olasılığından daha büyük olasılık sağlayan hece değiştirmeleri yapılarak yerine önerilebilecek aday sözcükler listesi oluşturulur. Bu listede bulunan sözcükler olasılıkları yükseltilebildiği sürece iyileştirilir.

Yöntem No	Y 9.7
Yöntem Adı	TrigramIleSozcukDuzelt
İşlevsel Tanımı	Bir sözcüğün hatalı olup olmadığının ve hatalı ise yerine gelebilecek sözcüklerin Trigram model kullanılarak bulunması
Yöntem Arayüzü	public bool TrigramIleSozcukDuzelt(string sozcuk, ref DataTable dtDuzeltmeler, int olasilikTipi, int esikDegeri, int n, int modelTipi)
Kullandığı Yöntemler	SozcukDogrumu, TrigramIleSozcukOlasiligi, SozcukAl, DuzeltilenlereEkle, SozcukIyilestirTrigram
Diğer Nesnelerden Kullandığı Yöntemler	Hece.SozcuktenHeceye, BirHece.BenzerlikDerecesi
Veri yapıları	DataTable dtDuzeltme, DataTable dtBigram, DataTable dtTrigram
Açıklama	Sözcük, içerdiği trigramlardan en az birinin sıklığı model için bulunan ya da kullanıcı tarafından verilebilen eşik sıklığı değerinden küçükse hatalı kabul edilir. Hatalı ise sözcüğün olasılığından daha büyük olasılık sağlayan hece değiştirmeleri yapılarak yerine önerilebilecek aday sözcükler listesi oluşturulur. Bu listede bulunan sözcükler olasılıkları yükseltilebildiği sürece iyileştirilir.

Yöntem No	Y 9.8
Yöntem Adı	SozcukIyilestirBigram
İşlevsel Tanımı	Bir sözcüğün yerine önerilmek üzere bulunmuş tüm aday sözcükler için oalsılıklarını yükseltecek değişikliklerin yapılması.
Yöntem Arayüzü	private void SozcukIyilestirBigram(DataTable dtDuzeltmeler, int olasilikTipi, int esikDegeri, int n, int modelTipi)
Kullandığı Yöntemler	SozcukDogrumu, BigramIleSozcukOlasiligi, SozcukAl, SozcukDogrumu
Diğer Nesnelerden Kullandığı Yöntemler	Hece.SozcuktenHeceye, BirHarf.BenzerlikDerecesi
Veri yapıları	DataTable dtDuzeltmeler
Açıklama	Aday sözcükler doğru bir sözcük olarak bulunmadıkları ya da daha fazla iyileştirme yapılamaz durumuna kadar hece değiştirilerek olasılıkları yükseltilir.

Yöntem No	Y 9.9
Yöntem Adı	SozcukIyilestirTrigram
İşlevsel Tanımı	Bir sözcüğün yerine önerilmek üzere bulunmuş tüm aday sözcükler için olasılıklarını yükseltecek değişikliklerin yapılması.
Yöntem Arayüzü	private void SozcukIyilestirBigram(DataTable dtDuzeltmeler, int olasilikTipi, int esikDegeri, int n, int modelTipi)
Kullandığı Yöntemler	SozcukDogrumu, BigramIleSozcukOlasiligi, SozcukAl, SozcukDogrumu
Diğer Nesnelerden Kullandığı Yöntemler	Hece.SozcuktenHeceye, BirHarf.BenzerlikDerecesi
Veri yapıları	DataTable dtDuzeltmeler
Açıklama	Aday sözcükler doğru bir sözcük olarak bulunmadıkları ya da daha fazla iyileştirme yapılamaz durumuna kadar hece değiştirilerek olasılıkları yükseltilir.

Yöntem No	Y 9.10
Yöntem Adı	BigramIleSozcukOlasiligi
İşlevsel Tanımı	Bigram olasılıkları kullanılarak sözcüğün olasılığı hesaplanır.
Yöntem Arayüzü	private double BigramIleSozcukOlasiligi(string[] heceListesi, int olasilikTipi, int modelTipi)
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtBigram, DataTable dtKuralliHece
Açıklama	Bir sözcüğün olasılığı kendisini meydana getiren bigramların olasılıklarınını çarpılmasıyla bulunur. Eğer sıfır sıklıklı bir bigram içeriyorsa fakat olasilikTipi Good-Turing yöntemi ile hesaplanan ise bu bigramın da oalsılığı sıfır olmayacağı için sözcük olasılığı sıfır bulunmaz. Ancak bunun için bigramın her iki hecesinin de Türkçe için üretilen hecelerin yer aldığı KurallıHeceler tablosunda bulunması şartı aranır.

Yöntem No	Y 9.11
Yöntem Adı	TrigramIleSozcukOlasiligi
İşlevsel Tanımı	Trigram olasılıkları kullanılarak sözcüğün olasılığı hesaplanır.
Yöntem Arayüzü	private double TrigramIleSozcukOlasiligi(string[] heceListesi, int olasilikTipi, int modelTipi)
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtTrigram, DataTable dtKuralliHece
Açıklama	Bir sözcüğün olasılığı kendisini meydana getiren trigramların olasılıklarınını çarpılmasıyla bulunur. Eğer sıfır sıklıklı bir trigram içeriyorsa fakat olasilikTipi Good-Turing yöntemi ile hesaplanan ise bu trigramın da olasılığı sıfır olmayacağı için sözcük olasılığı sıfır bulunmaz. Ancak bunun için trigramın her hecesinin Türkçe için üretilen hecelerin yer aldığı KurallıHeceler tablosunda bulunması şartı aranır.

Yöntem No	Y 9.12
Yöntem Adı	SozcukDogrumu
İşlevsel Tanımı	Sözcüğün içerdiği n-gramların sıklığına göre hatalı olup olmadığı kontrol edilir.
Yöntem Arayüzü	private bool SozcukDogrumu(string[] heceListesi, ref int index, int esikDegeri, int n)
Veri yapıları	DataTable dtBigram, DataTable dtTrigram,
Açıklama	Sözcüğün içerdiği n-gramlardan en az birinin olasılığı eşik değerinin altındaysa hatalı sözcük olarak bulunur. Sözcük içindeki en düşük sıklığa sahip olan n-gram hatay neden olan n-gram olarak işaretlenir.

Yöntem No	Y 9.13
Yöntem Adı	HeceCikar
İşlevsel Tanımı	Hece listesinden belirtilen sıradaki hece çıkarılır.
Yöntem Arayüzü	private string[] HeceCikar(string[] heceListesi, int index)
Veri yapıları	string[] heceListesi

Yöntem No	Y 9.14
Yöntem Adı	HeceEkle
İşlevsel Tanımı	Hece listesinde belirtilen sıraya verilen hece eklenir.
Yöntem Arayüzü	private string[] HeceEkle(string[] heceListesi, int index, string hece)
Veri yapıları	string[] heceListesi

Yöntem No	Y 9.15
Yöntem Adı	DuzeltilenlereEkle
İşlevsel Tanımı	Verilen sözcük düzeltmeler listesine eklenir.
Yöntem Arayüzü	private bool DuzeltilenlereEkle(DataTable dtDuzeltmeler, string sozcuk, double olasilik, int hecesayisi)
Veri yapıları	DataTable dtDuzeltmeler

Yöntem No	Y 9.16
Yöntem Adı	SozcukAl
İşlevsel Tanımı	Hece listesi birleştirilerek sözcük oluşturulur.
Yöntem Arayüzü	private string SozcukAl(string[] heceListesi)
Veri yapıları	string[] heceListesi

4.2.10 TumHece Nesnesi

Bu nesne Tükçedeki tüm hecelerin üretilmesi amacıyla geliştirilmiştir. Nesnenin içerdiği yöntemler aşağıda listelenmiştir.

Yöntem No	Y 10.1
Yöntem Adı	ListeyiAl
İşlevsel Tanımı	Türkçe için tüm hecelerin üretilmesi.
Yöntem Arayüzü	public void ListeyiAl(bool kuralli)
Kullandığı Yöntemler	HeceYarat, HecelereEkle, HeceYaz
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Yaz
Veri yapıları	DataTable dtHece
Açıklama	"kuralli" parametresine göre ya tüm heceler ya da sadece tanımlı kurallara uyan heceler üretilir, veritabanına kaydedilir.

Yöntem No	Y 10.2
Yöntem Adı	HeceYarat
İşlevsel Tanımı	Veri Tablosu dtHecenin yaratılması
Yöntem Arayüzü	private void HeceYarat(bool kuralli)
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisim.Oku
Veri yapıları	DataTable dtHece

Yöntem No	Y 10.3
Yöntem Adı	HecelereEkle
İşlevsel Tanımı	Veri Tablosu dtHece'ye hece eklenmesi
Yöntem Arayüzü	private void HecelereEkle(string hece)
Veri yapıları	DataTable dtHece

Yöntem No	Y 10.4
Yöntem Adı	HeceYaz
İşlevsel Tanımı	Hece listesinin veritabanına yazılması
Yöntem Arayüzü	private void HeceYaz(bool kuralli)
Diğer Nesnelerden Kullandığı Yöntemler	VeriErisimi.Yaz
Veri yapıları	DataTable dtHece
Açıklama	"kuralli" parametresine göre veritabanındaki TabKuralliHece ya da TabTumHece tablosuna dtHece'deki kayıtlar yazılır.

4.2.11 SozcukUretici Nesnesi

Bu nesne oluşturulmuş olan n-gram modeller kullanılarak rastgele Tükçe sözcükler üretilmesi amacıyla geliştirilmiştir. Nesnenin içerdiği yöntemler aşağıda listelenmiştir.

Yöntem No	Y 11.1
Yöntem Adı	Listele
İşlevsel Tanımı	N-gram modeller kullaılarak Türkçe için rastgele sözcükler üretilmesi.
Yöntem Arayüzü	public DataTable Listele(int adet, int n, int uzunluk)
Veri yapıları	DataTable dtSozcukListesi, DataTable dt, DataTable dtSozcukBasi
Açıklama	"adet" parametresinde belirtildiği kadar sözcük, içerdiği birim sayısı (modelin tipine göre harf ya da hece) "uzunluk" parametresini geçmeyecek şekilde, rastgele ngramlar seçilerek üretilir. "n" parametresi ngramın derecesini belirtir.

5. BÖLÜM
TESTLER VE SONUÇLAR

Giriş bölümünde, Türkçe için geliştirilecek olan hece tabanlı istatistiksel n-gram modellerin yazım hatası bulma ve düzeltme uygulamalarında kullanılabileceği savı öne sürülmüştür. Bu tezde, hem bigram hem de trigram modeller geliştirilip örneklendirilerek bu sav kanıtlanmıştır

Tükçenin bitişken yapısı nedeniyle sahip olduğu çok sayıda farklı sözcüğü yönetmede, sözcük tabanlı n-gram modellerin başarımının kısıtlı olacağı kolayca tahmin edilebilmektedir. Bu nedenle Türkçe için bugüne kadar yapılmış olan çalışmalarda n-gram modeller, sözcüklerin biçimbirimsel çözümlemesi yapıldıktan sonra kökler ve ekler esas alınarak geliştirilmiştir. Bunun için detaylı biçimbirimsel çözümleme gerekmektedir.

Heceler Türkçe bir sözcük içinden ayrıştırılması daha kolay olan birimlerdir. Üstelik hecelerin art arda gelme olasılıklarının kullanılması, ses kurallarının da kendiliğinden ele alınmasını sağlamıştır.

N-gram modellerde birim açısından bir diğer seçenek olan harflerin, geliştirilen sistem üzerinde hece yerine seçimli olarak kullanılması sağlanmış, yine bigram ve trigram modeller harf tabanlı olarak da geliştirilmiştir. Hem bigram hem de trigram modellerde, hata bulma ve düzeltme başarımı açısından hece tabanlı modellerin daha iyi başarı sağladığı gözlenmiştir.

Kullanılan birim ne olursa olsun, eğitim kümelerinin doğal olarak sonlu sayıda sözcük içermesi nedeniyle, veri seyrekliği problemi karşımıza çıkmaktadır. Bu problem tüm n-gram modeller için bir düzleştirme algoritması ile modelin olasılık yoğunluğunun tüm olası n-gramlar arasında paylaştırılmasını zorunlu kılmaktadır.

Bu çalışmada, Tablo 3.1'de verilmiş olan eğitim kümelerinin hepsi için n-gram modeller oluşturulmuştur. Tablo 3.1'deki birinci ve üçüncü örnek kümeler, Orta Doğu Teknik Üniversitesinin "Laboratory for the LcsL Computational Studies METU of Language" ftp sunucusundan alınmıştır. 1. Örnek Arkeoloji ile ilgili bir

metin olup, en küçük eğitim kümesi olduğu için sadeleştirmek açısından bu tez boyunca örnekler bu eğitim kümesinden elde edilmiş olan model esas alınarak verilmiştir. 2. Örnek Prof. Dr. Eşref Adalı'nın bir kitabıdır. 3. Örnek çeşitli haber metinleri içermektedir. 4. Örnek bu tez çalışması için derlediğimiz bir eğitim kümesidir. Bunun için Bilkent Üniversitesi Türk Edebiyatı Bölümünde yapılmış olan edebiyat ile ilgili tez çalışmaları alınmış ve tek bir kümede toplanmıştır. 5. Örnek İstanbul Teknik Üniversitesi Doğal Dil İşleme Grubunun sağladığı bir kümedir; çeşitli kaynaklardan toplanmış Türkçe metinleri içermektedir.

Tüm eğitim kümeleri için, metnin bütünü eğitim amacıyla kullanılmamış, bir kısmı test amaçlı kullanılmak üzere ayrılmıştır. Örneğin bu tez çalışması kapsamında derlenen 4. örnek kümede, 46 adet tez çalışması bir eğitim kümesi olarak hazırlanmış, geriye kalan 1 adet tez çalışması test kümesi olarak kullanılmıştır.

Yapılan bir dizi test sonucu Türkçe için şu sonuçlara varılmıştır; hece tabanlı n-gram modeller ile yazım hatası bulma oranı yaklaşık 89%, hata düzeltme oranı yaklaşık 43% olarak bulunmuştur. Bu oranlar eğitim kümesinin niteliğine ve boyutuna gore değişebildiği için ortalama değerler verilmiştir. Good-Turing yöntemi ile olasılık düzleştirme işlemi yapıldıktan sonar tekrarlanan testlerde hata düzeltme oranı 46%'ya çıkarılmıştır. Ayrıca ilk başta 4% civarında olan yanlış alarmlar (doğru bir sözcüğün hatalı biçimde değiştirilmesi) olasılıklar düzleştirildikten sonra 0%'a yaklaşmıştır.

Bölüm 3.8'de anlatıldığı şekilde Türkçe hece yapısı kurallarına göre hece ekleme ve çıkarma işlemleriyle iyileştirmeler yapıldığında, yazım hatası bulma oranı 92%, yazım hatası düzeltme oranı ise 75% değerlerine kadar çıkarılabilmiştir.

Hece tabanlı bigram bir model, harf tabanlı bigram bir modele göre yazım hatası bulmada ortalama 10%, yazım hatası düzeltmede 15% fazla başarı sağlamaktadır.

Doğal olarak eğitim kümesinin boyutu büyüdükçe test aşamasında daha iyi sonuçlar elde edilmiştir. Ancak eğitim kümesi büyüdükçe, model oluşturma aşamasının süre açısından başarımı düşmektedir. Tablo 5.1'de örnek eğitim kümeleri için sözcük sayısına gore model oluşturma süreleri gösterilmiştir. Kullanılmış olan makinenin özellikleri şöyledir; Intel Pentium III işlemci, 733 Mhz; 256 MB RAM.

Tablo 5.1: Sözcük Adedine Göre Eğitim Kümelerinden Hece Tabanlı N-gram Model Oluşturma Süreleri

Toplam Sözcük Adedi	Toplam Bigram Adedi	Bigram Model Oluşturma Süresi
3,682	13,898	~ 15-20 sn
36,586	139,918	~ 1 dk
774,762	2,829,134	~ 27 dk
5,413,671	19,758,946	~ 3 saat

Bu tez kapsamında heceleme işlemi gerçekleştirildikten sonra n-gram modellerin oluşturulma aşaması için geliştirilmiş olan uygulama dilin özelliklerinden bağımsız olduğu için başka dillerde de birim olarak harfler ya da heceler seçilerek kullanılabilecek şekilde genelleştirilebilir.

6. BÖLÜM
KAYNAKLAR

[1] **SUZEP**, Selçuk Üniversitesi Uzaktan Eğitim Programı - Türk Dili Ders notları

http://farabi.selcuk.edu.tr/suzep/

[2] **Hengirmen, M.**, 2002. Türkçe Temel Dilbilgisi, Engin Yayın Evi, Ankara.

[3] **Solak A., Oflazer K.**, 1993. Design and Implementation of a Spelling Checker for Turkish

[4] **Oflazer K.**, Güzey C., 1994. Spelling Correction in Agglutinative Languages

[5] **Güngör T**., 1995. Computer Processing of Turkish: Morphological and Lexical Investigation

[6] **Oflazer K.**, 1996. Error Tolerant Finite State Recognition with Applications to Morphological Analysis and Spelling Correction

[7] **Hakkani-Tür D. Z.**, 1996. Statistical Language Modeling for Turkish, Ph.D., Bilkent University, 9/96 -- 8/00

[8] **Tür. G.**, 1996. A Statistical Information Extraction System for Turkish, Ph.D., Bilkent University, 9/96 -- 8/00

[9] **Golding, A., Roth D.,** 1999. A Winnow based approach to Context-Sensitive Spelling Correction. Machine Learning. 34 (1-3). 1999. 107--130.

[10] **Manning C. D., Schütze H.**, 1999. Foundations of Statistical Natural Language Processing, The MIT Press, Cambridge, Massachusetts, London, England.

[11] **Jurafsky D., Martin J. H.**, 2000. Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, Prentice Hall, New Jersey.

[12] **Gale W. A, Sampson G.**, 1995. Good-Turing Frequency Estimation Without Tears. Journal of Quantitative Linguistics 2(3): 217-237

[13] **Stanley F. C.**, 1996. Building Probabilistic Models for Natural Language Computer Science Deparment, Phd Thesis, Harvard University

[14] **Damereau F.**, 1964. A technique for computer detection and correction of spelling errors. Communications of the ACM, 7:171-176, 1964

[15] **Atwell, E. Elliott, S.**, 1987. Dealing with ill-formed English text. In The Computational Analysis of English: A Corpus-Based Approach. R. Garside, G. Leach, G. Sampson, Ed. Longman, Inc. New York.

[16] **Church, K. W., Gale, W. A.**, 1991. Probability scoring for spelling correction. In Stat. Comp. 1., 93-103.

[17] **Gale, W. A., Church K. W.**, 1990, Estimation Procedures for language context: Poor estimates are worse than none. In Proceedings of Compstat-90 (Dubrovnik, Yugoslavia), 69-74. New York: Springer-Verlag.

[18] **Mays E., Damerau F. J., Mercer R. L.**, 1991. Context based spelling correction, Information Processing and Management. 27 (5). 1991. 517--522.

[19] **Gale, W., Church, K. and Yarowsky, D.**,1995. Dis-crimination decisions for 100,000 dimensional spaces. Annals of Operations Research, 55, 323-344.

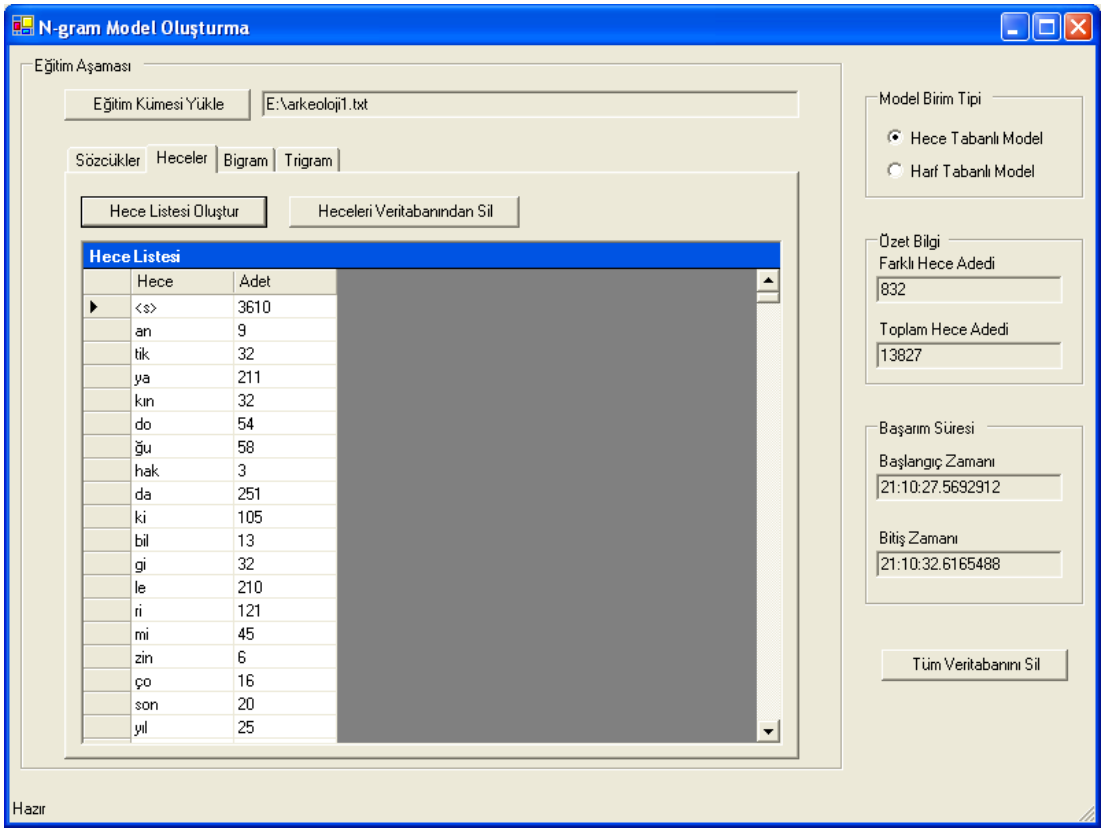
[20] **Yarowsky, D.,** 1994. Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, 88-95. Las Cruces, NM.

[21] **Golding, A.,** 1995 A Bayesian hybrid method for context-sensitive spelling correction. In Proceedings of the Third Workshop on Very Large Corpora, 39-53. Boston, MA.

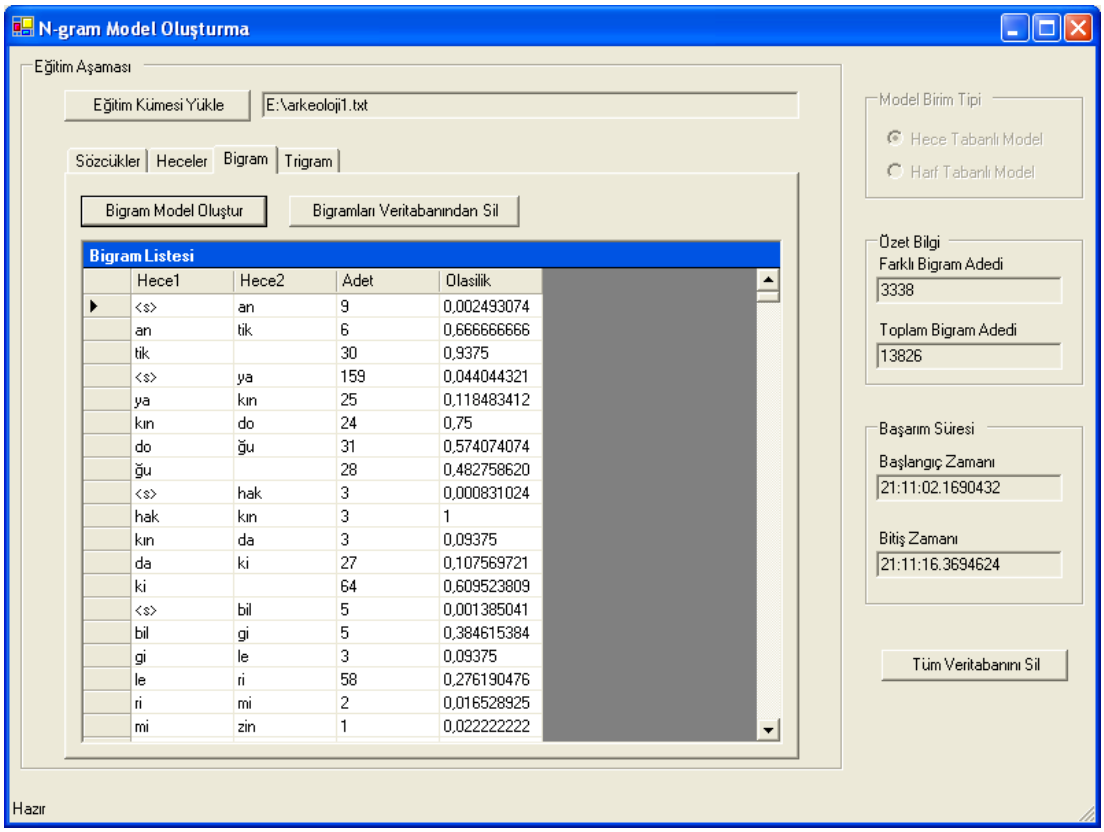
[22] **Golding, A. and Schabes, Y.**, (1996). Combining Trigram-based and Feature-based Methods for Context-Sensitive Spelling Correction. In Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, 71-78. Santa Cruz, CA.

EK-A

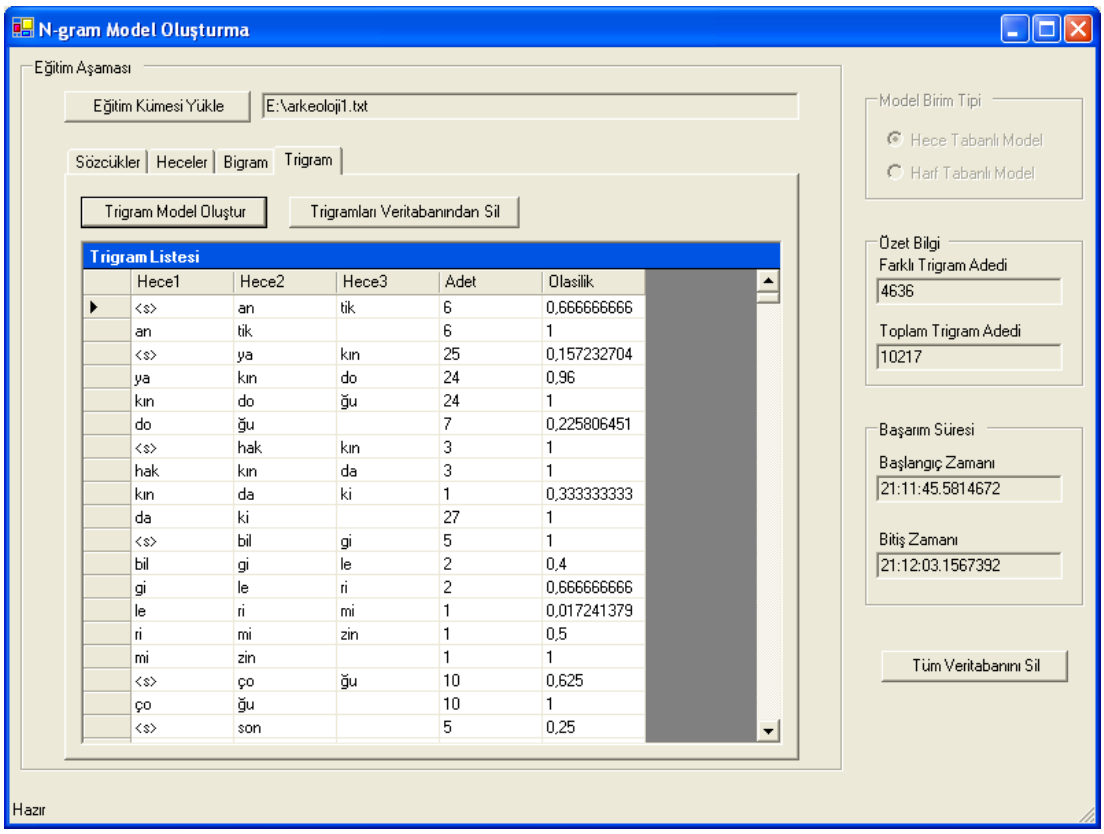
EKRAN GÖRÜNTÜLERİ



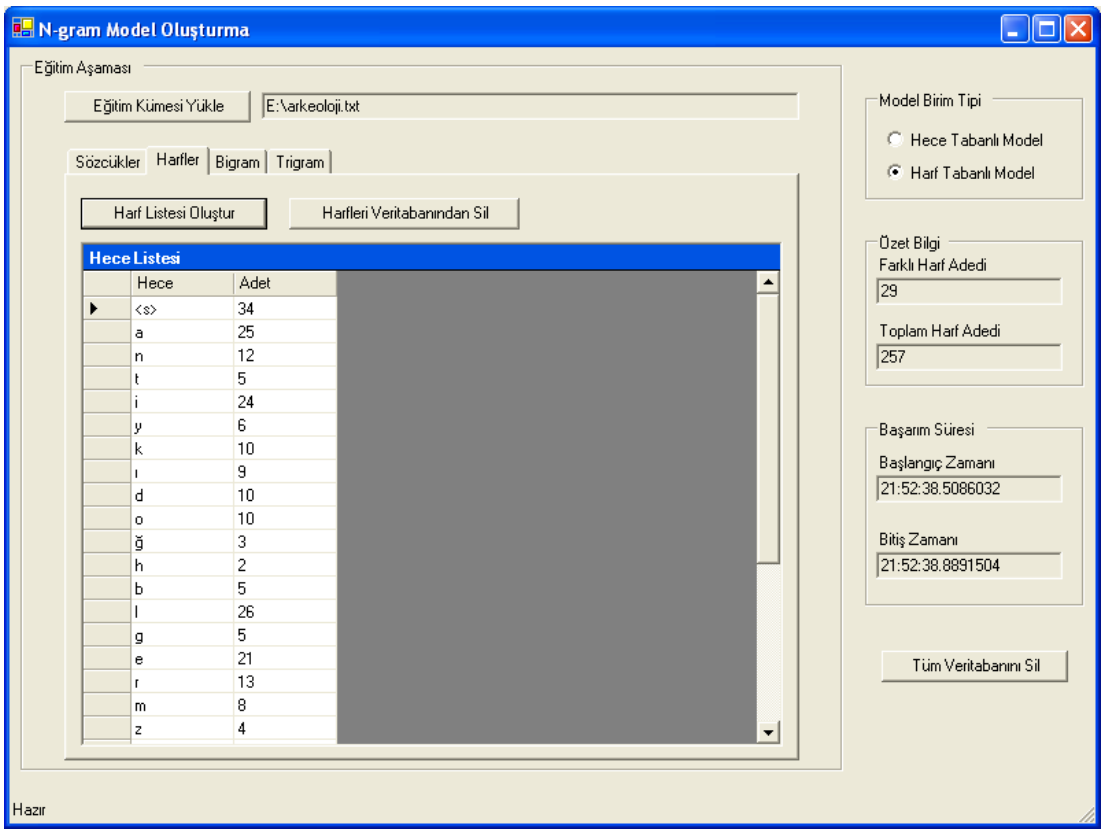
Şekil A.1: N-Gram Model Oluşturma Ekranı; Hece Listesi Oluşturma



Şekil A.2: N-Gram Model Oluşturma Ekranı; Bigram Model Oluşturma



Şekil A.3: N-Gram Model Oluşturma Ekranı; Trigram Model Oluşturma



Şekil A.4: N-Gram Model Oluşturma Ekranı; Harf Listesi Oluşturma