T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

DOĞAL DİL İŞLEMEDE ÇİZGESEL VE OLASILIK TABANLI BİR OTOMATİK ÖĞRENME UYGULAMASI

Hayri Volkan AGUN
Yüksek Lisans Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Yrd. Doç. Dr. Erdem UÇAR
Yardımcı Danışman: Yrd. Doç. Dr. Yılmaz KILIÇASLAN
2008
EDİRNE

T.C.

TRAKYA ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

DOĞAL DİL İŞLEMEDE ÇİZGESEL VE OLASILIK TABANLI BİR OTOMATİK ÖĞRENME UYGULAMASI

Hayri Volkan AGUN

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANA BİLİM DALI

Bu tez *17.06.2008* tarihinde aşağıdaki jüri tarafından kabul edilmiştir.

Yrd. Doç. Dr. Erdem UÇAR

(Danışman)

Yrd. Doç. Dr. Yılmaz KILIÇASLAN

(Yardımcı Danışman)

Yrd. Doç. Dr. Tahir ALTINBALIK

Yüksek Lisans Tezi
Trakya Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Bölümü

ÖZET

Bu tez çalışmasında Türkçe'deki sözdizimsel özelliklerin öğrenilmesi için çizge tabanlı bir otomatik öğrenme modeli sunulmaktadır. Çalışmada bir derlem kullanılarak tasarlanan çizge modeli eğitilmiş ve girilen bir cümle için doğru sözdizimsel etiketler bu model aracılığıyla çıkarılmıştır. Modelin tasarımı sırasında, olasılık tabanlı çizge modeli olan Saklı Markov Modelleri ve çizge teorisinden yararlanılmıştır. Sunulan çalışmada diğer olasılık tabanlı etiketleme algoritmalarından ve istatistiksel doğal dil işleme çalışmalarından farklı olarak Türkçe'nin biçimbilimsel özelliklerinin de kullanılabildiği olasılık tabanlı bir çizge modeli geliştirilmiştir. İlk olarak, ODTÜ-Sabancı Ağaç derleminden model için belirlenen bağlantılara göre bir çizge üretilmiş, daha sonra bu çizge üzerinden sözdizimsel öğelerin bulunabileceği Saklı Markov Modeli oluşturulmuş ve bu modelin üzerinde Viterbi algoritması uygulanarak bir cümle için sözdizimsel öğelerin bulunması sağlanmıştır. Modelin testi için N-Kere Çapraz Doğrulama algoritması kullanılarak başarı ölçülmüştür. Karmaşık derlem çizge modelinden Saklı Markov Modelinin bulunması için çizge teorisinde kullanılan Subdue çizge eşleme algoritmasından yararlanılmıştır. Saklı Markov Modeli ve çizge arama algoritmalarını birlikte kullanılarak daha karmaşık ilişkiye sahip öğeleri (sözdizimsel ve biçimbilimsel ilişkiler gibi) öğrenme için gereken model yapısı oluşturulmuştur. Karmaşık ilişkilerin, sonuç çıkarma ve otomatik öğrenme metotlarının bir arada kullanarak öğrenilmesi, ileride kavram uzayının öğrenilmesi doğrultusunda yapılabilecek çalışmalar için bir alt yapı oluşturmaktadır.

Tezin organizasyonu şu şeklidedir. İlk bölüm Türkçe'nin karakteristiği ve istatistiksel doğal dil işleme konularını, ikinci bölüm çalışmanın konusu olan çizge algoritmalarını, üçüncü bölüm uygulamada kullanılan Saklı Markov Modellerini ve dördüncü bölüm uygulamayı ve sonuçlarını, beşinci bölüm ise yorumları içermektedir.

Anahtar Kelimeler: Saklı Markov Modelleri, Türkçe için Sözdizimsel Etiketleme, Düzleme Teknikleri. Kümeleme, Çizge Madenciliği

Master Thesis
Trakya University Graduate School of
Natural and Applied Sciences
Department of Computer Engineering

ABSTRACT

In this thesis, a model based on combinatorial and probabilistic graphical approaches is proposed for learning of syntactic tor m s for Turkish sentences. A Treebank is used to train a designed probabilistic graphical model and syntactic tags are inferred for a Turkish sentence from this model. Hidden Markov Models and Graph Theory constitute the framework for this model and application. In this proposed model, in a way different from other probabilistic tagging methods and statistical natural language processing applications, a probabilistic graphical model has been developed for syntactic tagging based on morphological features of Turkish language. In the application, firstly a graph model has been constructed from METU-Sabanci Treebank based on certain relations; secondly, a Hidden Markov Model which was extracted from the graph model has been created and trained by the Viterbi algorithm in order to find syntactic features of a given sentence. In order to test the model the N-Fold Cross Validation algorithm is used. When extracting the Hidden Markov Model from the complex Treebank graph model the Subdue graph matching algorithm is used. In conclusion, it is observed that graph models and graph mining algorithms can be a new model in learning of complex relations such as syntactic and morphological relations. Since this study offers an exemplary case where discrete mathematical models and machine learning algorithms are used together, it theoretically supports conceptual space learning studies.

The organization of the thesis is as follows: First chapter presents the characteristics of Turkish and gives an account of statistical natural language processing applications. The second chapter includes the graph algorithms which are used in this study. The third chapter offers information about Hidden Markov Models and language smoothing techniques. The forth chapter reports on the application and its results. The fifth chapter includes a conclusion and an evaluation of the results.

Keywords: Hidden Markov Models, Syntactic Tagging for Turkish, Language Smoothing, Clustering, Graph Mining

TEŞEKKÜR

Bu çalışmanın hazırlanması esnasında bana yol gösteren, bu alanda çalışmam için beni teşvik eden, yardımlarını ve desteğini benden esirgemeyen değerli danışmanlarım Yrd. Doç. Dr. Erdem UÇAR ve Yrd. Doç. Dr. Yılmaz KILIÇ ASLAN ‘a teşekkür ederim.

Çalışmalarım sırasında değerli katkılarıyla bana ilham veren ve ortak çalışmalar yaptığımız arkadaşlarım Yrd. Doç. Erdinç Uzun, Arş. Gör. Özlem Aydın ve Arş. Gör. Edip Serdar Güner'e, ayrıca çalışabilmem için gerekli ortamı sağlayan tüm mesai arkadaşlarıma çok teşekkür ederim.

İÇİNDEKİLER

BÖLÜM 1

GİRİŞ

Olasılık tabanlı doğal dil işleme metotları, kural tabanlı metotların eksiklerini tamamlayıcı özellikler taşımaktadır. Olasılık tabanlı yapılan birçok çalışma metnin uygulamaya göre semantik veya sözdizimsel olarak etiketlenmesi üzerinde durmaktadır. Bunların en güzel örneklerinden son yıllarda üzerinde sıkça durulan süper etiketleme işlemidir. İlk olarak *Bangalore* (Bangalore & Joshi, 1999) kural tabanlı sözdizimsel ayrıştırma işlemine yardımcı olacak şekilde bir metot önermiştir. Bu yaklaşımda, *Bangalore* çok fazla tekrar eden sözdizimsel ağaç yapılarını kullanarak yapılan etiketlemenin, kural tabanlı doğal dil işleme metotlarına hız ve doğruluk açısından büyük katkı sağladığını Penn Ağaç Derlemi (*Penn Treebank)* üzerinde yaptığı çalışmalarla kanıtlamıştır.

Türkçe'nin cümle yapısının serbest kelime sıralamasına sahip olması, bitişken bir dil olması, Türkçe için sağlam sözdizimsel gramer ve ayrıştırıcının ortaya çıkmaması bu konuda yapılan uygulamanın temel sebebini oluşturmaktadır. Amaç, Türkçe'nin sözdizimsel olarak sağlam bir şekilde ayrıştırılabilmesi için olasılık tabanlı bir etiketleme modeli geliştirmektir. Ancak bu çalışmanın ileride süper etiketlemeye, sözdizimsel veya semantik modellere yardımcı şekilde tasarlanmış olması, çalışmanın sadece bir etiketleme çalışması olmadığını göstermektedir. Tez kapsamında, ileride yapılması muhtemel çalışmalar için, sözdizimsel etiketleme modeli üzerinden sözdizimsel ağaç yapılarının oluşturulması ve ağaç yapısı yardımıyla cümlenin kural tabanlı ayrıştırılması öngörülmektedir.

Tezde, çizge arama teknikleri, çizge teorisi, Saklı Markov Modelleri, yumuşatma algoritmaları ve süper etiketleme konuları araştırılmıştır. Ayrıca, uygulamada kullanılan algoritmalar özetlenmiştir. Buna göre Subdue çizge arama algoritması, Kneser-Ney yumuşatma algoritması ve Viterbi algoritması üzerinde durulmuştur. Tezde ayrıca Yüksek Boyutlu Gömme çizge gösterim tekniğine de değinilmiştir. Uygulamada, bir derlem kullanılarak sözdizimsel etiketleme yapılmıştır. Diğer çalışmalardan farklı olarak, çizge teorisi ve çizge teknikleri ile bir derlemden çizge modeli oluşturulmuştur. Bu model kullanılarak bir cümledeki her bir kelime için en uygun sözdizimsel etiketler bulunmuştur.

Uygulamada, tasarlanan çizge modelinin eğitimi için ODTÜ-Sabancı Ağaç Derlemi kullanılmıştır. Bu ağaç derleminde bulunan ilişkiler ilk önce bir çizge modeline aktarılmış, daha

sonra bu modelin olasılıkları hesaplanmıştır. Ağaç derleminin tutulması için tasarlanan çizge modeli, derlemde bulunan kelimeleri, sözdizimsel ve biçimbilimsel özellikleri içermektedir. Tüm bu bilgiler çizgede düğümler üzerinde tutulmaktadır. Bu derlem bilgilerinin aralarında kurduğu sözdizimsel, biçimbilimsel ve kelime ilişkileri ise düğümler arasındaki bağlantılarda tutulmaktadır. Örneğin "Boğaziçi" kelimesi derlemde "İsim" sözdizimsel özelliği ile sıkça geçmektedir. Bu durumda tasarlanan çizgede "Boğaziçi" kelimesi ile "isim" sözdizimsel özelliği arasında bir bağlantı kurulmaktadır. Benzer şekilde "Boğaziçi" kelimesi derlemde "Köprüsü" kelimesi ile birlikte geçtiğinden "Boğaziçi" ile "Köprüsü" arasında sonraki bağlantısı yer almaktadır. Bu yaklaşıma göre derlemden elde edilen aynı tür bilgiler için çizgede (kelime - kelime, sözdizimsel - sözdizimsel, biçimbilimsel - biçimbilimsel) sonraki bağlantısı yapılmıştır. Aynı yaklaşımla derlemdeki farklı öğeler arasındaki ilişkiler için, çizgede özellik bağlantısı oluşturulmuştur. Tüm bu bağlantıların sayıları ve olasılıkları hesaplanmıştır.

Modelin çok fazla bağlantı içermesi bizim için karmaşık ilişkileri bulmakta bir avantaj sağlamaktadır. Ancak daha özel konular için bu karmaşıklık gereksizdir. Bu çalışmada yapılan uygulama sözdizimsel etiketleme olduğu için karmaşık çizge modelinde yapılan bağlantılar sınırlandırılmalıdır. Bunu için çizgede "Subdue" eşleme algoritması kullanılarak arama yapılmış ve sadece kelime - sözdizimsel, kelime - kelime, sözdizimsel - sözdizimsel bağlantılar ve bu bağlantıları yapan düğümler bulunmuştur. Sonuç olarak bir karmaşık modelden istenen düğümler ve bağlantıları elde edilerek örnekleme yapılmıştır.

Bulunan tüm örnekler kullanılarak, Viterbi algoritması ile girilen bir cümlenin kelimelerinin sözdizimsel etiketleri bulunmuştur. Bir kelime birden fazla sözdizimsel öğeyle ilişkilendirildiği için hangi sözdizimsel öğenin daha uygun olduğunun bulunması işlemi sırasında olasılığın maksimum olmasına dikkat edilmiştir. Kısaca, bir kelimenin uygun sözdizimsel etiketinin bulunması kelime - kelime, kelime - sözdizimsel ve sözdizimsel - sözdizimsel bağlantıların olasılıklarına bakılarak yapılmaktadır. 1.1'de girilen bir cümle ve bu cümledeki kelimelerin biçimbilimsel analizi verilmektedir. 1.2'de bu cümlenin etiketleme sonucu gösterilmektedir.

(1.1)

Cümle:

"'Ali Ayşe'ye kitabı verdi."

Biçimbilimsel Analiz:

"Ali+isim Ayşe+isim+(e hali) kitap-isim+(i hali) ver+(di'li geçmiş zaman kipi)."

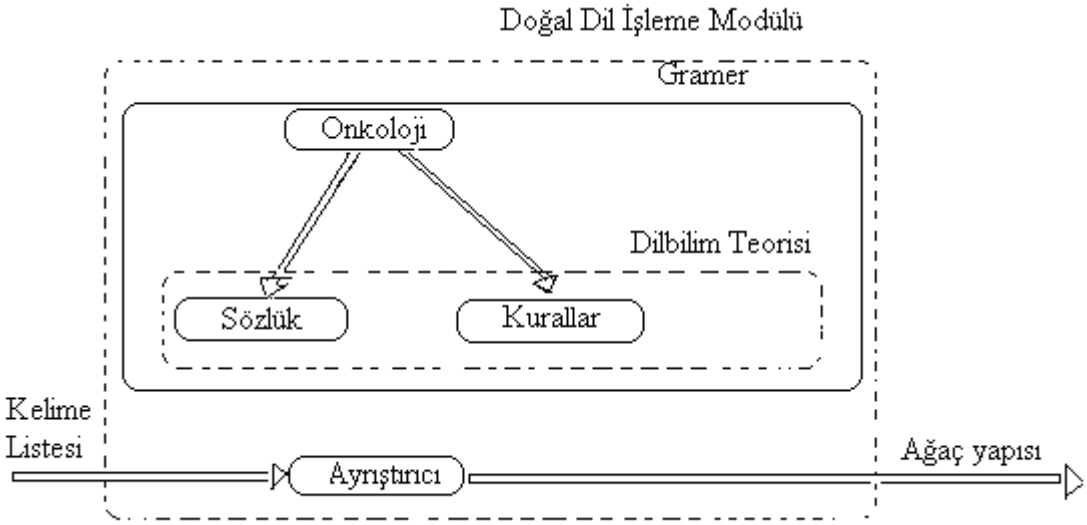
(1.2)

Etiketleme Sonucu:

Al i	Ayşe	kitap	verdi
Özne	Dolaylı Tümleç	Nesne	Fili

1.1 Doğal Dil İşleme Çalışmaları

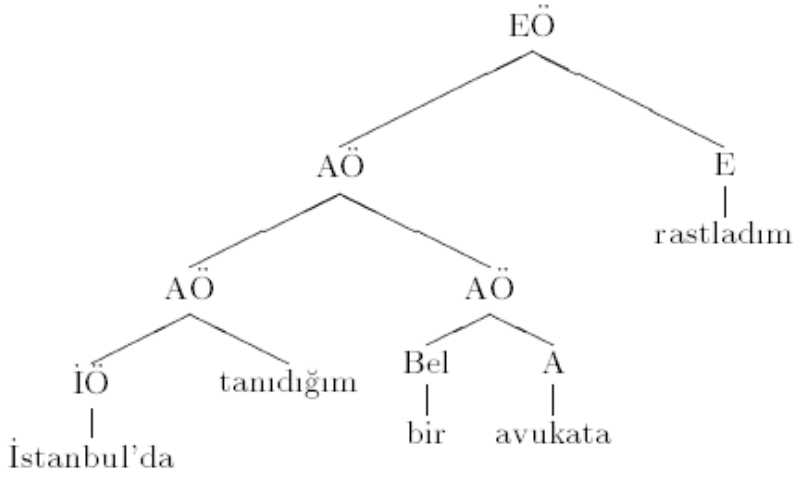
Doğal dil işleme için yapılan çalışmalar, kural tabanlı ve olasılık tabanlı olmak üzere iki gruba ayrılmaktadır. Kural tabanlı bir doğal dil işlemede kullanılan algoritmalar bir metnin özetini çıkarmada ve bir metni bir dilden başka bir dile çevirmede ortak yapıların oluşturulmasında kullanılabilir. Kural tabanlı algoritmaların gerçek metinler üzerinde yüzde yüz başarıyla çalışması mümkün olmamaktadır. Çünkü günlük metinlerde sözcükler eksik yazılmış olabilir ya da cümlenin grameri tam doğru olmayabilir. Ayrıca metni incelemede kullanacağımız gramer kuralları dilin bütününü kapsayacak düzeyde gelişmemiş olabilir. Bu faktörler göz önüne alındığında, olasılık tabanlı doğal dil işleme algoritmaları kural tabanlı algoritmaların eksik kaldığı noktaları tamamlamaktadır. Geliştirilen olasılık tabanlı modellerde bir kelimenin eksik olması veya cümlenin yazım kurallarına aykırı olması analiz sonucunu ölümcül bir şekilde etkilememektedir. Kural tabanlı modeller her ne kadar yüzde yüz başarılı olmasa bile doğal dilin analizi için gereklidir. Şekil 1.1'de kural tabanlı bir doğal dil işleme modülü gösterilmektedir.



Şekil 1.1 Kural tabanlı doğal dil işleme modülü

Şekil 1.1'de gösterilen kural tabanlı modül gramer, sözlük ve ontoloji modüllerini kullanan bir ayrıştırıcı ile dilin analizini yapmaktadır. Bu analiz sonucu çoğu zaman Şekil 1.1'de belirtildiği gibi ağaç yapısı şeklinde ifade edilmektedir. Sonucun ağaç yapısı ile ifade edilmesinin sebebi dildeki öğelerin arasındaki ilişkileri göstermektir. Örneğin 1.3'de belirtilen bir kelime dizisinin (cümle) böyle bir yapı ile işlenmesi sonucunda oluşan ağaç yapısı Şekil 1.2'de gösterilmiştir.

İstanbul'da tanıdığım bir avukata rastladım. (1.3)



Şekil 1.3 Cümlenin analizi sonucunda oluşan ağaç yapısı ifadesi

Kural tabanlı doğal dil işlemede kullanılan yöntemler benzer şekilde olasılık modellerinde de kullanılmaktadır. Ancak, gramer kuralları ve sözlükteki dilbilimsel özellikler olasılıkla ifade edilmekte ve işlemler bu olasılıklar üzerinden yapılmaktadır. Olasılık modelleri sonuç olarak benzer ağaç yapıları üretebilirler. Olasılık tabanlı mimarilerde ve metin analizlerinde çoğu zaman ağaç yapıları üretilmesi tercih edilmez. Bunun yerine öğelerin ağaç yapılarının belli parçaları elde edilir. Bu işlem, etiketleme veya çoklu etiketleme *(multi tagging)* olarak adlandırılır. Çoklu etiketleme işlemi bir metin için metnin barındırdığı kelimelerin ve/veya cümlelerin dilbilimsel özelliklerinin bulunması işlemidir. Etiketleme işlemi için olasılık tabanlı mimarinin başta o dildeki işaretlenmiş (etiketlendirilmiş) metinler kullanılarak eğitilmesi zorunludur. Bu eğitim, yapılan birçok çalışmada olduğu gibi ağaç derlem yapıları, hiyerarşik dilbilimsel ve/veya ansiklopedik sözlükler veya elle işaretlenmiş metin grupları kullanılarak yapılabilir, işlenecek dile ve uygulamanın türüne ait böyle veri kaynaklarının olmaması olasılık tabanlı modeller için bir engel teşkil etmektedir.

Son zamanlarda Internet'in devreye girmesiyle dokümanların HTML içerisinde HTML özellikleriyle etiketlenmesi ve arama motorlarının gelişmesi verinin gruplanarak toplanmasını kolaylık sağlamaktadır. Bunun dışında daha önceden etiketlenen derlemler yani ağaç derlem yapıları yeni metin gruplarının etiketlenmesini ve etiketleme kontrolünü yapan kişinin işini kolaylaştırmaktadır.

Doğal dil işleme çalışmaları için tüm faktörler göz önüne alındığında, hem kural tabanlı hem de olasılık tabanlı mimariler için verinin etiketlenmesi çok önemli bir iş olmaktadır. Bunun ilk kanıtı olarak birçok olasılık tabanlı doğal dil işleme çalışmasında, otomatik öğrenme metotları kullanılarak sözdizimsel öğelerin etiketlenmesi gösterilebilir. Süper etiketleme ile cümlenin sözdizimsel olarak etiketlenmesinin yanında ağaç yapısının da bir kısmı elde edilir. Bu kural tabanlı doğal dil işleme modülleri için örneğin ayrıştırma modülü için büyük kolaylık sağlar.

Kural ve olasılık tabanlı yapıların kullanabileceği veya bunların birlikte çalışabileceği mimariler genelde ardışık olarak çalışırlar. İlk önce kural tabanlı model işleyerek girdideki bulunabilecek ipuçlarını en iyi şekilde ortaya çıkarır. (Örneğin kelimenin biçimbilimsel analizini yapabilir.) Daha sonra olasılık tabanlı model, girdi metni için bu ipuçlarını kullanarak etiketleme işlemini yapar. (Örneğin, yanlış yazılmış

bir kelimede bulunan biçimbilimsel öğeleri ve cümlenin bütününü değerlendirerek bu kelimenin kategorisini veya sözdizimsel formunu tahmin edebilir.) Son olarak kural tabanlı model olasılık tabanlı modelin çıktılarını kullanarak veriyi işler ve sonucu en iyi şekilde kullanıcıya döndürür.

Olasılık veya istatistik tabanlı birçok doğal dil işleme çalışması yapılabilir. Bunların hepsi aslında daha önce söz ettiğimiz gibi birer çoklu etiketleme işlemidir. Örnek vermek gerekirse, anlam belirginleştirmesi *(word sense disambiguation)* (Yarowsky, 1995), sözcük türünün saptanması *(part of speech tagging)* (Cutting, Kupiec, Pedersen, & Sibun., 1992), semantik etiketleme *(semantic tagging)* (Segond, Schiller, Grefenstette, & Chanod, 1997), artgönderim çözümleme *(anaphora resolution)* (Ge, Hale, & Charniak, 1998), süper etiketleme *(süper tagging)* (Clark & Curran, 2004), kök ve ek analizi (*morphological analysis*) ve ayrıştırma *(parsing)* (Eryigit & Oflazer, 2006) gibi olasılık tabanlı yaklaşımlar aslında işledikleri veriye çeşitli türlerde etiketler atayan uygulamalardır.

1.2 Türkçe'nin Karakteristiği

Türkçe benzerlik itibariyle Ural-Altay[1] dil ailesine aittir. Bu dil ailesinde Türkçe'den başka Türkçe'ye en yakın diller Moğolca ve Tunguzcadır. Bunların dışında Korece ve Japonca dilleri de Türkçe'ye yakın dil gruplarıdır.

Türkçe bitişken (agglutinative) bir dildir, Türkçe'de ekler, belli kurallar doğrultusunda cümlenin yapısında kelimenin aldığı role (özne, yüklem) göre, köke eklenerek kelimeyi oluştururlar. Bu biçimbirimler köke eklenirken, kelime cümle içerisindeki rolüne tam anlamıyla kavuşur.

Türkçe'de kelime sıralamasının Özne-Nesne-Yüklem biçiminde olduğu varsayılır. Ancak kelimelerin yerlerinin değiştirilmesi mümkün olduğu için görece

[1] Ural-Altay dil ailesi ile ilgili bilgi http://tr.wikipedia.org/wiki/Ural-Altay_dilleri İnternet adresinde bulunmaktadır.

serbest bir kelime sıralamasına sahiptir. Bu açıdan Türkçe, Fince ve Japonca gibidir (Boz, 1994).

1.3 Türkçe'de Doğal Dil İşleme Çalışmaları

Türkçe için yapılan doğal dil işleme çalışmalarının çoğu, Türkçe'nin sözdizimsel ve biçimbilimsel analizine odaklanmıştır. Bu çalışmalar biçimbilimsel düzeyde kural tabanlı olup sözdizimsel düzeyde olasılık tabanlı çalışmalardır. Örneğin Türkçe için yapılan sözdizimsel seviyedeki ayrıştırma işlemlerinde olasılık tabanlı bağımlı gramer *(dependency grammer)* kullanılmış ancak yapılan biçimbilimsel çalışmalarda otomatlar kullanılmıştır. (Oflazer K. , 1994) (Oflazer & Kuruoz, 1994) (Nivre, Hall., Nilsson, Eryiğit, & Marinov, 2006).

Türkçe'nin serbest kelime sıralamalı olması ve sözcük yapısının eklenen son ekler ile değişkenlik taşıması bilgisayar tarafından işlenmesini zorlaştıran önemli bir faktördür. Türkçe'nin işlenmesi sırasında en çok karşılaşılan problemlerden bir tanesi cümle veya öbek içerisindeki görevlerinin saptanması problemidir. Bununla birlikte, başka kelime öbekleri içerisindeki kelime öbeklerinin işlevlerinin ve yapılarının saptanması, ilk problemin sonucu olarak ikincil bir sorun yaratmaktadır.

1.4 Etiketleme

Etiketleme işlemi *(tagging)* doğal dil işlemede en önemli konularından biridir. Kavram olarak bir öğenin ya da öğe kümesinin özelliğinin bulunması doğal dil işlemede, işaretleme veya etiketleme olarak bilinir. Bu kavram, otomatik öğrenmede sınıflandırma veya kümeleme olarak da bilinmektedir.

Doğal dil işlemede dilbilimsel öğelere (örneğin sözcüklere), konusuna ve cümledeki görevine göre aşağıdaki etiketler atanabilir.

- Kelimenin kategorisi: isim, fiil, sıfat, zarf, ...
- Kelimenin anlam sınıfı: canlı, cansız, ...
- Kelimenin sözdizimsel özelliği: nesne, özne, yüklem, ...

Cümledeki kelimeler için yapılan etiketleme işlemi cümlenin ayrıştırılmasına, ağaç formasyonuna dönüştürmeye ve bağlam içerisindeki görevini bulmaya yardımcı olacaktır.

Etiketleme için kullanılan olasılık tabanlı yapılar, olasılık tabanlı çizge ya da ağaç modelleri ve istatistiksel hipotez veya bağlantı tabanlı olmayan modeller olmak üzere ikiye ayrılabilir. Olasılık tabanlı çizge modellerinde, en çok göze çarpan *Bayesian* ağlarıyken istatistiksel modellerde hipotez testleridir.

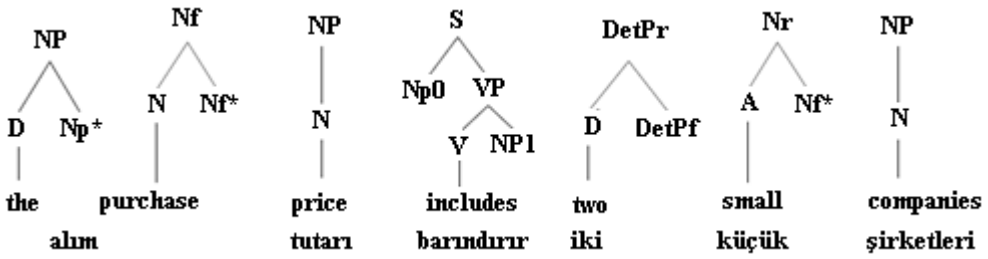
1.5 Süper Etiketleme İşlemi

Süper etiketleme, etiketleme işleminin ağaç formasyonu şeklinde sözdizimsel olarak yapılmasıdır. Bir anlamda ayrıştırıcının görevini yerine getirmediği söylenebilir. Ancak sözdizimsel ayrıştırıcıdan farklı olarak, ağaç yapısını bir bütün olarak vermeyip en olası ağaç formasyonunu bulmaya çalışır. Kural tabanlı bir ayrıştırıcıda ise ayrıştırma işlemi tam olarak yapılır. Ancak cümle tam bir ağaç formasyonuna sahip olacak şekilde girilmemiş ise kural tabanlı sözdizimsel ayrıştırıcı, ayrıştırma işlemini yapamaz. Süper etiketleme böyle bir durumda en yüksek olasılıklı birleşime sahip süper etiketle bulduğundan, ayrıştırıcının bu etiketleri kullanarak ayrıştırma işlemini yapması doğru ağaç formasyonunu bulmasına yardımcı olur. Kural tabanlı ayrıştırıcı, bir sonuç döndüremese bile, süper etiketleme işlemi yaklaşık bir ağaç formasyonu döndürdüğünden kullanıcıya cümle hakkında bir bilgi sağlamış olur.

Süper etiketleme işlemi, ayrıştırma işlemine çok fazla gerek duymayan metin madenciliği uygulamalarında kullanılır. Çoğu zaman Korece ve Çince gibi karmaşık yapıdaki kural tabanlı ayrıştırma işleminin zor olduğu dillerde ayrıştırma işlemi için kullanılmaktadır (Bangalore & Joshi, 1999).

Şekil 1.2'de süper etiketleme işlemi için bir örnek verilmiştir. Bu örnekte girilen bir cümle için bulunan ağaç forrnasyonundaki etiketler yer almaktadır, Bu etiketler dilin yapısı için daha önceden oluşturulmuş olası sözdizimsel ağaç formasyonlarıdır. Etiketlerin ağaç formasyonlarının tam olmadığı, eksik kelime öğeleri barındırdığı Şekil 1.2'de, "*" şeklindeki sözdizimsel özelliklerle gösterilmektedir. "*" ile belirtilen

öğelerin varlığı bu formasyonun böyle bir öğeyi beklediğini ancak öğenin süper etiketleme işleminde eşleştirilemediğini göstermektedir.



Şekil 1*2* Cümlenin süper etiketlenmesinin sonucu

Böyle bir etiketleme sonucunda ayrıştırma işlemi çok basit bir hale gelecektir (Bangalore & Joshi, 1994).

1.6 Çizge Modelleri

Çizge modelleri doğal dil işleme çalışmalarında en çok olasılık tabanlı mimarilerde karşımıza çıkmaktadır. Doğal dil işlemede, etiketleme işlemlerinde kullanılan Saklı Markov Modelleri olasılık tabanlı çizge modellerindendir. Ancak çizge teorisi ve kesikli matematiksel yapılar doğal dil işlemede son zamanlarda kullanılmaya başlanmıştır. Doğal dil işlemede, ilk çizge uygulamaları anlam belirginleştirmesinde ve anlamasal ontolojilerin öğrenilmesinde ortaya çıkmaktadır. Çizge algoritmalarının öğrenmede kullanıldığı mimarilere genel olarak metin çizgeleri *(text graphs)* adı verilmektedir (Nuutila & Törmä, 2003). Bu tür çizge mimarileri, bizim uygulamamızda derlemin saklanmasında ve derlemden örnekleme yapılmasında kullanılmıştır.

BÖLÜM 2

ÇİZGE ALGORİTMALARI

Bu tez çalışmasında, çizge teorisine ilişkin çizge eşleme algoritmaları ve çizge gösterim teknikleri kullanılmıştır. Çizge eşleme için k-tane tam olmayan eşleme yöntemleri araştırılmış ve bu yöntemlerden "Subdue" algoritması uygulamada kullanılmıştır (Lawrence & Diane, 1993). Kullanılan çizge gösterim tekniklerinden "Yüksek Boyutlu Gömme" (*High Dimesional Embedding*) tekniği ve çizge gramer (*graph grammer*) yapısı bu bölümde anlatılmaktadır.

2.1 Çizge Algoritmaları ve Kesikli Matematiksel Yapılar

Çizge algoritmaları ve kesikli matematiksel yapılar doğal dil işlemede kullanılmasa da, bilginin tutulması ve işlenmesi açısından büyük bir alt yapı sağlamaktadırlar. Çizge algoritmaları ve kesikli yapıların, otomatik öğrenmeye de katkısı büyüktür. Çizge algoritmaları ile otomatik öğrenme yapılması, birleşimsel En İyileme (*combinatorial optimization*) olarak adlandırılmaktadır. Bunun dışında son yıllarda ortaya çıkan çizge madenciliği (*graph mining*) yine çizge teorisi ve otomatik öğrenme (*machine learning*) kavramlarım birleştirmektedir (Palmer, Gibbons, & Faloutsos, 2002).

Çizge yaklaşımları ile anlambilimin ifade edilmeye çalışılması çizge gramerini ve bu gramerin kullanıldığı çizge ifade tekniklerini ortaya çıkarmıştır (Palmer, Gibbons, & Faloutsos, 2002). Kavramların çizge modelleri ile öğrenilmesi. Kavram Uzayı Öğrenme teorisinin ortaya atmıştır (Ickjai Lee Portier, 2007). Bu teoriye göre öğrenme işlemi kavramı ifade eden ilişkilerin öğrenilmesidir. Kavram öğrenimi literatürde uygulama olarak, bağlantı tabanlı yaklaşımlarla ve Özdüzenleyici Harita *(Self Organizing Maps)* uygulamaları ile ilişkilendirilmektedir (Berg & Schuemie, 1999).

2.2 Çizge Eşleme Algoritmaları

Çizgi eşleme algoritmaları incelendiğinde probleme bağlı olarak çok fazla yöntemden bahsedilebilir. Bu yöntemlerin her biri NP-Zor[2] *(NP-Hard)* karmaşıklık düzeyine sahip olduğundan bu alandaki iyi yöntemler karmaşıklık düzeylerine göre belirlenmektedir.

2.2.1 İzomorfik eşleme

Temelde bir çizge eşlenirken en basit yöntem çizgen in izomorfizm eşlemesini yapmaktır. İzomorfik eşleme iki çizgenin aynı sayıda düğüme sahip olması ve her bir düğümün bağlantılarının diğer çizgedeki bir düğümün bağlantılarına karşılık gelmesi olarak özetlenebilir. Şekil 2.1'de iki izomorfik çizge gösterilmektedir.

Şekil 2.1 İki izomorfik çizge

Bu iki çizge görünüşte farklı olsa da aslında yapısal olarak aynıdır. Yani Çizge 1' deki tüm bağlantı ve düğümlerin üzerinde döngüye girmeden gezeceğimiz farklı yolların en az bir tanesi Çizge 2'nin aynı şekilde gezilmesi ile oluşan yollardan en az birine eştir.

Çizge izomorfizmini bulan temel algoritmalardan bir tanesi Ullmann algoritmasıdır (Ullmann, 1976). Ullmann algoritmasına göre,

$G_\alpha ve\ G_\beta$ İki çizge olsun. Bu çizgelerin komşuluk matrisleri ise sırasıyla $A = [a_{ij}]$ ve $B = [b_{ij}]$ olsun. Öyle bir M' matrisi tanımlayalım ki M' matrisi A

[2] Bir problemin polinom-zamanda çözümünün gerçekleşmesinin belirsiz olduğu durumlar için kullanılır. NP-Zor problemler çoğu zaman polinom düzeyindeki bir zamanda çözülemez. Örneğin n adet veri için problem çözümünde n^{100} işlem yapılıyorsa bu NP (polinom-zamanda) karmaşıklık düzeyine sahiptir. Ancak bu problem n^n işlem yapıyorsa o zaman NP-Zor (üstsel-zamanda) karmaşıklık düzeyine sahiptir.

matrisinin satırlarından ve matrisinin sütunlarından oluşsun ve elemanları 1 veya 0 olsun. Bu matriste her satır ve sütün sadece bir tane 1 barındırsın.

(2.1)

$$C = [c_{ij}] = M'(M'B)^T$$

olacak şekilde

(2.2)

$$\forall i_{1\leq i\leq p_\alpha} \bigwedge \forall i_{1\leq i\leq p_\alpha} (a_{ij} = 1) \Longrightarrow (c_{ij} = 1)$$

Yukarıdaki durum sağlanıyorsa $G_\alpha ve\ G_\beta$ çizgeleri izomorfiktir. Bu formüllere göre M' matrisini türetmek için ilk başta bir M^0 matrisi aşağıdaki koşulu sağlayacak şekilde hesaplanır.

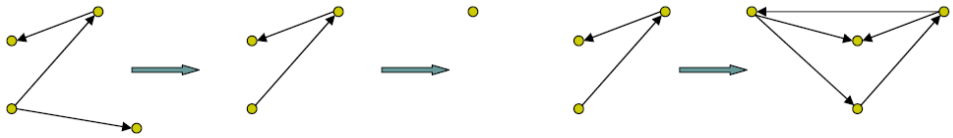
$m_{ij}^0 = 1$; Eğer G_β 'nın j inci düğümünün derecesi $\geq G_\alpha$ 'nın i inci düğümünün derecesi

$m_{ij}^0 = 0$; Diğer durumda

Her bir M^1 matrisi M^0 matrisinin her bir satırında ve sütününda sadece bir tane 1 bırakacak şekilde tüm birleşimler sıralanarak bulunur. Sonuç olarak M' (2.1) de yerine yazılır ve C matrisi bulunur. Eğer (2.2) deki koşul sağlanırsa iki çizge izomorfiktir.

2.2.2 Levenshtein uzaklığı

Levenshtein uzaklığı karakter katarı eşlemede ve çizge algoritmalarında sıklıkla kullanılmaktadır. Bu yöntemde bir çizgenin karşılaştırıldığı diğer bir çizgeye benzetilmek için yapılacak minimum değişiklik temel alınarak eşleme yapılır. Şekil 2.2'de bir çizgenin yeni düğümler ve bağlantılar eklenerek değiştirilmesi örnek olarak gösterilmektedir.



Şekil 2.2 Çizgeye düğüm ve bağlantı ekleme sonucu yapılan benzetim

Şekil 2.2'de bir çizgeye sırasıyla yeni düğüm ve bağlantılar eklenmektedir. Eklenecek veya silinecek bağlantı ve düğümlerin bulunması ile bir çizgeyi başka bir çizgeye dönüştürecek toplam dönüşüm uzaklığı hesaplanır. Bu uzaklık aynı zamanda dönüştürülen çizge ile hedef çizge arasındaki benzerliği vermektedir.

Levenshtein uzaklığı doğal dil işleme ve otomatik öğrenmede kullanılmaktadır (Neuhaus & Bunke, 2006). Doğal dil işleme çalışmalarında Levenshtein uzaklığı en çok anlam belirginleştirmesinde kullanılmaktadır.

2.2.3 Subdue çizge eşleme algoritması

Subdue çizge eşleme algoritması bir çizgede bulunan en sık tekrarlanan kavramların bulunmasında ve çizge eşlemede kullanılmaktadır. Subdue algoritması kullanılarak geliştirilen uygulamalar çizge madenciliği ve kavram öğrenimi üzerinde yoğunlaşmaktadır.

Subdue algoritması bir k-tane tam olmayan çizge eşleme *(inexact graph matching)* metodudur. Bu tür metotlarda yapılan eşleme sonuçları maksimum benzerlikten minimum benzerliğe doğru sıralanır. Benzerlik kıstası Subdue algoritmasında maksimum sıkıştırma prensibine dayanmaktadır (Cook & Holder, 2000).

Subdue algoritmasında kullanılan sıkıştırma adımları şunlardır:

1. Benzersiz düğümler elimizdeki çizge içerisinde bulunur, Bu düğümlerin yaptığı tüm benzersiz bağlantılar bulunarak, birer çizge yapısına çevrilerek düğüm ve bağlantı şeklinde saklanır.
2. Bulunan benzersiz ait çizgelerin her biri için 2.3, 2.4, 2.5'deki denklemler kullanılarak etki değerleri hesaplanır.

Çizge_*Hacmi* = Tüm Düğümlerin Sayısı + Tüm Bağlantıların Sayısı (2.3)

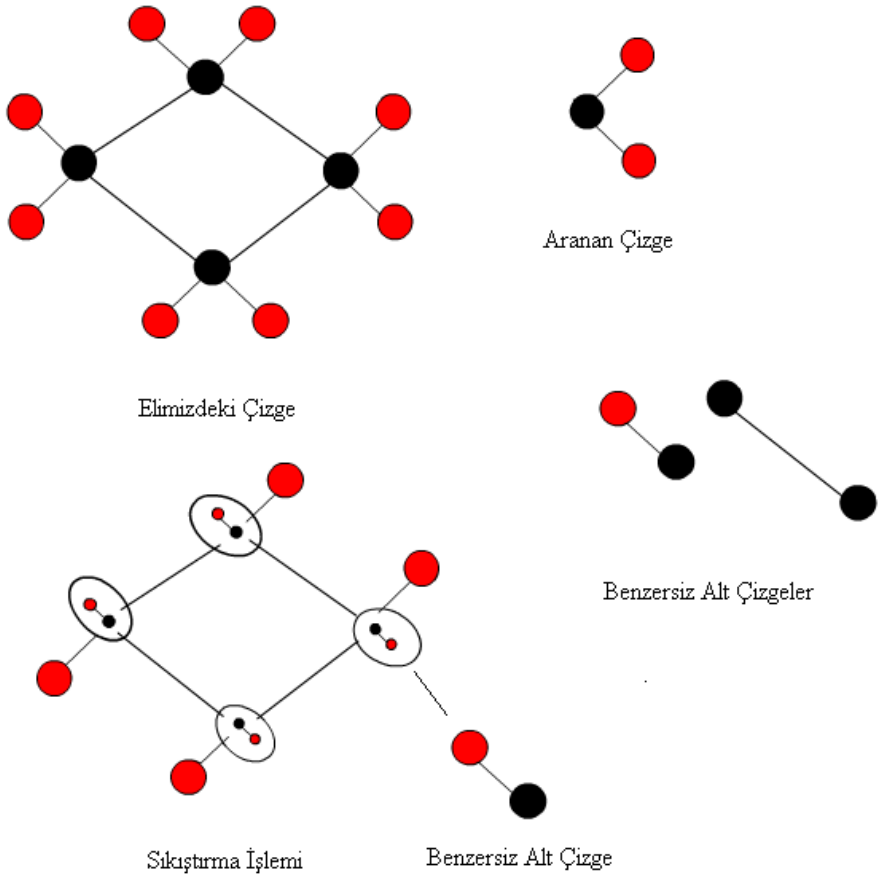
$$Sıkıştırılan\_Çizge = Girilen\ Çizge \cap^{sıkıştır} BenzersizÇizge \quad (2.4)$$

$$Etki\_Değeri = \frac{Girilen\_Çizge\_Hacmi}{Benzersiz\_Çizge\_Hacmi + Sıkıştırılan\_Çizge\_Hacmi} \quad (2.5)$$

3. En büyük B tane etki değerine sahip benzersiz çizge kullanılarak çizge sıkıştırılır
4. Yeni oluşan çizge üzerinde aynı işlem tekrarlanır.

Sıkıştırma adımları verilen Subdue algoritması bir çizge de geçen en olası yapıyı her tekrarda bulmaktadır. Bu algoritma kullanılarak çizge eşlemesi şu şekilde yapılmaktadır.

1. Karşılaştırılacak her iki çizge için benzersiz alt çizgeler bulunur.
2. Bulunan benzersiz alt çizgeler birebir karşılaştırılır. Aynı olan benzersiz alt çizgeler saklanır.
3. Aynı olan benzersiz alt çizgelere göre her iki çizge yukarıdaki kavram bulma adımları kullanılarak sıkıştırılır. Sıkıştırma işlemi yapılmadan önce sıkıştırılacak düğüm ve bağlantılar saklanır.
4. Yukarıdaki tüm adımlar tekrarlanır. Eğer birinci adımda benzersiz alt çizge yoksa veya ikinci adımda aynı olan benzersiz alt çizge yoksa durulur. Sıkıştırmada kullanılan her bir benzersiz alt çizge sıkıştırma işleminin tersi olacak şekilde saklanan düğümler ve bağlantıları açılır.
5. Açılan tüm alt çizgeler bir veya birden fazla tek bir çizge içerdiğinden her iki çizge arasındaki ortak çizgeler hacimlerine göre bulunmuş olur. Sıkıştırma işleminde kullanılan en son benzersiz alt çizge bulunan en büyük ortak alt çizgeyi verir. Sıkıştırılmış olan en son çizgede sıkıştırma işleminin yapıldığı düğümler ise ortak çizgenin bulunduğu yeri verir. Şekil 2.3'de sıkıştırmada oluşan ara öğeler gösterilmektedir.

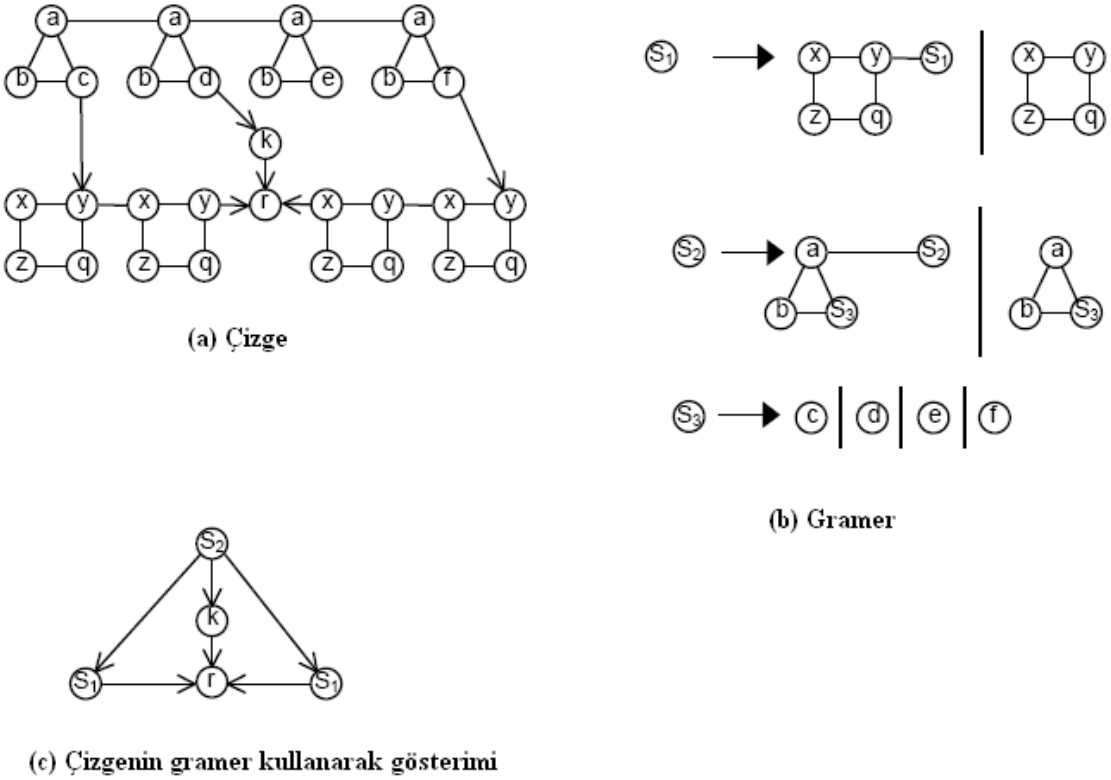


Şekil 2.3 Subdue algoritmasında sıkıştırma işleminde oluşan ara öğeler

Şekil 2.3'de elimizdeki bir çizgenin SUBDUE metodu ile sıkıştırılmasında oluşan ara formasyonlar gösterilmektedir. Bunlar sırasıyla çizge, aranan çizge ve benzersiz alt çizgelerdir. Subdue algoritması bu öğeleri kullanarak sıkıştırma işlemini gerçekleştirmektedir.

2.2 Çizge Grameri

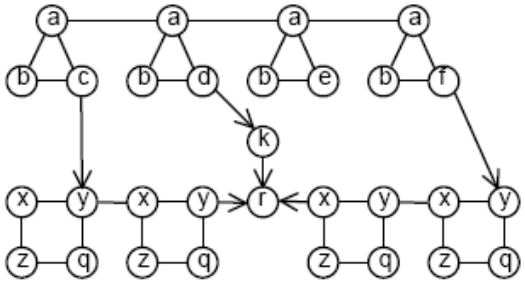
Tüm çizgeler çizge gramerleri kullanılarak türetilebilir. Çizge grameri çizgenin barındırdığı tekrarlayan alt öbeklerin kural şeklinde gösterilmesi elde edilir. Şekil 2.4'de, girilen bir çizge için gramer yapısı gösterilmektedir.



Şekil 2.4 Bir Çizge ve bu çizge'nin grameri

Şekil 2.4'de bir çizge, gramer, ve çizgenin gramer kullanılarak gösterimi bulunmaktadır. Bu şekilde Şekil 2.4 (a)'da belirtilen çizge Şekil 2.4 (b)'de gösterilen gramer kuralları ile ifade edilebilir. Bu gramer kuralları ile ifade edilen çizge ise Şekil 2.4 (c)'de gösterilmiştir.

Bir önceki bölümde açıklanan Subdue algoritması çizge gramerinin bulunmasında da kullanılmaktadır (Cook L. H., 2002). Bu bölümde Subdue algoritmasının çizge gramerini bulurken kullanılan adımları şekillerle gösterilmektedir.

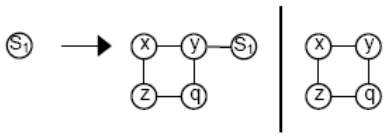


Şekil 2.5 Çizge Yapısı

Yukarıdaki çizge yapısında en sık geçen yani en yüksek etki değerine sahip benzersiz alt çizgeler saptanarak çizge grameri bulunmaktadır (bakınız Bölüm 2.2.4).

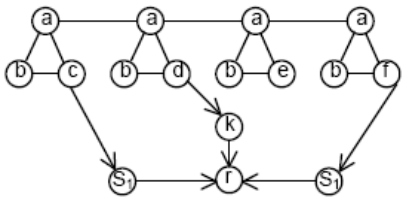
Şekil 2.6 Benzersiz Alt Çizge

Adım 1 : İlk kuralın bulunması



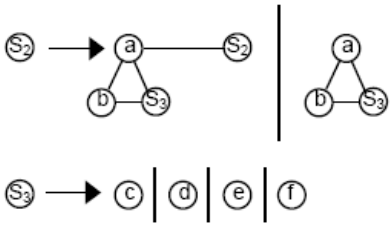
Şekil 2.7 Benzersiz alt çizge kullanılarak oluşan çizge grameri

Adım 2: Sıkıştırma işlemi ile oluşan yeni çizge yapısı ve bulunan yeni gramer kuralı

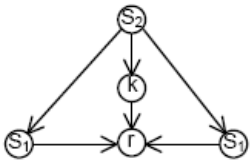


Şekil 2.8 Gramer kuralı kullanılarak sıkıştırılmış çizge

Şekil 2.9 Benzersiz Alt Çizgeler



Şekil 2.10 Benzersiz ait çizgelerin kullanılması ile oluşan çizge grameri



Şekil 2.11 Sıkıştırma işlemi oluşan yeni çizge ve çizgenin gramer kullanılarak ifadesi

Adım 1 ve adım 2'de gösterilen şekillerde Subdue algoritması kullanılarak bir gramer elde edilmekte ve bu gramer ile çizge gösterimi yapılmaktadır. Çizge grameri oluştururken çizgede en sık geçen yapıları bulduğumuz için Subdue algoritmasındaki sıkıştırma işlemi kullanılarak çizge grameri oluşturulabilir.

Çizge gramerleri sadece çizgeleri kavramsal olarak ifade etmek veya sıkıştırmak için kullanılmaz. Çizge gramerleri çizgelerin üzerinde yapılacak işlemleri göstermek için de kullanılmaktadır. Bu tür gramerler çizge dönüşüm gramerleri olarak adlandırılmaktadır.

2.3 Yüksek Boyutlu Gömme Gösterim Tekniği

Çizge gösterim teknikleri çizge teorisinin bir parçasıdır. Bir çizgenin gösterimi düğümlerin ve bağlantıların analiz edilmesini gerektirir. Farklı çizge türleri için farklı gösterim metotları bulunmaktadır (Di Battista, Eades, Tamassia, & G. Tollis, 1998). Yüksek Boyutlu Gömme tekniği de çizge gösterim tekniklerinden bir tanesidir (Koren, 2002). Literatürdeki çalışmalarda bu teknik daha çok kümeleme analizi için kullanılmaktadır.

Yüksek Boyutlu Gömme (YBM) yönteminde, ilk önce bir çizge yüksek boyutlu bir uzaya eşlenir ya da yüksek boyutlu bir uzaya gömülür. Daha sonra bu çizge tekrar iki ya da üç boyutlu bir uzayda gösterilecek biçime çevrilir. Aşağıda bu çizim tekniği için gerekli algoritma adımları verilmiştir.

1. K boyutlu bir vektör k tane merkez düğümünü tutacak şekilde oluşturulur.
2. Her merkez düğümü bir birinden en uzak k tane düğümü barındıracak şekilde seçilir. Bu işlem için ilk merkez düğüm rastgele seçilir, ikinci veya üçüncü merkez düğümler bir önce seçilen merkez düğümünden en uzak olan düğüm olarak seçilir. Sonuçta her bir düğüm için k boyutlu bir koordinat sistemi yaratılmış olur. Bir merkez düğümü u_i ile ifade edilmektedir. Buna göre bir düğümün ifadesi x_i ile ifade edilmektedir. Bir düğüm tüm merkez düğümlere olan uzaklıkları tutacağından aşağıdaki şekilde tanımlanmaktadır.

$$x_i = \{d_{u_i c_1}, d_{u_i c_2}, \dots, d_{u_i c_k}\}$$

   Burada $d_{u_i c_j}$, u_i düğümünün bulunduğu c_j merkezine olan uzaklığıdır. x_i ise n satırdan oluşan X matrisinin i'inci satırıdır.
3. $n \times k$ Boyutlu X matrisinin iki veya üç boyuta tekrar çevirilmesi için lineer bir kombinasyondan yararlanılır. X matrisi ilk önce normalleştirilir.

$$X' = X - \frac{ee^T X}{n}, \quad e = \{1,1,..,1\} \tag{2.6}$$

X matrisi bulunduktan sonra birbirine bağlı olan v_1 ve v_2 lineer vektörlerinin farklılaştırılması gerekir. Bunun için bu iki vektörün birbirine dik olması gerekir. 2.7'deki formülle diklik koşulu verilmiştir.

$$(X'v_1)^T X'v_2 = v_1^T (X'^T X')v_2 = 0 \tag{2.7}$$

$$\frac{v_i^T (X'^T X')v_i}{\|v_i\|^2} \rightarrow max, i = 1,2, \dots \tag{2.8}$$

2.7 ve 2.8'deki koşulların sağlanması için v_1 ve v_2, $X'^T X'$ $k \times k$ simetrik matrisinin en büyük iki özvektörü olarak seçilmelidir. Bu iki bağımsız vektörün seçilmesi işlemi, ana bileşenler çözümlemesi olarak bilinir. Seçilen v_1 ve v_2 vektörleri gerekli olan diklik koşulunu sağlar. Bulunan bu iki vektör ile iki yada üç boyutlu çizim yapılırken $X'v_1$ ve $X'v_2$ vektörleri koordinat olarak alınır.

BÖLÜM 3

OLASILIK TABANLI ALGORİTMALAR

3.1 Saklı Markov Modelleri

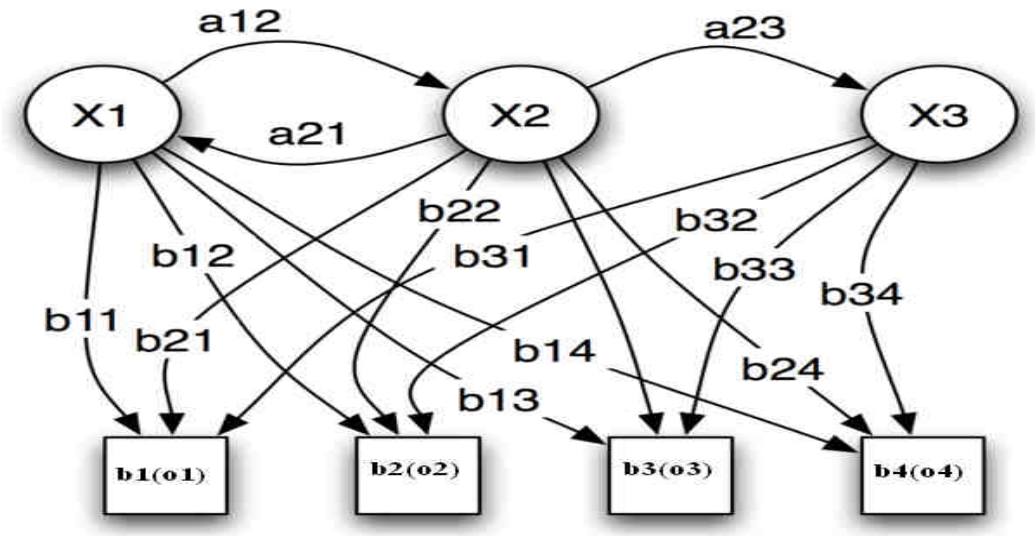
Saklı Markov Modeli (SMM) doğal dil işleme, ses tanıma, video işleme ve bunun gibi zamana bağlı değişkenlerin olduğu alanlarda kullanılan bağlantı tabanlı bir modeldir. SMM yapısı itibariyle Markov Model teorisine dayanmaktadır. Markov Model yapısı ardışık olarak gelen düğümlerin (çizge modeli) olasılığının bulunmasında kullanılır. Markov model yapısında düğümler sadece gözlemlenen öğeleri ifade eder. Markov modelinde gözükmeyen bir işlemin modelinden söz edilemez. Tüm Markov modeli gözlemlenen olayla bire bir örtüşür. Ancak SMM, gözlemleyemediğimiz ve varsaydığımız bağlantılar için saklı düğümler oluşturarak bağlantıları bizim varsayımımıza göre yapar. Bir Markov Modeli ile bir denklemin matematiksel bir modeli çıkarılırken, SMM ile bu denkleme yakınsayan bir model çıkarılır (Manning & Schutze, Foundations of Statistical Natural Language Processing, 1999).

SMM iki stokastik süreç içerir. İlk olan Markov süreci, zaman ile ilgili değişikliklerde kullanılır ve durumları içeren bir Markov zinciri üretir. Diğer süreç gözlemlenebilir olan özellik parametrelerini veya gözlemler denilen rastgele değişkenleri içerir.

SMM'in yapısı (Şekil 3.1) bir durumlar zincirinden meydana gelir. SMM zinciri üzerindeki her durum kelimenin bir parçasına karşılık gelir. Her durum bir diğerine geçişlerle bağlıdır. Geçişler, geçiş olasılıklarına (a_{ij}) bağlı olarak durum değiştirmeye imkân verir. Durumlara iliştirilen sürüm (*Emission*) olasılıkları (b_i) bir öznitelik vektörünün, referansın belirli bir zaman aralığıyla olan spektral benzerliğini gösterir. Sistem girdisine göre oluşturulan öznitelik vektörleri dizisine bağlı olarak, model üzerinde birinci durumdan başlayan farklı yollar izlenebilir. Bazı durumların tekrarı veya atlanması kullanıcının konuşma hızındaki değişimlere sistemin adaptasyonunu

sağlar. Bir kelimenin tanınabilmesi için referans olarak alınan durumdan itibaren izlenen yolun en son duruma kabul edilebilir bir olasılıkla ulaşması gereklidir.

Bir SMM modeli her anda durumu değişen birimleri olan bir sonlu durum makinesidir. Her *t* ayrık zaman anında, *i* durumundan *j* durumuna geçiş gerçekleşir ve gözlem vektörü o_t yoğunluk vektörü b_j (o_t) ile dışarı verilir. Bundan başka *i* durumundan *j* durumuna geçiş aynı zamanda rastgeledir ve a_{ij} yoğunluğu ile olur. Şekil 3.1'de üç durum soldan sağa SMM atlamasız olarak verilmiştir.



Şekil 3.1 SMM yapısı

Tam bir SMM modeli belirlenmesi iki model parametresi N ve M' in, gözlem sembollerinin ve üç set olasılık ölçümleri A, B, π'in belirlenmesini gerektirir. Bu parametrelerin tanımı şöyledir:

1. N parametresi, SMM' deki durum sayısıdır. Ayrı durumlar $\{1,2,\ldots,N\}$ olarak tanımlanır, *t* anındaki durum q_t olarak gösterilir.

2. M parametresi her durumda bulunan farklı gözlem sembollerinin sayısıdır. Gözlem sembolleri modellenen sistemin fiziksel çıktısı olarak gösterilir. Ayrı gözlem sembolleri $O = \{o_1, o_2, \ldots, o_m\}$ ile gösterilir.

3. $A = \{a_{ij}\}$ matrisi durum geçiş olasılık dağılımıdır. Burada a_{ij}, *i* durumundan *j* durumuna geçiş olasılığıdır.

$$a_{ij} = P(q_{t+1} = j | q_t = i); 1 \leq i, j \leq N \quad (3.1)$$

i' den *j*' ye tek bir geçişle ulaşılamıyorsa tüm *i ve j* değerleri için $a_{ij} = 0$ olur.

4. $O = \{o_1, o_2, \dots, o_T\}$ Gözlem sembolleri seti olsun. $B = \{b_j(o_t)\}$ Matrisi gözlem sembol olasılık dağılımıdır.

$$b_j(o_t) = P(o_t | q_t = j); 1 \leq t \leq T \quad (3.2)$$

$i, j = 1,2, \dots, N$ Durumunda sembol dağılımını tanımlar. Ses tanıma probleminde, gözlem sembolleri özellik parametresi vektörleridir.

5. $\pi = \{\pi_i\}$ vektörü başlangıç durum dağılımıdır ve 3.3'deki gibi hesaplanır.

$$\pi_i = P(q_1 = i); 1 \leq i \leq N \quad (3.3)$$

Denklem 3.3'deki π_i değerinin bulunması ile olasılıkların hesaplanması için gereken model yapısı oluşturulmuş olur. SMM' de yukarıdaki parametrelerin hesaplanması için sınırlılık ilkesinden *(Limited Horizon)* yararlanılmıştır (Manning & Schutze, Foundations of Statistical Natural Language Processing, 1999). Denklem 3.4'de bu hesaplama kriteri verilmektedir.

$$P(X_{t+1} = s_k | X_1, .., X_t) = P(X_{t+1} = s_k | X_t) \quad (3.4)$$

SMM' de üç problem için üç algoritma vardır. Bunlar aşağıda özetlenmektedir.

1. İlk problem gözlemlenen bir serinin $O = (o_1, \dots, o_t)$ verilen bir model için $\mu = (A, B, \Pi)$ olasılığının hesaplanması problemidir. Bu Forward algoritması ile çözümlenir.

   Herhangi bir durum serisi için $X = (X_1, \dots, X_{T+1})$ gözlem olasılığı aşağıda verilmiştir.

   $$P(O|X, \mu) = \prod_{t=1}^{T} P(O_t|X_t, X_{t+1}, \mu) \tag{3.5}$$

   $$= b_{x_1 x_2 o_1}\ b_{x_2 x_3 o_2}\ \dots b_{X_T X_{T+1} O_T}$$

   Bu olasılığın tüm durumlar için hesaplanması polinom-zamanı karmaşıklık düzeyine sahip olduğundan hesaplanması zordur (Manning & Schutze, Foundations of Statistical Natural Language Processing, 1999).

2. İkinci problem gözlemlenen bir serinin $O = (o_1, \dots, o_t)$ verilen bir model için $\mu = (A, B, \Pi)$ olasılığının maksimum olacağı durumların bulunması problemidir. Bu Viterbi algoritması ile çözülür (Merialdo, 1994).

3. Üçüncü problem gözlemlenen serilerin gözlemlenme olasılığını maksimum yapacak model parametrelerinin bulunması işlemdir. Bu

   $$\underset{\mu}{argmax}\ P(O_{egitim}|\mu) \tag{3.6}$$

   olasılığının hesaplanması ile bulunmaktadır. Bauch – Welch algoritması gözlemin maksimize edilmesi için tasarlanmış bir eğitim algoritmasıdır. Bu optimizasyon tekniğinin SMM'lere uyarlanan hali En İyileme *(Expectation Maximization)* metodudur. Sonraki kısımlarda her üç problem için kullanılan algoritmalar özetlenmektedir.

3.1.1 Forward ve Backward Algoritması

Forward algoritması gözlemlenen durumların verilen bir modele göre baştan sona doğru olan gerçekleşme olasılıklarını hesaplar. Backward algoritması ise tam ters yönde çalışarak bu olasılığı hesaplar. Forward değişkeni $\alpha_t(i)$ şu şekilde tanımlanır.

$$\alpha_t(i) = P(o_1 o_2 .. o_t, q_t = i|\mu) \tag{3.7}$$

Örneğin, *t* anında *i* durumunda μ modeli verilen kısmi gözlem dizisi $o_1 o_2 o_3 \ldots o_t$'nin olasılığı, 3.7 verilen $\alpha_t(i)$ değeri, 3.8, 3.9 ve 3.10'da belirtilen adımlar kullanılarak hesaplanır.

1. *Başlangıç*

$$\alpha_t(i) = \pi_i b_i(o_1) \qquad 1 \leq i \leq N \tag{3.8}$$

2. *Sonuç Çıkarma*

$$\alpha_{t(i)} = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}) \qquad 1 \leq t \leq T-1, 1 \leq j \leq N \tag{3.9}$$

3. Sonuç

$$P(O|\mu) = \sum_{i=1}^{N} \alpha_T(i) \tag{3.10}$$

3.8 ve 3.9'da bulunan $\alpha_t(i)$ değerleri 3.10'da yerine konulduğunda gözlem olasılığı bulunur. Benzer olarak, Backward değişkeni $\beta_t(i)$ şöyle tanımlanabilir:

$$\beta_t(i) = P(o_{t+1} o_{t+2} \ldots o_T|q_t = i, \mu) \tag{3.11}$$

Backward değişkeninin ($\beta_t(i)$), *t+1* anından t anına kadar olan değerinin hesaplanması 3.12 ve 3.13'de verilmiştir.

1. Başlangıç:

$$\beta_t(i) \qquad 1 \leq i \leq N \tag{3.12}$$

2. Sonuç Çıkarma:

$$\beta_{t(i)} = \sum_{j=1}^{N} \beta_{t+1}(i) a_{ij}\, b_j(o_{t+1}) \qquad t = T-1, T-2, .., 1 \qquad 1 \le i \le N \qquad (3.13)$$

$\beta_t(i)$ nin hesaplanması forward algoritmasındaki gibi $N^2 t$ karmaşıklık düzeyine sahiptir.

3.1.2 Viterbi algoritması

SMM modelinin hesaplanmasında en önemli problem en uygun durum dizileri tahminidir. Verilen gözlem dizisi ile ilgili en uygun durum dizisini bulmak için birkaç yol vardır. Çeşitli uygunluk kıstasları tanımlanabilir. Amaç $P(q, O, \mu)$ 'yı maksimize eden durum dizisini bulmaktır. Çözüm için dinamik Viterbi algoritması geliştirilmiştir. 3.14'deki denklem ile bu formülasyon verilmiştir.

$$\delta_{t(i)} = \max_{q_1, q_2, \ldots, q_{t-1}} P(q_1 q_2 \ldots q_{t-1,} q_t, o_1, .., o_{t-1} q_t = j | \mu) \qquad (3.14)$$

$q = (q_1 q_2 \ldots q_T)$ durum dizisi $O = (o_1, o_2, \ldots, o_t)$ dizisi gözlendiğinde olabilecek en iyi durum dizisidir. Diğer bir ifade ile, $q = (q_1 q_2 \ldots q_T)$ durum dizisi en yüksek olasılıkla $O = (o_1, o_2, \ldots, o_t)$ dizisini üretir. Aşağıda Viterbi algoritmasının adımları gösterilmiştir.

1. Başlangıç:

$$\delta_j(1) = \pi_j, \qquad 1 \le j \le N \qquad (3.15)$$

2. Sonuç Çıkarma

$$\delta_j(t+1) = \max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo_t} \ , \ \ 1 \le j \le N \qquad (3.16)$$

İzlenilen yolun bulunması için indekslerin tutulması 3.17'deki gibi olmaktadır.

$$\varphi_j(t+1) = \text{ar}\max_{1 \le i \le N} \delta_i(t) a_{ij} b_{ijo_t} \ , \ \ 1 \le j \le N \qquad (3.17)$$

3. Sonlandırma ve izlenilen yolun okunması3.18, 3.19'de verilmektedir.

$$\hat{q}_{T+1} \ = \ \text{ar}\max_{1 \le i \le N} \delta_i(T+1) \qquad (3.18)$$

$$\hat{q}_t = \varphi_{\hat{q}_{t+1}}(t+1 \tag{3.19}$$

Gözlem olasılığın hesaplanması 3.20'de verilmektedir.

$$P(\hat{q}) = \max_{1 \le i \le N} \delta_i(T+1) \tag{3.20}$$

3.19 ve 3.20'deki hesaplamalar yapıldığında $\hat{q}$ dizisi bize izlenilen yolu yani saklı Markov Düğümlerini, $P(\hat{q})$ ise bu yolun olasılığını vermektedir.

3.1.3 Forward - Backward algoritması (Baum - Welch Algoritması)

Forward-Backward algoritması bir En İyileme algoritmasıdır (Dempster, Laird, & Rubin, 1977). Bir olasılık tabanlı modelin parametrelerinin tahmin edilmesinde kullanılır. Aşağıda bu algoritma için gerekli formüller verilmiştir.

$p_t(i,j), 1 \le t \le T, 1 \le i,j \le N$ aşağıda hesaplandığı şekliyle t zamanında O çıktısını i durumundan j durumuna giderken gözlemlemenin olasılığı olsun.

(3.21)

$$p_t(i,j) = P(X_t = i, X_{t+1} = j | O, \mu) \text{(3.3.1)}$$

$$= \frac{P(X_t=i, X_{t+1}=j, O|\mu)}{P(O|\mu)}$$

$$= \frac{\alpha_i(t) a_{ij} b_{ijo_t} \beta_j(t+1)}{\sum_{m=1}^{N} \alpha_m(t)\beta_m(t)} \implies \frac{\alpha_i(t) a_{ij} b_{ij}(o_t) \beta_j(t+1)}{\sum_{m=1}^{N}\sum_{n=1}^{N} \alpha_m(t) a_{mn}\, b_{mn}(o_t)\beta_n(t+1)}$$

3.21'deki açılıma göre toplam olasılık 3.22'deki gibi hesaplanır.

$$\zeta_i(t) = \sum_{j=1}^{N} p_t(i,j) \tag{3.22}$$

Eğer süreç indeksi üzerinde 3.22'de hesaplanan değeri toplarsak beklenen geçişlerin sayısını elde ederiz. Bu toplama işlemi 3.23'de verilmiştir.

(3.23)

$$\sum_{t=1}^{T} \zeta_i(t) = durum\ i\ ve\ gözlem\ O\ için\ beklenen\ geçiş\ sayısı$$

$$\sum_{t=1}^{T} p_i(t) = durum\ i\ ve\ durum\ j\ arasında\ gözlem\ O\ için\ beklenen\ geçiş\ sayısı$$

3.21 ve 3.22'deki formülleri ve 3.23'deki açılımı kullanarak yüksek olasılık veren model parametrelerini 3.24' deki gibi güncelleyebiliriz.

(3.24)

$$\hat{\pi} = \{t = 1\ zamanında\ durum_i nin\ beklenen\ olasılığı\}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} p_t(i,j)}{\sum_{t=1}^{T} \zeta_i(t)}$$

$$\hat{b}_{ij}(o_k) = \frac{\sum_{\{t:o_t=k, 1\le t\le T\}} p_t(i,j)}{\sum_{t=1}^{T} p_t(i,j)}$$

Sonuçta $\mu = (A, B, \Pi)$ den $\hat{\mu} = (\hat{A}, \hat{B}, \hat{\Pi})$ modeli elde edilir. Bu çıkarsama işlemi birden fazla tekrarlanarak model parametrelerindeki değişimin belli bir η değerinin altına düşmesi ile son bulmalıdır. Sonuçta, en son elde edilen $\hat{\mu} = (\hat{A}, \hat{B}, \hat{\Pi})$ modeli eğitimde kullandığımız yapıya benzer olan veriler (parametreler) için en uygun sonucu verecektir.

3.2 Kavram Uzayının Öğrenilmesi

Kavramların öğrenilmesi doğal dil işleme literatüründe çok önceden beri yer alan bir konu olmasına rağmen, kavram uzayının öğrenilmesi literatüre yeni girmiştir. Kavram uzayının öğrenilmesinde en çok gözetimsiz öğrenme metotları kullanılmaktadır. Gözetimsiz öğrenme, kavramları elde edilen ham veriden her hangi bir eğitim yapmadan öğrenilmesi için kullanılan bir yöntemdir. Gözetimsiz öğrenme algoritmaları içerisinde benzerlik üzerine kurulu olan *K-Means* (MacQueen, 1967) ve Hiyerarşik Kümeleme (Johnson, 1967) algoritmaları kavramların öğrenilmesi ya da kümelenmesi için bize bir çatı oluşturmaktadır. Fakat kavram uzayının öğrenilmesinde kümeleme algoritmaları yeterli olmamaktadır. Çünkü sınıflama ve kümeleme algoritmaları elde edilen sonucun yorumlanması işini gözlemciye bırakmaktadır (Anderson, Michalski, Carbonell, & Mitchell, 1985). Kavram uzayı sembolik, kavramsal ve bağlantı tabanlı bir model üzerinde durmalıdır (Gardenfors, 2000).

3.3 Yumuşatma Algoritmaları (Düzleme Algoritmaları)

Yumuşatma algoritmaları *(Language Smoothing)* Saklı Markov Modellerinde ve *n-grams* doğal dil modellerinde özellikle kullanılmaktadır (Manning & Schutze, 1999). Yumuşatma algoritmalarının amacı, seyrek olan durumların olasılıklarını tahmin işlemlerini zorlaştırmaması için güncelleyerek geçme olasılıklarını arttırmaktır.

Yumuşatmada veya matematikte kullanılan adıyla düzlemede en büyük problem hangi olasılığın ne kadar artacağı ve artan olasılığın diğer olasılıkları etkileyip etkileyemeyeceği sorunudur. Bu problem çan eğrisi sisteminde notu çok düşük olan öğrencilerin notunu arttırmak gibidir. Bir öğrencinin notunun değişmesi diğer öğrencilerin çan eğrisindeki notunu etkiler. Dolayısıyla olasılıkların güncellenmesi işlemi belli bir koşula göre yapılmalıdır. Değişen bir olasılığın diğer olasılıkları ne kadar etkileyeceği bir problemdir.

Olasılığın sıfır olduğu durumlar maksimum beklenti durumlarının olasılıklarını sıfır yapacağından, bu durumların belli oranlarda yükseltilmesi gerekir. Örnek olarak 3.25'te bir cümledeki ikili kelime grupları için olasılıklar verilmiştir.

(3.25)

$$P(ali|\cdot) = 0.1, \quad P(bir|ali) = 0, \quad P(bardak|bir) = 0.4,$$
$$P(su|bardak) = 0.4, \quad P(içti|su) = 0.1, \quad P(\cdot\,|içti) = 0.1$$

3.25 de ikili olasılıkları verilen "ali bir bardak su içti." cümlesinin olasılığı bigram modeline göre her bir ikili öğenin olasılıklarının çarpımıdır. 3.25'de verilen olasılıklar göz önüne alındığında sonuç 3.26'daki gibi olmaktadır.

(3.26)

$$P(cümle) = P(ali|\cdot) \times P(bir|ali) \times P(bardak|bir) \times P(su|bardak) \times P(içti|su) \times P(\cdot\,|içti)$$

$$P(cümle) = 0$$

Cümlenin olasılığının sıfır olması bir problem olduğundan bu olasılığı sıfır yapan kelime birleşimlerinin sıfır olmayacak şekilde değiştirilmesi gerekir. Ancak bu değişim gerçekleşirken hangi birleşimin olasılığının ne kadar artacağının bulunması yumuşatma algoritmaları ile mümkündür.

Cümlenin olasılığının sıfır olması bir problem olduğundan bu olasılığı sıfır yapan kelime birleşimlerinin sıfır olmayacak şekilde değiştirilmesi gerekir. Ancak bu değişim gerçekleşirken hangi birleşimin olasılığının ne kadar artacağının bulunması yumuşatma algoritmaları ile mümkündür.

Yumuşatma algoritmalarında en basit yöntem ekleme yöntemidir. Olasılığı sıfır yapan değerlerin birleşim sayısı yani birlikte olma sayısı bir arttırılır veya belli bir oranda arttırılır. Ancak bu yöntem yumuşatmada tavsiye edilmemektedir. Bunun nedeni bir birleşimin olasılığının ne kadar artacağının belirlenmesi birleşimde bulunan öğelerin tümüne bağlı olmasıdır. Dolayısıyla birlikte geçme sayısı arttırılacak öğenin diğer öğeler üzerinde nasıl bir etki yaratabileceği hesaplanarak sayısının arttırılması gerekir.

Bu kısımda ekleme yönteminden daha üstün olan yumuşatma algoritmalarına ilişkin iki metot anlatılacaktır. Bunlar sırasıyla Jelinek-Mercer Smoothing ve Kneser-Ney Smoothing metotlarıdır.

3.3.1 Jelinek-Mercer düzlemesi *(Jelinek-Mercer Smoothing)*

Jelinek-Mercer düzleme algoritmasında olasılıkların güncellenmesi işlemi başka modellerden gelen bilgi ile yapılır (Jelinek & Mercer, 1980). Örneğin *Bigrams*[3] (ikili öğe modeli) modelinde tahmin yapılırken bir önceki öğe göz önüne alınır. Eğer önceki öğe ile şimdiki öğe daha önce hiç birlikte geçmemişse birlikte geçme olasılıkları sıfır olacak ve öğe dizisinin tüm olasılığı diğer ikili öğelere bakılmaksızın sıfır olacaktır. Bunu engellemek için *Unigrams* (tekli öğe modeli) modeli kullanılarak şimdiki öğenin olasılığının belli bir yüzdesi *Bigrams* modelinden gelen olasılığın belirli bir yüzdesi ile toplanır. 3.27'de Jelinek-Mercer metodunun denklemi verilmektedir.

[3] Bigrams ve Unigrams modelleri *n-grams* model teorisi olarak bilinir. Bu model teorisinde amaç bir öğeden sonra gelecek öğeyi tahmin etmektir. *Bigrams*'da ardışık iki öğenin gelme olasılığı, Unigrams'da ise tek bir öğenin gelme olasılığı tahmin etme işleminde kullanılır.

(3.27)

$$P(w_i|w_{i-1}) = \lambda_1 P(w_i|w_{i-1}) + \lambda_2 P(w_i)$$

3.27'deki denkleme göre şimdiki öğe w_i ile bir önceki öğe w_{i-1} arasındaki *Bigrams* olasılığı sıfır olduğunda ve $w_i{}'$nin *Unigrams* olasılığı sıfır olamayacağından dolayı ikisinin toplamı, sonucun sıfır olmasını engelleyecektir. Tablo 3.3'de bununla ilgili bir örnek için olasılık değerleri verilmiştir. Bu $\lambda_1 = 0.5 \ ve \ \lambda_2 = 0.2$ olarak alınmıştır.

Bigrams	$P(ali\| \cdot) = 0.1$	$P(bir\|ali) = 0$	$P(bardak\|bir) = 0.4$	$P(su\|bardak) = 0.4$	$P(içti\|su) = 0.1$	$P(\cdot \|içti) = 0.1$
Unigrams	$P(ali) = 0.1$	$P(bir) = 0.3$	$P(bardak) = 0.2$	$P(su) = 0.5$	$P(içti) = 0.9$	
Cümle	ali	bir	bardak	su	içti	

Tablo 3.3 Bir cümledeki kelimelerin ikili ve birli olasılıkları

Tablo 3.3 kullanılarak 3.28'deki hesap yapıldığı zaman $P(cümle)$değerini sıfır yapacak bir durumun olmadığı gözükmektedir.

(3.28)

$$P(cümle) = \big(0.5 \times P(ali|\cdot) + 0.2 \times P(ali)\big) \times (0.5 \times P(bir|ali) + 0.2 \times P(bir) \times \cdots)$$

İki modelin (*Bigrams* ve *Unigrams*) kullanılmasının amacı sadece seyrek oluşan durumlara bakarak karar vermemek ve daha sık oluşan durumları da değerlendirerek olasılığı arttırmaktır (Manning & Schutze, Foundations of Statistical Natural Language

Processing, 1999). Ancak yukarıdaki metoda göre karşımıza çıkan ilk problem λ değerlerinin nasıl belirleneceğidir. λ değerleri iki şekilde belirlenebilir. Bunlardan ilki En İyileme metodunu kullanarak olasılığı maksimum yapan λ değerlerinin seçilmesi yöntemi, ikincisi ise N-Kere Çapraz doğrulama algoritması (*N-Fold Cross Validation*) kullanılarak rastgele λ değerlerinden doğrulamada en çok başarılı olan λ değerlerinin seçilmesi yöntemidir.

3.3.2 Kneser-Ney düzlemesi *(Kneser-Ney Smoothing)*

Bir önceki metot göz önüne alındığında, Kneser-Ney Smoothing metodunda amaç, yüksek olasılıkta oluşması beklenen bir durum için daha düşük λ değeri vermektir. Bunu yapmak için belirli bir oranda olasılık değeri eksiltilir. Düşük olasılıklar ise belirli bir oranda yükseltilir. Ancak tüm bu eksiltme ve yükseltme işlemleri belirli koşullara göre yapılır. Buna göre yüksek seviyeli modelde *(Bigrams)* olasılıklar düşük ise düşük seviyeli modeldeki *(Unigrams)* olasılıkların düşürülmesi gerekir.

Örneğin bir derlemde 'Boğaziçi Üniversitesi' kelime öbeği 'Boğaziçi Köprüsü' kelime öbeğine oranla daha sık geçebilir. Ancak 'Üniversitesi' kelimesinin çoğu zaman 'Boğaziçi' kelimesinden sonra geldiğini ve tek başına geçme sıklığının 'Köprüsü' kelimesinden tek başına geçme sıklığından çok daha az olduğunu düşünürsek Jelinek-Mercer smoothing metodu 'Boğaziçi Köprüsü' kelime öbeği için "Boğaziçi Üniversitesi" kelime öbeğine verdiği olasılıktan daha yüksek bir olasılık verecektir. 'Boğaziçi Köprüsü' kelime öbeğinin olasılığı bir önceki Jelinek-Mercer smoothing metodunda hesaplandığından daha düşük olmalıdır. Dolayısıyla Jelinek-Mercer metoduna göre düşük seviyeli *(Unigrams)* modelinin olasılığı yalnızca yüksek seviyeli *(Bigrams)* modelin olasılığı düşük ise devreye girmelidir. Buna göre "Köprüsü" kelimesinin olasılığı ile 'Boğaziçi Köprüsü' kelime öbeğinin olasılığı toplanırken 'Köprüsü' kelimesinin olasılığı eksiltilerek toplanmamalıdır. Bu metot için kullanılan denklemler 3.29 ve 3.30'da verilmektedir.

(3.29)

$$N_{1+}(\cdot \text{öğe}_i) = |\{\text{öğe}_{i-1} : sayı(\text{öğe}_{i-1}\text{öğe}_i) > 0\}|$$

(3.30)

$$N_{1+}(\cdot\,\cdot) = \sum_{\text{öğe}_i} N_{1+}(\cdot\ \text{öğe}_i)$$

3.29 ve 3.30'daki formüllere göre $N_{1+}(\cdot\ \text{öğe}_i)$, öğe$_i$ 'den önce gelen tüm öğelerin sayısını döndürmektedir. $N_{1+}(\cdot\,\cdot)$ ise bu işlemi bilinen tüm öğeler için yapmaktadır. Yukarıdaki işlemler sonucunda tüm öğeler için o öğeden önce gelen tüm farklı öğelerin sayılarının toplamlarını bulmuş oluruz. 3.31'de 3.29 ve 3.30'daki denklemler kullanılarak düşük seviyeli modeldeki bir öğenin olasılık dağılımını hesaplayan bir denklem verilmiştir.

(3.31)

$$P_{KN}(\text{öğe}_i) = \frac{N_{1+}(\cdot\text{öğe}_i)}{N_{1+}(\cdot\cdot)}$$

3.32'de ise yüksek seviyeli modeldeki bir öğenin olasılık dağılımını hesaplayan denklem gösterilmektedir.

(3.32)

$$P_{KN}(\text{öğe}_i|\text{öğe}_{i-1}) = \frac{\max\{sayı(w_{i-1}) - \delta, 0\}}{\sum_{\text{öğe}_i} sayı} + \frac{\delta}{\sum_{\text{öğe}_i} sayı(\text{öğe}_{i-1})} \times N_{1+}(\text{öğe}_{i-1}\ \cdot)$$

3.31 'e göre ikili öğelerin olasılığı δ eksilteme değeri kullanılarak hesaplanmıştır. δ değeri herhangi bir sayı olabilir.

Literatürde Kneser-Ney algoritmasının değiştirilmiş sürümleri kullanılmaktadır. Olasılık modellerinde en iyi performans veren algoritmanın bu olduğu belirtilmektedir (Chen & Goodman, 1998).

BÖLÜM 4

SÖZDİZİMSEL ETİKETLEME: BİR UYGULAMA ÇALIŞMASI

Bu bölümde sözdizimsel etiketleme uygulamasındaki üç temel bileşen anlatılmaktadır. Bunlar kısaca; veri kümesinin oluşturulması, gerekli olan kısımların veri kümesinden çıkarılması ve elde edilen verinin işlenmesidir. Ayrıca algoritmayı eğitmede kullanılan ODTÜ-Sabancı Türkçe Ağaç Derlemi ve yapılan testler yine bu bölümde anlatılmaktadır.

4.1 ODTÜ - Sabancı Türkçe Ağaç Derlem Yapısı

ODTÜ-Sabancı derlemi, bağlam grameri kullanılarak işaretlenmiş bir derlemdir. Türkçe'nin ekli yapısından dolayı işaretlemeler biçimbilimsel çözümleme ile çıkarılan yapı kullanılarak elde edilmiştir. Bunun birinci sebebi, eklerin cümlenin sözdizimsel yapısını değiştirmesidir. Bu derlemin XML formatında işaretlenmiş bir örneği Şekil 4.1'de gösterilmektedir.

```
<?xml version="1.0" encoding="windows-1254" ?>
<Set sentences="1">
<S No="1">
<W IX="1" LEM="" MORPH=" " IG='[(1,"ama+Conj")]' REL="[4,1,(S.MODIFIER)]"> Ama </W>
<W IX="2" LEM="" MORPH=" " IG='[(1,"yol+Noun+A3sg+Pnon+Acc")]' REL="[3,1,(OBJECT)]"> yolu </W>
<W IX="3" LEM="" MORPH=" " IG='[(1,"bil+Verb+Neg+Prog1+A1sg")]' REL="[4,1,(SENTENCE)]"> bilmiyorum </W>
<W IX="4" LEM="" MORPH=" " IG='[(1,".+Punc")]' REL="[,( )]"> . </W>
</S>
</Set>
```

Şekil 4.1 ODTÜ - Sabancı Ağaç Derleminin XML gösterimi

Şekil 4.1'deki "Ama yolu bilmiyorum" cümlesinde her bir "<W>" etiketi, kelimeyi ve bu etiketin "IG" özelliği ise onun biçimbilimsel çözümleme sonuçlarını göstermektedir. "'Rel" özelliği ise bu kelimenin sözdizimsel olarak başka hangi kelimenin biçimbilimsel yapısını nitelediğini belirtmektedir. Çoklu biçimbilimsel yapıya sahip olan bir kelimeyi barındıran cümle örneği Şekil 4.2'de gösterilmektedir.

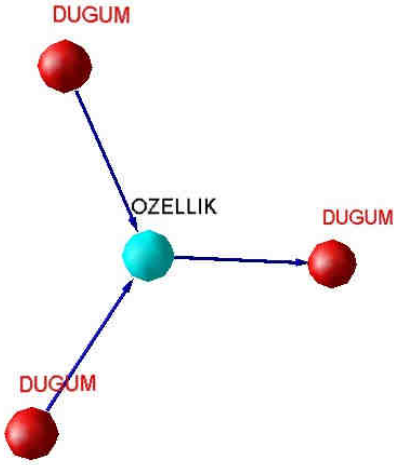
```
<S No="4">
<W IX="1" LEM="" MORPH=" " IG='[(1,"arka+Noun+A3sg+Pnon+Dat")]' REL="[2,2,(DATIVE.ADJUNCT)]"> Arkaya </W>
<W IX="2" LEM="" MORPH=" " IG='[(1,"tara+Verb")(2,"Verb+Pass+Pos+Narr+A3sg")]' REL="[3,1,(SENTENCE)]"> taranmış </W>
<W IX="3" LEM="" MORPH=" " IG='[(1,".+Punc")]' REL="[,( )]"> . </W>
```

Şekil 4.2 Birden fazla biçimbilimsel analizi taşıyan cümle örneği

Şekil 4.2'deki "Arkaya taramış" cümlesinde "taramış" kelimesinin <IG> etiketi bu kelimenin iki faklı biçimbilimsel analiz yapısını göstermektedir. Bu cümlede "Arkaya" kelimesinin "Rel" özelliği kısmında "2,2" değeri ile ikinci kelimenin ikinci biçimbilimsel analiz yapısını sözdizimsel bir değer olan "Dative. Adjunct" ile nitelediği anlaşılmaktadır.

4.2 Veri Kümesinin Oluşturulması

Veri kümesi ODTÜ-SABANCI Ağaç Derlemi olarak seçilmiştir. Bu uygulamadaki veri kümesi anlambilimsel bilgi içermemesine rağmen anlambilimsel kavramların istatistiksel olarak elde edilmesinde kullanılabilmektedir. Dolayısıyla bu verinin anlambilimsel kavramları elde etmeye yönelik en uygun şekilde tutulması ileriki çalışmalar için önemlidir. Bu yapı için en uygun tasarım bir çizge modelidir. Tasarlanan çizge modeli Şekil 4.3'de gösterilmektedir.



Şekil 4.3 Veri kümesinin ve ilişkilerin tutulacağı çizge modeli

Tasarlanan yapı diğer çizge yapılarından farklı olarak bir düğüme gelen ve o düğümden çıkan bağlantıların bilgisini düğüm üzerinde tutmaktadır. Bu bir düğüme gelen bağlantıların ve bu düğümden çıkan bağlantıların olasılığını tutmayı kolaylaştırmakla birlikte, bağlantıların bir düğüm olarak tutulmasına olanak sağlamaktadır.

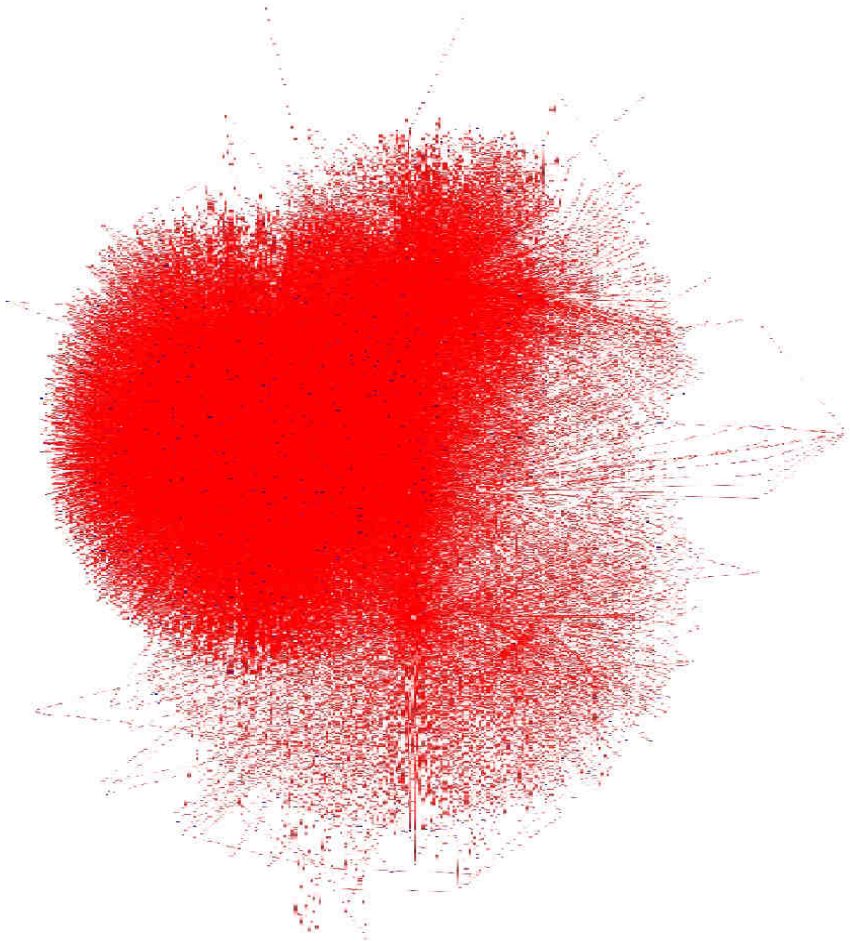
Yapılan araştırmada çizge yapısının ve bağlantı modellerinin kavramları tutmak ve işlemekte iyi bir araç olduğu saptanmış ancak mantıksal düzeyde yetersiz olduğu Gardenfors tarafından belirtilmektedir (bkz. Bölüm 3.2). Bu konuda yapılan bir çalışmada kavramların bağlantı modellerini kullanarak başarılı bir şekilde öğrenilebileceği gösterilmektedir (Berg & Schuemie, 1999). Yapılan bu çalışmadan yola çıkarak, çizge modelinde bilgiyi bağlantıda tutmak yerine bir düğümde tutmanın ileride bağlantı bilgilerini de başka bağlantılar ile ilişkilendirebilmemiz açısından yararlı olacağı anlaşılmaktadır. Bu modeli kullanarak veri kümesinin oluşturulması işlemi aşağıda açıklanmaktadır.

1) ODTÜ-SABANCI Ağaç Derleminin hataları düzeltildi.
    a. XML verilerinde etiket eşleme hataları düzeltildi.
    b. Yanlış sözdizimsel ve biçimbilimsel özellik barındıran yapılar düzeltildi.

2) XML verisi okunarak her bir bilginin ait olduğu düzeye göre çizge düğümleri yaratıldı.
    a. Sözdizimsel, biçimbilimsel ve her türlü değer bilgisi için üç düzey yaratıldı. Sözdizimsel düzeyde sözdizimsel etiketlerin düğümleri, biçimbilimsel düzeyde biçimbilimsel etiketlerin düğümleri ve değer düzeyinde kelime ve karakter gruplarının metin içerisindeki yalın değerleri için düğümler yaratıldı.
    b. Her düğüm arasında 'barındırma', 'sonrasında gelme' ve 'özelliği olma' durumlarına göre bağlantı düğümleri oluşturuldu.

3) Oluşturulan çizge, çizge gösterim biçimine (bkz. Ek-1) dönüştürüldü. Bu biçim kullanılarak okunan dosya matematik programlama (*Wolfram Mathematica*) ortamında Yüksek Boyutlu Gömme tekniği kullanılarak çizildi (bkz. Bölüm 2.3).

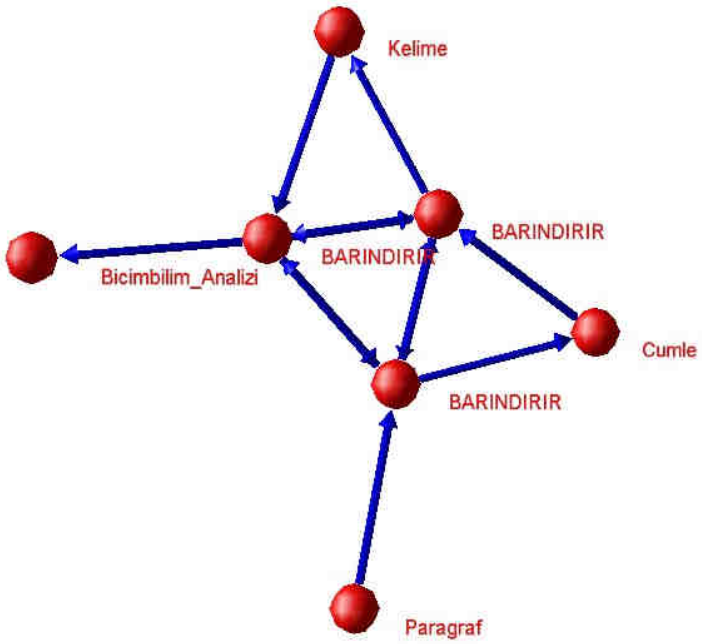
Şekil 4.4'de, anlatılan adımların uygulanması sonucunda oluşan çizge şekli gösterilmektedir. Bu çizgede çok fazla bilgi yer aldığı için şekil üzerinde bilginin gösterimi mümkün olmamaktadır. Ancak YBG'nin kullanılması bize bilgi kümesindeki benzerlik dağılımını verdiği için yararlıdır.



Şekil 4.4 ODTÜ-SABANCI Ağaç Derleminin YBG tekniği kullanılarak gösterimi

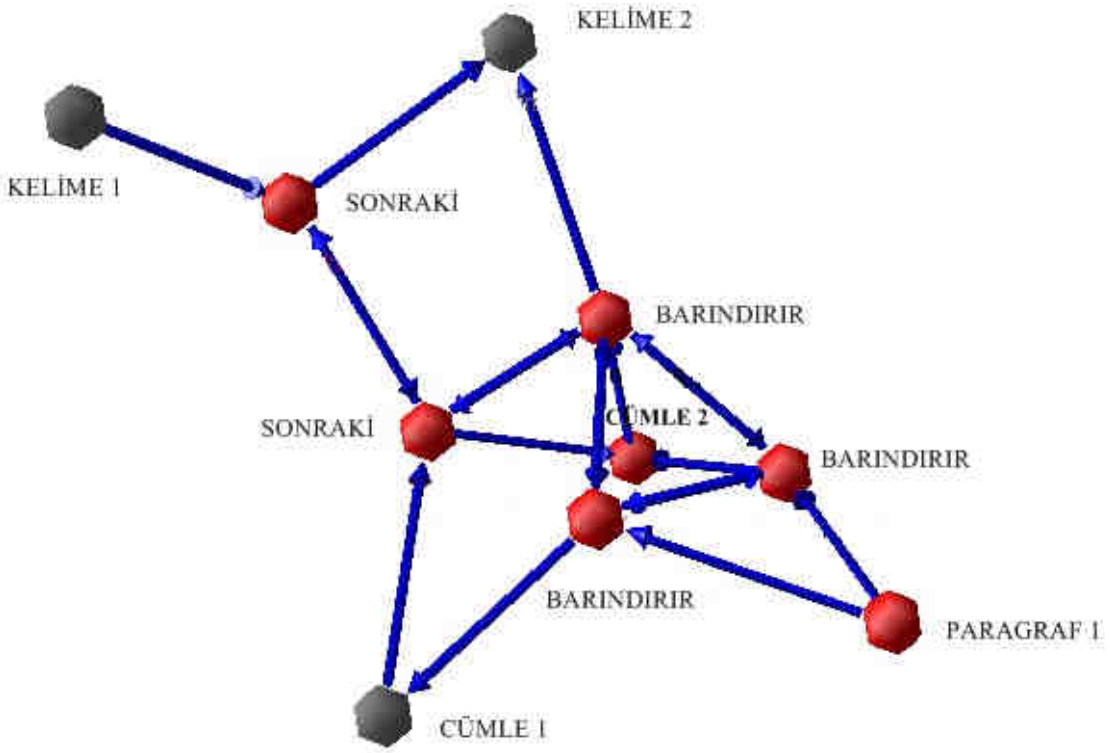
Şekil 4.4'de gösterilen çizge şeklinden, veri kümelerindeki bağlantıların belirli noktalarda yoğunlaştığı veya seyrekleştiği anlaşılabilir. Bunun sebebi elimizdeki veri kümelerinin çoğunluğunun birbirine çok benzemesidir. Veri kümeleri olarak kelime, biçimbilimsel ve sözdizimsel öğeler gösterilebilir.

Çalışmada, düğümlerin ve bağlantıların belirlenmesi ile tasarlanan bir çizge modeli Şekil 4.5'de gösterilmektedir. Tasarlanan çizgenin özelliklerinden sadece 'Barındırır' özelliği bu şekil üzerinde gösterilmiştir. 'Barındırır' özelliği dışında model 'Sonraki' ve 'Özellik' öğelerini barındırmaktadır. Bu öğeler sırasıyla bir cümlenin kelimeleri barındırması, kelimelerden sonra başka kelimelerin gelmesi ve "bir kelimenin sözdizimsel özelliği cümlenin nesnesidir" şeklinde modelde soyutlanmaktadır. Şekil 4.6'da modelin 'Sonraki' özelliğini de barındıran şekli gösterilmektedir. Şekillerin daha anlaşılır olması için 'Sonraki' ve 'Barındırır' düğümleri birden fazla kez yaratılmıştır. Uygulamada 'Sonraki', "Sarındırır" ve 'Özellik' düğümleri birer tanedir.



Şekil 4.5 Cümle, Kelime ve Paragraf ilişkilerini içeren basit model görüntüsü

Şekil 4.5'de 'Barındırır' ilişkisi Paragraf ile Cümle, Kelime ile Biçimbilimsel analiz ve Cümle ile Kelime arasındadır.



Şekil 4.6 Öğelerin arasındaki iki farklı ilişkinin model görüntüsü

Şekil 4.6'da 'Paragraf 1' in 'Cümle l' i barındırdığı, 'Cümle l' den sonra yine 'Paragraf 1' in barındırdığı 'Cümle 2'nin geldiği 'Sonraki' ve 'Barındırır' özelliklerinden ve 'Kelime 1' den sonra 'Kelime 2'nin geldiği 'Sonraki' özelliği ile bağlantılı olmasından anlaşılmaktadır.

4.3 Verinin Çıkarılması

Çizgesel yapıda tutulan veri çok fazla bilgi içerdiğinden çizge eşleme algoritması kullanılarak verinin sadece istenilen kısmı, uygulamada işlenmek için çıkarılmıştır. İstenen verinin elde edilmesi bu kısımda anlatılmaktadır. Buna göre, verinin yoğunlaştırılarak elde edilmesi işlemi üç aşamayı kapsamaktadır. Bunlar "Düğümlerin Olasılıklarının Güncellenmesi", "Bağlantı Çıkarsaması" ve "Düğümlerin Aranması". Bu işlemler alt bölümlerde sırasıyla açıklanmaktadır.

4.3.1 Düğümlerin olasılıklarının güncellenmesi

Tüm düğümler ve düğüm bağlantılarının olasılık değerlerinin hesaplanması işlemi kısaca aşağıdaki formüllerle yapılmaktadır.

Düğüm olasılığının hesaplanması:

(4.1)

$$p(dugum_x) = \frac{Sayı[dugum_x]}{\sum_i^{Dugumler \in duzey\_dugum_i} Sayı[dugum_i]}$$

Bağlantı olasılığının hesaplanması:

$$p(baglanti_i, dugum_x) = \frac{Sayı[baglanti_i]}{\sum_j^{Baglantilar_x \in tip\_baglanti_j} Sayı[baglanti_j]} \quad (4.2)$$

Tüm olasılıkların hesaplanması:

$$\{\forall\, dugum_x \in Uzay:\ p(dugum_x) \wedge \forall baglantı_i \in dugum_x:\ p(baglanti_i, dugum_x)\} \quad (4.3)$$

4.1, 4.2 ve 4.3'deki hesapların yapılabilmesi için her düzeydeki toplam düğümlerin sayısı, farklı öğelerle belirtilen düğümlerin her birinin sayısı ile birlikte her düğümün sahip olduğu bağlantıların tipinin sayısı bilinmelidir. Genel olarak 2 tip temel bağlantı vardır. Ayrıca bizim uygulamamız için 3 tip ek bağlantı vardır. Bunlar aşağıda belirtilmiştir.

Temel Bağlantı Tipleri: *Düğüme Gelen, Düğümden Çıkan*

Uygulama İçin Bağlantı Tipleri: *Sözdizimsel, Biçimbilimsel, ve Kelime*

Düğümlerin bulunduğu düzeyler: *Sözdizimsel, Biçimbilimsel* ve *Kelime* olmak üzere üç tiptedir.

Düğümlerin ve bağlantıların olasılık hesaplarında kısaca bir düğümün hangi düzeyde olduğu, bu düğümün bağlantılarının hangi düzeye gittiği veya hangi düzeyden geldiği belirleyici olmaktadır. Bu yaklaşım olasılıkların hesaplanması açısından daha güvenilirdir. Çünkü düzeyler arasında düğüm sayıları ve bağlantı sayıları açısından büyük farklar vardır.

Olasılıkların güncellenmesi işleminde tüm olasılıklar hesaplandıktan sonra, tüm bağlantılar için Kneser-Ney algoritması uygulanarak yeni olasılıklar hesaplanmıştır. Bu hesaplamada iki olasılık göz önüne alınmıştır. Birincisi, bir bağlantının olasılığı (4.1.4.2 ve 4.3'de hesaplandığı gibi), ikincisi bağlantılı öğelerin ya da düğümlerin olasılığıdır. Bu iki olasılık Kneser-Ney algoritmasına girdi olarak alınmış ve sonuçta bağlantı olasılığı bulunmuştur. Kneser-Ney algoritmasında eksiltme (veya arttırma) değeri olan δ için 4 sayı değeri alınmıştır. Çalışmada bu algoritmadaki δ değerinin 4 seçilmesi tamamen tesadüfîdir. Fakat en uygun δ değerinin saptanması için çeşitli çalışmalar mevcuttur (Chen & Goodman, 1998).

4.3.2 Bağlantı çıkarsaması

Çizge teorisindeki çıkarsama yöntemlerinden bir tanesi çizge renklendirmesidir (Nicolas & Coste, 1997). Çizge renklendirmesi düğümleri bağlantılarına göre gruplandırmakta kullanılır. Bunun dışında Kafes ya da *Trellis* metotları, *Max-Sum* algoritması çıkarsamada kullanılabilecek yöntemlerdir (Cormen, Leiserson, Rivest, & Stein, 2001) (Bishop, 2006).

Bu çalışmadaki amaç olmayan bağlantıları yaratarak veri seyrekliğini azaltmaktır. Bu şekilde bir bağlantı çıkarsaması yapmak için aşağıdaki adımlar izlenmiştir.

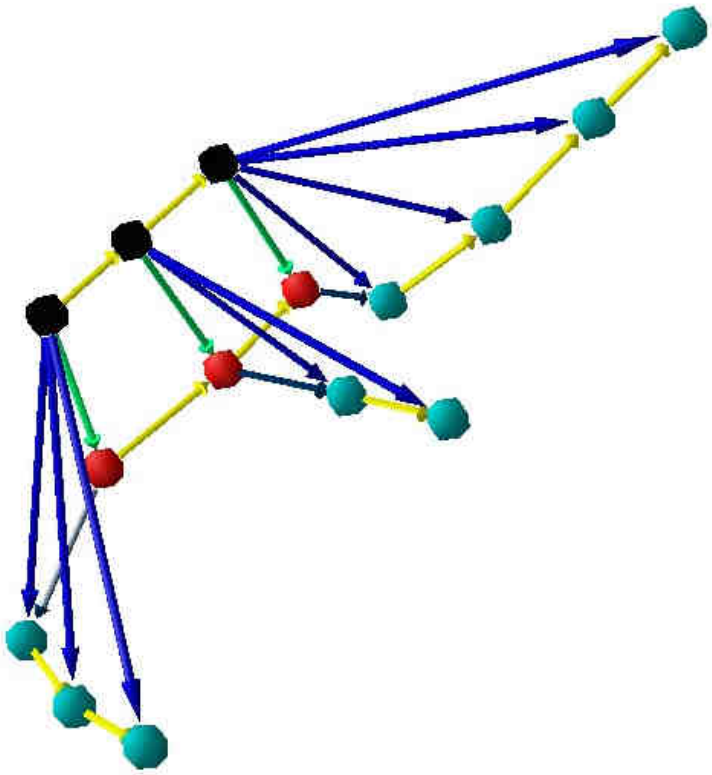
1. Aşağıdaki faktörlere göre düğümler kümelere ayrıldı.
    - Düğümün bulunduğu düzey: kelime, biçimbilimsel, sözdizimsel
    - Düğümün bağlantılarının tipleri: Temel Bağlantı Tipleri (Düğüme Gelen, Düğümden Çıkan) ve Uygulama Bağlantı Tipleri (kelime, biçimbilimsel, sözdizimsel)
    - Bağlantı olasılıkları.

2. Her bir kümenin sahip olduğu düğümlerin bağlantılarının olasılığı tiplerine göre büyükten küçüğe sıralandı. Her tip bağlantının olasılığı büyükten küçüğe doğru o bağlantıya sahip olmayan düğümlere bağlantı olarak eklendi.
3. Yeni eklenen bağlantılar küme içerisindeki ortalama olasılıklarına atandı.

4.3.3 Düğümlerin aranması

Elimizdeki çizge yapacağımız uygulama için çok fazla bilgi taşıdığından gereksiz karmaşıklığı önlemek için ve gerektiğinde daha özel testler yapabilmemiz için verinin bir kısmı çizge eşleme algoritması kullanılarak çıkartılmıştır. Kullanılabilecek eşleme tekniklerinin çizgenin düğümlerine bakmadan çizgenin şekilsel olarak eşlenmesi, çizgenin hem şeklinin hem de düğümlerinin barındırdığı verilerin eşlenmesi ve olasılık tabanlı eşleme yapabilecek kapasitede olması gerektiği ileride uygulamanın geliştirilmesi açısından yararlı olacağı düşünülmüştür. Bu bağlamda Subdue algoritması basit ve yukarıdaki özellikleri barındırdığı için tercih edilmiştir. Aşağıda çizge eşleme için gereken adımlar belirtilmektedir.

1) Aramak istediğimiz çizgesel yapı tasarlandı. Şekil 4.7"de tasarlanan çizge yapısı gösterilmektedir. Bu yapıda sarı renkte olan çizgiler sonraki özelliğinde olan bağlantıları, mavi renkte olan çizgiler sözdizimselden biçimbilimsele olan bağlantıları, koyu gri renkte olan çizgiler ise kelimeden biçimbilimsele olan bağlantıları, yeşil renkte olan bağlantılar ise sözdizimselden kelimeye olan bağlantıları göstermektedir. Siyah renkli düğümler sözdizimsel, yeşil renkli düğümler biçimbilimsel ve kırmızı renkli düğümler kelime öğelerini simgelemektedir. Bu öğelerin arasındaki tüm bağlantılar şartlı olasılık özelliğini belirtmektedir. Bu durumda tüm modele bakıldığında, sözdizimsel öğeler gözlemlenen biçimbilimsel ve kelime öğelerine şartlı bağlıdır.



Şekil 4.7 Tüm çizgede arama yapacağımız çizge modeli

2) Tüm çizge içerisinde, arama algoritması kullanılarak eşleme yapıldı. Kullanılan *arama* algoritması "Subdue"[4] olarak bilinen çizge eşleme algoritmasıdır. Subdue algoritmasında düğümlerin değerlerinin birebir eşlemesi yerine düğümlerin düzeylerinin birebir eşlemesi yapıldı.

Eşleme sonucunda Şekil 4.7 dekine benzer birden fazla (benzersiz) çizge sonucu elde ettik. Elde edilen bu çizge ile tüm verinin sadece Biçimbilimsel, Sözdizimsel ve Kelime düzeyleri için Sonraki ve Özellik bağlantı tiplerini barındıran düğümler ve bu düğümlerin bağlantıları alınmış oldu.

4.4 Verinin İşlenmesi

Oluşturulan veri kümesinde sözdizimsel etiketlemenin yapılabilmesi için Saklı Markov Model teorisinden yararlanılmıştır (bkz. Bölüm 3). Saklı Markov Teorisi

[4] Subdue algoritması ve C kodu http:\\www.subdue.org adresinde bulunmaktadır.

kullanılarak doğal dil üzerine yapılan birçok çalışma mevcuttur. En başta konuşma tanıma üzerine yapılan çalışmalar gelmektedir (Riis., 1998). Doğal dil üzerine yapılan çalışmalar ise daha çok biçimbilimsel öğelerin ve/veya sözcük türünün saptanması üzerinde durmaktadır.

Araştırmacılar kelimelerin biçimbilimsel yada sözdizimsel etiketlenmesinde Saklı Markov Modellerin kural tabanlı olanlar kadar başarı elde etmesi için ek çalışmalar yapmışlardır (Manning & Schutze, 1999). Bu konuda yapılan çalışmalarda Saklı Markov Modellerinin eğitilmesi gerektiğini öne süren ilk Jelinek adlı araştırmacı bir eğitim metodu geliştirmiştir (Jelinek, 1985).

Saklı Markov Modellerinde maksimum olasılıkları bulmak için Viterbi algoritması geliştirilmiştir. Bu algoritma bir tablo üzerinde dinamik programlama prensibi ile işlediğinden kafes algoritmalarından (*Trellis Algorithms*) daha hızlıdır. Ancak kafes algoritmaları gibi tüm sonucu vermemektedir. Birçok etiketleme işlemi Viterbi algoritmasını kullanmaktadır (Merialdo, 1994).

Uygulamada sözdizimsel öğelerin bulunması amaçlanmıştır. Bir önceki bölümde bahsettiğimiz Şekil 4.7 ile gösterilen çizge modelinde bu bilgi sözdizimsel öğeler ile kelime öğeleri arasındaki bağlantılar ile belirtilmiştir. Bu bağlantıların olasılıklarının güncellenmesi ve model üzerindeki yerlerinin elde edilmesi, verilerin çıkarılması kısmında (bkz. Bölüm 4.2) anlatılmıştır. Uygulamada sadece sözdizimsel ve kelime modeli arasındaki bağlantılar kullanıldığından işlem etiketleme işlemine çok benzemektedir.

Bilinmeyen kelimeler her türlü bağlantı tabanlı model için sorun teşkil etmektedir. Bilinmeyen ya da çok seyrek kelimelerin işlemleri aksatmaması için bir yumuşatma metodu olan Kneser-Ney algoritması seçilmiştir. Bilinmeyen bir kelime için ortalama kelime sayısı alınmıştır. Bu 4.4'deki formül ile hesaplanmaktadır.

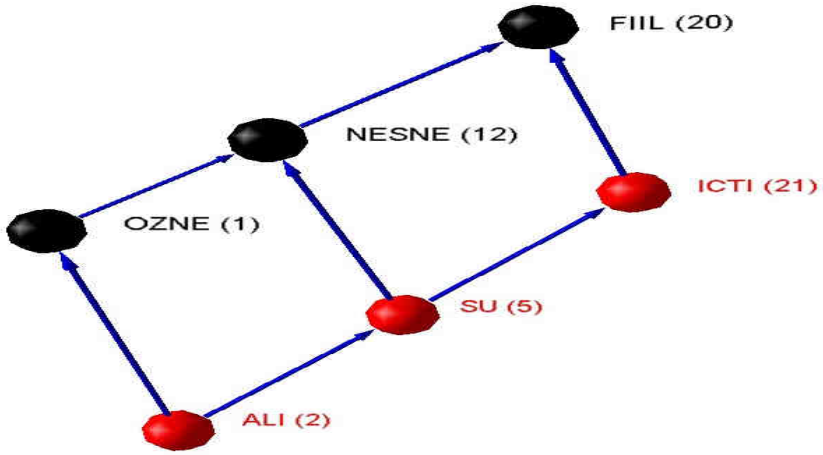
(4.4)

$$ortalama = \frac{\sum_{kelime_i}^{tüm\_kelimeler} sayı(kelime_i)}{kelime\_sayısı}$$

Yukarıdaki formüle göre ortalama kelime sayısı tüm kelimelerin sayısının (her bir kelime sayısının toplamının) benzersiz kelime sayısına bölümü ile çıkarılır. Verinin işlenmesi ile ilgili yapılan tüm hesaplamalar aşağıda adımlar halinde belirtilmektedir.

1. Bilinmeyen kelimelerin sayısı için çizge yapısına yalancı bir kelime (pseudo) olan "kelime" değeri bir düğüm olarak eklendi. Bu düğüm ile diğer tüm kelime düğümleri sonraki bağlantı tipiyle ilişkilendirildi.

2. Hızı arttırmak için derlemde bulunan tüm kelimeler indeksleri kullanılarak çizge teorisindeki gibi komşuluk matrisleri oluşturuldu. Örneğin, Şekil 4.8'deki çizge için Tablo 4.1'de komşuluk matrisi verilmiştir.



Şekil 4.8 Örnek Çizge

		1	**2**	**...**	**5**	**...**	**12**	**...**	**20**	**21**
	1	0.0	0.0	**...**	0.0	**...**	**0.7**	**...**	0.0	0.0
	2	**0.5**	0.0	**...**	**0.6**	**...**	0.0	**...**	0.0	0.0
	...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
	5	0.0	0.0	**...**	0.0	**...**	**0.8**	**...**	0.0	**0.9**
	...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
	12	0.0	0.0	**...**	0.0	**...**	0.0	**...**	**0.7**	0.0
	...	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**	**...**
	20	0.0	0.0	**...**	0.0	**...**	0.0	**...**	0.0	0.0
	21	0.0	0.0	**...**	0.0	**...**	0.0	**...**	**0.9**	0.0

Tablo 4.1 Çizge Komşuluk Matrisi

Şekil 4.8 ile gösterilen örnek çizgede düğümlerin indeks değerleri belirtilmiştir. Bu değerlerin matris formunda satır veya sütun indeks numaralarına karşılık gelecek şekilde gösterilmesiyle Tablo 4.1' deki komşuluk matrisi oluşmaktadır. Bu matris formunda bağlantıların olasılık değerleri değer olarak bağlantının olduğu düğümlerin indekslerine karşılık gelen hücrelere atanmıştır. Bu matris formu özellik ve sonraki bağlantı tiplerini barındırmakla birlikte tüm düğümleri kapsamaktadır. Büyük bir derlem için bu matris formu çok büyük, olasılık değerleri ise çok küçük olmaktadır.

3. Veri girişinden gelen kelimeler, çizgede aranarak kelimeleri temsil eden düğümlerin indeks değerleri döndürüldü. Eğer kelime bilinmiyorsa yani çizgede yoksa yalancı kelimenin ("kelime") indeks değeri döndürüldü.
4. İndeks değerleri kullanılarak 4.5'de adımları verilen Viterbi algoritması ile etiketleme işlemi tamamlandı.

$$kelime^{i} \in Girilen\ Tum\ Kelimeler:$$
$$etiket^{j} \in Matristeki\ tüm\ etiket\ indeksleri:$$

$$\delta_{i+1}(etiket^{j}) = maksimum_{1\le k\le Tüm\ Kelime\ İndeksleri}\ [\![\delta_i(etiket^{k}) \times P(etiket^{j}|etiket^{k}) \times P(kelime_{i+1}|etiket^{j})\ ]\!]$$

$$\varphi_{i+1}(etiket^{j}) = argmaks_{1\le k\le Tüm\ Kelime\ İndeksleri}[\![\delta_i(etiket^{k}) \times P(etiket^{j}|etiket^{k}) \times P(kelime_{i+1}|etiket^{j})\ ]\!]$$

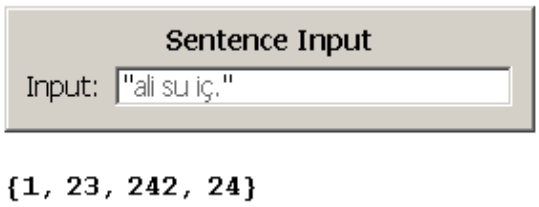
$$etiketler_{grilen\ kelime\ sayısı+1} = argmax_{1\le j\le tüm\ etiket\ sayısı}[\![\delta_{grilen\ kelime\ sayısı+1}(etiket^{j})]\!]$$
$$\{tüm\ etiketler\ için: etiketler_j = \varphi_{j+1}(etiketler_{j+1})\}$$

$$cümle\ olasılığı = P(kelime_1, kelime_2, ..) = maksimum_{1\le j\le tüm\ etiket\ sayısı}[\![\delta_{n+1}(etiket^{j})]\!]$$

Bu algoritma ile her bir kelime için, kelimeden önce gelen tüm etiketlere ve o kelimeye bakılarak, maksimum olasılıklı etiket bulunmuş oldu. Etiketleme işlemi için kelimelerin birbirinden bağımsız olduğu, ve bir kelimenin aldığı sözdizimsel etiket ile ilişkisi olduğu kabul edildi. Dolayısıyla, koşullu olasılık durumunun hesabı bizim çizge modelimizdeki kelime ve sözdizimsel özelliklerin bağlantı olasılık değerleri ile aynı oldu.

Kullanıcıdan istenen cümlenin etiketlerinin bulunması sırasında girilen bir cümle için anlatılan adımlardan sadece üçüncü ve dördüncü adımlar çalışmaktadır. Birinci ve ikinci adımlar ilk başta program çalıştığı zaman yapıldığından çalışma hızı kullanım anında girilen tüm cümleler için makul olmaktadır.

Şekil 4.9'da girilen bir örneğin sonuçları gösterilmiştir. Bu örnekte sırasıyla girilen bir cümlenin indekslerinin bulunması ve kelimelere atanabilecek en uygun etiketin bulunması şekillerle gösterilmektedir. Oluşan ara tablolarda ise kelime sırası sütun indeksleri ile ve etiketler ise satır indeksleri ile belirtilmektedir.



Şekil 4.9 Cümle giriş ekranı ve kelimelerin indeks karşılığı

Şekil 4.9'da girilen bir cümle ve bu cümledeki kelimelere karşılık gelen çizge düğümlerinin indeks numaraları gösterilmektedir. Şekil 4.5 ve 4.6'da ise Viterbi algoritmasında kullanılan (en uygun sözdizimsel etiketin bulunmasında kullanılan) δ ve φ tablosu gösterilmektedir.

0.000332336	0	0	0	0
0.289465	0	0.00862577	0	3.4491×10^{-6}
0.102692	0	0	0	0.0000137964
0.0249252	0	0	0	0
0.17215	0.0170435	0	0	3.4491×10^{-6}
0.0179462	0	0	0.000589797	0
0.168162	0.0148444	0	0	6.89821×10^{-6}
0.104021	0	0	0	0
0.0156198	0	0	0	0
0.0232635	0	0	0	0
0.0136258	0	0	0	0
0.0242606	0	0	0	3.4491×10^{-6}
0.00598205	0	0	0	0
0.00531738	0	0	0	0
0.00465271	0	0	0	0
0.0169492	0	0	0	0
0.00631439	0	0	0	0
0.00232635	0	0	0	0
0.00199402	0	0	0	0

Şekil 4.10 $\delta\ tablosu$ (matrisi)

0	1	1	1	1
0	2	7	2	6
0	3	3	3	6
0	4	4	4	4
0	7	5	5	6
0	6	6	2	6
0	2	7	7	6
0	8	8	8	8
0	9	9	9	9
0	10	10	10	10
0	11	11	11	11
0	12	12	12	6
0	13	13	13	13
0	14	14	14	14
0	15	15	15	15
0	16	16	16	16
0	17	17	17	17
0	18	18	18	18
0	19	19	19	19

Şekil 4.11 $\varphi\ tablosu$ (matrisi)

Şekil 4.10 ve Şekil 4.11 ile belirtilen tablolarda her bir satır indeksi bir etiketi ve her bir kolon indeksi ise bir kelimeyi ve onun sırasını simgelemektedir. Örneğin o tablosunda birinci kelime ikinci indekse karşılık gelmekte ve 0.0170435 olasılık değeri ile beşinci etiketi almaktadır. Aşağıda toplam olasılık ve etiket indeksleri verilmiştir. Bu etiket indeksleri tablodaki indeks numaralarından çizgenin içerisindeki indeks numaralarına çevrilerek verilmiştir.

```
{0.0000137964, {30, 12, 29, 13}}
```

Şekil 4.12 Etiketlemenin indeks çıktısı

Sonuç olarak bulunan etiketlerin çizgedeki indeks numaraları düğümlerin karakter katarı değerlerine dönüştürüldüğünde bu cümle için sonuç Şekil 4.13'deki gibi olmaktadır.

4.5 Testler ve Sonuçları

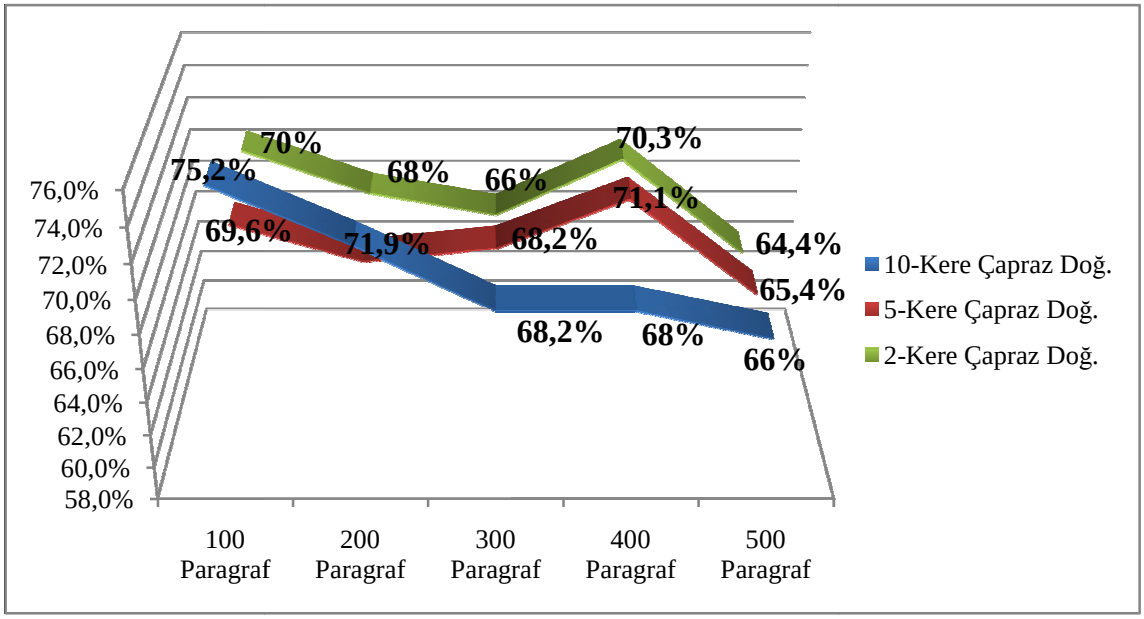
Sistemlerin performansı değerlendirilirken genelde bütünsel oranlı doğruluk (recall), kısmi oranlı doğruluk (precision) ve F-ölçümü (F-Measure) kriterleri kullanılmaktadır (Manning & Schutze, 1999). Ancak bu çalışmada performans ölçümü için en uygun olan doğruluk (accuracy) bir kriter olarak seçilmiştir. Buna göre, doğruluk

(4.14)

$$doğruluk = \frac{Doğru\_Etiketleme\_Sayısı}{Yapılan\_Tüm\_Etiketleme\_Sayısı}$$

olacak şekilde tanımlanmaktadır. Sistemin başarısı 4.14'de belirtildiği gibi doğru etiketlemelerin yapılan tüm etiketlemelere olan oranı olarak belirlenmiştir.

Yapılan tüm uygulamanın olası eğilimlerini yani özel durumlara göre başarının değişimini minimum tutmak için N-Kere Çapraz Doğrulama algoritması kullanılarak test yapılmıştır. N-Kere Çapraz Doğrulama algoritması ile test verisi başta N tane kümeye ayrılmaktadır. Her küme verisi için geri kalan kümelerdeki veriler kullanılarak eğitim yapılmış bir öğrenme modelinin başarısı ölçülmektedir. Genel başarı yapılan tüm testlerin başarılarının ortalaması olarak alınır. Bu algoritma ile eğitim gerektiren tüm öğrenme modelleri için daha güvenilir bir başarı sonucu elde edilmektedir. N-Kere Çapraz Doğrulama algoritmasında seçilen küme sayısına ve örnek sayısına göre yapılan testlerin sonuçlan Çizelge 4.1 'de gösterilmektedir.



Çizelge 4.1 Doğrulama sayısı ve kullanılan veri büyüklüklerine göre yapılan çapraz doğrulama testlerinin sonuçları

Çizelge 4.1'de sırasıyla 100, 200, 300, 400 ve 500 adet paragraf üzerinde yapılan çeşitli testlerin sonuçlarının başarı oranlan verilmektedir. Her bir veri kümesi için değişik sayıda çapraz doğrulama yapan test yapılmıştır. Çapraz doğrulama sayılarına göre yapılan testler mavi, kırmızı ve yeşil renklerdeki çizgilerle gösterilmektedir.

Paragraf sayısının artması çizgede tutulan cümle sayısını arttırmaktadır. Artan cümle sayısı ile eğitimde kullanılan veri sayısı arttığı için sonuçlar daha güvenilir olmaktadır. Ayrıca N-Kere Çapraz Doğrulama algoritmasının doğası gereği çapraz doğrulama sayısı arttıkça eğitim için kullanılan veri miktarı artmaktadır. Eğitim için kullanılan veri sayısı ile başarı doğru orantılı olduğundan artan çapraz doğrulama sayısı ile başarının arttığı söylenebilir.

Yapılan tüm testlerde en tutarlı sonuçlar 10-Kere Çapraz Doğrulama algoritması ile elde edilmiştir. Çünkü 10-Kere Çapraz Doğrulama algoritması ile değişik sayıdaki paragraflar üzerinde yapılan testlerde sonuçlarda değişim çok az olmaktadır. Buna göre 500 paragraftık veri kümesinin 10-Kere Çapraz Doğrulama algoritması ile yapılan testin sonucu (%66,5) en tutarlı sonuç olarak kabul edilebilir.

Bilindiği kadarıyla Türkçe üzerine yapılan başka sözdizimsel etiketleme uygulamaları olmadığı için yapılan sözdizimsel etiketlemede elde edilen %66,5'lik doğruluk değerinin bu uygulama için bir alt sınır olacağı kabul edilebilir. Bundan sonraki çalışmalar bu başarı oranım arttırmaya yönelik olmalıdır.

BÖLÜM 5

SONUÇ

Çalışmada, Viterbi algoritması. Subdue çizge eşleme algoritması ve Kneser-Ney yumuşatma algoritması kullanılarak bir sözdizimsel etiketleme modeli oluşturulmuştur. Tasarlanan çizge yapısı ile bilginin tutulması sağlanmış, Kneser-Ney yumuşatma algoritması ile bu çizge yapısındaki bağlantıların olasılık değerleri güncellenmiş ve Subdue eşleme algoritması ile bu modelden belirlenmiş bir modele göre örnek çizgeler üretilmiştir. Örnek çizgeler üzerinde Saklı Markov Modellerinde kullanılan Viterbi algoritması ile örnek çizgenin olasılığını maksimum yapan en uygun sözdizimsel etiket düğümleri bulunmuştur. Yüksek Boyutlu Gömme çizge gösterim tekniği kullanılarak tüm çizge verisi görselleştirilmiştir.

Sonuç olarak yapılan bu çalışmada sözdizimsel etiketlerin bulunmasında sadece kelime düzeyinden gelen bilginin yeterli olmadığı saptanmıştır. Kneser-Ney yumuşatma algoritması bile kelime düzeyindeki seyreklikleri gidememiştir. Ancak tüm çalışma veriyi bir model üzerinde tuttuğundan ileride yapılacak biçimbilimsel, sözdizimsel, semantik etiketleme çalışmaları için bir alt yapı sunmaktadır.

Görselleştirme işleminde kullanılan Yüksek Boyutlu Gömme tekniği ile görselleştirilen çizge modelinde ilk olarak çizgedeki düğümler arasındaki farklılıkların az olduğu gözlemlenmiştir. Bu veri kümesinin büyük bir bölümünün birbirine çok yakın bir alanda toplanmasından anlaşılmaktadır.

Bu mimari ile ileride yapılacak çalışmalar için ilk amaçlanan biçimbilimsel düzeyde ve kelime düzeyindeki bilginin birlikte kullanılarak başarının artırılmasıdır. Daha sonraki çalışmalar, daha özel durumlar için olacaktır. Örneğin fiillerin alt öğeleme listelerinin[5] (*Subcategorization Frames*) bulunması (Kilicaslan, Uzun, Agun, & Ucar, 2007) ve cümleler üzerinde tematik rollerin (*Thematic Roles*) etiketlenmesi ileride

[5] Türkçede bir fiil yanma belli sayıda kelime ve bu kelimelere bağlı hal ekleri almaktadır. Bu ekler o fiilin alt öğeleme listesi olarak kabul edilmektedir. Türkçe için alt öğeleme listesinin bulunması diğer dillere göre çok daha zordur.

yapılması planlanan çalışmalardır. Doğal dil işleme çalışmalarından farklı olarak bilgi erişim sistemlerinde (*Information Retrieval Systems*) başarının artırılması için bu mimari kullanılabilecektir. Tasarlanan yapının ilerideki çalışmalarda kullanılabilecek öğelere sahip olması (kavram öğrenme, çizge eşleme, olasılık tabanlı etiketleme) sunulan tezin ileriye dönük bir çalışma olduğunun bir göstergesidir.

KAYNAKLAR

Anderson, J. R., Michalski, R. S., Carbonell, J. G., & Mitchell, T. M. (1985). *Machine Learning: An Artificial Intelligence Approach* (Cilt 1). Morgan Kaufmann.

Bangalore, S., & Joshi, A. K. (1994). Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. *In Proceedings of the International Conference on Computational Linguistics.* Kyoto, Japan.

Bangalore, S., & Joshi, A. K. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics , 20* (3), 331-378.

Berg, J. v., & Schuemie, M. J. (1999). Information Retrieval Systems using an Associative Conceptual Space. *Proceedings of the ESANN'99.* D-Facto Publishing.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning* (2nd b.). Springer.

Boz, K. O. (1994). Turkish Natural Language Processing Initiative: An overview. *Proceedings of the Third Turkish Symposium on Artifical.* Ankara, Turkey: Middle East Technical University.

Chen, S., & Goodman, J. (1998). *An empirical study of smoothing techniques for language modeling.* Technical Report TR–10–98, Harward University, Center for Research in Computing Technology.

Clark, S., & Curran, J. R. (2004). The importance of supertagging for wide-coverage ccg parsing. *Proceedings of the International Conference on Computational Linguistics (COLING).* Geneva, Switzerland.

Cook, D. J., & Holder, L. B. (2000). Graph-Based Data Mining. *IEEE Intelligent Systems* , 32-41.

Cook, L. H. (2002). Concept formation using graph grammars. *KDD Workshop on Multi-Relational Data Mining.*

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2001). *Introduction to Algorithms.* MIT Press.

Cutting, D., Kupiec, J., Pedersen, J., & Sibun., P. (1992). A Practical part-of-speech tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*, (s. 133-140). Trento, Italy.

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society , Series B* (39(1)), 1–38.

Di Battista, G., Eades, P., Tamassia, R., & G. Tollis, I. (1998). *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice Hall.

Eryigit, G., & Oflazer, K. (2006). Statistical Dependency Parsing for Turkish. *EACL*.

Gardenfors, P. (2000). *Conceptual Spaces: The Geometry of Thought.* Cambridge: MIT Press.

Ge, N., Hale, J., & Charniak, E. (1998). A statistical approach to anaphora resolution. *Proceedings of the Sixth Workshop on Very Large Corpora.*

Ickjai Lee Portier, B. (2007). An Empirical Study of Knowledge Representation and Learning within Conceptual Spaces for Intelligent Agents. *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference* (s. 463-468). Melbourne, Qld.: IEEE.

Jelinek, F. (1985). Markov source modeling of text generation. *The Impact of Processing Rechniques on Communication* , 569-598.

Jelinek, F., & Mercer, R. (1980). Interpolated estimation of markov source parameters from sparse data. (E. Gelsema, & L. Kanal, Dü) *Pattern Recognition in Practice* , 381-397.

Johnson, C. S. (1967). Hierarchical Clustering Schemes. *Psychometrika* , *2*, 241-254.

Kilicaslan, Y., Uzun, E., Agun, V., & Ucar, E. (2007). Automatic Acquisition of Subcategorization Frames for Turkish with Purely Statistical Methods. *International Symposium on Innovations in Intelligent Systems and Applications.* Istanbul.

Koren, H. D. (2002). Graph Drawing by High Dimensional Embedding. *Proceedings of 10th International Symp. Graph Drawing*, (s. 207-217).

Lawrence, B. H., & Diane, J. C. (1993). Discovery of inexact concepts from structural data. *Transcations on Knowledge and Data Engineering.* IEEE .

MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability. 1*, pp. 281-297. Berkeley: University of California Press.

Manning, C., & Schutze, H. (1999). *Foundations of Statistical Natural Language Processing.* Massachusetts: MIT Press.

Manning, C., & Schutze, H. (1999). *Foundations of Statistical Natural Language Processing.* Massachusetts: MIT Press.

Manning, C., & Schutze, H. (1999). *Foundations of Statistical Natural Language Processing.* Massachusetts: MIT Press.

Merialdo, B. (1994). Tagging English Text with a Probabilistic Model. *Computational Linguistics , 20* (2), 155-171.

Neuhaus, M., & Bunke, H. (2006). Edit distance-based kernel functions for structural pattern classification. *Pattern Recogn. , 39* (10), 1852-1863.

Nicolas, F., & Coste, J. (1997). Regular inference as a graph coloring problem. *Workshop on Grammatical Inference, Automata Induction, and Language Acquisition.* Nashville: ICML.

Nivre, J., Hall., J., Nilsson, J., Eryiğit, G., & Marinov, S. (2006). Labeled Pseudo-Projective Dependency Parsing with Support Vector Machines. *Proceeding of the 10th Conference on Computational Natural Language Learning.* NewYork, USA.

Nuutila, E., & Törmä, S. (2003). Introduction to Text Graphs. *Proceedings of the Third Finnish/Baltic Sea Conference on Computer Science Education.* Koli, Finland: University of Helsinki Department of Computer Science.

Oflazer, K. (1994). Two-level description of Turkish morphology. *Literary , 9* (2).

Oflazer, K., & Kuruoz, I. (1994). Tagging and Morphological Disambiguation of Turkish Text. *Proceedings of the Fourth Conference on Applied Natural Language Processing* (s. 144-149). Stuttgart: ACL.

Palmer, C., Gibbons, P., & Faloutsos, C. (2002). Data mining on large graphs. *Proceddings of International Conference on SIGKDD*, (s. 81-90).

Riis., S. K. (1998). *Hidden Markov Models and Neural Networks for Speech Recognition.* Denmark: PhD thesis, Technical University of Denmark.

Segond, F., Schiller, A., Grefenstette, G., & Chanod, J. (1997). An experiment in semantic tagging using hidden markov model tagging. *Proceedings of the ACL/EACL Workshop on Automatic Information Extraction and Building of Lexical Semantic Resources* (s. 78--81). New Brunswick, New Jersey: Association for Computational Linguistics.

Ullmann, J. R. (1976). An Algorithm for Subgraph Isomorphism. *J. ACM* , 31-42.

Yarowsky, D. (1995). Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, (s. 189-196). Cambridge.

EK-1

ÇİZGE GÖSTERİM FORMATI

Bu bölümde GraphML çizge gösterim formatı açıklanmaktadır. Çeşitli çizge gösterim formatları arasından en çok kullanılanı GraphML formatıdır. Her çizge gösterim formatı gibi GraphML de düğüm ve bağlantı etiketlerini, renklerini, tiplerini ve bunun gibi verileri saklamada bize tanımlı bir format sunar. Bir gösterim formatı kullanmanın amacı başka programların elimizdeki çizge verisini göstermesini sağlamaktır. Aşağıda GraphML formatında bir çizge örneği verilmiştir.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns
     http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="undirected">
    <node id="n0"/>
    <node id="n1"/>
    <node id="n2"/>
    <node id="n3"/>
    <edge source="n0" target="n2"/>
    <edge source="n1" target="n2"/>
    <edge source="n2" target="n3"/>
  </graph>
</graphml>
```

Tekrar eden belirli özelliklerin (örneğin renk bilgisi) değerinin veya konuya ilişkin başka özelliklerin yazılması aşağıdaki formatta olmaktadır,

```
<key id="d1" for="edge" attr.name="weight" attr.type="double"/>
```

Yukarıda bağlantılar için tanımlanmış "weight" adına sahip bir "double" tipinde özellik kümesi belirtilmiştir.

Diğer bir özellik kümesi ise aşağıda belirtilmektedir.

```
<key id="d0" for="node" attr.name="color" attr.type="string">
    <default>yellow</default>
</key>
```

Yukarıdaki özellik kümesinin adı "color" ve değer tipi "karakter katan" olarak belirlenmiş ve düğümler üzerinde kullanılabileceği belirtilmiştir. Düğümlerde renkleri göstermek için yukarıda tanımlanan özellik kümesi kullanılabilir; örneğin mavi bir düğüm yaratmak için aşağıdakiler yazılabilir.

```
<node id="n4">
       <data key="d0">blue</data>
</node>
```

Yukarıdaki bağlantıda <data key = "d0">, özellik kümesinden "d0" anahtar sözcüğü ile belirlenmiş olan özelliği nitelemekte ve ona "blue" renk değerini atamaktadır. Bağlantılara etiket yada değer atamak için <data key="d1"> ile daha önce tanımladığımız "d1" özellik kümesini bağlantıya ekleyerek ona bir değer verebiliriz. Aşağıda bu işlem için yazılması gereken bir örnek verilmiştir.

```
<edge id="e6" source="n3" target="n4">
      <data key="d1">1.1</data>
</edge>
```

Yukarıdaki örnekte "e6" anahtar sözcüğü ile tanımlanmış olan bir bağlantıya "d1" özellik kümesine ilişkin bir bilgi eklenmiştir. Bu bağlantı için "d1" özellik kümesinin değeri "1.1" olarak belirlenmiştir.