

Doğal Dil İşleme (NLP)

İçindekiler

- [SORU 1: Term Document Matrix \(Kavram Metin Masfufu\)](#)
- [SORU 2: Vektör Uzay Modeli \(Vector Space Model\)](#)
- [SORU 3: POS Tagger \(Metin Parçası Etiketleme\)](#)
- [SORU 4: TF-IDF](#)
- [SORU 5: SimHash \(Benzerlik Özeti\)](#)
- [SORU 6: n-gram](#)
- [SORU 7: Evrimsel Diller \(Evolutionary Languages\)](#)
- [SORU 8: Dilin Derecesi \(Rate of Language\)](#)
- [SORU 9: Özyineli Diller \(Recursive Languages\)](#)
- [SORU 10: Özyineli Geçiş Ağları \(Reursive Transition Networks\)](#)
- [SORU 11: Gellish \(Kontrollü Doğal Dil\)](#)
- [SORU 12: Turing Makinesi \(Turing Machine\)](#)
- [SORU 13: Özyineli Sayılabilir Diller \(Recursively Enumerable Languages\)](#)
- [SORU 14: Chomsky Hiyerarşisi \(Chomsky Hierarchy \)](#)
- [SORU 15: Mana Ağları \(Sematic Webs, Anlamsal Ağ\)](#)
- [SORU 16: Muntazam Diller \(Formal Languages\)](#)
- [SORU 17: Anlamsal Bağ \(Semantic Link\)](#)
- [SORU 18: Sözdizim \(Syntax\)](#)
- [SORU 19: Anlambilimsel Tertip \(Semantic Composition\)](#)
- [SORU 20: İstatistiksel Dilbilim \(Probabilistic Linguistic\)](#)
- [SORU 21: Augmented Transition Network \(ATN, Uzatılmış Geçiş Ağı\)](#)
- [SORU 22: Definite Clause Grammer \(Belirli Cümle Dilbilgisi DCG\)](#)
- [SORU 23: Pragma \(Edimbilim, kullanımbilim, Fiili, Ameli\)](#)
- [SORU 24: Allen Fasıla Mantığı \(Allen's Interval Logic\)](#)
- [SORU 25: Haber \(Predicate\)](#)
- [SORU 26: Şekli Mantık \(Kipler Mantığı, Modal Logic\)](#)
- [SORU 27: Zamansal Mantık \(Temporal Logic\)](#)
- [SORU 28: Dilbilgisel Bakış \(Grammatical Aspect\)](#)
- [SORU 29: Kelimebilimsel Bakış \(Lexical Aspect\)](#)
- [SORU 30: Reichenbach Zaman Analizi \(Reichenbachian Tense Analysis\)](#)
- [SORU 31: Zamani \(Temporal, Zamansal, Zamane, Mevcut\)](#)
- [SORU 32: Zaman Sırası \(Sequence of Tenses\)](#)
- [SORU 33: Uyum \(Agreement, Kabul, Bağıt, Mutabakat\)](#)
- [SORU 34: İçerikten Bağımsız Gramer \(context free grammer, CFG\)](#)
- [SORU 35: İçerikten bağımsız dil \(Context Free Language, CFL\)](#)

[SORU 36: Kelime \(Lexeme\)](#)

[SORU 37: Soru Cevaplama \(Question Answering, QA\)](#)

[SORU 38: Kesinlik Zarfları \(Katı Zarflar, Sentential Prepositions\)](#)

[SORU 39: Koşaç \(Mafsal, Haber Edatı, Copula\)](#)

[SORU 40: Hazf \(Eksilti, Ellipsis\)](#)

[SORU 41: Gösterim İşlemi \(Projection Operator\)](#)

[SORU 42: Parçalama Ağacı \(Parse Tree\)](#)

[SORU 43: Türkçe için TimeML](#)

[SORU 44: Cümle Zamanları \(Tense\) ve Bakış \(aspect\)](#)

[SORU 45: Sıralama Algoritmaları \(Sorting Algorithms\)](#)

[SORU 46: TimeML](#)

[SORU 47: HTML+TIME](#)

[SORU 48: OWL Time \(OWL Zaman, Web Varlıkbilim Dili Zaman\)](#)

[SORU 49: TTML \(Time Tabling Markup Language, Zaman Çizelgeleme İşaretleme Dili\)](#)

[SORU 50: Hitabe \(Nutuk, Söylev, Discourse\)](#)

[SORU 51: Terminoloji Çıkarımı \(Terminology Extraction\)](#)

[SORU 52: Külliyyat \(corpus\)](#)

[SORU 53: Eş Atf \(Coreference\)](#)

[SORU 54: Bilgi Çıkarımı \(Information Extraction\)](#)

[SORU 55: Geçişsiz Fiiller \(intransitive verbs\)](#)

[SORU 56: Geçişli Fiiller \(transitive verbs\)](#)

[SORU 57: İkili dil \(bigram\)](#)

[SORU 58: Kelime Bilim \(lexicology, vocabulary\)](#)

[SORU 59: Ekleme \(apposition\)](#)

[SORU 60: Dönüştü \(anaphora\)](#)

[SORU 61: İlgî Belirsizliği \(reference ambiguity\)](#)

[SORU 62: Yapısal Belirsizlik \(Structural Ambiguity\)](#)

[SORU 63: Kelime-İfade Belirsizliği \(word-sense ambiguity\)](#)

[SORU 64: Belirsizlik \(ambiguity, muğlaklık, ikircimlik\)](#)

[SORU 65: Bağlaç \(conjunction\)](#)

[SORU 66: Edat \(preposition\)](#)

[SORU 67: zarf \(adverb\)](#)

[SORU 68: Sıfat \(adjective\)](#)

[SORU 69: Fiil \(verb\)](#)

[SORU 70: Somut \(müşahhas\) isim \(concrete noun\)](#)

[SORU 71: Özel isim \(hususî isim, proper noun\)](#)

SORU 72: Cins İsim (common noun)

SORU 73: Soyut İsim (mücerret, abstract noun)

SORU 74: isim (noun)

SORU 75: Sayılabilir İsimler (count noun)

SORU 76: kütle adı (mass noun)

SORU 1: Term Document Matrix (Kavram Metin Masfufu)

Basitçe seyrek bir matris üzerinde terim sayıları tutulmaktadır. İki boyutlu bir matrisin bir boyutunun metinlere diğer boyutunun da terimlere ayrıldığını düşünün. Bu matriste, her metinde o terimden kaç tane olduğunun sayısı bulunacaktır.

Basitçe aşağıdaki şekilde iki metnimiz bulunduğunu kabul edelim:

metin 1 : java bazı bilgisayar mühendisliği bölümlerinde eğitimi verilen bir programlama dilidir.

metin 2 : bazı bilgisayar mühendisliği projelerinde java programlama dili kullanılmaktadır.

Yukarıdaki iki metin için aşağıdaki şekilde bir matris olacaktır:

Terimler \ Metinler	Metin 1	Metin 2
Java	1	1
Bazı	1	1
Bilgisayar	1	1
Mühendisliği	1	1
Bölümlerinde	1	0
Eğitimi	1	0
Verilen	1	0
Bir	1	0
Programlama	1	0
Dilidir	1	0
Projelerinde	0	1
Dili	0	1
Kullanılmaktadır	0	1

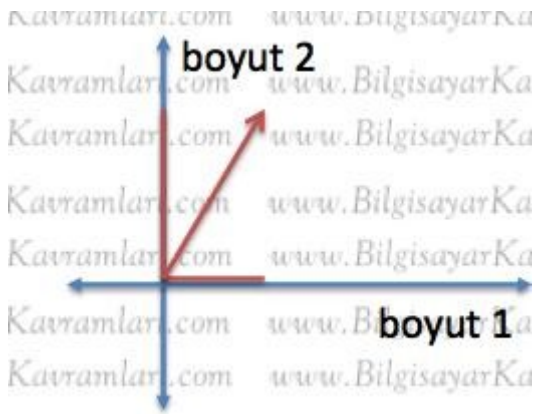
Yukarıda örnek olarak verilen bu tabloda, sadece iki metin alındığı için ve metinlerde geçen kelimeler birbirine çok yakın olduğu için sıfır sayısı fazla olmamıştır. Ancak okuyucu, metin sayısı arttıkça ve metinler arasındaki benzer kelimeler azaldıkça sıfır sayısının artacağını ve masfufun, daha seyrek olacağını görmelidir.

Yukarıdaki bu matris, metin madenciliğine temel teşkil etmekte ve bu matris üzerinde yapılan çeşitli işlemlerle metin madenciliği fonksiyonları çalıştırılabilir.

SORU 2: Vektör Uzak Modeli (Vector Space Model)

Bu modelleme yöntemine göre herhangi bir metin veya metin parçasını vektörel olarak göstermek ve bir uzay içerisinde modellemek mümkündür.

Örneğin bir yöneyi (vektör, vector) her boyuttaki değeri ile göstermemiz gerekir. Misal olarak [1,2] vektörünü iki boyutlu uzayda aşağıdaki şekilde çizilebilir:



İki boyutlu uzayda ilk boyut ve ikinci boyut yukarıdaki temsili resimde gösterilmiştir. Buna benzer şekilde bir vektörün sonsuz sayıda boyutu olabilir. [1,2,1,3,2,1,2,3,2,1,2,2,1,3] şeklinde 14 boyutlu bir vektörden bahsedebiliriz. Bu vektörün gösterimi veya çizilmesi gerçekten gerekmemektedir çünkü amaç birbiri ile ilişkisi olmayan boyutlarda farklı değerlere sahip bir vektörü temsil etmektir. Buradaki önemli nokta bu boyutların arasında ilişki bulunmamasıdır. Yukarıdaki temsili şekilde de ilk okulda öğrenilen ve birbirine dik olan eksenlerin ilişkilerinin olmaması kabulüne dayanılarak çizim yapılmıştır. Diğer bir deyişle birinci boyutun kaç olduğunun 2. boyuta veya ikinci boyutun kaç olduğunun 3. boyuta bir etkisi yoktur.

Bir metnin vektörel olarak gösterilmesi ise çok farklı şekillerde olabilir. Örneğin sık kullanılan yöntemlerden birisi terim frekansının sayılmasıdır (term frequency). Bu konuda daha önceden yayınlanan yazıya bakılarak bu işlemin hangi adımlarla yapıldığı görülebilir.

Diğer yöntemler ise bunlarla sınırlı kalmamak şartıyla, Metin Etiketleme (POS Tagging), n-gram sayımı, veya LSA (Gizli Anlambilimsel Tahlil, Latent Semantic Analysis) şeklinde sayılabilir.

Vektör uzaylarının kullanılmasının bazı avantajları, doğrusal cebir (linear algebra) kullanılarak işlenebilen veri yapılarının elde edilmesi, ikilik tabandaki sayılar yerine ağırlıkların hesaba katılabilir olması, vektörler arasında tanımlı olan bütün fonksiyonların metinler arasında da tanımlanabilir olması (örneğin kosinüs benzerliği (cosine similarity) gibi), metinler üzerinde sıralama (ranking) fonksiyonlarının çalıştırılabilir olması, metnin tamamı yerine bir parçası üzerinde çalışabilir olması şeklinde sayılabilir.

Bunun yanında bazı dez avantajlarına bakacak olursak. Genelde bu tip özellik vektörlerinin çıkarılması sonucunda çok yüksek miktarda özellik içeren veriyle uğraşmak gerekir. Örneğin imdb sinema yorumları içeren web sitesinden çektiğimiz 62,000 yorum için ingilizcedeki 160,000 civarında farklı kelimenin kullanıldığını ve bu kelimeleri tutan özellik vektörümüzün 7GB civarında veri olduğunu ve bu veriyi bilgisayarın RAM'ine yüklemek bile işlemek için yer kalmadığını ve özel yöntemler geliştirmemiz gerektiğini burada belirtebilirim.

Terim frekansının yanında anlambilimsel yaklaşımlar kullanılarak bu boyutların azaltılması mümkündür.

Vektör uzay modelinin en büyük dez avantajlarından birisi, metin boyutu uzadıkça kullanışsız hale gelmesidir. Çünkü uzayan metinler birbirine benzemeye başlar ve metinleri ayırt etmede önemli rol oynayan kelime farklılıkları azalır. Ayrıca kelimeler arasındaki benzerlikler göz ardı edilmektedir. Örneğin ağaç kelimesi ile ağaçlar kelimesi farklı iki kelimedir ve farklı metinlerde geçmesi halinde iki metin birbirinden farklı olarak sınıflandırılacaktır. Oysaki metinler arasında anlamsal bağlantılar olabilir.

Ayrıca metin içerisindeki kelimelerin sırası da vektör uzay modelinde kaybedilmektedir. Örneğin bir kelimenin nerede geçtiği önemsizdir.

SORU 3: POS Tagger (Metin Parçası Etiketleme)

Yazıya başlamadan önce belirtmek isterim ki ne yazık ki Türkçede tam başarılı kodlanmış, eğitilmiş ve çalışan bir etiketleme yazılımı bu yazıyı yazarken yoktu (veya ben bilmiyordum). Umarım yakın gelecekte bu şekilde bir çerçeve yazılım (framework) geliştirilerek bu konuda çalışan kişilere imkan sağlayacaktır.

Kelime etiketleme işlemi klasik bir gövdeleme (Stemmer) işleminden biraz daha karmaşık yapıdadır. Aslında işlem basitçe bir kelimenin tipini bilmektir. Örneğin kelime isim, fiil, sıfat, bağlaç gibi sınıflardan hangisine aitse o sınıfı etiket olarak koymak işlemidir.

Aslında basit bir işlem olarak görülmesine karşılık bu işlemde olasılıklar devreye girer. Örneğin “ koyun” kelimesinin kökü hayvan olan koyun olabileceği gibi “koy” fiilinin bir çekimi olup gövdesi “koyun” da olabilir. Ancak bu iki kelime arasındaki farkı anlamak önceki ve sonraki kelimelere bakarak mümkündür. Örneğin “kitaplarınızı masalarınızın üzerine koyun” cümlesindeki koyun kelimesinin fiil olarak etiketleneceği, buna mukabil “çoban merada tek bir koyun otlatıyordu” cümlesindeki koyun kelimesinin ise hayvan olan koyun olduğu anlaşılmaktadır.

Part of speech (POS, konuşmanın bir kısmı) kullanılarak bu etiketleme yapılabilir. Örneğin “koyun otlatıyordu” veya “üzerine koyun” şeklinde ikişer kelimenin alıntılanması aradaki farkın çözülmesi için yeterli olacaktır.

Bu ve benzeri durumlar için genelde kullanılan yaklaşım HMM (hidden markov model, gizli markov zinciri) yaklaşımıdır ve olasılıkların hesaplanmasına dayanır.

Örneğin İngilizce için çok daha olgun yapılan çalışmalarda bir belirtecin (determiner) ardından %40 oranında isim %40 oranında sıfat ve %20 oranında bir sayı geldiğini tespit etmişlerdir.

Bu durumda herhangi bir belirteçten sonra fiil gelmeyeceği bellidir. Örneğin ingilizcede isim olan koşu (run) veya fiil olan koşmak (run) anlamlarında kullanılan aynı kelime, “the run” şeklinde bir belirteçten sonra geliyorsa fiil olamayacak ve isim olarak sınıflandırılacaktır.

Yukarıdaki HMM yaklaşımında önemli bir ihtiyaç, doğru şekilde ve çok yüksek sayıdaki metinlerde eğitim yapılmasıdır. Yani bu oranların çıkarılması için bir insan tarafından etiketlenmiş metinlerin verilmesi ve sistemin eğitilmesi şu anda uygulanan en yaygın yöntemdir.

Ayrıca bu yöntemlerin çeşitli dinamik programlama (dynamic programming) yöntemleri ile hızlandırılması ve performans artışı mümkündür.

İncelediğim çok sayıdaki etiketleyici arasından en beğendiğim ve JAVA dilinde yazılmış olan ve bir kaç projede kullandığım stanford kütüphanesine aşağıdaki bağlantıdan erişilebilir:

<http://nlp.stanford.edu/software/tagger.shtml>

SORU 4: TF-IDF

TF-IDF kavramı IR (information retrieval, bilgi getirimi) gibi konuların altında bir sıralama (ranking) algoritması olarak sıkça geçmektedir.

İngilizcedeki Term Frequency – Inverse Document Frequency (Terim frekansı – ters metin frekansı) olarak geçen kelimelerin baş harflerinden oluşan terim basitçe bir metinde geçen terimlerin çıkarılması ve bu terimlerin geçtiği miktara göre çeşitli hesapların yapılması üzerine kuruludur.

Klasik olarak TF yani terimlerin kaç kere geçtiğinden daha iyi sonuç verir. Kısaca TF-IDF hesabı sırasında iki kritik sayı bulunmaktadır. Bunlardan birincisi o anda ele alınan dokümandaki terimin sayısı diğeri ise bu terimi külliyatta içeren toplam doküman sayısıdır.

Örnek:

Konuyu bir örnek üzerinden açıklayalım:

Örneğin 100 dokümandan oluşan bir külliyatımız olsun ve TF-IDF hesaplamak istediğimiz kelime de “şadi” olsun. Bu durumda birinci dokümana bakıp “şadi” kelimesinin kaç kere geçtiğini sayarız. Diyeli ki 4 kere geçiyor olsun. Ardından külliyatımızdaki 100 dokümandan kaçında “şadi” kelimesi geçiyor diye bakarız. Diyelim ki 10 dokümanda bu kelime geçiyor olsun (dikkat edilecek husu kelimenin geçip geçmediğidir diğer dokümanlarda kaç kere geçtiğinin bir önemi yoktur).

Şimdi TF ve IDF değerlerini ayrı ayrı hesaplayacağız ve sonra bu iki değeri çarpacağız, önce TF hesabına bakalım:

TF hesabı için ihtiyacımız olan bir diğer değer ise o andaki dokümanda en fazla geçen terim sayısıdır. Örneğin o anda baktığımız dokümanda en fazla geçen terimimizin sayısı da 80 olsun.

İlk hesaplama dokümanda bizim ilgilendiğimiz kelimenin en fazla geçen kelimeye oranıdır. Yani kelimemiz 4 kere geçtiğine ve en fazla geçen kelimemiz de 80 kere geçtiğine göre ilk oranımız (ki bu oran aynı zamanda TF (term frequency, terim frekansı) olarak tek başına da anlamlıdır)

$TF = 4 / 80 = 0,05$ olarak bulunur.

Ardından IDF değerini hesaplayalım. Bunun için basit bir bölme işlemi yapılacak ve logaritması alınacaktır.

$IDF = \log (\text{Toplam Doküman sayısı} / \text{Terimi içeren doküman sayısı})$

Buna göre IDF için toplam 100 dokümandan 10 dokümanda aradığımız kelime “şadi” geçtiğine göre

$IDF = \log (100 / 10) = \log (10) = 1$ olarak bulunacaktır.

IDF hesabı sırasında bir iki noktaya dikkat etmek gerekir. Öncelikle logaritmanın tabanının bir önemi yoktur. Amaç üssel fonksiyonun tersi yönde bir hesap yapmaktır. Doğal logaritma kökü e, 2 veya 10 gibi sayılar en çok kullanılan değerlerdir. Genelde TF-IDF değerinin kıyas için kullanıldığını ve diğer terimlerin TFIDF değerleri ile kıyaslandığını düşünecek olursak hepsinde aynı tabanın kullanılıyor olması sonucu değiştirmeyecektir.

Diğer dikkat edilecek bir husus ise IDF hesabı sırasında geçen “terimi içeren doküman sayısı” değeridir. Bu değer hesaplama sırasında paydada yer almaktadır ve bu değer 0 (sıfır) olma ihtimali vardır. Bu durumda sonuç sıfıra bölüm belirsizliğine götürebileceğinden genelde bu değere 1 eklemek sıkça yapılan bir programlama yaklaşımıdır.

Neticede elde ettiğimiz $TF = 0,05$ ve $IDF = 1$ değerlerini çarpıyoruz ve terimimizin TF-IDF değeri aşağıdaki şekilde bulunuyor:

$TF-IDF = TF \times IDF = 0,05 \times 1 = 0,05$

Yukarıda kullandığımız formülleri aşağıdaki şekilde de açıkça yazmak mümkündür:

$$w_{i,d} = tf_{i,d} \times \log(n / df_i)$$

Yukarıdaki gösterimde, i terimi için ve d dokümanı için hesaplama yapılmaktadır. Öncelikle TF hesaplanır ki bu basitçe terimin o dokümanda kaç kere geçtiğinin en fazla geçen terime oranı şeklinde hesaplanabilir:

$$tf_{i,d} = aza(fr_{i,d} / df_i)$$

Yani i terimi için d dokümanındaki terim frekansı (term frequency), i teriminin d dokümanındaki tekrar sayısının o dokümandaki en yüksek tekrar sayısına sahip terimin tekrar sayısına oranıdır. Veya bu oranların en yükseğidir.

Yukarıda verilen TF-IDF formülünde ayrıca n toplam doküman sayısını df ise doküman frekansını vermektedir ve df aslında i teriminin kaç farklı dokümanda geçtiğinin sayısıdır.

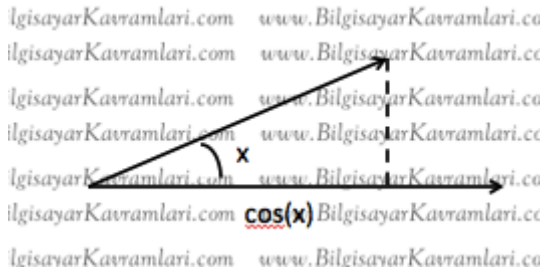
Son olarak TF-IDF yönteminin diğer yöntemlere göre farkını açıklamaya çalışalım. TF-IDF ile bir terimin kaç kere geçtiği kadar kaç farklı dokümanda da geçtiği önem kazanır. Örneğin sadece bir dokümanda 100 kere geçen bir terimle 10 farklı dokümanda onar kere geçen terimin ikisi de aslında toplamda 100 kere geçmiştir ancak TF-IDF ikincisine yani daha fazla dokümanda geçene önem verir.

SORU 5: SimHash (Benzerlik Özeti)

Bilgisayar bilimlerinde, özellikle metin işlemenin yoğun olduğu, arama motoru gibi uygulamalarda dosyaların veya web sitelerinin birbirine olan benzerliğini bulmak için kullanılan bir algoritmadır.

Algoritmaya alternatif olarak klasik hash fonksiyonları kullanılabilir. Yani, örneğin iki sayfasının ayrı ayrı hash değerleri alınıp bu değerleri karşılaştırmak mümkündür. Ancak simhash algoritması, bu yönetime göre daha fazla hız ve performans sunar.

Sim hash algoritması, iki dosyayı birer vektör olarak görür ve bu vektörler (yöney, vector) arasındaki cosinüs (cosine) bağlantısını bulmaya çalışır.



Yukarıdaki şekilde temsil edildiği üzere iki dokümanın ayrı ayrı birer vektör olması durumunda, aralarında $\cos(x)$ olarak gösterilen bir açı ile bağlantı bulunması mümkündür.

Algoritma, öncelikle işlediği metindeki kelimelerin ağırlıklarını (weight) çıkarmakta ve buna göre kelimeleri sıralamaktadır.

Sıralanan her kelimeye, b uzunluğunda, yegane (unique) değer döndüren bir fonksiyon kullanılır. Örneğin her kelime için farklı bir hash sonucu döndüren fonksiyon kullanılır.

b boyutundaki bir vektörün ağırlık değeri hesaplanırken, her kelimedeki 1 değeri için +1 ve 0 değeri için -1 değeri ağırlığa eklenir.

Son olarak üretilen ağırlık vektöründeki + değerler 1, 0 ve - değerler ise 0 olarak çevirilir.

Örnek

Yukarıdaki algoritmanın çalışmasını bir örnek üzerinden anlatalım. Algoritmanın üzerinde çalışacağı metin aşağıdaki şekilde verilmiş olsun:

www bilgisayar kavramları com bilgisayar kavramlarının anlatıldığı bir bilgisayar sitesidir ve com uzantılıdır

Yukarıdaki bu metni, algoritmanın anlatılan adımlarına göre işleyelim:

İlk adımımız, algoritmadaki kelimelerini ağırlıklarının çıkarılmasıdır. Bu adımı çeşitli şekillerde yapmak mümkündür ancak biz örneğimizde kolay olması açısından kelime frekanslarını (tekrar sayısı, frequency) kullanacağız. Buna göre metindeki kelimelerin tekrar sayılarına göre sıralanmış hali aşağıda verilmiştir:

bilgisayar 3 com 2 kavramları 1 kavramlarının 1 anlatıldığı 1 bir 1 www 1 sitesidir 1 ve 1 uzantılıdır 1

Yukarıda geçen her kelime için bir parmak izi (fingerprint) değeri üretiyoruz. Bu değerin özelliği, kelimeler arasında yegane (unique) bir değer bulmaktır. Bu değer, herhangi bir hash fonksiyonu üzerinden de üretilebilir. Biz örneğimizde kolalık olması açısından her kelime için rast gele bir değer kendimiz atayacağız. Ancak gerçek bir uygulamada rast gele değerlerin kullanılması mümkün değildir. Bunun sebebi, aynı kelimenin tekrar gelmesi halinde yine aynı değerin üretilmesi zorunluluğudur. Bu yazıdaki amaç algoritmayı anlatmak olduğu için birer hash sonucu olarak rast gele değerler kullanılacaktır.

bilgisayar 10101010 com 11000000 kavramları 01010101 kavramlarının 10100101 anlatıldığı 11101110 bir 01011111 www 11110001 sitesidir 10101110 ve 00001111 uzantılıdır 00100010

3. adımda, yukarıdaki değerleri topluyoruz. Toplama işlemi sırasında 1 değerleri için +1 ve 0 değerleri için -1 alıyoruz.

10101010
11000000
01010101
10100101
11101110
01011111
11110001
10101110
00001111
00100010

2 0 2 -4 0 2 2 0

Son olarak, yukarıdaki değerleri ikilik tabana çeviriyoruz: 10100110 bu değer bizi simhash sonucumuz olarak bulunuyor.

Örneğin yeni bir dosyayı daha işlemek istediğimizde, bu dosyadaki kelime yoğunluğuna göre yukarıda bulduğumuz simhash değerine yakın bir değer çıkmasını bekleriz.

Diyelim ki yeni bir dosyada da sadece “bilgisayar kavramları com” yazıyor olsun. Bu yazının sim hash değerini bularak karşılaştırmaya çalışalım:

bilgisayar 10101010 com 11000000 kavramları 01010101

10101010

11000000

01010101

1 1 -1 -1 -1 1 1 1

Değerin ikilik tabana çevrilmiş hali : 11000111

Orjinal dokümandan çıkardığımız simhash değeri ile farklı olan bit sayısı 3'tür. Bunun anlamı yukarıdaki bilgisayar kavramları com yazısının orjinal yazıya 3 mesafesinde yakın olduğudur.

SORU 6: n-gram

Verilen bir dizilimdeki (sequence) tekrar oranını bulmaya yarayan yöntemdir. İsmi n ve gram kelimelerinin birleşiminden oluşmaktadır. Buradaki n, tekrarın kontrol edildiği değerdir. Gram ise bu tekrarın dizilim içerisindeki ağırlığını ifade etmek için kullanılmıştır.

Örneğin bir [dizgi \(string\)](#) içerisindeki n-gram değerini bulmak isteyelim ve buradaki n değeri 2 olsun (n = 2)

“wwwbilgisayarkavramlaricomsadievrenseker”

Yukarıdaki yazının 2-gram değerleri aşağıda verilmiştir:

ww	2	ka	1	ad	1
wb	1	av	1	di	1
bi	1	vr	2	ie	1
il	1	ra	1	ev	1
lg	1	am	1	re	1
gi	1	ml	1	en	1
is	1	la	1	ns	1
sa	2	ri	1	se	1
ay	1	ic	1	ek	1
ya	1	co	1	ke	1
ar	2	om	1	er	1
rk	1	ms	1		

Yukarıdaki değerler üzerinde çalıştığımız [dizginin \(string\)](#) 2şer komşuluktaki alt kümesidir.

Benzer şekilde aynı dizginin 3-gram değerleri aşağıda verilmiştir:

www	1	kav	1	sad	1
wwb	1	avr	1	adi	1
wbi	1	vra	1	die	1

bil	1	ram	1	iev	1
ilg	1	aml	1	evr	1
lgi	1	mlla	1	vre	1
gis	1	lar	1	ren	1
isa	1	ari	1	ens	1
say	1	ric	1	nse	1
aya	1	ico	1	sek	1
yar	1	com	1	eke	1
ark	1	oms	1	ker	1
rka	1	msa	1		

Benzer şekillerde sayı arttırılarak istenilen bir n-gram değeri listelenebilir. Bu değerlerin bazıları literatürde özel olarak isimlendirilmektedir.

Örneğin bu isimlendirme listesi aşağıda verilmiştir:

- 1-gram > unigram
- 2-gram > bigram
- 3-gram > trigram

3'ten büyük değerler için sadece n-gram terimi kullanılır.

n değerine bağlı olarak dizilimin alacağı ihtimal uzayı artmaktadır. Örneğin 29 harfli Türkçe için unigram bir işleme sonucunda 29 farklı alternatifte göre dizilimin dağılımı ortaya çıkar. Yani listemizin sonucunda 29 farklı ihtimal ve bu ihtimallerin her birisinin kaçar kere tekrarlandığı bulunur.

n değerini arttırdıkça bu ihtimal uzayı büyür ve yine 29 harfli Türkçe için (Sadece alfabedeki harflerin olduğunu kabul ediyoruz) 29^n olarak bulunabilir.

Yukarıda anlatıldığı üzere bir [dizginin \(String\)](#) üzerinde n-gram işlemesi yapan kod aşağıda verilmiştir:

```

2  import java.util.*;
3  class sayac{
4      int num;
5      String ngram;
6      public sayac(){}
7      public sayac(int num, String ngram){
8          this.ngram = ngram;
9          this.num = num;
10     }
11 }
12 public class test {
13     public static void main(String[] args) {
14         String d = "wwwbilgisayarkavramlaricomsadievrenseker";
15         Vector v = new Vector();
16         int n = 3;
17         for(int i = 0; i < d.length()-n+1; i++){
18             String x= d.substring(i,i+n);
19             boolean buldu = false;
20             for(int k = 0; k < v.size(); k++){
21                 sayac s = (sayac)v.elementAt(k);
22                 if(x.equals(s.ngram)){
23                     buldu = true;
24                     s.num++;
25                 }
26             }
27             if(!buldu)
28                 v.add(new sayac(1,x));
29         }
30         for(int i = 0; i < v.size(); i++){
31             sayac s = (sayac)v.elementAt(i);
32             System.out.println(s.ngram+ " "+ s.num);
33         }
34     }
35 }

```

Yukarıdaki kodda, iki adet sınıf kullanılmıştır. sayac isimli sınıf, bulunan n-gramları tutmak için kullanılan bir veri ünitesidir. Bu veri ünitesi, test sınıfında kullanılan [vektör \(vector\)](#) içerisinde tutulmakta ve bulunan her n-gram'ın kaç tane olduğunu saymaktadır.

Kodun 17 ile 34. satırları arasında verilen [dizgi \(String\)](#) önce verilen n boyutunda parçalara ayrılmakta (kodun 18. satırındaki String kütüphanesinden substring fonksiyonu marifeti ile) ardından bu n boyutundaki parça veri yapımız olan Vektör içerisinde var mı diye bakılmaktadır (kodun 20 ile 26. satırları arasında). Şayet yoksa vektör eklenmekte (kodun 27,28. satırları) varsa da kaç tane olduğu değeri artırılmaktadır (kodun 24. satırı).

SORU 7: Evrimsel Diller (Evolutionary Languages)

Evrimsel diller temel olarak bir doğal dilin (natural language) geçirmiş olduğu evrimi ve bu evrimin dilde yaptığı değişiklikleri inceler. Evrimsel dil çalışmalarının bilgisayar bilimlerindeki yeri, dilbilim ve doğal dil işleme ile ilgilenen insanların dillerdeki bu değişimleri bilgisayar dünyasına uygulama merakıyla başlamıştır. Bu anlamda bilgisayar bilimleri için evrimsel diller literatürde, yapay zeka (artificial intelligence), insan makine

ilişkisi (man machine interaction) veya makine öğrenmesi (machine learning) konuları altında incelenebilir.

Evrimsel dil çalışmalarında karşılaşılan en büyük zorluk dillerin geçirdiği değişimler ile ilgili kaynaklara erişimdedir. Örneğin konuşulan bir dilin geçirdiği değişimler ile ilgili bir belge ya da bir kanıt bulmak imkansızdır. Ayrıca bu alanda ampirik (deneysel, emprical, tecrübi) çalışmalar yapmak da ne yazık ki mümkün değildir (tabi bir grup insanın dilini manipüle edip dilde kalıcı değişiklikler yapmak gibi şeyleri saymazsak)

Evrimsel dillerle bilgisayar bilimleri ilgilenmeden çok önce dilbilim konusunda çalışan ve psikoloji ve sosyolojinin bu alana bakan disiplinler arası alanları ilgilenmiştir. Günümüzde bu konuda bilişsel bilim (cognitive science), psiko-dilbilim (psycholinguistic) veya nöro-dilbilim (neurolinguistic) alanlarında çalışılmakta ayrıca sosyal olguların incelendiği ve toplumsal dönüşümleri çalışan evrimsel antropoloji (evolutionary anthropology) alanında da çeşitli çalışmalar yapılmaktadır.

Günümüzde evrimsel diller üzerinde yapılan çalışmaların dayandığı basit ama temel ilke, insanların iletişim ihtiyaçları doğrultusunda yapmış oldukları icatlar olarak özetlenebilir. Kısaca dil insanların iletişimi için vardır ve dilde bu iletişim ihtiyacı bir şekilde karşılanamazsa, dili kullanan insanlar, dile yeni eklentiler yaparak (veya mevcut kelime ve kavramların anlamlarını değiştirerek) bu ihtiyaçlarını çözerler.

Bilgisayar bilimlerinin de dahil olduğu evrimsel dilbilim çalışmalarında amaç daha çok dillerdeki evrimin bilgisayarlar tarafından algılanmasını sağlayacak bir model geliştirilmesidir. Yani insanlar dil üzerinde gerek kendi ihtiyaçlarını karşılayacak üretimler yaparken ve gerek de başka kişiler tarafından yapılmış üretilere uyum gösterirken belirli bir zeka örneği gösterirler. İşte bilgisayar bilimleri de bu zeka örneğinin bilgisayarlar tarafından modellenip modellenemeyeceğini veya nasıl modelleneyeceğini bulmaya çalışır.

Bu anlamda yapılan çalışmalar uyumlu sistemler (adaptive systems, adaptif sistemler, tetabık tertipler) olarak nitelendirilebilir. Tam bu noktada akıcı inşa dilbiliminden (fluid construction grammer kısaca FCG olarak literatürde geçmektedir) bahsetmekte yarar vardır. Luc Steels tarafından geliştirilen bu dilbilgisinde, bir dili her konuşanın (her ajanın (agent)) aslında farklı bir şey konuştuğunu kabul ile işe başlanır. İsmindeki akıcı (fluid) kelimesi de buradan gelmektedir. Yani dil sürekli bir akış halinde (sürekli bir evrim halinde) düşünülebilir. Bu anlamda kullanılan en geniş modelleme yöntemlerinden birisi olan FCG, bilgisayarlar tarafından işlenebilen (üretilebilen ve anlaşılabilen) bir yapıya sahiptir.

SORU 8: Dilin Derecesi (Rate of Language)

Veri işilemede (veri güvenliği veya veri tabanı gibi teorilerde) bir verinin içinde bulunduğu dilin (language) derecesinden bahsedilebilir. Burada derece (rate) ile kastedilen verinin değişim oranını bulmaktır. Basitçe verinin entropisinin (dağınımının, entropy) verinin uzunluğuna bölümü ile elde edilir

$$r = H(M) / N$$

Yukarıdaki formülde r, dilin derecesini, H(M) derecesini sorguladığımız dili oluşturan verinin entropisini (burada M veriyi veya mesajı temsil etmekte) ve N ise verinin uzunluğunu göstermektedir.

Bu yazı şadi evren şeker tarafından yazılmış ve bilgisayar kavramları.com sitesinde yayınlanmıştır. Bu içeriğin kopyalanması veya farklı bir sitede yayınlanması hırsızlıktır ve telif hakları yasası gereği suçtur.

Örneğin dilimizde sadece cinsiyet bilgilerini tutmak için bir dil inşa edersek. Dilimizde sadece “kadın” ve “erkek” kelimeleri olacaktır. Buradaki kelimelerin boyu 5’tir (yani formüldeki N).

Entropi yazısından hatırlanacağı üzere verimizin entropisi ise 1’dir. Bu değer $\log_2 2 = 1$ olarak hesaplanabilir. Diğer bir deyişle 2 farklı bilgiyi tek bir bit ile tutabiliriz. $2^1 = 2$ farklı bilgi tutar.

Dolayısıyla bu hayali dilin derecesi :

$$r = 1 / 5 = 0.20$$

olarak bulunacaktır.

SORU 9: Özyineli Diller (Recursive Languages)

Özyineli diller matematik, mantık veya bilgisayar bilimlerinde geçen [muntazam dillerden \(formal language\)](#) birisidir. Genellikle kararverilebilir yani [Turing makinesi \(Turing machine\)](#) tarafından işlenebilir diller olarak kabul edilirler.

Özyineli diller [Chomsky hiyerarşisinde](#) yer almamaktadır.

Bir özyineli dili tanımlamak için iki önemli tanım yapılır. Birincisi dilin içerdiği alfabeden üretilen [güç kümesinin \(power set\) özyineli alt kümesi \(recursive subset\)](#) olmasıdır.

İkincisi ise [Turing makinesi](#) tarafından kabul edilebilir bir [muntazam dil \(formal language\)](#) olmasıdır. Yani Turing makinesi bu dili kabul edecek ve bu dil dışındaki durumlarda [duracaktır \(halting\)](#).

Bütün özyineli diller aynı zamanda [özyineli sayılabilirlerdir](#). Bütün CFG veya düzenli ifadeler özyineli dil olarak kabul edilebilir.

SORU 10: Özyineli Geçiş Ağları (Reursive Transition Networks)

Veri modellemede kullanılan bir ağ şeklidir. Esas itibarıyla [içerikten bağımsız dillerin \(context free grammars\)](#) görsel gösterimi için kullanılabilirler. Ağların yapısı [uzatılmış geçiş ağlarına \(augmented transition network\)](#) benzemekle birlikte en büyük farkı ve isminin özyineli olmasının da sebebi ağın kendini tekrarlama özelliğidir.

Daha basitçe bir [içerikten bağımsız dil \(CFG\)](#) S devamlısı (nonterminal) ile başlayan bir kurallar listesidir. Bu listedeki her kural bir devamlıdan (nonterminal) bir sonluya (terminal) doğru yapılan bir açıklamadır. Özyineli geçiş ağlarında bu açıklamada (yani [CFG](#) kurallarının sağ tarafında) da S devamlısı (nonterminal) bulunabilmektedir.

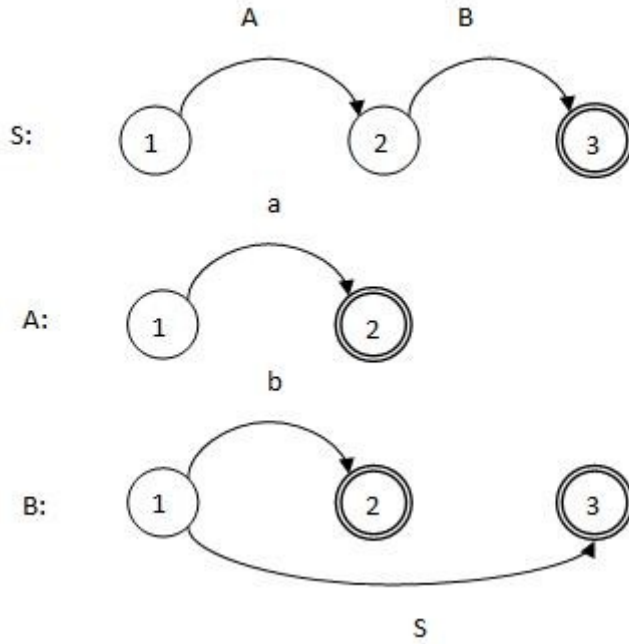
Örneğin aşağıdaki kuralları ele alalım:

$$S \rightarrow A B$$

$A \rightarrow a$

$B \rightarrow b \mid S$

Yukarıdaki son kuralda S bağılangıç devamlısına bir bağlantı kurulmuştur ve aşağıdaki şekilde görselleştirilebilir:



Yukarıdaki ağ yapısı verilen CFG örneğinin çizilmiş halidir. Buradaki çizimden de görüleceği üzere ağ yeniden başa döneme özelliğine sahiptir.

Doğal dil olarak yorumlanacak olursa bir cümlenin içinde alt cümlelerin bulunması bu ağ ile gösterilebilir. Örneğin aşağıdaki örnek cümleyi ele alalım:

Ali geldiğini söylemeyi unuttu.

Yukarıdaki örnekte 3 cümle iç içedir.

Ali gelmek

Ali söylemek

Ali unutmak

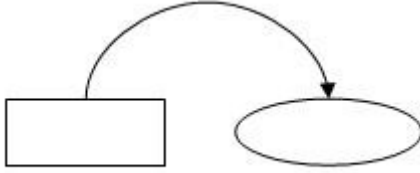
Basit bir yaklaşımla bir cümleyi özne – yüklem olarak tanımlayacak olursak yukarıdaki örnek cümlenin uyduğu yapı aşağıdaki şekilde olabilir:

C: Ö Y

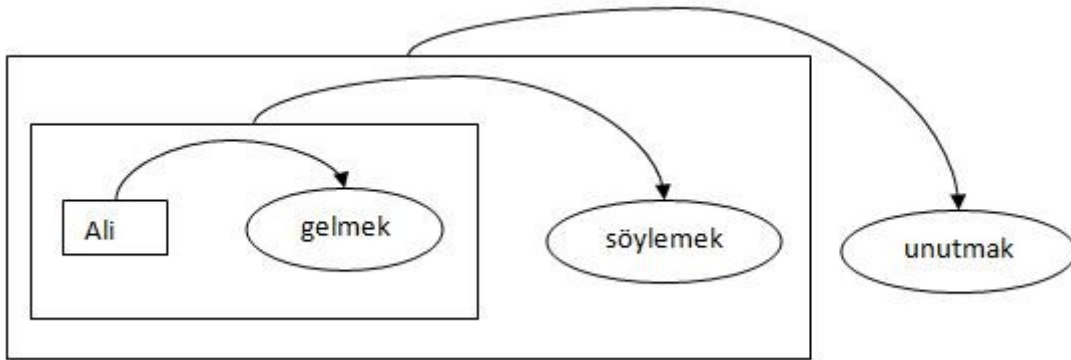
Ö : “Ali”

Y: gelmek | söylemek | unutmak

Yukarıdaki CFG yapısını şayet özyineli geçiş ağıımız (recursive transition network) ile gösterecek olursak:



Örneğin basit bir Özne-Yüklem ilişkisini yukarıdaki şekilde gösterelim. Yani özneleri dikdörtgen ve yüklemeleri yuvarlak ile gösterelim. Bu durumda örneğimiz:



Görüldüğü üzere özyineli ağıımızla gösterilen örnekte unutmak fiilinin başındaki isim kelime grubu “ali geldiğini söylemeyi” yine bir cümledir ve yapısı itibariyle bir önceki şekilde gösterilen şablonu içermektedir. Benzer şekilde söylemek fiilinin başındaki isim grubu da bir geçiş ağı (transition network) özelliği taşımaktadır

SORU 11: Gellish (Kontrollü Doğal Dil)

Gellish dilleri sınırları ve kuralları insanlar tarafından belirlenen ve istisnası bulunmayan dillerdir. Bu anlamda programlama dilleri de dahil olmak üzere çeşitli kullanım alanları vardır. Etimolojik olarak Genel Mühendislik Dili (Generic Engineering Language) kelimelerinin baş harflerinden oluşan kelime günümüzde mühendislik uygulamalarından farklı alanlarda da kullanılmaktadır.

Genellikle karmaşaya yer verilmek istenmeyen açık ve net olunması gereken yerlerde kullanılan gellish dilini örneğin sağlık, acil durumlar, afet yönetimi gibi direktif belirtici yazılarda veya makine ve yazılımların dökümantasyonlarında görebiliriz. Buradaki amaç dil bilgisi düşük olabilecek daha az eğitilmiş kişilerin kolayca anlamasını sağlamak veya acil durumlarda panik halinde olabilecek kişilerin anlama güçlüklerinin ve hatalarının en aza indirgenmesidir.

Gellish dillerini oluşturan iki grup vardır:

- Kelimeler

- Kelimeler arası ilişkiler

Öncelikle kesin ve net kelimelerin, belirsizliğe yer bırakmadan anlamlarının çıkarılması gerekir. Doğal dillerde bir kelime birden fazla anlama gelebilmekte hatta bazı durumlarda mecaz veya ima ile sözlükte bile hiç yeri olmayan anlamlarda kullanılabilmektedir. Gellish dilinin birinci adımı bu belirsilikleri ortadan kaldırmak ve her kelimeye tek bir anlam ve her anlama tek bir kelime bulmaktır.

Bu kelimeleri içeren sözlüğe akıllı sözlük (Smart dictionary) ismi verilmektedir. Buradaki kelimeler anlamsal olarak ilişkilendirilmiştir. Hatta kelimelerin anlam ilişkileri sonucunda bir [ontoloji \(ontology\)](#) çıkmaktadır.

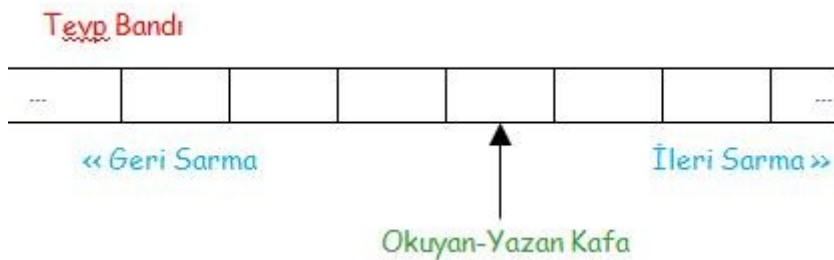
Gellish dilleri doğal dillerin bir alt kümesi olarak düşünülebilir. Örneğin Gellish Türkçe (Gellish Turkish) denilince Türkçede bulunan kelimeler ve dilbilgisi kuralları kapsamında kavramları ve bu kavramlar arası ilişkileri belirleyen bir dil kast edilmektedir. Bu dil zaten Türkçe için olan şeylerden oluşacaktır. Bu anlamda Esperanto gibi yeniden türetilmiş diller ile karıştırılmamalıdır. Ancak [owl \(web ontology language\)](#) gibi [anlamsal ağ \(Semantic web\)](#) gösterim dilleri ile karşılaştırılabilir.

SORU 12: Turing Makinesi (Turing Machine)

Bilgisayar bilimlerinin önemli bir kısmını oluşturan [otomatlar \(Automata\)](#) ve [Algoritma Analizi \(Algorithm analysis\)](#) çalışmalarının altındaki dil bilimin en temel taşlarından birisidir. 1936 yılında Alan Turing tarafından ortaya atılan makine tasarımı günümüzde pek çok teori ve standardın belirlenmesinde önemli rol oynar.

Turing Makinesinin Tanımı

Basitçe bir kafadan (head) ve bir de teyp bandından (tape) oluşan bir makinedir.



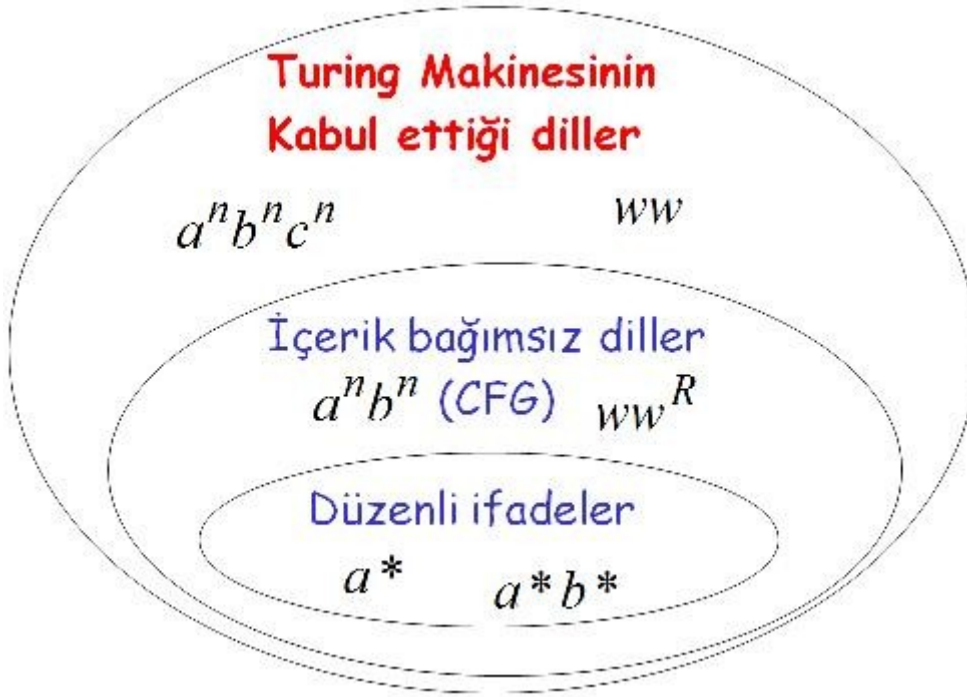
Makinede yapılabilecek işlemler

- Yazmak
- Okumak
- Bandı ileri sarmak
- Bandı geri sarmak

şeklinde sıralanabilir.

Chomsky hiyerarşisi ve Turing Makinesi

Bütün teori bu basit dört işlem üzerine kurulmuştur ve sadece yukarıdaki bu işlemleri kullanarak bir işin yapıp yapılamayacağı veya bir dilin bu basit 4 işleme indirgenip indirgenemeyeceğine göre diller ve işlemler tasnif edilmiştir.



Bu sınıflandırma yukarıdaki venn şeması ile gösterilmiştir. Aynı zamanda [chomsky hiyerarşisi \(chomsky hierarchy\)](#) için 1. seviye (type-1) olan ve Turing makinesi ile kabul edilebilen diller bütün tip-2 ve tip-3 dilleri yani içerik bağımsız dilleri ve düzenli dilleri kapsamaktadır. Ayrıca ilave olarak içerik bağımsız dillerin işleyemediği (üretmediği veya parçalayamadığı (parse)) $a^n b^n c^n$ şeklindeki kelimeleri de işleyebilmektedir. Düzenli ifadelerin işleyememesi konusunda bilgi için [düzenli ifadelerde pompalama savı \(pumping lemma in regular expressions\)](#) ve [içerik bağımsız dillerin işlemeyemesi için de içerik bağımsız dillerde pompalama savı \(pumping lemma for CFG\)](#) başlıklı yazıları okuyabilirsiniz.

Turing Makinesinin Akademik Tanımı

Turing makineleri literatürde akademik olarak aşağıdaki şekilde tanımlanır:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, \diamond, F)$$

Burada M ile gösterilen makinenin parçaları aşağıda listelenmiştir:

Q sembolü sonlu sayıdaki durumların [kümesidir](#). Yani makinenin işleme sırasında aldığı durumardır.

Γ sembolü dilde bulunan bütün harfleri içeren alfabeyi gösterir. Örneğin ikilik tabandaki sayılar ile işlem yapılıyorsa $\{0,1\}$ şeklinde kabul edilir.

Σ sembolü ile makineye verilecek girdiler (input) [kümesi](#) gösterilir. Girdi [kümesi](#) dildeki harfler dışında bir sembol taşıyamayacağı için $\Sigma \subseteq \Gamma$ demek doğru olur.

δ sembolü dilde bulunan ve makinenin çalışması sırasında kullanacağı geçişleri (transitions) tutmaktadır.

\diamond sembolü teyp bandı üzerindeki boşlukları ifade etmektedir. Yani teyp üzerinde hiçbir bilgi yokken bu sembol okunur.

q_0 sembolü makinenin başlangıç durumunu (state) tutmaktadır ve dolayısıyla $q_0 \subseteq Q$ olmak zorundadır.

F sembolü makinenin bitiş durumunu (state) tutmaktadır ve yine $F \subseteq Q$ olmak zorundadır.

Örnek Turing Makinesi

Yukarıdaki sembolleri kullanarak örnek bir Turing makinesini aşağıdaki şekilde inşa edebiliriz.

Örneğin basit bir kelime olan a^* [düzenli ifadesini \(regular expression\)](#) Turing makinesi ile gösterelim ve bize verilen aaa şeklindeki 3 a yı makinemizin kabul edip etmediğine bakalım.

Tanım itibariyle makinemizi aşağıdaki şekilde tanımlayalım:

$$M = \{ \{q_0, q_1\}, \{a\}, \{a, x\}, \{q_0 a \rightarrow a R q_0, q_0 x \rightarrow x L q_1\}, q_0, x, q_1 \}$$

Yukarıdaki bu makineyi yorumlayacak olursak:

Q değeri olarak $\{q_0, q_1\}$ verilmiştir. Yani makinemizin iki durumu olacaktır.

Γ değeri olarak $\{a, x\}$ verilmiştir. Yani makinemizdeki kullanılan semboller a ve x'ten ibarettir.

Σ değeri olarak $\{a\}$ verilmiştir. Yani makinemize sadece a girdisi kabul edilmektedir.

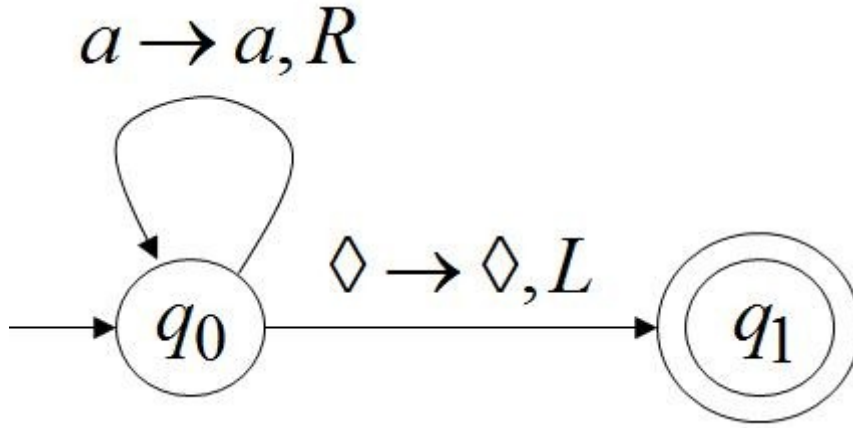
δ değeri olarak iki geçiş verilmiştir $\{q_0 a \rightarrow a R q_0, q_0 x \rightarrow x L q_1\}$ buradaki R sağa sarma L ise sola sarmadır ve görüleceği üzere Q değerindeki durumlar arasındaki geçişleri tutmaktadır.

\diamond değeri olarak x sembolü verilmiştir. Buradan x sembolünün aslında boş sembolü olduğu ve bantta hiçbir değer yokken okunan değer olduğu anlaşılmaktadır.

q_0 ile makinenin başlangıç durumundaki hali belirtilmiştir.

F değeri olarak q_1 değeri verilmiştir. Demek ki makinemiz q_1 durumuna geldiğinde bitmektedir (halt) ve bu duruma gelmesi halinde bu duruma kadar olan girdileri kabul etmiş olur.

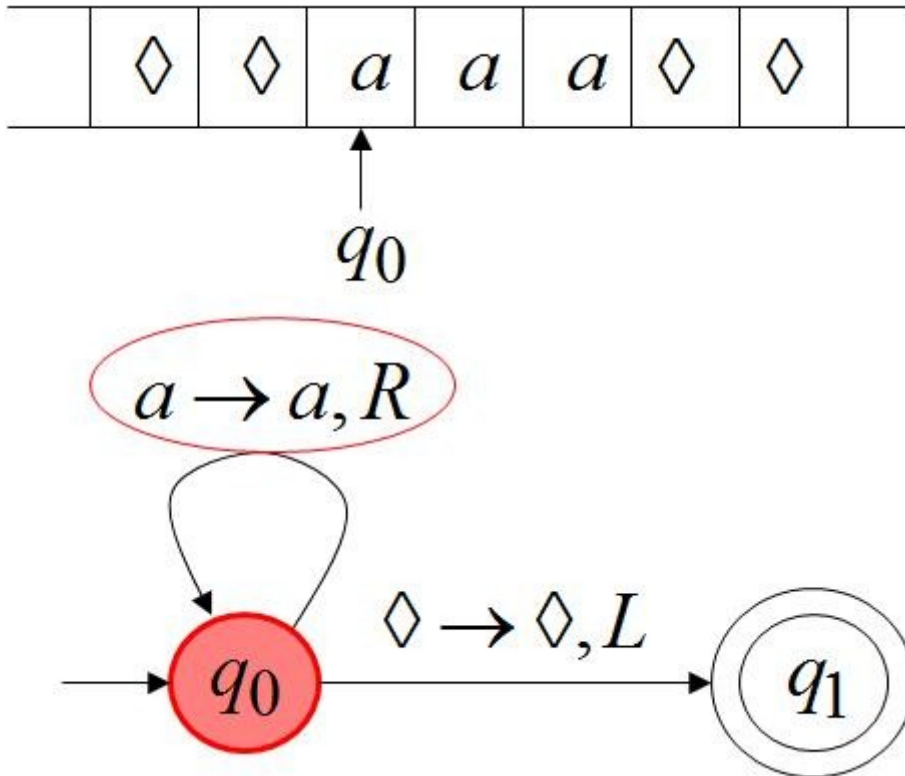
Yukarıdaki bu tanımları görsel olarak göstermek de mümkündür:



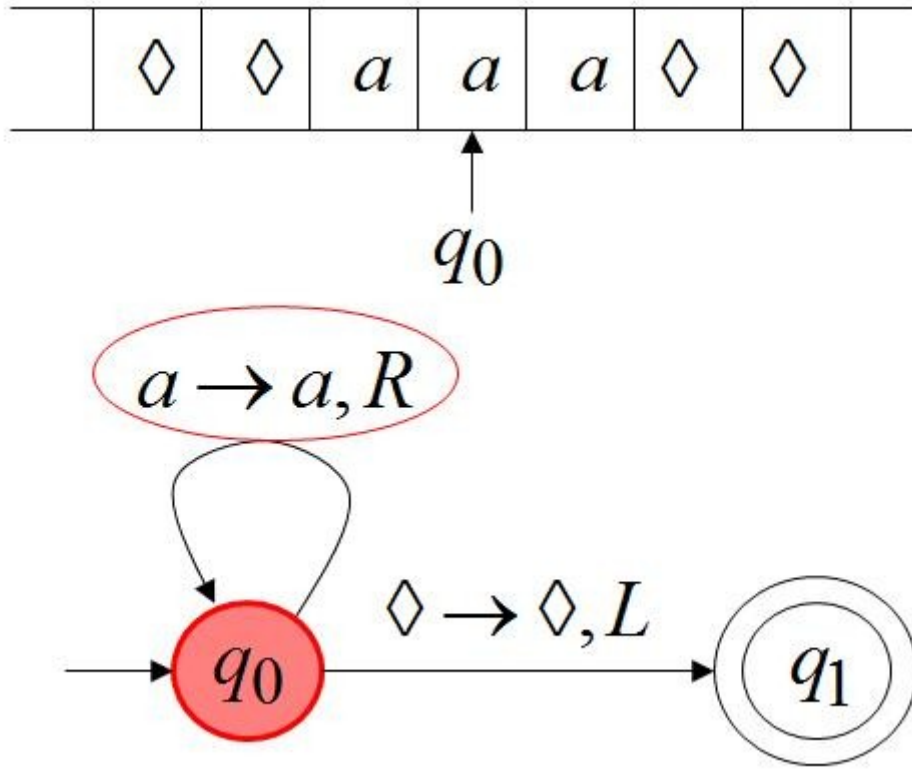
Yukarıdaki bu temsili resimde verilen turing makinesi çizilmiştir.

Makinemizin örnek çalışmasını ve bant durumunu adım adım inceleyelim.

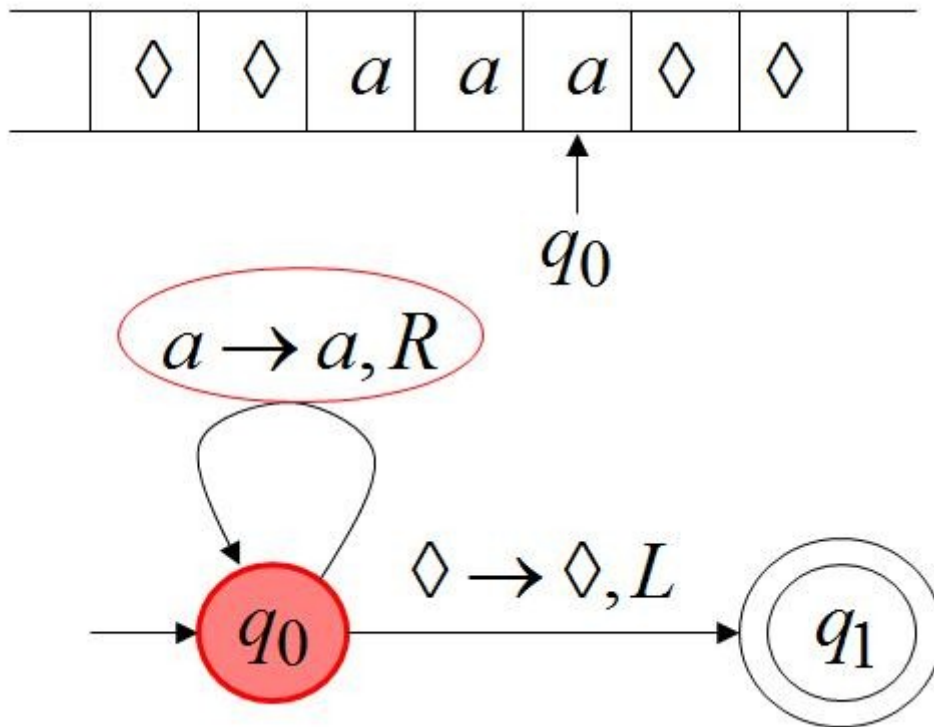
Birinci adımda bandımızda aaa (3 adet a) yazılı olduğunu kabul edelim ve makinemizin bu aaa değerini kabul edip etmeyeceğini adım adım görelim. Zaten istediğimiz de aaa değerini kabul eden bir makine yapabilmektir.



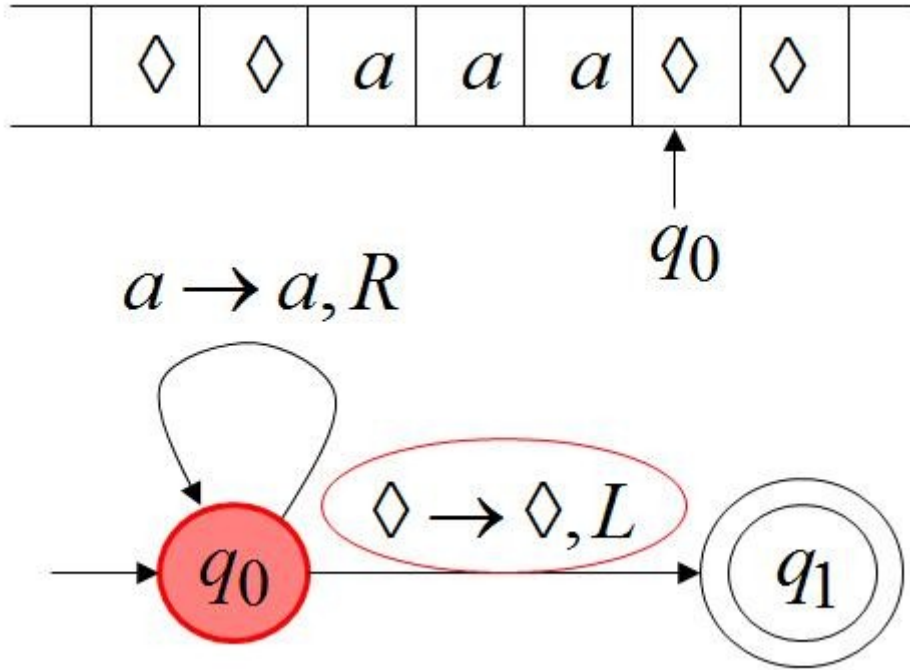
Yukarıdaki ilk durumda bant üzerinde beklenen ve kabul edilip edilmeyeceği merak edilen değerimiz bulunuyor. Makinemizin kafasının okuduğu değer a sembolü. Makinemizin geçiş tasarımına göre q_0 halinde başlıyoruz ve a geldiğinde teybi sağa sarıp yine q_0 durumunda kalmamız gerekiyor.



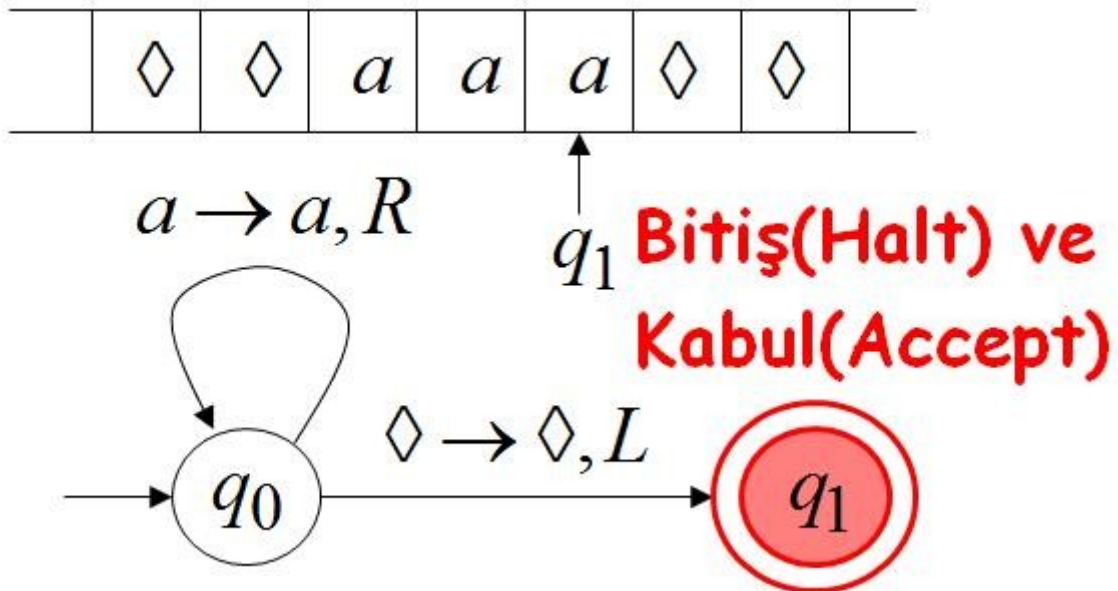
Yeni durumda kafamızın okuduğu değer banttaki 2. a harfi ve bu durumda yine q_0 durumundayken teybi sağa sarıp yine q_0 durumunda kalmamız tasarlanmıştır



3. durumda kafamızın okuduğu değer yine a sembolü olmakta ve daha önceki 2 duruma benzer şekilde q_0 durumundayken a sembolü okumanın sonucu olarak teybi sağa sarıp q_0 durumunda sabit kalıyoruz.



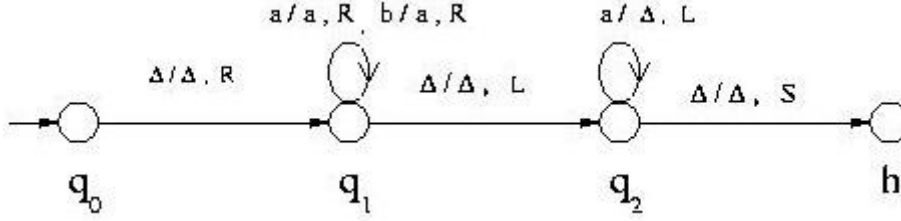
4. adımda teypten okuduğumuz değer boşluk sembolü x oluyor. Bu değer makinemizin tasarımında q_1 durumuna gitmemiz olarak tasarlanmış ve teybe sola sarma emri veriyoruz.



Makinenin son durumunda q_1 durumu makinenin kabul ve bitiş durumu olarak tasarlanmıştı (makinenin tasarımındaki F kümesi) dolayısıyla çalışmamız burada sonlanmış ve giriş olarak aaa girdisini kabul etmiş oluyoruz.

2. Örnek

Hasan Bey'in sorusu üzerine bir örnek makine daha ekleme ihtiyacı zuhur etti. Makinemiz $\{a,b\}$ sembolleri için çalışsın ve ilk durum olarak bandın en solunda başlayarak bandta bulunan sembolleri silmek için tasarlansın. Bu tasarımı aşağıdaki temsili resimde görülen otomat ile yapabiliriz:



Görüldüğü üzere makinemizde 4 durum bulunuyor, bunlardan en sağda olan h durumu bitişi (halt) temsil ediyor. Şimdi bu makinenin bir misal olarak “aabb” yazılı bir bandta silme işlemini nasıl yaptığını adım adım izah etmeye çalışalım.

Aşağıda, makinenin her adımda nasıl davranacağı bant üzerinde gösterilmiş ve altında açıklanmıştır. Sarı renge boyalı olan kutular, kafanın o anda üzerinde durduğu bant konumunu temsil etmektedir.

◇	◇	a	a	b	b	◇
---	---	---	---	---	---	---

q0 durumunda başlıyoruz. Ve boşluk ile bandı sağa sarıyoruz:

◇	◇	a	a	b	b	◇
---	---	---	---	---	---	---

a veya b değeri okundukça bant sağa sarılmaya devam ediyor ve q1 durumunda kalıyoruz.

◇	◇	a	a	b	b	◇
---	---	---	---	---	---	---

◇	◇	a	a	b	b	◇
---	---	---	---	---	---	---

Okunan değer b ise banda geri a değeri yazılıyor

◇	◇	a	a	a	b	◇
---	---	---	---	---	---	---

www.bilgisayarkavramlari.com

◇	◇	a	a	a	a	◇
---	---	---	---	---	---	---

En sağda boşluk değerini okuyunca (◇) Sağa sarma işlemi bitiyor ve geri dönüyoruz

◇	◇	a	a	a	a	◇
---	---	---	---	---	---	---

◇	◇	a	a	a	◇	◇
---	---	---	---	---	---	---

◇	◇	a	a	◇	◇	◇
---	---	---	---	---	---	---

◇	◇	a	◇	◇	◇	◇
---	---	---	---	---	---	---

◇	◇	◇	◇	◇	◇	◇
---	---	---	---	---	---	---

Tekrar boşluk (◇) görülünce makine bitiyor.

Geri sarma işlemi sırasında a değerleri silinmiş oluyor

Netice olarak Hasan Bey'in sorusuna temel teşkil eden ve örneğin q1 üzerindeki döngülerden birisi olan b/a,R geçişi, banttan b okunduğunda banta a değerini yaz manasındadır.

SORU 13: Özyineli Sayılabilir Diller (Recursively Enumerable Languages)

Muntazam dillerden (formal languages) birisi olan ve bu özelliği ile Mantık, Matematik ve Bilgisayar bilimlerinin çalışma alanına giren bir dil çeşididir. Sınıflandırma olarak Chomsky Hiyerarşisinde (Chomsky Hierarchy) 0. seviye olan (Type 0) bu dile uygun bütün diller birer düzenli ifade (regular expression) ile gösterilebilir.

Muntazam dil (formal language) olması dolayısıyla dilde üretilen her özyineli sayılabilir kelime kümesi (recursively enumerable set) bütün kelimelerden çıkarılabilecek güç kümesinin (power set) bir alt kümesi olmak zorundadır. Bu anlamda bir özyineli sayılabilir dili aşağıdaki üç farklı tanımla tanımlamak mümkündür:

Bir dilde bulunan alfabeden üretilebilen bütün kelimeleri kabul eden dil yani Σ sembolü ile dilimizdeki alfabe yi yani harfler kümesini gösterecek olursak L ile gösterilen dilimiz Σ^* ile üretilebilen kelimeler kümesidir.

[Turing makinesi \(Turing machine\)](#) ile işlenebilen veya [hesaplanabilir bir fonksiyon \(computable function\)](#) bulunabilen dildir. Burada dikkat edilecek nokta dilin sonsuz olabileceğidir. Yani dilimizde sonsuz sayıda tekrar bulunabilir. [Turing makinesi](#) ve [hesaplanabilirlik teorisi \(computability theory\)](#) dikkatle okunacak olursa dilin sonsuz olmasının bir sakıncası yoktur. Teorik olarak sonlu zamanda bitiyor olması yeterlidir ve dil sonsuz bile olsa sonlu zamanda işleyen bir makine veya fonksiyon bulunabilir.

Şayet bir dilde üretilen bir [dizgi \(string\)](#) için bir [Turing makinesi \(turing machine\)](#) veya [hesaplanabilir bir fonksiyon \(computable function\)](#) bulunuyorsa, diğer bir ifadeyle ürettiğimiz [turing makinesi](#) şu üç durumdan birisini yapıyorsa:

- [bitmek \(halt\)](#)
- [döngü \(loop\)](#)
- red etmek (reject)

bu durumda bu dil özyineli sayılabilir dildir denilebilir.

Özyineli sayılabilir diller temel olarak [chomsky hiyerarşisinde \(chomsky hierarchy\)](#) bulunan bütün diğer dilleri kapsar. Bu anlamda hiyerarşinin en alt seviye dildir ve diğer dillere göre çok daha esnektir.

SORU 14: Chomsky Hiyerarşisi (Chomsky Hierarchy)

Bilgisayar bilimlerinin özellikle dil alanında yapılan çalışmalarında [muntazam dilleri \(formal languages\)](#) tasnif etmek için kullanılan bir yapıdır. Literatürde Chomsky–Schützenberger hiyerarşisi olarak da geçmektedir.

Bilindiği üzere ([muntazam diller \(formal languages\)](#) veya [CFG](#) yazısından da okunabileceği üzere) [muntazam dillerin](#) dört özelliği bulunur. Bunlar özellikle [içerikten bağımsız dillerin \(context free languages\)](#) da temelini oluşturmuştur.

- sonlular (terminals)
- devamlılar (nonterminals)
- üretim kuralları (devamlıların değerlerini belirleyen geçiş kuralları)
- Başlangıç devamlısı

Örneğin aşağıdaki [içerikten bağımsız dilbilgisini \(context free grammar\)](#) ele alalım

$$S \rightarrow AB$$

$$A \rightarrow Aa \mid \varepsilon$$

$$B \rightarrow b$$

Yukarıdaki bu [CFG](#) tanımındaki sonlular (terminals) $\{a,b,\varepsilon\}$, devamlılar (nonterminals) $\{S,A,B\}$ olarak tanımlanır. üretim kuralı olarak (production rules) S,A,B 'nin açılımlarını

gösteren ve \rightarrow sembolü ile belirtilen satırlar bulunmaktadır. Son olarak başlangıç devamlısı (nonterminal) değeri olarak S kabul edilmiştir. Başlangıç devamlısı böyle bir kural bulunmamasına karşılık genelde S harfi (start kelimesinden gelmektedir) ile gösterilmekte ve ilk satırda bulunmaktadır.

Yukarıdaki bu CFG şayet düzenli ifadeye (regular expression) çevrilirse a^*b şeklinde yazılabilir. Aslında burada anlatılan değer $a^n b$ şeklinde de gösterilebilen istenildiği kadar a değeri alan (hiç almaya da bilir) ve sonra mutlaka b ile biten dildir.

Chomsky yukarıdaki şekilde tanımlanan muntazam diller (formal languages) için bir sınıflandırmaya gitmiş ve 4 seviye belirlemiştir.

- Type-0 (tip 0) Sınırlandırılmamış diller (unrestricted grammars)
- Type-1 (tip 1) İçerik duyarlı diller (context-sensitive grammars)
- Type-2 (tip 2) İçerikten bağımsız diller (context free languages)
- Type-3 (tip 3) Düzenli diller (regular grammars)

şeklinde 4 seviye isimlendirilmiştir. Bu seviyelerde iler gidildikçe (seviye arttıkça) dili bağlayan kurallarda sıkılaşmakta ve dil daha kolay işlenebilir ve daha belirgin (deterministic) bir hal almaktadır.

Tip-0 dilleri basitçe turing makinesi (turing machine) tarafından çalıştırılabilen ve bir şekilde bitecek olan dillerdir (hesaplanabilir diller (computable languages)). Örneğin özyineli sayılabilir diller (recursive enumerable languages) bu seviyeden kabul edilir.

Tip-1 dilleri için içerik duyarlı diller örnek gösterilebilir. Basitçe $\alpha A \beta \rightarrow \alpha \gamma \beta$ şeklindeki gösterime sahip bir dildir. Buradaki A devamlı (nonterminal) ve α, γ, β birer sonlu (terminal) terimdir. Bu sonlulardan α ve β boş harf (yani ϵ veya bazı kaynaklarda λ) olabilir ancak γ mutlak bir değere sahip olmalıdır (yani boş olamaz) Ayrıca bu tipte

$$S \rightarrow \epsilon$$

şeklinde bir kurala da izin verilmektedir. Burada S devamlısının sağ tarafta olmaması gerekmektedir. Bu diller doğrusal bağlı otomatlar (linear bounded automaton) ile gösterilebilir. Doğrusal bağlı otomatlar, turing makinelerinin özel bir halidir ve Turing makinesinde bulunan bantın sabit uzunlukta olduğu (çalışmanın sabit zaman sonra sona ereceği) kabul edilir.

Tip-2 diller ise CFG ile ifade edilebilen içerikten bağımsız dillerdir (context free languages). Bu dillerin özelliği $A \rightarrow \gamma$ şeklinde gösterilmeleridir. Buradaki γ değeri sonlular (terminals) ve devamlılar (nonterminals) olabilmektedir. Bu diller aşağı sürüklemeli otomatlar (push down automata PDA) tarafından kabul edilen dillerdir ve hemen hemen bütün programlama dillerinin temelini oluşturmaktadırlar. (programlama dillerinin neredeyse tamamı bu seviye kurallarına uymaktadırlar)

Tip-3 diller ise düzeli ifadeler (regular expressions) ile ifade edilebilen (veya üretilebilen veya parçalanabilen (parse)) dillerdir. Bu dillerin farkı

$$A \rightarrow \alpha \text{ ve}$$

$$A \rightarrow \alpha B$$

şeklindeki kurallar ile gösterilebilmeleridir.

Yukarıdaki bu tanımları bir tabloda toplamak gerekirse:

Seviye	Dil Örneği	Otomat Uygulaması	Kuralları
Type-0	Recursively enumerable	Turing machine	$\alpha \rightarrow \beta$
Type-1	Context-sensitive	Linear-bounded non-deterministic Turing machine	$\alpha A \beta \rightarrow \alpha \gamma \beta$
Type-2	Context-free	Non-deterministic pushdown automaton	$A \rightarrow \gamma$
Type-3	Regular	Finite state automaton	$A \rightarrow \alpha$ ve $A \rightarrow \alpha B$

Yukarıdaki seviyeler bütün dilleri kapsamak için yeterli değildir ayrıca yukarıda gösterilen seviyelere giren yukarıdaki tablo dışında diller de bulunmaktadır.

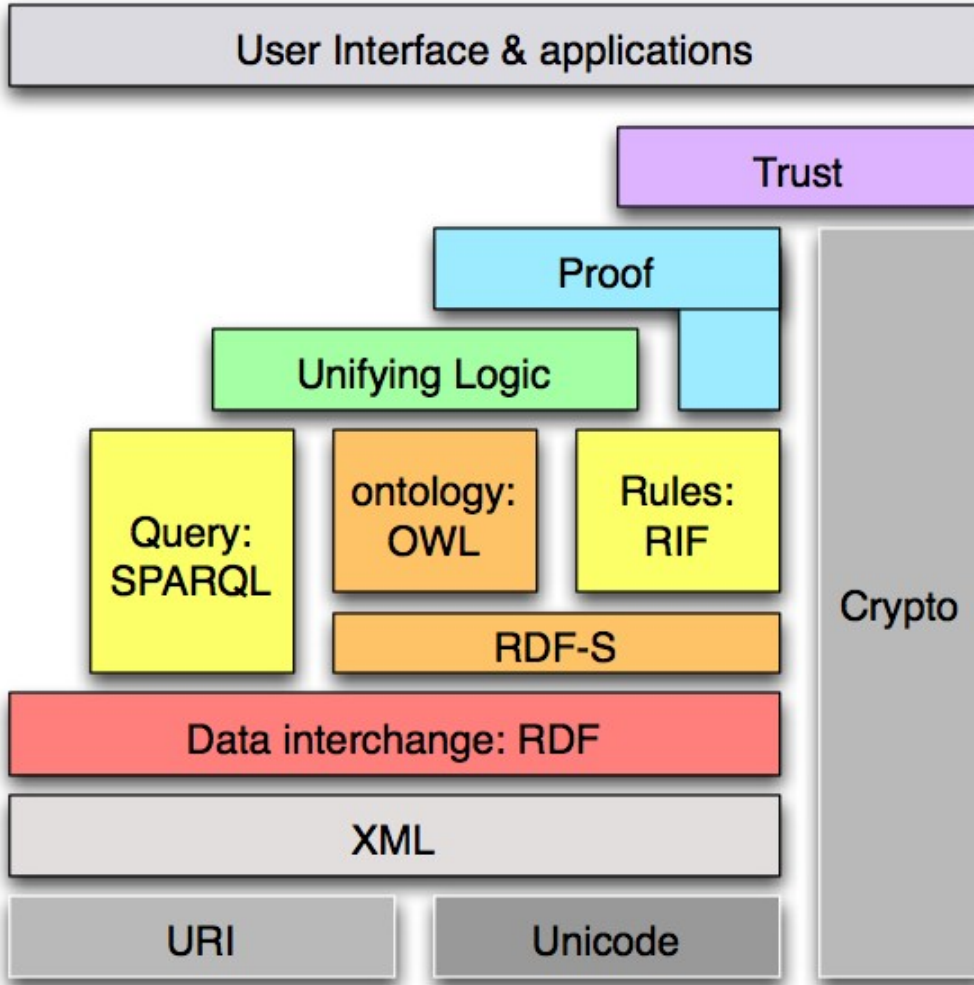
SORU 15: Mana Ağları (Sematic Webs, Anlamsal Ağ)

İnternetin (world wide web) bir alt uzayı olarak düşünülebilecek anlambilimsel ağlar, internet üzerinde bulunan ve doğal dilde yapılan yayınlara bir alternatiftir.

Anlambilimsel ağlar, bir bilgi kaynağının makinelere (bilgisayarlar) tarafından işlenebilecek ve bu işleme sonucunda anlamı tam olarak anlaşılabilen ağlardır.

Mânâ ağları (anlambilimsel ağlar) üzerinde yapılan çalışmalarda henüz tam bir kesinlik yoktur. Farklı çalışma grupları, farklı standart ve alanlarda çalışmaktadırlar. Ancak [Kaynak Tanım Çerçevesi \(resource description framework\)](#) ve XML kullanımları neredeyse standartlaşmıştır. Örneğin [varlıkbilim \(onthology\)](#) seviyesinde yapılan [web ontology language \(OWL\)](#) çalışması [RDF](#) kullanımının bir örneği olarak görülebilir. Ayrıca [malumat seviyesi gösterimler \(knowledge representation\)](#) için çeşitli alternatifler de bulunmaktadır.

Mânâ ağlarının çıkış sebebi bilgisayarların malumat seviyesinde (knowldege) işlem yapma yetersizliğidir. Bir bilgisayarın işleyebileceği ve işlemi sonucunda çıkarabileceği anlam ile insanların anladığı anlam çok farklıdır. Örneğin “maymun” kelimesinin hangi dilden Türkçeye girdiğini araştırmak ve cevap olarak Asurcadan girdiğini bulmak insan için ulaşılabilir bir araştırma süreci iken, bilgisayarların bu konuda işlem yapmaları çelişik açıklamaları temizlemeleri ve bir malumat sahibi olmaları günümüz teknolojisinde insana göre oldukça ilkindir. İşte bu sebeplerden oldayı İnternet üzerinde [HTML](#) ile yapılan ve insanların okuyup anladığı sayfaların yanında bilgisayarların anlayabildiği mana ağları türemiştir.



Yukarıda W3C'nin sitesinden alınmış mana ağı yığını (semantic web stack) resmi görülmektedir. Yukarıdaki değişik katmanlarda bulunan teknolojiler w3c tarafından standartlaştırılmış teknolojilerdir.

Yukarıdaki [yığına \(stack\)](#) bakılınca en alt seviyede URI (Unique resource identifier) seviyesi görülmektedir. Bu seviye mana ağlarının bulunduğu servis sağlayıcılarının internet üzerindeki adresleridir. Yine alt seviyelerden Unicode ile anlatılmak istenen ise verinin işlenmesi ve iletimi sırasında kullanılacak olan kodlama standardıdır.

Örneğin XML dosyalarının saklanması ve iletimi sırasında [XML dosyalarının](#) UTF (unicode transormation table) tablolarına uygun olması istenmektedir.

Yukarıdaki şekilde dikkat edilebilecek bir nokta ise Crypto katmanının en üst seviyeye kadar çıkıyor olmasıdır. Bunun sebebi kullanılacak olan dijital imza (digital signature) ve benzeri güvenlik uygulamalarının her katmandan kontrol edilebiliyor olması ve katmanların (örneğin [XML](#) veya [RDF](#)) güvenlik parçaları barındırmamasıdır. Yani örneğin [XML teknolojisinin](#) içerisinde doğrudan bir güvenlik çözümü söz konusu değildir. Yine kullanıcı katmanı (user interface & application) ile geri kalan mana ağı yığını (semantic web stack) arasında bulunan güven (Trust) katmanı da bu amaca hizmet etmektedir ve kullanıcının şifre katmanı (crypto) ile erişimini ayarlamaktadır.

SPARQL yine w3c tarafından standartlaştırılmış [RDF](#) sorgulama dilidir. Yani bir RDF dökümanını işleyerek sorgulamaya yarayan dillerden birisidir.

SORU 16: Muntazam Diller (Formal Languages)

Kısaca istisnası olmayan dillere muntazam dil diyebiliriz.

Muntazam diller bilgisayar bilimlerinde, mantıkta ve dilbilim (linguistic) çalışmalarında kullanılan bir dil ailesidir. Dilde bulunan bütün öğeler ve dilin ulaşabileceği sınırlar belirli kurallar dahilinde tanımlanabiliyorsa bu dillere muntazam dil ismi verilir.

Bu anlamda bilgisayar bilimlerinde bulunan bütün programlama dillerini bu ailede düşünmek mümkündür.

Temel olarak bir dildeki harfleri bu harflerden oluşabilecek [kelimelerin oluşma kurallarını \(morphology\)](#) ve bu [kelimelerin dizilimini \(syntax\)](#) muntazam bir şekilde belirleyebilen kurallarımız varsa. Ve bu kuralların istisnası yoksa bu dilin bir muntazam dil olduğu söylenebilir.

Muntazam dilimizi tanımlamak için öncelikle dilin en ufak ögesi olan harflerden başlamak gerekir.

Alfabe ve Harf tanımı

Bir dilde kullanılan sembollerin her birisine harf denilir. Bu harflerin listesine alfabe (bazı kaynaklarda abece) denilir.

Örneğin [haber mantığını \(predicate logic\)](#) ele alacak olursak bu mantığın ifade ettiği dil için $\wedge, \neg, \forall, ,) , ($ sembolleri kullanılır. Ayrıca mantığımızdaki değişkenleri ve önermeleri gösteren x_0, x_1, x_2, \dots şeklinde sonsuz harf bulunmaktadır.

Kelime (word) tanımı

Bir alfabedeki sembollerin (harflerin) o dildeki kelimebilim(morphology) kurallarına uygun olarak oluşturduğu dizilime kelime ismi verilir. Kelimeler dildeki anlamlı en küçük birimlerdir ve tek başlarına sadece bir birimlik bilgi ifade ederler. Daha karmaşık bilgi ifadeleri için cümlelere ihtiyaç duyulur. Ne yazık ki dilbilimde bir kelimenin anlattığı bilgi ile bir cümlemin anlattığı bilgi arasında kesin bir çizgi bulunmamaktadır. Örneğin bazı doğal dillerde tek kelime ile anlatılan bir duygu veya olay, başka dillere bir kelime grubu hatta bazan bir cümle ile çevrilebilmektedir.

Muntazam diller için de aynı durum söz konusudur. Dilin tasarımında hangi bilginin yeterli varlık oluşturacağı ve ne kadar detaya inileceği kararlaştırılır.

Hatta bir dilin var olması için sözdizim (syntax) seviyesinin bulunması gerekmez. Yani sadece kelimeler olan ve kelimelerin dizilmesi gerekmeyen diller de bulunabilir.

Bir dilde üretilebilecek kelimelerin belirli kurallarla tanımlanması ve bu kuralların dışına çıkan istisna bulunmaması durumunda bu dile muntazam dil denilebilir.

Örneğin [düzenli ifadeleri \(regular expressions\)](#) ele alırsak, bu ifadelerin kullanılması ile dilde oluşabilecek sonsuz sayıdaki kelime belirlenebilir.

$(a + b)^*$ şeklindeki bir düzenli ifadeden a ve b harfleri kullanılarak (istenilen sıra ve istenilen sayıda) kleene yıldızı sayesinde boş kümeyi de kapsayan bütün kelimeler üretilebilir.

Muntazam Dil tanımı

Yukarıdaki dilin öğelerini tanımladıktan sonra dili tanımlamak mümkündür.

Σ sembolü ile dilimizdeki alfabeyi yani harfler kümesini gösterecek olursak L ile gösterilen dilimiz Σ^* ile üretilebilen kelimeler kümesidir.

Bu tanıma ilave olarak muntazam dil kavramının matematik ve bilgisayar bilimlerinde geçen pek çok “dil” kelimesini karşıladığını söyleyebiliriz. Yani aslında matematik ve bilgisayar bilimlerinde geçen “dil” kelimesi aslında tam olarak “muntazam dile” karşılık gelmektedir.

Örneğin [içerikten bağımsız dil \(context free language\)](#) terimi aslında içerikten bağımsız muntazam dil anlamında kullanılmaktadır. Benzer şekilde düzenli diller (regular languages) terimi aslında düzenli muntazam dil (regular formal languages) anlamında kullanılmaktadır. Dolayısıyla matematik ve bilgisayar bilimlerinin uğraştığı bütün dilleri ve kullandıkları bütün dilleri muntazam dil (formal language) olarak tanımlamak mümkündür. Bunun tek istisnası bilgisayar bilimlerindeki yapay zeka konusunun altında geçen doğal dil işleme konusudur.

Muntazam dil çalışmaları

Muntazam diller üzerinde yapılan çalışmalardan birincisi bir muntazam dilin nasıl tanımlanacağıdır. Bu alanda yapılan bazı çalışmalar sonucunda elde edilen yaklaşımları aşağıdaki şekilde sayabiliriz:

- Muntazam bir dilbilgisi (formal grammer) tarafından üretilen dillere Muntazam dil (formal language) ismi verilir. Bu tanım [Chomsky Hiyerarşisi](#) tarafından sınıflandırılan dilbilgileri ile gösterilebilir.
- [Düzenli ifadelerin \(Regular expressions\)](#) bir örneği ile üretilebilen dillere muntazam dil ismi verilir.
- Bir otomat tarafından üretilen ([sonlu durum otomatu \(finite state automat\)](#) veya [Turing Makinesi \(Turing Machine\)](#) gibi) dillere muntazam dil ismi verilir.

Yukarıdaki bu tanımlama çalışmalarının yanında muntazam diller ile ilgili aşağıdaki sorulara da cevap aranmaktadır.

- Tanımlama yeteneği. Bir dilin tanımlayabilme gücü ve tanımlayabildiği dilin büyüklüğü nedir? Aynı güce ve yeteneğe sahip ikinci bir dil farklı bir şekilde olabilir mi? Bütün dilleri gösteren bir dil yapılabilir mi? şeklindeki soruları bu grup altında toplayabiliriz.
- Algılama yeteneği. Verilen bir kelimenin verilen bir dile ait olup olmadığının algılanması yeteneğidir. Elimizde muntazam bir dilbilgisi (grammer) olduğunu kabul edelim. Bu dilden olan veya olmayan bir kelime verildiğinde bu kelimenin analiz edilmesi ve dile ait olup olmadığının algılanması en verimli nasıl yapılabilir? sorusuna cevap arayan çalışmalardır.

- Karşılaştırma yeteneği. Elimizde iki dil bulunsun. Bu iki dilin birbirine göre farklarının algılanılması, bir dilde olup diğer dilde olmayan veya iki dilde ortak olan kelime ve cümlelerin tespiti gibi konularda çalışılan çalışma grubudur.

Yukarıdaki bu problemler genel olarak algoritma analizi (analysis of algorithms) altındaki karmaşıklık teorisi (complexity theory) alanın çalışma konularına girmektedir.

Diller üzerinde tanımlı işlemler

Diller üzerinde de farklı işlemlerin yapılması mümkündür. Aşağıda bu işlemler ve tanımları verilmiştir:

Dillerin kesişimi (intersection): Temel olarak bir dili o dildeki kurallar ve alfabe marifetiyle üretilebilecek kelimeler kümesi olarak tanımlarsak, iki dilin kesişimi aslında iki dilde üretilebilecek kelimelerin kesişimi olmuş olur. $L_1 \cap L_2$ şeklinde gösterilir.

Üleştirme işlemi (concatenation): İki dilden üleştirme yapmak için birinci dilden bir kelime ile ikinci dilden bir kelimenin arka arkaya eklenmesi ile yeni bir kelime oluşturulması kastedilir. Yani L_1 dilinden k kelimesi ile L_2 dilinden l kelimesi alınıp kl veya lk şeklinde kelime üretme işlemidir. L_1L_2 şeklinde gösterilir.

Tümleyen işlemi (Complement): Bir dildeki [güç kümesinden \(power set\)](#) o dildeki dilbilgisine (grammar) göre çıkarılabilen kelimelerin farkıdır. Yani alfabede olan harflerle yazılabilecek bütün kelimelerin, dilde izin verilen kelimelerle [fark kümesidir](#). $\neg L$ şeklinde veya \overline{L} şeklinde gösterilir.

Kleene Yıldızı (Kleene Star): Bir dildeki alfabe üzerinde boş küme dahil olmak üzere üretilebilecek bütün kelimeler kümesidir. Bir anlamda güç kümesi (power set) ismi de verilebilir. Kullanımı aslında [düzenli ifadelerden \(regular expressions\)](#) alınmıştır.

Ters işlemi (reverse): Bir dildeki herhangi bir kelimenin yazılışının terse çevrilmiş halidir. Örneğin “abc”nin tersi “cba” şeklindedir. Daha akademik olarak iki kuralla tanımlanabilir.

- Şayet dilimizdeki boş kelimeyi ϵ sembolü ile gösterirsek $\epsilon^R = \epsilon$ olur yani boş kelimenin tersi yine boş kelimedir.
- Boş kelime dışındaki herhangi bir kelime için $w = x_1 \dots x_n$ şeklinde n adet harf ile yazıldığını düşünelim, bu durumda tersi $w^R = x_n \dots x_1$ olur

Bu durumda muntazam bir dil L için, $L^R = \{w^R \mid w \in L\}$ olarak tanım yapılabilir.

SORU 17: Anlamsal Bağ (Semantic Link)

Bilgisayar bilimlerinde yapay zeka konusunda özellikle de doğal dil işleme ile ilgili yapılan çalışmaların önemli bir kısmını [anlambilim \(semantics\)](#) kaplar. Kısaca bir metin veya ortamdan elde edilen bilginin anlamını çıkarmak ve bu anlamı kullanışlı hale getirmek [anlambilimin \(semantics\)](#) çalışma alanına girmektedir.

Anlambilimsel bağlar ise bu çıkarımı ve gösterimi yapılan bilgilerin birbirine nasıl bağlandığını tutmaktadır. Bir anlamda iki etki alanı (domain) arasındaki geçişi gösteren [fonksiyonlar \(functions\)](#) olarak kabul edilebilirler.

Bu bağların kendileri de ayrıca anlambilimsel çalışmanın bir parçası olmaktadır. Örneğin aşağıdaki cümleyi ele alalım:

“Ali, Ayşe ile Ahmetin oğludur”

Yukarıdaki bu cümlede üç kişiden bahsedilmektedir : Ali, Ahmet, Ayşe

Şayet Ahmet’in erkek ismi ve Ayşe’nin de kadın ismi olduğu biliniyorsa yukarıdaki cümleden çıkarılabilecek anlamlardan bazıları aşağıdadır:

- Ali’nin annesi Ayşedir
- Ali’nin babası Ahmettir
- Alinin yaşı Ahmetin yaşından küçüktür.
- Ayşenin yaşı Alinin yaşından büyüktür.

Ayrıca yukarıdaki cümlelere ilave olarak [istatistiksel dilbilim \(probabilistic linguistic\)](#) ile olaya yaklaşırsak:

- Ahmetin karısının ismi Ayşedir.
- Ayşenin kocasının ismi Ahmettir.

Sonuçlarına varılabilir. Buradaki gizli kabul (hidden premises) Ahmet ile Ayşenin evli olduklarıdır.

Yukarıdaki bu çıkarımların tamamı anlambilimsel bağ (semantic link) örnekleridir. Yani anlamını çıkardığımız üç varlık (Ali,Ahmet ve Ayşe) arasındaki bağları göstermektedirler.

Anlambilimsel bağlar, [Anlambilim ağlarının \(Semantic web ve Semantic Network olarak geçmektedir\)](#) temelini oluşturmaktadırlar.

Benzer çıkarımlar daha makine diline yakın olan formal diller için (formal languages) yapılabilmektedir.

Örneğin XML dökümanlarında benzer çıkarımlar yapılabilir. Bir kütüphanenin [XML](#) dosyasını düşünelim. Muhtemelen kitap bilgisi tutulurken yazar, yayın evi, yayın yılı gibi ilave bilgilerle tutulacaktır.

<kitap yazar=”Şadi Evren ŞEKER” isim=”Programlama ve Veri Yapılarına Giriş” />

şeklinde bir satırdan oluşan XML girdisini yorumlarsak.

- “Programlama ve Veri Yapılarına Giriş” isimli bir kitap vardır
- “Programlama ve Veri Yapılarına Giriş” isimli kitabı “Şadi Evren ŞEKER” yazmıştır.
- “Şadi Evren ŞEKER” bir kitap yazmıştır.

benzeri sonuçlarına varılabilir.

Benzer durumlar farklı ortamlarda yapılan analiz çalışmaları ile çıkarılabilir. Buradaki amaç ulaşılan anlambilimsel neticedir.

SORU 18: Sözdizim (Syntax)

Temel olarak bir dilde (language) tanımlı olan öğelerin (kelime, işlem, sembol yada değerlerin) anlamlı bir dizilim oluşturmasıyla ilgilenen bilimdir.

Örnekler

Örneğin Türkçe için aşağıdaki cümle anlamlı bir cümledir:

"Ali okula geldi"

yukarıdaki cümlede herhangi bir yazım hatası ve dizilim sorunu bulunmamaktadır. Bununla birlikte:

"okula Ali geldi"

cümlesi de Türkçe açısından uygun bir cümle iken

"geldi okula Ali"

cümlesi hem devrik hem de anlamını yitirmiş bir cümledir. Benzer durum matematiksel işlemler için de aşağıdaki şekilde gösterilebilir:

(3 + 4)

Yukarıdaki matematiksel işlem dizilim açısından sorunsuzken:

(3 4 +)

Yukarıdaki halde de yazılabilir (postfix notation).

3) (4 +

Ancak yukarıdaki yazılış dizilim açısından hatalıdır.

Yukarıdaki örneklerde görüldüğü üzere her dilin kendisine göre bir dizilimi ve dizilime göre farklı anlamları bulunmaktadır. İşte sözdizim (syntax) bu konuda çalışmaktadır.

Anlam çıkarımı

Sözdizim çalışmalarının önemli bir kısmı da dilin işlendiği bilgisayar tarafından girilen cümleye (Statement) anlam verilmesidir. Cümlelerin anlamlarını çalışan [anlambilim \(Semantics\)](#) açısından genelde cümlenin bir anlamını gösteren model bulunmakta ve anlam çıkarımında hedeflenen amaç girilen cümleyi bu modele çevirmektedir.

Bu çevirim için çeşitli yaklaşımlar mevcuttur. Örneği [yacc](#) gibi sözdizim (syntax) analizi yapan ve makine dilleri (machine languages) için başarılı bir şekilde çalışan ve yapısında [içerik bağımsız dil \(context free language\)](#) barındıran, kod üretme programları bulunmaktadır. [Yacc](#) ve benzeri kod üreticilerinde (code builder) hedeflenen dil genelde [CFG](#) ile gösterilebilen bir makine dilidir ve başarı oranı oldukça yüksektir.

Buna karşılık doğal dillerin analizi ve [semantik gösterimi \(semantic representation\)](#) için tam başarı ne yazık ki neredeyse imkansızdır. Ayrıca doğal dilin analizi sonucunda elde edilen sonucun gösteriminde de ciddi problemler bulunmaktadır.

Cümle üretimi (Syntax Generation)

Anlam çıkarımına benzer şekilde, ifade edilmek istenen anlamın üretilmesi de bir problemdir. Örneğin 4 sayısını toplama işlemi ve pozitif tamsayılar kümesi ile göstermek istersek:

1+3
2+2
3+1

gösterilmesinin tamamı üretmek istediğimiz değer olabilir. Benzer şekilde doğal dil üretimi (natural language generation) yukarıdaki örneğe göre oldukça karmaşık bir üretim yapısını desteklemek zorundadır. (Yukarıdaki örnekte yaşanan [karmaşıklık \(ambiguity\)](#) çözümü için belirli sonlu otomat (deterministic finite automata) kullanılabilir, burada sadece probleme örnek verilmiştir)

Örneğin Ali'nin okula gelmesinden, ders dinlemesinden ve eve dönmesinden bahsetmek isteyelim. Muhtemel bir bilgisayar çıktısı:

Ali okula geldi. Ali ders dinledi. Ali eve döndü.

Şeklinde olabilir. Ancak dikkat edileceği üzere insanlar konuşurken bu tip cümleler sıralamazlar. Örneğin

Ali okula geldi. O ders dinledi. O eve döndü.

Şeklindeki cümleler daha insana yakındır. ([dönüşü \(anaphora resolution\)](#) kullanılmıştır)

Benzer şekilde:

Ali okula geldi, ders dinledi ve eve döndü.

Cümlesi insan kullanımına çok daha yakındır ([ellipsis \(hazf\)](#) kullanılmıştır)

Yukarıda görüldüğü üzere doğal dil üretimi sırasında da söz dizimden faydalanılmaktadır.

Şablon Yapısı (templates)

Bu sözdizim çalışması yaklaşımına göre dilde bulunan cümlelerin anlamlarının anlaşılması (veya bu dilde bir cümle üretimi) için çeşitli şablonlar kullanılır. Bir söz dizim çalışmasının en temelini oluşturan şablon kullanımı (çeşitli kaynaklarda çerçeve (Frame) olarak da isimlendirilmektedir). Oldukça katı(rigid) bir yapıya sahiptir. Anlamak için şablona mutlak uyum şarttır. Ürettiği sonuçlar ise ancak şablon yapısında olmaktadır.

Kategorik Dilbilgisi (Categorical Grammer)

Sözdizim çalışmalarında dilin modellenmesi sırasında kullanılan bir dilbilgisi örneğidir. Bu yapıda dilde bulunan öğeler anlamsal olarak gruplanır ve modellenir. Örneğin “bahçeli okul”

şeklindeki bir tamlamada aslında iki farklı kelime vardır. Kategorik dilbilgisinde bu iki kelimenin oluşturduğu gruba dilbilgisindeki bir kategori ismi verilir.

Genelde çok sık kullanılan kategoriler NP ve VP olarak da geçen isim kelime grubu (noun phrase) ve fiil kelime grubudur. (verb phrase)

Bağılılık Dilbilgisi (Dependency Grammar)

Dilbilgisindeki kelime veya kelime gruplarının tanımlanması ve sıralarının belirlenmesine ilave olarak şayet bu kategoriler arasında bir bağlantı tanımı yapılıyorsa bu dilbilgisi çeşidine bağılılık dilbilgisi ismi verilir.

Basitçe bir öznenin, bir fiilin faili (agent) olması durumu bu grup gramer tanımındandır.

Örneğin cebirsel sözdizimler (algebraic syntax) bu gruptan sayılabilir.

Fonksiyonel Dilbilgisi (Functionalist Grammar)

Bu dilbilgisi grubunda ise dilde bulunan kelimelerin ve kelime gruplarının kullanım amaçları ve üstlendikleri görevler değerlendirilir.

Örneğin “At, tay doğurdu” cümlesindeki doğurmak fiili üretken bir fiildir ve modellenen dünyaya yeni bir varlık ekler. Buna karşılık “Ali peynir yedi” fiili tüketken bir fiildir ve modellenen dünyadan bir varlık eksilir. Bu durumda iki fiili iki ayrı grupta ele almak mümkündür.

Stokastik Dizilimler (Stochastic Grammars)

[İstatistiksel dilbilimin \(Probabilistic Linguistic\)](#) çalışma alanı olan bu sözdizim çeşidinde çeşitli olasılık ağları (örneğin [markov modelleri \(markov chain\)](#)) vasıtasıyla dizilim modellemesi yapılır.

SORU 19: Anlambilimsel Tertip (Semantic Composition)

Doğal dil işleme çalışmaları sırasında bir metinden ([derlem \(corpus\)](#)), paragraf, cümle veya kelimedenden çıkarılan anlamın bilgisayar tarafından bir şekilde modellenmesi gerekmektedir.

Bu modelleme sırasında kullanılan gösterim çeşitlerine anlambilimsel tertip ismi verilir. [Anlambilimsel dilbilgisi \(semantic grammar\)](#) kadar kesin kuralları olmayan bu gösterim şekillerinde çoğu zaman doğru veya yanlış kaygısı güdülmeden sadece bir bilginin gösterimi hedeflenir.

Anlambilimsel tertipleri [cümle gösterimlerinin \(Syntactic representation\)](#) ötesinde fonksiyonel ve çalışabilir tertipler olarak görmekte yarar vardır.

Örneğin en çok kullanılan ve en etkili gösterimlerden birisi [birinci derece haber mantığıdır \(first order predicate logic\)](#). Bu mantığa göre örneğin aşağıdaki şekilde bir gösterim yapılabilir.

“Ali uyudu” şeklindeki bir cümleyi ele alalım. Bu cümleden aşağıdaki şekilde birinci derece haber mantığı gösterimi elde edilebilir:

uyudu (Ali) .

Basitçe uyudu isminde bir fonksiyonumuz varsa bu fonksiyonun parametresi de Ali olmaktadır. Bu durum özellikle birinci derece haber mantığını kullanan diller için çok önemlidir. Örneğin PROLOG yukarıdaki bu satırı doğrudan kod olarak kabul edip çalıştırabilir.

Farklı bir örnek olarak lambda matematiğinde (lambda calculus) modelleme yapmaya çalışalım.

“Ali peynir yedi” şeklindeki bir cümleyi lambda matematiği ile modellersek:

(λa) (λb) yedi (a,b)

şeklinde gösterebiliriz. Yukarıdaki bu gösterimde modelimizde bulunan [değişkenler \(variables\)](#) birer lambda değişkeni olarak tanımlanmıştır. Yani yukarıdaki gösterime göre a ve b isminde iki değişken yedi fonksiyonunun parametresidir. Benzer bir durum için:

(λa) yedi (a,peynir)
(λb) yedi (ali,b)

şeklinde bir gösterim de yapılabilir. Yukarıdaki ilk satırda “peynir yemek” eylemi modellenirken ikinci satırda Ali tarafından yenilen şeyler modellenmiştir.

Lambda matematiğindeki bu gösterim de LISP (veya Scheme) gibi bu gösterimi doğrudan kullanan diller için oldukça önemlidir.

Yukarıdaki örneklere ilave olarak modelin daha detaylı hale getirilmesi de mümkündür. Örneğin yukarıdaki cümlelerde fiil zaman, şahıs, kip gibi bilgilerinden arındırılmadan olduğu gibi fonksiyon haline çevrilmiştir. Oysaki bu bilgilerin de modelde birer gösterimi bulunabilir.

uyumak (Ali, geçmiş) .

Yukarıdaki birinci derece haber mantığında gösterilen satıra göre Ali uyuma eylemini geçmişte yapmıştır. Elbette bu gösterim daha önceki gösterime göre daha detaylıdır ve doğal dil işlemesi yapacak bilgisayar için daha kıymetlidir çünkü ilk gösterimde bilgisayarın olay zamanı hakkında fikri yokken artık olayın zamanını parametre olarak almaktadır.

Ancak burada bir problem olayın modellenmesi sırasında kullanılan yöntemin ne kadar bilgi içerebileceğidir. Örneğin yukarıdaki gösterime göre uyuma eyleminin geçmişte olduğunu tutuyoruz ancak bilindiği üzere “uyudu” fiili aslında söyleyen kişinin uyuma eylemi yapılırken orada olduğunu göstermektedir (örneğin uyumuş fiili de geçmiş zamanlıdır). Dolayısıyla bu ilave bilginin de tutulması gerekir.

Yukarıdaki basit örnekten anlaşılacağı üzere bir anlambilimsel model geliştirildiğinde ve kullanılırken doğal dilde çok basit ve kısa ifadelerle anlatılabilen bilgilerin analiz edilip çeşitli alternatifler ile bir ortamda tutulması gerekir.

SORU 20: İstatistiksel Dilbilim (Probabilistic Linguistic)

Bilgisayar mühendisliğinin bir alt kolu olan yapay zeka çalışmalarında amaçlardan birisi de insan gibi davranan veya insanı anlayarak yorum yapabilen yazılımlar elde etmektir. Bu amaçla makine insan sınırının (man machine boundary) üzerinde çeşitli çalışmalar yapılmıştır. Bu çalışmaların önemli bir kısmı da insanın kendini ifade yöntemi olan ve diğer insanlarla iletişim kurmasına yarayan dilbilim üzerindedir.

Dilbilim çalışmaları ve bilişim (cognition, cognitive) konusunda yapılan çalışmalar, yani insanın bilgmesi ve bildiğini kullanması ile ilgili yapılan çalışmaların bilgisayarlar üzerine uyarlanması doğal dil işleme (natural language processin) alanının çalışma konusudur.

Bu alanda çeşitli yaklaşımlar bulunmaktadır. Bu yaklaşımlardan birisi de bu yazının konusu olan istatistiksel dilbilim yaklaşımıdır. Bu yaklaşıma göre bilgisayarların doğal dildeki bilgileri anlamasını istatistik bilimine dayandırmak düşünülmüştür.

Öncelikle dilin ve dildeki öğelerin bilişsel (cognitive) yaklaşımını incelemeye çalışalım. Örneğin bir otoyolun kenarında durduğunuzu ve ortalama 50km hızla geçen arabaların arasından karşıya geçmek istediğinizi düşünelim. Böyle bir yolda geçmiş tecrübeleriniz ve o anlık gözlemlerinizle yola kontrolsüz bir şekilde atlarsanız ezilip zarar göreceğiniz (bir arabanın çarpacağı) neredeyse kesindir.

Tam bu anda yanınıza birisi gelip size karşıya geçmek için düğmeye basabileceğinizi ve bu düğme sayesinde arabalara kırmızı ışık yanarak arabaları durdurabileceğinizi söylüyor olsun. Bu bilgi benzer şekilde trafik ışıklarının üzerinde yazılı da olabilir. Normalde belki de böyle bir bilgiyi elde etmeseniz deneyerek bulmanız veya kullanmanız mümkün olmayabilir ancak bu bilgi size bir dil aracılığıyla ulaşabiliyor ve siz başka birisinin size ilettiği bu bilgiyi öğrenerek kullanabiliyorsunuz.

İşte mevcut gözlemlerle veya geçmiş tecrübelerle öğrenilemeyen ancak dil vasıtasıyla (yazılı veya sözlü) iletilen bilgi, bilgisayar dünyasındaki doğal dil işleme çalışmalarının ve bilişsel bilimin (cognitive science) temelini oluşturur. Tam bu noktada artık bu bilginin istatistik ile ilişkisini tanımlayabiliriz.

Örneğin doğal dilden bir kelime ele alalım ve anlamını nasıl anladığımızı veya bilgisayarın nasıl anlayabileceğini tartışalım. Kelimemiz “uzun” olsun. Görüldüğü üzere uzun kelimesi belirli bir geçmiş bilgiyi gerektirir. Her insanın uzun kelimesi ile düşündüğü farklı bir tecrübesi veya kafasında canlanan bir sembol olabilir. Ancak uzun kelimesini tanımlayın desek genelde herkes “NORMALDEN uzun (mesafe boy veya uzunluk birimi olarak fazla) olan” şeklinde bir tanım yapabilir. Bu tanımda dikkat edilecek nokta normal kelimesidir. Demek ki herkesin kafasında bir de normal kavramı var ve hepimizin bildiği üzere bu normal kavramı istatistiksel bir birikimdir ve muhtemelen herkes kendi geçmişindeki normal ile ölçer. Örneğin “uzun boylu insan” tamlamasındaki uzun için insanlar geçmiş insan boylarına göre bir normal kavramı elde edip bununla karşılaştırabilirler.

Benzer bir örnek de güncel konularda olabilir. Örneğin bu yazının yazıldığı gün itibariyle milliyet gazetesinin web sitesindeki başlıklardan birisi “Musevi yandaşları tahrandaki büyük bir gösteri düzenliyor” şeklindeydi. Haberin detayı belki siz bu yazıyı okurken anlaşılmayacak kadar geçmişte kalmış olabilir ancak gündem itibariyle iranda seçimler yeni yapıldı ve

Ahmedinejad seçimleri kazandır. Rakibi olan Musevi'nin taraftarları ise İranda çeşitli yerlerde gösteri yapıyor ve seçimi protesto ediyor.

Şimdi gündemi bilmeyen bir gözle bu başlığı analiz edelim. Burada bilinmesi gereken [isimler](#):

- Musevi
- tahrân
- gösteri

Bu isimlerin dışında betimleyici (tamlayan) görevinde ve yüklem görevinde çeşitli ilave kelimeler de var. Peki bir bilgisayar sizce yukarıdaki isimler için ne düşünür.

Musevi: Hz. Musa'nın isminde türeyen kutsal kitabı Tevrat olan ve Yahudi ırkının inancı olarak gelen dini inanış yada Mir Hüseyin Musevi ismindeki son seçimlerde reformcu kanadın temsilcisi İranlı.

Tahrân : Farsçada sıcak yer anlamına gelen kelime yada 1943 senesinde Churchill, Stalin ve Roosevelt'in, müttefiklerin Batı Avrupa'ya çıkarma yapmasına karar verdikleri konferansın adı yada İran'ın başkenti.

Gösteri: Bu kelimeyi tek başına düşündüğümüzde çok çeşitli anlamlarda yerde ve zamanlarda düşünebiliriz.

Peki biz insan olarak yukarıdaki bu isimleri nasıl anlıyoruz. Elbette yukarıdaki bu isimler için belki de bütün bu bilgileri biliyoruz ancak okuduğumuz zaman gündemi de biliyorsak tek bir anlam veriyor ve diğer bütün ihtimalleri eliyoruz.

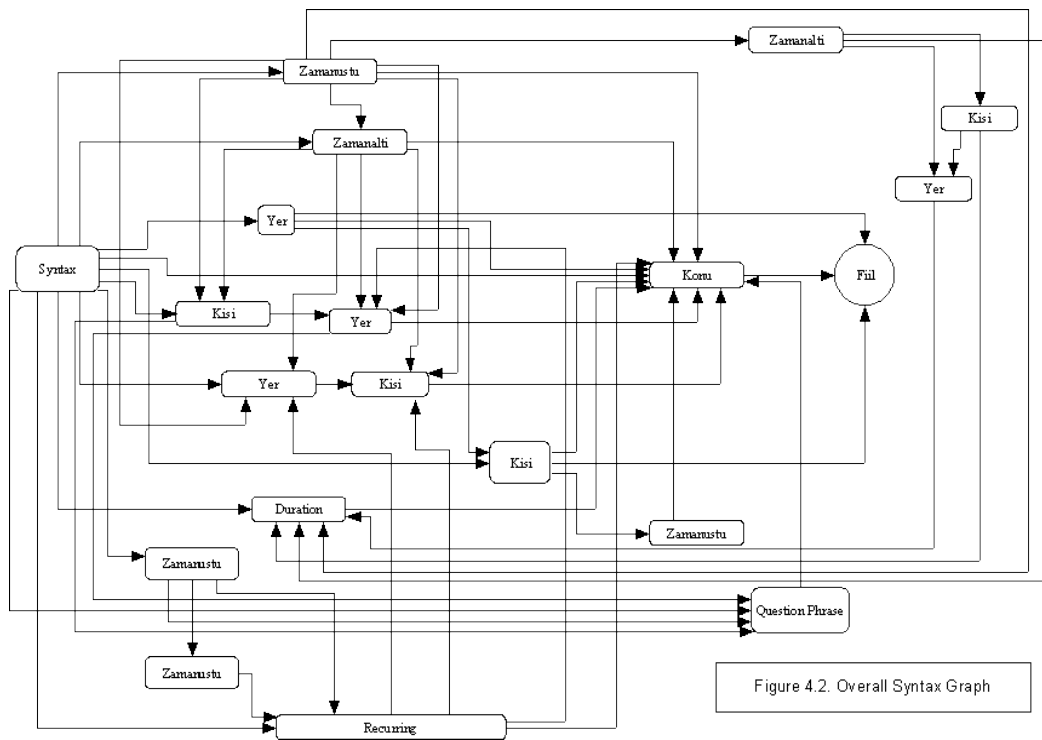
Bunun sebebi aslında bu isimleri ele aldığımızda bütün bu ihtimalleri eşit düşünüyor ancak bağlantılı isimlerle ve gündemle birlikte kelimenin bazı isimlerini azaltıyor olmamızdır. Dolayısıyla aslında bu cümlenin anlamını analiz ederken istatistiksel olarak kelimenin güncel kullanımına göre bir anlam yüklüyoruz. Hatta bu durum bazı kelimelerde o kadar çok olmaktadır ki zaman içinde kelimenin anlamı kaymakta ve aynı kelime farklı zamanlarda farklı anlamlara gelmektedir. Bunun sebebi kelimeyi kullanan kişilerin kelimeye yüklediği anlamın değişmesidir.

Aslında her kelimenin sözlüklerdeki anlamı o kelimeyi kullanan kişi sayısına göre belirleniyor demek yanlış olmaz. Dolayısıyla çok uç bir bakış açısına göre bütün dilbilim ve bütün anlam bilim istatistikten ibarettir bile denilebilir.

SORU 21: Augmented Transition Network (ATN, Uzatılmış Geçiş Ağı)

Bilgisayar bilimlerinde özellikle de yapay zeka konusunda ve buna bağlı diğer alt dallarda (örneğin doğal dil işleme) kullanılan bir graf teorisi (graph theory) gösterimidir. Kelime anlamı olarak uzatılmış geçiş ağı (tehir-i intikal şebekesi) denilen ağların amacı toplanan bilgilere göre bir karar vermek ve karar verme işlemi sırasında da bir belirsizlik (ambiguity) bulunuyorsa, karar verme işlemini yeterli bilgi toplanana kadar tehir etmektir (geciktirmektir, augment). Ağın ismi de buradan gelmektedir.

Örneğin yukarıdaki şekilde Türkçe cümleler ile randevuları tutan bir takvim programı uygulaması olan yüksek lisans tezinden alınma doğal dildeki kelime grupları arasındaki geçişleri gösteren bir ATN görülmektedir.(tezin detaylarına www.shedai.net/tusa adresinden erişilebilir)



Input: "on ocak ikibin ile on mart ikibiniki arasında haftada bir ali bey ile bahçeli okulda saat onda ikişer saatlik toplantılar var"
 (There are meetings with mister ali at the school with garden between tenth of January two thousand and tenth of march two thousand and two with the period of two weeks and each one is two hours long at ten o'clock)

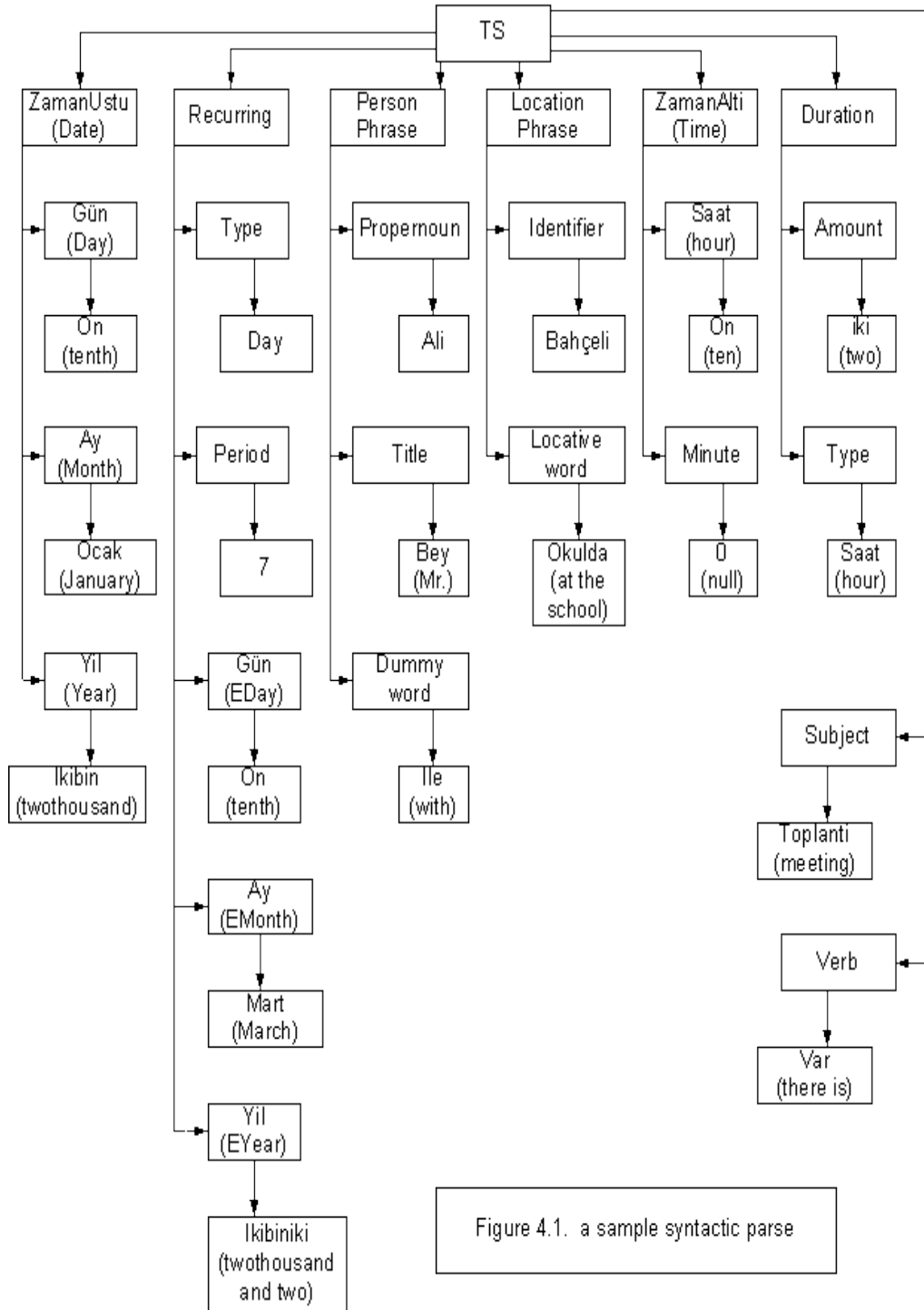


Figure 4.1. a sample syntactic parse

Yukarıda ise bir önceki şekilde görülen ATN kullanılarak “on ocak ikibin ile on mart ikibiniki arasında haftada bir ali bey ile bahçeli okulda saat onda ikişer saatlik toplantılar var” cümlesinin [parçalama ağacı \(parse tree\)](#) gösterilmiştir.

SORU 22: Definite Clause Grammer (Belirli Cümle Dilbilgisi DCG)

Kısaca DCG olarak da adlandırılan bu yapılar, doğal dil işleme konusunda şablon (template) yapılarından daha farklı ve daha insana yakın bir yapıya sahiptir. DCG'ye göre cümlede bulunan yazıların birer [dizgi \(String\)](#) veya harf olarak görülmesinden öte bu yazıların anlamsal olarak gruplanması beklenir.

Örneğin

“Ali okula doğru emin adımlarla yürüyordu”

Cümlesindeki öğeleri inceleyecek olursak:

Ali : Özne

Okula doğru : Yer yön zarfı (tümleç)

Emin adımlarla : Hal durum zarfı (tümleç)

Yürüyordu : Yüklem

Şeklinde sıralayabiliriz. Bu sıralamada dikkat edilirse kelimeler veya kelime grupları farklı gruplara bölünerek anlamsal bir sınıflandırmaya gidilmiştir.

Yukarıdaki sınıflandırmaya aykırı kelime grupları da bulunur. Örneğin

“doğru emin” kelimeleri cümlemin bir parçasıdır ancak insan olarak bu kelimelerin anlamsız bir grup olduğunu söyleyebiliriz.

Ayrıca yukarıdaki kelime gruplarını

Ali: İsim grubu

Okula doğru : isim grubu

Emin adımlarla: isim grubu

Yürüyordu : fiil grubu

Olarak da sınıflandırmak mümkündür. Burada da anlamsal olarak tamlamalar ve belirteçlerden yararlanarak kelimeler gruplanmış ancak bu defa [anlamlarına göre \(lexical analysis\)](#) değil [yapılarına göre \(grammatical analysis\)](#) sınıflandırılmıştır.

Yukarıdaki iki sınıflandırmada da aslında kullanılan belirli bir dilbilim söz konusudur. DCG yaklaşımında bu dilbilim dili oluşturan grupların nasıl dizileceğini belirleyen [içerik bağımsız dilbilgisidir \(Context Free Grammar\)](#). Yukarıdaki cümleyi içeren bir [CFG](#) aşağıdaki şekilde yazılabilir:

C  İG FG

İG  İG İG | İG

İG → B İ | İ B | İ

FG → F

B → “doğru” | “emin”

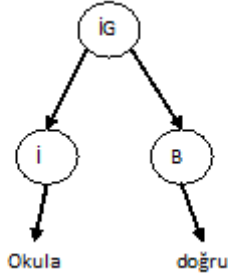
İ → “okul” | “Adım” | “Ali”

F → “yürümek”

Yukarıda verilen CFG ve dolayısıyla DCG için C, cümle; İG, isim grubu ; B, belirteç ; İ, isim; FG, fiil grubu ; F , fiil şeklinde kısaltılmıştır.

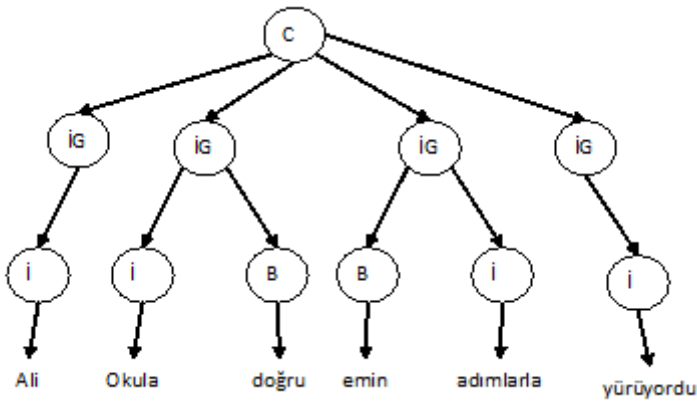
Buna göre örneğin

“Okula doğru” tamlaması için:



Yukarıdaki şekilde isim ve belirteç ve daha sonra da isim grubu denilebilir.

Benzer şekilde ilk örneğimiz olan “Ali okula doğru emin adımlarla yürüyordu” cümlesini aşağıdaki şekilde parçalama (parse) mümkündür.



Yukarıda görülen [parçalama ağacı \(parse tree\)](#), ilgili CFG’nin örnek için uyarlanmış halidir.

SORU 23: Pragma (Edimbilim, kullanımbilim, Fiili, Ameli)

Genel olarak dildeki cümlelerin ve [kelimelerin](#) anlattıklarından daha ötede bulunan anlamı ifade eder. Örneğin bir kişiye “saatin var mı?” diye bir soru sorulursa buradaki anlam aslında saatin kaç olduğunun sorulmasıdır. Yani bu cümledeki pragmatik ifade saatin sorgulanmasıdır. Yukarıdaki bu soruya kişi “Evet var” şeklinde bir cevap verirse sorudaki pragmayı kaçırmış olur.

Doğal dil işleme çalışmalarında pragma önemli bir rol oynar. Bir bilgisayarın pragmatik yapıyı anlayamaması durumunda doğru cevap vermesi beklenemez. SPEECH ACTS

Pragma konusundaki önemli kavramlardan birisi de (îmâ, IMPLICATURE)’dur. Buna göre bir kişinin kurduğu cümleden yapılabilecek çıkarımlar yine pragma çalışmasının parçasıdır. Örneğin

“Benim iki çocuğum var”

Cümlesinden benim çocuğum olmadığı veya tek çocuğum olduğu çıkarımları hatalı olur. Ancak bu cümleyi duyan birisi benim 3 çocuğum olduğunu söylerse bu iddia yukarıdaki cümle ile çelişmez. Yani 3 çocuğu olan birisi için 2 çocuğu olduğu da doğrudur.

“Benim sadece iki çocuğum var”

Cümlesi bu anlamda daha doğrudur. Ancak pragma çalışmasına göre yukarıdaki ilk cümleden de benim 3 çocuğum olduğu sonucunu çıkarmamak gerekir.

Diğer önemli bir konu da önkabuldür (presupposition). Örneğin

“kızım çok zekidir”

Cümlesine göre bir kızım olduğu kabul edilmiştir.

Önkabul ile ilgili bir durum da bu kabulün iptalidir (cancel). Örneğin

“Bu sene bütün notlarım sınıf ortalamasına eşit”

“Çünkü sınıfta tek öğrenci var”

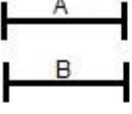
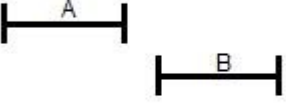
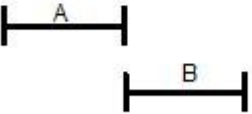
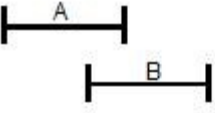
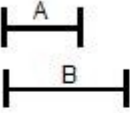
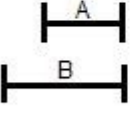
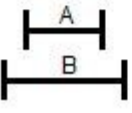
Yukarıdaki ilk cümlede sınıf ortalaması ile kişinin sınıfı [grup ismi \(mass noun\)](#) kabul edeceği ve sonucunda kalabalık (birden fazla) olacağını düşünmesi beklenir. İkinci cümle ile bu ön kabul iptal edilerek sınıfın tek kişilik olduğu söylenmiştir.

Pragma çalışmaları ayrıca [Söylev \(Discourse\)](#) çalışmalarını da kapsar. Bu çalışmaların amacı birden fazla cümlelerin birbirine bağlanma şeklini incelemektir.

SORU 24: Allen Fasıla Mantiğı (Allen's Interval Logic)

1983 yılında James F. Allen tarafından ortaya atılan ve zaman modellemek için kullanılan mantıktır. Bu mantık bilgisayar bilimlerinde zaman çıkarımına dayalı doğal dil çalışmalarında önemli bir modelleme şeklidir. Mantık basitçe olayları ve olaylar arasındaki ilişkileri modellemek üzerine kuruludur. Bu

ilişki şekillerini göstermek amacıyla bir tablo hazırlanmıştır. Aşağıda bu tablo görülebilir:

	A ile B eşit B ile A aynı zamanda
	A, B'den önce B, A'dan sonra
	A'yı B takip eder B, A tarafından izlenir
	A ile B çakışır B, A ile çakışır
	A, B'yi başlatır B, A tarafından başlatılır
	A, B'yi bitirir B, A ile biter
	A esnasında B olur B, A'yı kapsar

Yukarıdaki tabloda, Allen fasıla mantığının 7 farklı durumu gösterilmiştir. Bu tablo kullanılarak herhangi bir olaylar arası bağlantı modellenilebilir.

Örneğin “Ali eve geldiğinde Ayşe uyuyordu” cümlesinde bulunan iki [eylem \(fiil, verb\)](#) (gelmek ve uyumak) birbirlerine göre 7. tip ilişkidir. Yani bir eylemin oluşu ve diğer eylemin oluşu aynı esnada olmuş ancak uyuma eylemi gelme eyleminden önce başlamış ve sonra bitmiştir.

Allen Zamansal Varlıkbilim (Allen’s Temporal Ontology)

Yukarıda açıklanan fasıla mantığının yanında Allen tarafından bir varlıkbilimsel yaklaşım da ortaya konmuştur. Bu yaklaşımda üç tip varlıkbilimsel öğeden söz etmek mümkündür:

Özellikler (Properties)

Eylemler (Events)

İşlemler (Processes)

Özellikler basitçe bir fasılanın bütün fasıllar üzerinde kılınması durumudur. Yani bir özellik veya eylemin bütün zamanlara mâl edilmesidir.

Örneğin “Ali zekidir” cümlesinde bir özellikten bahsedilmekte ve bu özelliğin zamanı bulunmamaktadır. Yani bütün zamanlar için geçerli bir önermedir.

Benzer şekilde “Ali iki gündür hasta” cümlesi de bir özellik olarak sınıflandırılabilir. Evet iki günlük süre bütün zamanları kapsamamaktadır ama mevcut [pragma](#) üzerine tanımlı olan bütün zamanlar iki günlük zaman dilimidir. Yani aynı cümle “Ali bir haftalık tatilden sonra iki gündür hasta” şeklinde değişecek olsaydı bu durumda hastalık özellik olmaktan çıkacak bir eyleme dönüşecekti.

Eylemler (olaylar , fiiller, events) sadece bir fasıla üzerinde tanımlı olan ve bütün fasılları kapsamayan fiillerdir. Örneğin “Ali öğleden sonra okula gitti” eyleminde olayın geçtiği süre öğleden sonra ve zaman kavramı sadece bununla sınırlıdır.

İşlemler (processes) ise bir alt fasılanın genele hakim olmasını ifade eden eylemdir. Örneğin “Ali bugün okula gidiyor” cümlesinde okula gitme eylemi, zamanın bütününe yayılmıştır.

Yukarıdaki bu üç sınıfı daha net ayırmak için şöyle bir sınıflandırma yapılabilir:

şayet bir fasılanın (interval) bütün alt fasılları kapsanıyorsa özellik (properties)

şayet bir faslı için bir fiil ifade ediliyor ama alt fasıllar için (subinterval) bir eylem ifade etmiyorsa eylem (event)

şayet bir fasılanın bazı alt fasılları için eylem ifade ediyorsa işlem (process) olarak sınıflandırılabilir.

SORU 25: Haber (Predicate)

Bilgisayar bilimlerinde önemli bir rol oynayan dilbilimi ve dil felsefesinin önemli unsurlarından birisidir. Bir cümlenin iki önemli unsurundan birisi olarak kabul edebiliriz. Haber-müpteda ilişkisi (Subject-Predicate) veya Özne-yüklem ilişkisi de denilebilir.

İçerik

1.	Haberin	dilbilimsel	incelemesi
2.	Dilbiliminde	haber	sınıfları
a.		Hal	haberleri
b.		kişisel	haberler
c.		Nevi	Haberler
3. Dağıtıcı ve birleştirici tip haberler			

Yani örneğin “Benim kalemim kırmızı” cümlesindeki “Benim kalemim” müptedayı (özne,subject) ifade ederken “kırmızı(dır)” kelimesi ise haberi ifade etmektedir.

Burada üretilen haber-müpteda birleşimlerinden kaziyeler mantığı inşa etmek (önergeler, prepositional logic) mümkündür. Dolayısıyla aslında yapılan ifadenin üzerine bir matematik inşası da mümkündür. Zaten böylede olmuştur ve Haber Cebiri (Predicate Calculus) ismi verilen ve özellikle yapay zeka programlamasında oldukça önemli bir yer tutan bir matematik dalı mevcuttur.

Bu yazıda öncelikle dilbilim ve doğal dil işleme açısından haber (predicate) konusu incelenecek ardından bu konu üzerine inşa edilmiş matematik (predicate calculus) incelenecektir.

1 Haberin Dilbilimsel İncelemesi

Güncel [anlambilimsel \(semantic\)](#) çalışmalara göre bir haber (predicate) doğruluğu ifade etmek için kurulan bir cümle parçasıdır. Yani kalemim kırmızıdır cümlesindeki ifade gibi. Ayrıca bu ifadeyi güçlendirmek veya genişletmek için çeşitli sıfat ve isimlerden yararlanılabilir. Müpteda-Haber şeklinde olan cümlelere isim cümlesi ismi de verilir.

Örneklerle durumu açıklamak gerekirse:

Ali uykudadır. (isim cümlesi, haberi hal durum zarfı)

Ali kitap okuyordur. (isim cümlesi doğrudan nesneyi işaret ediyor, Şimdiki zaman)

Ali parkta. (isim cümlesi, yer yön zarfı)

2 Dilbiliminde haber sınıfları

Dilbiliminde haberler anlamları bakımından çeşitli sınıflarda incelenebilir. Greg N. Carison tarafından yapılan bu sınıflama aşağıda verilmiştir.

a Hal haberleri (State level predicates, s-l predicates , h-s haber)

Bir hali, durumu bildiren haberlerdir. Ali uykulu. Ali aç. Ali kokmuş. cümlelerinde olduğu gibi bir durum belirtirler. Buradaki durum belirtisi bir zamanla kısıtlıdır. Yani Alinin uykulu olması aç olması veya korkmuş olması geçici bir haldir.

b Kişisel haberler (Individual level predicates, I-l predicates, k-s haber)

Bir kişilik durumunu belirtirler. Örneğin Ali hızlıdır. Ali akıllıdır. Cümlelerinde olduğu gibi. Bu tip haberler hal haberlerinin genleştirilmesi ile de elde edilebilir. Örneğin Ali korkaktır cümlesi bir önceki örnekte bulunan Ali korkmuş haberinin genleştirilmesidir.

Bu grupta bulunan haberler kişi ile özdeşleşmiş haberlerdir ve ilgili kişiyi betimlerler.

Dilbilgisi (grammatic) olarak bu kelimeleri sınıflandırmak ve s-l veya i-l şeklinde yorumlamak ne yazık ki mümkün değildir. Kelimelerin taşıdıkları anlamlar bu grupta önemli rol oynar.

Örneğin geçmiş zamanda ifade edilen

- Ali uykuluydu
- Ali cesurdu

Cümlelerinden ilki h-s haber iken ikincisi k-s haber olmuştur.

c Nevi haberler (Kind level predicates, k-l predicates, n-s haber)

Bu haber grubunda belirtilen haberin bir varlığın çeşidini veya şeklini betimlemesi beklenir. Bu grubun kişisel haberlerden ayrılması için bir çeşit bir cins hakkında haber bildirmesi gerekir.

Kediler uysaldır

cümlesinde olduğu şekliyle bir cinsin bir nevîni belirtmektedir.

Örneğin insanlar hırslıdır cümlesini kişisel seviye haberden ayıran özellik insanın genellenmiş bir nevî oluşundandır. Benzer cümle Ali hırslıdır şeklinde olsaydı k-s haber olacaktı.

Yukarıdaki örneklerden de anlaşılacağı üzere k-s haber ile n-s haber ifade ettikleri faile göre sınıflanmaktadır.

3 Birleştirici ve dağıtıcı haberler

Haberler ifade ettikleri varlığa göre birleştirici veya dağıtıcı olabilirler. Örneğin bir haberde varlıkların birleşmesi bir başkasında ise varlıkların dağılması ifade edilebilir.

Örneğin Gazete dağıtmak, Dünyaya yayılmak, Haber yaymak, Oyuncaklarını dağıtmak gibi dağıtıcı tip haberlerin yanında

Mektupları toplamak, Oyuncaklarını topladı, Bulmacayı birleştirmek gibi birleştirici tip haberler de mevcuttur.

Haberlerin dağıtıcı veya birleştirici olması haberin ifade ettiği anlama göre sınıflandırılabilir. Ayrıca birleştirici tip habelerin ifade ettikleri varlıklar çoğul iken dağıtıcı tip haberlerin ifade ettiği varlıklar tekil veya çoğul olabilir.

SORU 26: Şekli Mantık (Kipler Mantığı, Modal Logic)

Mantığın bir türü olan şekli mantığında şekiller (modal) bir [kaziyenin \(önerme\)](#) doğruluğunu göstermek için kullanılır. Genel olarak şekil mantığında gösterilen 3 tip şekil bulunur:

- olabilirlik (possibility)
- ihtimal (probability)
- gereklilik (necessity)

Doğal dil açısından ve dilbilim gözüyle şekli mantığa bakarsak, aslında yukarıdaki bu liste ve şekil mantığının dayanağı İngilizcede açıkça kullanılan ve kelime zamana ve bakışlarını etkileyen modallara bağlıdır. Yani İngilizcede “*eventually, formerly, can, could, might, may, must, ought, need*” gibi modalların her birisi bu yukarıda sayılan üç tipten birisine girmektedir.

Türkçe için tam bir karşılığı olmayan bu şekillerin karşılıkları kipler ve fiil zamanları ile karşılanmaktadır.

Mantık bilimi açısından konu incelendiğinde aslında dört tip işlemin (operator) üzerinde tanımlı olduğu bir mantıktan bahsedilmektedir. Bu işlem tipleri aşağıdaki şekilde sıralanabilir:

- Alethic (gerekirlik)
- Deontic (Ahlaksal)
- Axiological (belistsel)
- Epistemic (Bilgisel)

Yukarıda sıralanan bu şekli mantık tipleri aşağıda açıklanmıştır.

Alethic (Gerekirlik) şekli

Bu şekli mantık türünde bir kaziyenin (önerme) yargılanması gerekirlik açısından yapılır.

Örneğin “Çember kare olamaz” cümlesindeki olamamazlık ile “Ali mühendis olamaz” cümlesindeki olamamazlık arasında fark vardır. Birincisi doğası itibariyle imkansız iken (olmaması gerekir iken) ikincisinin olmamasının gerekirliliğinden sadece bir yorum veya kanaat olarak bahsedilebilir. Dolayısıyla ilk kaziye için alethic (gerekirlik şeklinden doğru) denilirken ikinci kaziye için non-alethic (gerekirlik şeklinden yanlış) denilebilir.

Klasik olarak gerekirlik şeklini üç başlıkta toplayabiliriz:

- olabilirlik (possible) : Kaziyenin yanlış olaması gerekmiyorsa
- gerekirlik (necessary) : Kaziyenin yanlış olma ihtimali yoksa
- şartlı (contingent) : Kaziyenin yanlış olması gerekmiyorsa ve aynı anda kaziyenin doğru olması da gerekmiyorsa

Yukarıdaki bu üç grup için kaziyelerde gerekirlik aranabilir. Buradaki üçüncü madde biraz karışık gelebilir. Bu maddeye göre bir kaziyenin (önerme) aynı anda hem doğru hem de yanlış olmaması durumu basitçe “doğruluğu mümkün ama gerekli değil” şeklinde açıklanabilir.

Gerekirlik şeklini ayrıca gerekirlik dünyası açısından da incelemek mümkündür. Yani bir kaziye şayet fiziksel kurallar dahilinde gerekirse bu gerekirliğe fiziksel gerekirlik (Physical alethic) ismi verilebilir. Benzer şekilde, fiziküstü (metafiziksel, tinsel) gerekirlikten de söz edilebilir. Örneğin “Aya yürüyerek gidip geldim” cümlesindeki gerekirlik fiziksel bir gerekirli şeklidir. Benzer şekilde “Allah ikidir” cümlesi İslami açıdan non-alethic (gerekirlik şeklinden yanlış) bir önermedir.

Epistemic (Bilgisel) şekli

Bu şekil bilgiye dayalı olarak kaziyelerin (önerme) doğruluğunu sınar. Daha basit bir ifadeyle kişinin konu hakkında emin olması veya önermesini kesin olarak bildiği temellere dayandırması sorgulanır.

Örneğin “Ali borsanın yükseleceğine inanıyor” cümlesi ile ” Alinin bütün bilgisine dayanarak borsa yükselebilir” cümlesi arasında fark vardır. İlk cümlede bir kanaat bir düşünce dillendirilirken ikinci cümlede kişinin bilgisine dayanarak kesin bir yargı ifade edilmektedir.

Yani ilk cümle örneğinde kişinin tam anlamıyla inanması beklenmez veya kişinin elinde aksini de gösteren durumlar bulunabilir. Ancak ikincisi kişinin bilgisine dayanarak kesinlik arz eder.

Örneğin “borsanın yükseleceğine inanıyorum” cümlesindeki bilgi şekli ile “Davud’un peygamber olduğuna inanıyorum” cümlesindeki bilgi şekli arasında fark vardır. Birinci bilgi tipinde fiziksel bilgilere dayanarak ulaşılmış bir önerme bulunurken ikinci tip cümlelerde tinsel (metafiziksel) bir inanış (kanaat, yargı, kaziye) bulunmaktadır.

Her iki cümle yapısında da ortaya konan yargı geçmiş bilgi ve tecrübeler üzerine inşa edilen bilgisel bir yargıdır. Dolayısıyla yargı kişiden kişiye değişmekle birlikte bütün insanlar aynı yargıda bulunsun bile doğru olduğu sonucuna varılmaz.

Bu ifadeleri şekli mantıkta (modal logic) göstermek için iki önemli işlem (operator) kullanılır.

L harfi ile gösterilen işlemlerde “X ...’e inanıyor” şeklinde önermeler ifade edilir.

M harfi ile gösterilen işlemlerde ise “X’in bütün bilgisine dayanarak Y doğru olabilir” şeklinde önermeler ifade edilir.

Bazı kaynaklarda L işlemi için \Box sembolü ve M işlemi için de \Diamond sembolü kullanılmaktadır. Bu işlemler kullanılarak kaziye denklemleri çözülebilir. Yani örneğimize geri dönecek olursak Ali borsanın yükseleceğine inanıyorsa (Lp) şeklinde gösterilebilir. Benzer şekilde “Alinin bütün bilgisine dayanarak borsa yükselecektir” cümlesini de (Mp) olarak gösterebiliriz.

Bu iki cümle arasında kişinin bilgisine dayanarak ortaya konan şekil farkı bulunmaktadır.

SORU 27: Zamansal Mantık (Temporal Logic)

Bilgisayar mühendisliğinin önemli parçalarından birisi de modellemedir. Çeşitli alanlarda veri modellemesi yapılan bilgisayar bilimlerinin, modellemeye ihtiyaç duyduğu bir konu da zaman modellemesidir. Yani [kaziyelerin \(önerme, predicate\)](#) ifade ettikleri zamanı modellemek için bir sistem geliştirilmesi gerekmektedir.

Aslında antik yunan ve Aristo zamanından beri üzerinde çalışılan zaman modellemesi konusunda yapılan ilk çalışmalar nispeten ilkel sayılabilecek birinci dereceden zamansal mantık modelleri üzerine kuruludur.

Bu mantığın durumsal veya evrensel değişkenler üzerine kurulu olanına [kaziyeler mantığı \(önermeler mantığı, propositiona logic\)](#) ismi verilir.

Yine aynı zamansal mantığın ikili ihtimaller ve mantık denklemleri üzerine kurulu olması durumunda da bool cebiri, ikili cebir (boolean algebra, binary logic) isimleri verilir.

Yukarıdaki bu üç farklı mantığı anlatmak için bir örnek üzerinde duralım. Örneğin “İnsanlar ölümlüdür” kaziyesinin doğruluğunu sorgularken zamansal bir bakış izlenmez ve bütün zamanlardaki doğruluğu sınanır.

Buna karşılık “Karnım aç” kaziyesinin doğruluğu zaman göre değişmektedir. Yani bazan aç ve bazan tok olabilirim. Bu durumda ise bütün zamanlar değil sadece anlık bir tahlil yapılır. Yani aynı anda hem aç hem de tok olamam dolayısıyla tahlil eden kişi belki bu çelişkiyi sınavabilir.

Yukarıdaki bu iki örnekte kullanılan mantığın farkını görmeliyiz. Birincisi kaziyeler mantığı (predicate calculus) olarak incelenebilirken ikincisi zamansal mantık (temporal logic) için bir örnek olabilir.

Ancak unutulmaması gereken nokta her mantık için doğruluğun sorgulanabilir olduğudur.

Mantıksal İşlemler (Logical Operators)

Kısacası Zamansal mantık için geçerli olan ik durum söz konusudur. Bunlardan birincisi mantıksa önermenin doğruluğunun sınanmasıdır. Bu işlem mantıksal operatörler (logical operators) ile yapılır ve bu operatörler

- ve (And) \wedge
- veya (or) \vee
- ise (implication) \rightarrow
- değil (not) \neg

şeklinde sıralanabilir.

Zamansal İşlemler (Temporal Operators)

Ayrıca zamana ait işlemler (operators) ise aşağıdaki tabloda verildikleri gibi sıralanabilir:

- Tekli işlemler (unary operators):
 - Sonra (Next)
 - Gelecek (Future)
 - Genel (Globally)
 - Hep (All)
 - Mevcut (Exists)
- İkili işlemler (Binary Operators):
 - -e Kadar (Until)
 - -den sonra (Release)

Yukarıda listelenen bu işlemleri aşağıdaki şekillerle göstermek ve örneklerle açıklamak mümkündür.

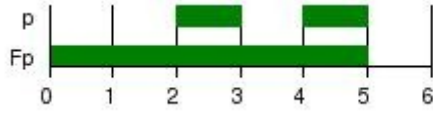


Yukarıdaki şekilden sonra operatörü anlaşılabilir. Buradaki 1-2 ve 4-5 aralığındaki eylemden sonra 2-3 ve 5-6 aralığındaki eylem gelmiştir.

İşlem (operatör) N harfi ile gösterilir (bazı kaynaklarda neXt kelimesindeki X harfi ile de gösterilir). Kaziyeye (önerme, haber, predicate) p harfi ile gösterilmiştir. Dolayısıyla ilk eylem

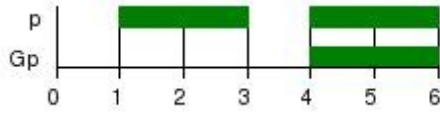
(1-2 aralığındaki eylem) bittikten sonra diğer eylem başlamıştır (2-3 aralığındaki eylem) yani p kaziyesi, Np'ye göre sonradır.

Örneğin “Şöyle bir olaydan sonra p gibi bir olay olacaktır” tipindeki cümleler bu gruba örnektir.



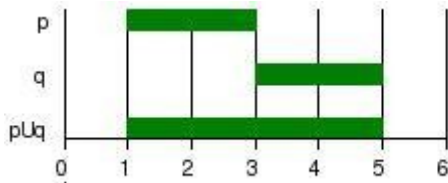
Yukarıdaki şekilde gelecek işlemi (future operator) görülebilir. İşlem F harfi ile gösterilir. Burada p önermesi Fp'ye göre ilerdir. Bazı kaynaklarda nihayet (finally) olarak da geçmektedir.

Örneğin “gelecekte bir zamanda p gibi birşey olacaktır” şeklindeki cümleler bu gruba örnektirler

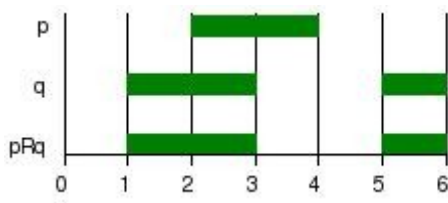


Yukarıdaki işlem genel zamanı ifade etmektedir. (Global operator) Önerme p sonradan gelen bir yolu ifade etmektedir.

Örneğin “p durumunda hep böyle olacaktır” şeklindeki cümleler bu mantık işlemine örnektirler.



Yukarıdaki örnek ikili işlemlerden (binary operators) -e kadar (until) ifadesi için çizilmiştir. Yani p ve q önermelerinden p'nin bittiği anda başlayan q ifadesi pUq ile gösterilebilir.



Yukarıdaki şekilde ise -den sonra (release) işlemi gösterilmiştir. Burada p ve q eylemleri için p'den sonra q anlamında işleme tabi tutulması söz konusudur.

Birleşik işlemler

Yukarıdaki mantık işlemlerinin bir kısmının aynı örnek üzerinde kullanılması da mümkündür. Aşağıda bir veya iki önerme için birden fazla işlemin aynı anda kullanıldığı örnekler verilmiştir.

SORU 28: Dilbilgisel Bakış (Grammatical Aspect)

Bilgisayar bilimlerinin güncel araştırma konularından olan doğal dil işleme (natural language processing) ve dil bilim (linguistics) konusundan önemli olan dilbilgisel bakış (grammatical aspect), kabaca bir fiilin zamanının başka bir fiil ile aynı bakışa sahip olmasıdır.

Yani örneğin “geldim” fiilinin zamanı ile “geliyordum” fiilinin zamanları geçmiş zamandır ancak bakışları aynı değildir. İkinci fiil Türkçe açısından birleşik zamanlı bir fiildir ve bu fiil şimdiki zamanın hikaye kipindedir. Buna karşılık geldim fiili hikaye geçmiş zamandır.

Türkçede kip ve zaman ayrımı İngilizceden farklıdır. Aslında geliyordum fiilinin eylemi geçmişte gerçekleşmiş ancak devamı (duration) devam etmiştir. Geldim fiilinin eylemi de geçmişte gerçekleşmiş ama devamı söz konusu olmamış eylem bitmiştir. Bu durumda aslında iki fiilin zamanı anlamsal olarak aynıdır. Ancak dilbilgisi açısından farklı isimlendirilirler. (Şimdiki ve geçmiş zaman olarak) bu isimlendirmeden kaynaklanan sorun doğal dil işleme çalışmalarında problem olmaktadır.

İngilizce için durum biraz daha basit mantıkla düşünülebilir.

Örneğin “I swam” (yüzdüm) ile “I was swimming” (yüzüyordum) cümlelerinin zamanları geçmiş zamandır ve gerçekten de ikisi de geçmişte olmuştur. Ancak İngiliz dilbilgisine göre sınıflandırma sonucunda ilk cümle (perfective) ikinci cümle ise (imperfective) olarak isimlendirilir. Yani zamanı aynı ancak bakışı (aspect) farklı fiillerdir.

Burada dikkat edilecek bir husus dilbilimsel bakış (grammatical aspect) ile [kelime bilimsel bakış \(lexical aspect\)](#) arasındaki farktır.

Kelime bilimsel bakışta anlam fiilin içinde bulunduğu cümle, daha önceki cümleler ve diğer etken faktörler (zarf, özne vs.) ile belirlenirken dilbilgisel bakışta zaman doğrudan fiilin dilbilgisel analizi ile belirlenir. Yani fiilin kökünün eklerinin veya doğrudan etkileyen zarfların etkisinden söz edilebilir.

Bakışlara (aspects) örnek olabilecek çeşitli dillerden toparlanmış aşağıdaki liste olayların (events) zamanı hakkında bilgi verebilir.

Tamam bakışı (Perfective) : Ali okula gitti (Ali has gone to the school). Türkçede dilbilgisel olarak tam karşılanmayan bu bakışa göre eylem tamamlanmış ve fiil hala etkisi altındadır. Yani Ali hala okuldadır.

İlerleme bakışı (Progressive, continuous) : Ali yemek yiyor. (Ali is eating) Bu bakış Türkçedeki basit şimdiki zamanlı fiil ile karşılanır. Buradaki anlam eylemin ilerlemekte olduğudur (duration)

Alışkanlık bakışı (Habitual) : Okuldan eve, evden okula gidiyorum. Bu cümledeki eylemden her gün aynı işin yapıldığı yani tekrarlı (recurring) bir eylem olduğunu çıkarabiliriz. Türkçede dilbilgisel olarak böyle bir bakış elde edilemez. Nitekim örnekte de kelime bilimsel bakış

kullanılmıştır (lexical aspect). İngilizce için “I walk home from work” (evden işe yürüyorum) cümlesinde de her gün işe yürüyerek gidildiği ve bunun bir habit (yaşam biçimi , alışkanlık) olduğu ifade edilmiştir.

Natamam bakış (Imperfective Aspect): Bir eylemin bitmediğini ifade eder. Yani eylemin tamam bakışında (perfective) olduğunun tersine bitmediğini gösterir. Bu durumda ya devam bakışı (progressive, continuous) veya alışkanlık bakışı (habitual) olan bütün bakışlar bu grupta kabul edilebilir. Yani daha önce verdiğimiz “Ali yemek yiyor” veya “Okuldan eve, evden okula gidiyorum” cümleleri bu nevi eylemin devamını söz konusu olduğu bakışlardır.

İhtimal bakışı (Prospective Aspect): Bir eylemin olma ihtimali üzerine inşa edilen bakıştır. “Ali elmayı yemek üzere” cümlesinde Alinin elma yeme eylemindeki olasılık vurgulanmıştır. Benzer şekilde “Ali birazdan elmayı yiyecek” cümlesi gelecek zamanlı bir fiil sahıptir ve bu cümlede de ihtimal bakışından söz edilebilir. İngilizce için de durum aynıdır. “Ali is about to eat the apple” veya “Ali is going to eat the apple” cümlelerinde olasılıktan bahsedilebilir.

Tamam öncesi Bakış (Recent Perfect Aspect): Bir eylem şayet tamam bakışına sahipse (perfective) ve cümlemiz bu tamamlanma eyleminden hemen öncesine bakıyorsa, bu bakışa tamam öncesi bakış diyebiliriz. Örneğin “Ali birazdan elmayı yiyecek” cümlesindeki bakışımız elmayı yeme eyleminden hemen öncesidir.

Tamam sonrası bakış (After Perfect Aspect): Bir eylemin tamamlanmasından (perfect) hemen sonraki bakıştır. Örneğin “Ali az önce elmayı yedi” cümlesinde biraz önce elmanın yendiğini (olayın tamamlandığını (perfective)) anlıyoruz ve cümle zamanımızın bakışı bunu gösteriyor.

Başlama Bakışı (Inceptive Aspect): Bir eylemin başlamak üzere olduğunu gösteren bakıştır. Örneğin “Ali yemeğe başlıyor” cümlesindeki bakış yemek eyleminin başlamak üzere olduğunu gösterir.

Nihayet Bakışı (Terminative Aspect): Bir eylemin bitmek üzere olduğunu gösteren bakıştır. Örneğin “Ali yemeğini bitiriyor” cümlesindeki bakış yemek eyleminin bitmek üzere olduğunu gösterir.

Devamsal Bakış (Continuative Aspect): Bir eylemin devam etmekte olduğunu gösterir. “Ali yemeğe devam etti” cümlesindeki bakışta yemek eyleminin devamı vurgulanmıştır. İlerleme bakışından farklıdır çünkü ilerleme bakışında eylemin devamı söz konusuysen devamsal bakışta eylemin devamının devamı söz konusudur. Yani eylemin bitmesi kesilmesi ara verilmesi gibi eylemlerden mahfuz olduğunu söyleyebiliriz.

Hazırlık bakışı (Inchoative Aspect) : Bir eylemin hazırlık aşamasını vurgular. Örneğin “Ali yemeğe hazırlandı” cümlesindeki anlam, yemek eylemine hazırlığı vurgulamaktadır. Yani yemek eylemi birazdan gerçekleşecektir ancak bakışımız bu eylemin öncesine çekilmektedir.

SORU 29: Kelimebilimsel Bakış (Lexical Aspect)

Bilgisayar bilimlerinin bir çalışma alanı olan doğal dil işleme (Natural Language Processing) konusunda özellikle zaman kavramı önemli bir yer tutar. Bir eylemin (fiil, event) zamanının tutulması için çeşitli sınıflandırma yöntemleri geliştirilmiştir.

Bu yöntemlerden birisi de [bakış'ın zamanının tespit edilmesidir](#) örneğin bu amaçla [Reichenbach zaman analizi](#) yöntemi kullanılan bir yöntemdir. Bu yazıda ise kelimebilimsel (lexical) açıdan fiillerin zamanlarının tasnifine bakılacaktır.

Buna göre aynı zamanı ifade eden bütün fiiler birbiri ile aynı kelime bilimsel bakışa sahiptir diyebiliriz. Örneğin:

- geldim
- gittim

fiillerinin kelimebilimsel bakışları aynıdır. Buna mukabil:

- Oturuyorum
- Yiyorum

fiillerinin kelimebilimsel bakışları aynı değildir. Bunun sebebi yemek eyleminin doğal bir sonu bulunmasıdır. Oturmak eylemi ise başka bir eylemle ikame edilebilir yani doğal olarak bitmez.

Bu durumda kelimelerin çeşitli özelliklerine göre sıralanması mümkündür. Konu üzerinde çalışma yapan Zeno Vendler fiilleri bu özelliklerini kullanarak dört gruba bölmüştür:

- activity (eylem)
- achievement (erişmek)
- accomplishment (nihayet)
- state (hal)

Bu sınıflardan eylemler ve erişme fiilleri, nihayet ve hal fiillerinde ayrılabilir. Bu ayrımda eylemin devamı önemli rol oynar.

Benzer şekilde Eylem ve erişme fiilleri de aralarında sınırlılık anlamında ayrılabilirler. Yani erişme fiillerinde eylem erişilen noktada biterken eylem fiillerinde süreklilik söz konusudur.

Yukarıdaki bu durumu aşağıdaki tablo ile ifade edebiliriz.

	Telic	Dynamic Devinimli	Durative Sürekli	Örnekler.
Stative	Atelic	-	+	(birşeyi) Bilmek (birşey) Edinmek
Activity	Atelic	+	+	Yürümek Boyamak
Accomplishment	Telic	+	+	İnşa etmek Mezun olmak
Achievement	Telic	+	-	(para,statu vb.) Kazanmak

Yukarıdaki tabloda kullanılan terim “Telic” antik yunancadan gelen ve bir eylemin sonucu olduğunu belirten terimdir. Doğal sonucu bulunmayan eylemler için Atelic terimi kullanılabilir.

Yukarıdaki tabloda ayrıca süreklilik kavramı da gösterilmiştir. Bir fiilin sürekliliği yapılan eylemin kesikli mi devamlı mı olduğuna bağlıdır. Yani örneğin yürümek sürekli bir eylemdir buna karşılık adım atmak kesikli bir eylemdir. Yürüme eylemini adımlamak olarak tanımlamak ise sürekli bir eylemi kesikli eylemler ile tanımlamak anlamına gelir.

SORU 30: Reichenbach Zaman Analizi (Reichenbachian Tense Analysis)

Bilgisayar bilimlerinin çalışma alanlarından birisi olan doğal dil işleme ve bu alana bağlı olarak çalışılan [soru cevaplama \(question answering\)](#) konularında zaman çıkarımı ve [olayların zamanlarının tespit edilmesi \(tense, modal, aspect\)](#) önemli bir yer tutar.

[Olay zamanlarının analiz edilmesi](#) sırasında yazılı metinlerden faydalandığı düşünülürse teorik olarak üç farklı zamandan bahsetmek gerekir:

1. olayın geçtiği zaman (E)
2. olayın yazıldığı veya anlatıldığı zaman (S)
3. referans zamanı (R)

Örneğin aşağıdaki cümleyi ele alalım:

“Ali gelip gerçekleri anlattığında çoktan talimat vermiştim.”

Yukarıdaki cümlede geçen fiilleri ayıklayacak olursak:

(“ (Ali gelip gerçekleri anlattığında)_R (çoktan talimat vermiştim)_E.”)_S

Yukarıda gösterilmek istenen bütün cümlelerin şu anda anlatıldığı veya okunduğudur. Ayrıca talimat verme eylemi tam olarak olayın geçtiği zamanda gerçekleşmiştir. Son olarak Alinin gelip gerçekleri anlatması eylemi olayın zamanına referans alınan bir olaydır. Yani bu cümlede bu iki olay arasında bir bağlantı kurulmuştur.

Dolayısıyla Reichenbach analizine göre yukarıdaki cümlede 3 farklı zamanın sıralaması aşağıdaki şekilde olacaktır:

$E < R < S$

Yani olayın geçtiği zaman, referans zamanından çok önce olmuştur, benzer şekilde olayın referans zamanı ise olayın anlatıldığı şimdiki zamandan öncedir.

Yukarıdaki örnekten yola çıkarak Reichenbach 13 farklı zaman dizilimi alternatifi getirmiştir. Bu dizilimler aşağıdaki gibi bir tabloda toparlanabilir.

İlişki	Reichenbach Zaman İsmi (Tense)	İngilizce Zaman İsmi	İngilizce Örnek	Türkçe Karşılığı
--------	--------------------------------	----------------------	-----------------	------------------

	Name)			
E<R<S	Anterior past	Past perfect	I had slept	Hikaye Geçmiş Zaman
E=R<S	Simple past	Simple past	I slept	Geçmiş Zaman
R<E<S				
R<S=E	Posterior past		I would sleep	Gelecek Zamanın Mazisi
R<S<E				
E<S= R	Anterior present	Present perfect	I have slept	Rivayet Geçmiş Zaman
S= R= E	Simple present	Simple present	I sleep	Geniş Zaman
S= R<E	Posterior present	Simple future	I will sleep <i>Je vais dormir</i>	YOK
S<E<R				YOK
S=E<R	Anterior future	Future perfect	I will have slept	YOK
E<S<R				
S<R=E	Simple future	Simple future	I will sleep <i>Je dormirai</i>	Gelecek Zaman
S<R<E	Posterior future		I shall be going to sleep	

Yukarıdaki tabloda 13 farklı zaman gösterilmektedir. Bu zamanların hepsinin ingilizce veya türkçe için birebir karşılıkları yoktur ancak konuşmacı (yada yazar) farklı şekillerde bu zamanları ifade etmektedir.

SORU 31: Zamani (Temporal, Zamansal, Zamane, Mevkuat)

Zamani kavramları açıklamak için kullanılan terimdir. Basitçe insanın zaman algısı ve bu algı üzerine kurulu olan felsefi ve yaşamsal düşünceleri geçmiş, şimdi ve gelecek üzerine kuruludur. Bu durumda zamani kavramlarda bu değerlerin etkisine şekillendirilmektedir.

Aslında din ve felsefede derin tartışmalar açmış bu konuya insanlığın var olduğu tarih boyunca hemen her çağda rastlanmaktadır ve neredeyse bütün filozofların çalışmalarında yer bulmuştur. Örneğin din açısından zamanın yaratılmış olması (tanrıdan başka yaratılmayanın olmayışı), insan olarak bizlerin kısıtlı bir zaman kavramı içerisinde yaşıyor olduğumuz sonucunu doğurmaktadır.

Gerek ontolojik gerek epistemolojik tartışmalara da sebep olan bu konuda örneğin Heidegger'in "Sein und Zeit" (zaman ve varlık) veya Nietzsche'nin Sonsuz Dönüş (Eternal Recurrence) çalışmaları örnek olarak gösterilebilir.

Bilgisayar bilimleri açısından konunun önemi özellikle yapay zeka çalışmalarında ön plana çıkmaktadır. Çalışma alanı olarak insan zekasını modellemek ve daha çok taklit etmek amacını belirlemiş olan yapay zeka, konu ile ilgili değişik alt alanlarda çalışmaktadır. Örneğin [soru cevaplama \(question answering\)](#) , doğal dil işleme (natural language processing) bu konulardan bazılarıdır.

Bu alıřmalardaki ama, insanın zaman algısını modelleyerek bu algı zerinden suni bir zeka retmek veya taklit etmektir.

Zaman kavramı ayrıca mhendisliėin pekok alanında bir parametre veya boyut olarak da kullanılır ancak bu kullanımlarda zamanın anlamının nemi yoktur. rneėin gerek zamanlı sistemlerde (real time systems) zamana baėlı olarak iřlerin takip edilmesi gerekir ancak burada zaman sadece takip edilen ve anlamıyla ilgilenilmeyen bir bilgidir.

SORU 32: Zaman Sırası (Sequence of Tenses)

Bilgisayar bilimlerinin bir alıřma konusu olan doėal dil iřlemede eřitli amalarla kullanılan zaman belirleme iřlemleri aısından nemli bir kavramdır. Basite bir birleřik cmledeki ana cmlenin ve alt cmlenin zamanlarının arasındaki uyumu aıklamak iin kullanılır.

rneėin

Ali eve geldiėinde Ayře uyuyordu

Ali eve geldiėinde Ayře uyumuřtu

Ali eve geldiėinde Ayře uyuyacaktı

Ali eve geldiėinde Ayře uyudu

Ali eve geldiėinde Ayře uyumuř

Ali eve geldiėinde Ayře uyuyor

Yukarıdaki cmle rneklerinde Ali'nin eylemi olan eve gelmek ile, Ayřenin eylemi olan uyumak arasındaki baėlantı anlamsal olarka farklıdır. Bu cmlelerin anlamlarındaki farklılık ana cmle olan ve Ayře'nin uyumasını ifade eden fiilin zamanından kaynaklanmaktadır. Yukarıda sadece bazı rnekleri verilen alternatiflerin bazıları anlamsız, bazıları ise eř anlamlı olduėu iin Trkede kullanılmaz. rneėin yukarıdaki rneklerden ilk nde uyuma eyleminin zamanı birleřik zamandır, son nde ise basit zamanlıdır.

Ayrıca yukarıdaki rneklerde alt cmlenin fiilinin zamanındaki deėiřikliklerde anlamı deėiřtirir:

Ali eve gelirken Ayře uyuyordu

Ali eve gelecekken Ayře uyuyordu

Ali eve geliyorken Ayře uyuyordu

řeklinde alt cmlede farklı zamanların kullanılması da olayın anlamını ve dolayısıyla zaman kavramını deėiřtirebilir.

Bu durumun anlařılması iin iki farklı grup altında inceleme yapmak gerekir:

- Basit zamanlı sıralamalar (Natural sequence)

- Birleşik zamanlı sıralamalar (Attracted sequence)

Zaman sırası literatürde aşağıdaki şekillerde de geçmektedir:

sequence of tenses, *consecutio temporum*, agreement of tenses, succession of tenses, tense harmony, backshifting

Basit zamanlı sıralamalar (Natural Sequence)

Bu zaman sıralamasında (sequence) ana cümlenin (superordinate) zamanı ile alt cümlenin (subordinate) zamanı arasında doğrudan bir bağlantı yoktur. Cümlelerin zamanları bağımsızca ve doğal olarak belirlenir.

Örneğin “Ali okula geldiğinde ayşe uyuyordu”

Yukarıdaki “gelmek” ve “uyumak” eylemleri arasında doğrudan bir zaman bağı yoktur. Örneğin “Ayşe uyuyacaktı” şeklinde bir cümle ile de bitebilirdi.

Birleşik zamanlı sıralamalar (Attracted sequence)

Bu cümle yapısında iki cümle arasında bir etkileşim söz konusudur. Örneğin “Okula hiç gelmemiş olmayı isterdim” cümlesindeki “okula gelmek” ve “istemek” eylemleri arasında zaman bağlantısı söz konusudur. “istemek” eylemi geçmiş zamanlı olduğu için “gelmek” eylemi de geçmiş zamanlı olmak zorundadır.

Örneğin bu cümle “Okula hiç gelmedi olmayı isterdim” yada “Okula hiç gelmez olmayı isterdim” şeklinde kurulduğundan cümlecikler arasındaki bağlantı bozulmaktadır.

SORU 33: Uyum (Agreement, Kabul, Bağıt, Mutabakat)

Bilgisayar bilimlerinin bir çalışma alanı olan doğal dil işleme (Natural language processing) konusunda ve dolayısıyla dilbilim (linguistic) konusunda kullanılan bir kavramdır. Bu kavrama göre bir dilde aynı anlama gelen kelimelerin sayısı, cinsiyet, kişi veya duruma göre farklı kelimelerle ifade edilmesidir.

Örneğin Türkçede çoğul kelimelerin kullanımı bir uyum örneğidir. [Kelimebilim \(lexicology\)](#) açısından aynı ifade olan “el” ve “eller” kelimelerini ele alalım. Bu iki kelime aynı anlamdaki varlığı işaret etmekte ancak ikinci kelime çoğul anlamda kullanılmaktadır. Bu durumda, Türkçede kelimenin sayısal olarak uyumundan (agreement) bahsedebiliriz.

Benzer şekilde “evim” ve “evimiz” kelimeleri arasında da kişi anlamında bir uyum vardır. Yani ilk kelime 1. tekil şahsın varlığını işaret ederken ikinci kelime 1. çoğul şahsın varlığını işaret etmektedir.

Cinsiyete göre de ayırım yapılan Türkçedeki bu kelimeler genelde birleşik kelime veya yabancı dillerden giren kelimelerdir. Örneğin bilimadamı , bilimkadını veya bilimsanı ayırımı birleşik kelimeye örnektir. Müdür veya müdüre, memur veya memure ayırımı Türkçeye giren arap-fars akımına örnek olabilir. Benzer şekilde kuzen veya kuzin ayırımı Türkçeye giren latin (fransız) akımına örnek olabilir. Bu örneklerin hepsinde kastedilen aynı varlık iken, kelimeler arasında sadece kastedilen varlığın cinsiyet farkı söz konusudur.

Aslında Türkçede de rastlanan ve genelde Arapça, Farsça gibi dillerden giren kelimelerde yaşanan bu durum, Latin, Sami, Germen veya Slav dil ailelerinde sıkça görülmektedir. Örneğin Japoncada da neredeyse az görülen kabullerin en sık görüldüğü dillerden birisi de Swahili dilidir.

SORU 34: İçerikten Bağımsız Gramer (context free grammer, CFG)

Bilgisayar bilimlerinde, dil tasarımı sırasında kullanılan bir gramer tipidir. Basitçe bir dilin kurallarını (dilbilgisini, grammer) tanımlamak için kullanılır.

Örneğin:

S -> a

Yukarıdaki dil tanımında bir büyük harfle gösterilen (S) bir de küçük harfle gösterilen (a) sembolleri bulunmaktadır. Bu satır, S devamlısının(nonterminal) a sonuncusuna(terminal) dönüştüğünü göstermektedir. Kısaca dildeki kuralları ifade etmek için büyük harfli semboller devamlıları (nonterminal) ve küçük harfli semboller sonucuları (terminal) ifade etmekte, -> ok işareti ise, işaretin solundaki devamlının (nonterminal), sağındaki sembolle gösterilebileceğini ifade etmektedir.

Bir [dili içerikten bağımsız \(context-free\)](#) yapan, o dilin bir [belirsiz aşağı sürüklememli otomat \(non deterministic pushdown automata\)](#) tarafından üretilebilir olmasıdır.

İçerikten bağımsız diller, programlama dilleri olarak sıklıkla kullanılmaktadır, bilinen çoğu programlama dili aslında birer içerikten bağımsız dil özelliğindedir ve bu dillerin kurallarının tanımlandığı gramerlerde içerikten bağımsız gramerlerdir (context free grammer).

Bu anlamda, [YACC](#) gibi programlama ortamlarında, bir dil tasarlamak ve içerikten bağımsız kurallar yazarak dili tanımlamak mümkündür.

CFG gösterimi ne yazık ki doğal diller (natural languages) için kullanılamaz. Ya da kullanılsa bile bir doğal dilin tamamını kapsayacak bir CFG gösterimi çıkarılamaz. Örneğin doğal dillerde [kelimebiliminin \(lexicology\)](#) bir parçası olarak sıkça kullanılan [uyum \(agreement\)](#) veya atıf (reference) kullanımları CFG ile gösterilemeyen özelliklerdir. Yani daha net bir ifadeyle CFG gösterimi için dildeki anlamların belirli olması gerekir. Çeşitli durumlarda belirsizlik içeren doğal diller için ise bu durum imkansızdır.

CFG tanımı

Temel olarak bir içerik bağımsız gramer dört özellik içermelidir. Bunlar sonlular (terminals), devamlılar (nonterminals), bağlantılar (relation) ve başlangıç sembolü (starting symbol) olarak sıralanabilir. Bir gramerin tanımı sırasında kullanılan bu kümeler aşağıdaki şekilde yazılabilir:

$$G = (V , \Sigma , R , S)$$

Bu gösterimdeki grammer (G) , V devamlıları, Σ sonluları, R bağlantıları, S ise başlangıç sembolünü göstermektedir.

Örneğin

$S \rightarrow aSb \mid ab$

şeklinde tanımlanan bir dilde:

$G = (\{S\} , \{a,b\} , \{S \rightarrow aSb \mid ab\} , S)$

gösterimi kullanılabilir.

SORU 35: İçerikten bağımsız dil (Context Free Language, CFL)

Bilgisayar bilimlerinde bir dilin tasarımı sırasında, içerik bağımsız bir gramer ile oluşturulması durumudur. Basitçe bir [aşağı sürüklemeli otomat \(push down automata\)](#) tarafından kabul edilen dil çeşididir. Bazı kaynaklarda bağlamdan bağımsız dil olarak da geçmektedir.

Örneğin çok meşhur $L = \{a^n b^n, n > 0\}$ dilini ele alalım. Bu dil örneğinin bu kadar meşhur olmasının ve önemli olmasının sebebi bir düzenli ifade (regular expression) ile yazılmasının imkansız oluşu ancak içerikten bağımsız dil ile yazılmasının mümkün olmasındandır.

Şimdi bu dilin aşağı sürüklemeli otomatını aşağıdaki şekilde çıkarabiliyoruz:

$$\begin{aligned} \delta(q_0, a, z) &= (q_0, a) \\ \delta(q_0, a, a) &= (q_0, a) \\ \delta(q_0, b, a) &= (q_1, x) \\ \delta(q_1, b, a) &= (q_1, x) \\ \delta(q_1, b, z) &= (q_f, z) \end{aligned}$$

$$\delta(state_1, read, pop) = (state_2, push)$$

Yukarıdaki [PDA \(push down automaton\)](#) tasarımında dikkat edilirse iki durum (state) arasındaki geçişler ile yukarıdaki dili tasarlamak mümkündür. Bu sayede bu dilin bir içerik bağımsız dil olduğu söylenebilir.

Ayrıca yukarıdaki tanımda kullandığımız ” içerik bağımsız bir gramer ile oluşturulması durumu” ifadesini de açıklayarak buna da bir örnek verelim ve dilimizin ($L = \{a^n b^n, n > 0\}$) CFG (context free gramer, içerik bağımsız gramer) karşılığını aşağıda yazalım:

$S \rightarrow aSb \mid ab$

Yukarıdaki yazılışta, dilin sonucu ab veya aSb olarak çıkacaktır ancak S devamlısı (nonterminal) bitmek için bir sonuncuya (terminal) ihtiyaç duyacaktır bu değer de yine ab olacaktır.

Sonuçta yukarıdaki gramer ile istenilen uzunlukta sırasıyla a ve b lerden oluşsan dil tasarlanabilir ve üretilen bütün dillerde a ’nın sayısı ile b ’nin sayısı eşittir.

SORU 36: Kelime (Lexeme)

Doğal dil işlemenin (natural language processing) en önemli parçası olan ve bir dildeki en küçük anlamlı birim olan İngilizcedeki “Lexeme” kelimesi, genelde “kelime” kelimesine karşılık olarak kullanılmaktadır. Bu tam olarak yanlış olmasa da yanlış anlaşılmaya açıktır.

Lexeme anlam olarak herhangi bir kelime değildir, daha çok kelimenin ifade ettiği anlamdır. Örneğin, “Gözlük” kelimesi ile “Gözlükler” aynı anlamdaki veya daha net bir ifadeyle aynı şeyayı ifade eden kelimelerdir. Buna karşılık “Gözlükçü” kelimesinin ifade ettiği varlık farklıdır. Dolayısıyla “Gözlük” veya “Gözlükler” veya “Göz protezi” gibi kelimeler aynı lexeme olarak kabul edilirken “Gözlükçü” farklı bir lexemedir.

Aynı anlamdaki bu kelimelerin tek bir kümede toplanması halinde bu kümeye hypernym ismi de verilir. Örneğin “Gözlük”, “Gözlükler”, “Göz protezi” kelimelerinin tamamını bir gözlük kümesinde toplarsak bu kümeye hypernym ismi verilebilir.

SORU 37: Soru Cevaplama (Question Answering, QA)

Doğal dil işleme (natural language processing) çalışmalarının bir parçası olan soru cevaplama çalışmalarında amaç, doğal dildeki bir soruya doğru cevap verebilmektir. Soru cevaplama çalışmalarını bir kaç farklı şekilde gruplamak mümkündür.

Unutulmaması gereken, soru cevaplama çalışmalarının hedefinde insan gibi davranabilen bilgisayar programları bulunmaktadır. Yani ulaşılmak istenen noktada aynı soruyu bir insana sorduğunuzda alabileceğiniz olsaydı bir cevabı bilgisayardan almak istersiniz. Daha fazla bilgi için [Turing Testine](#) bakabilirsiniz.

Soru alanına göre QA (Question domain)

Çoğu zaman çalışmaların kolaylaştırılması için soruların belirli bir metin içerisinde cevaplanması istenir. Yani bir insanın hayatı boyunca öğrendiği şeylerin bilgisayar tarafından bir insan gibi cevaplanması her ne kadar bu çalışmayı yapan kişilerin hayali olsa da henüz bu seviyeye ulaşmak için çok erkendir. Bunun yerine, günümüz çalışmalarında, bilgisayara yine doğal dilde yazılmış bir metin verilerek bu metin içerisinde sorulan soruların cevaplanması beklenmektedir. Bu anlamda soru cevaplama iki farklı soru alanı belirlenebilir:

- açık alanlı sorular (open domain)
- kapalı alanlı sorular (close domain)

İlk gruba örnek olarak belirli bir konudaki soruları düşünebiliriz. Örneğin sadece ilaç sektörüne yönelik sorular veya spor sorularını düşünebiliriz.

Örneğin “Türkiye’nin en çok kazanan oyuncusu kimdir?” sorusuna sinema konusunda farklı bir cevap veya spor konusunda farklı bir cevap verilebilir. Şayet soruların alanları tanımlıysa bu problem ortadan kalkar, aksi halde soru cevaplayan program bu ayrımı yapabilmelidir.

Cevap tipine göre QA

Bazı soruların cevapları kısa bir kelime ile verilebilen ve genelde belirli cevaplardır. Örneğin “Türkiye Cumhuriyeti’nin ilk Cumhurbaşkanı kimdir?” sorusunun tek bir cevabı vardır ve herkes bu soruya aynı cevabı verir.

Ancak “Türkiye Cumhuriyetinin ilk Cumhurbaşkanını nasıl birisidir?” gibi bir sorunun cevabı belirli bir cevap değildir. Yani sorunun her muhatabı farklı cevaplar verebilir. Bu durumda cevabın belirliliğine(deterministic) göre soru cevaplama çalışmalarını ayırmak mümkündür. Dolayısıyla cevap tiplerine göre sorular ikiye ayrılabilir:

- Belirli Cevaplar (Deterministic Answers)
- Belirsiz Cevaplar (Nondeterministic Answers)

Soru Cevaplama Yöntemine Göre:

Soru cevaplama sırasında kullanılan yöntemlere göre yapılan çalışmaları iki ayrı grupta toplamak mümkündür.

- Sığ Yöntemler (shallow methods)
- Derin Yöntemler (deep methods)

Şayet kullanılan yöntem cümle içerisindeki basit yapıların yakalanması, cümle içinde geçen bazı kelimelerden sorunun anlamının çıkarılması gibi tekniklere dayanıyorsa bu cevaplama yöntemine sığ yöntem ismi verilmektedir.

Şayet kullanılan yöntem, kelimelerde başlayarak kelimebilim (morphology), [cümle dizilimi \(syntax\)](#) ve [anlambilim \(semantics\)](#) gibi aşamalara ayrılıyorsa hatta, [hazf \(ellipsis\)](#), [dönüştü \(anaphora\)](#), [eş atıf çözümü \(coreference resolution\)](#), [ilgi belirsizliği \(reference ambiguity\)](#), [ontoloji](#) ve benzeri problemleri de çözüyorsa sığ yöntem ismi verilebilir.

SORU 38: Kesinlik Zarfları (Katî Zarflar, Sentential Prepositions)

Bilgisayar bilimlerinin bir alt dalı olan yapay zekanın çalışma alanlarından doğal dil işleme (natural language processing) konusunda kullanılan bir dilbilim (linguistic) terimidir.

Bu [zarf \(adverb\)](#) tipleri, cümledeki bir alt cümle ile ana cümle arasında bağlantı kurmaya yarar. Örneğin “önce, sonra, çünkü, dolayısıyla” gibi kelimeler bu gruptan sayılabilir.

“Ayşe hazırlandıktan sonra okula gitti. “

Yukarıdaki cümlede iki ayrı olay “Ayşenin okula gitmesi” ve “Ayşenin hazırlanması” durumları söz konusudur. Arada bağlamak için kullanılan “sonra” kelimesi bu iki durumu bağlamış ve iki durumun da birbirine göre açıklanmasını sağlamıştır. Yani hem Ayşenin okula gitmeden hazırlandığını hem de Ayşenin hazırlandıktan sonra okula gittiğini açıklayarak iki olayı da birbirine göre açıklamıştır.

“Alinin yemek yedikten sonraki konuşması çok eğlencelidir”

Yukarıdaki cümlede bu yazının konusu olan katî zarf “sonraki” kelimesidir ve iki alt cümleyi birbirine bağlamıştır. Buradaki ana cümle olan “çok eğlencelidir” cümlesi bir [koşaç \(copula\)](#) örneğidir. İlk kısım olan ve ana cümleyi betimlemeye yarayan “alinin yemek yeme” eylemi ise bir alt cümledir (subordinate).

Aslında katî zarflarla birbirine bağlanan cümleler ayrıldıklarında çıkan durum bir [hazf \(ellipsis\)](#) örneği olarak kabul edilebilir.

Örneğin yukarıdaki cümleyi iki cümle şeklinde açarsak :

“Ayşe hazırlandıktan sonra okula gitti. “

1. “Ayşenin okula gitmesi”
2. “Ayşenin hazırlanması”

Görüldüğü üzere Ayşe öznesi iki ayrı cümlede tekrarlanmalıdır.

SORU 39: Koşaç (Mafsal, Haber Edatı, Copula)

Bilgisayar bilimlerinde önemli bir konu olan yapay zekanın bir alt dalı olan doğal dil işleme konusunda kullanılan bir terimdir. Basitçe bir dildeki özneyi, o dildeki zarflara veya (adverbs) veya [kaziyelere \(önerme, predicate\)](#) bağlamaya yarayan kelimelerdir.

Copula kelimesi latinedeki bağlama (bu yüzden çiftleşme anlamında da kullanılmaktadır) anlamından gelmektedir. Örneğin yazılım mühendisliğindeki [bağlanma \(coupling\)](#) konusu ile aynı kökten gelmektedirler.

Haber edatları genelde bir cümlede bildirimde bulunmaya yararlar. Örneğin “Ali sınıfın en çalışkanıdır”. Buradaki -dır eki bildirimde bulunmaktadır. Bu bildirimin üzerine kaziye(önerme) isnad edilmesi hasebiyle bu edatlara, edad-ı isnad ismi de verilmektedir.

Genelde isimlerin (ve ismin bir türü olan sıfatların) sonuna -dır eki eklenerek elde edilirler ve bir beyanda bulunmaya yararlar.

İngilizcede bu görevi “is”, “are” gibi kelimeler karşılamaktadır. Kısaca olma fiili (verb of being) isim de verilebilir.

Anlambilimsel olarak (semantic) bir kümeye veya alt kümeye aidiyet de bildirirler.

Örneğin:

Şadi bir öğrencidir. (öğrenci kümesine aitlik bildirir)

Evren ve Ahmet okuldadır. (yer zarfı ile fiil olan Evren ve Ahmet bağlanmıştır ve okuldakiler kümesine aidiyet bildirir)

Bu bağlama eyleminin neredeyse tamamı bir fiil ile de yapılabilir. Örneğin “Şadi öğrenciliğe başladı” cümlesindeki başlamak veya “Şadi öğrenci oldu” cümlesindeki olmak gibi fiiller eklenerek bu haber edatlarındaki öznelerin durumunu bildiren ve bu özneleri fail konumuna sokan fiil eklemeleri mümkündür.

Türkçede ayrıca -dır ekinin eklenmediği ancak haber edadı anlamında kullanıldığı isim cümleleri de kurmak mümkündür.

Örneğin “Şadi bir öğrenci” cümlesi aslında eksiktir ancak Türkçedeki günlük kullanımda bu tip cümlelere rastlanabilir.

Türkçede haber edatlarını olusuzu “değil” kelimesi ile yapılır.

Örneğin :

Şadi bir öğrenci değil

Evren ve Ahmet okulda değil

cümlelerinde yukarıdaki örneklerin tam tersi mana bulunmaktadır.

Türkçede haber edatlarının karıştırıldığı bir durum da edilgen fiillerin sonundaki -dır ekidir. Örneğin “Şadi ders anlatmaktadır” cümlesinde aslında bir fiil bulunmakta ve buradaki -dır eki sadece edilgenlik halinin devamlılığını ifade amacıyla eklenmiştir. Buradaki devamlılık “Şadi ders anlatmakta” cümlesi ile farkı incelenerek bulunabilir.

SORU 40: Hazf (Eksilti, Ellipsis)

Sözlük anlamı olarak giderme, kaldırma manalarına gelen bu kelime bir cümlede gerekli olan bütün bilgileri bulundurmuyup bir takım bilgileri önceki ve sonraki cümlelere atfederek kaldırmaya verilen isimdir. Bilgisayar bilimlerinde, doğal dil işleme sırasında yaşanan problemlerden birisidir ve çeşitli çözüm yöntemleri geliştirilmiştir.

Örneğin “Bana, işletim sistemleri dersini alan öğrencilerin listesini ver. Veriyapılarının da ver. Bir de veritabanınıninkisini ver. “

Yukarıdaki cümle tam olarak kullanıldığında: “Bana, işletim sistemleri dersini alan öğrencilerin listesini ver. Veriyapıları derini alan öğrencilerin de listesini ver. Bir de veritabanı dersini alan öğrencilerin listesini ver. “

Şeklinde kullanılması gerektirir. Ancak gerek cümleleri kuran, gerek de okuyan veya dinleyen taraf bu tamamlama olmadan da hazf edilmiş cümleleri anlamaktadır ve noksan olan bilgiyi geçmiş cümlelerde tamamlamaktadır.

SORU 41: Gösterim İşlemi (Projection Operator)

İlişkisel cebirde (relational algebra) kullanılan işlemlerden birisidir. Basitçe bir küme üzerinden herhangi bir kolonun seçilmesi işlemini gerçekleştirir. Sembolü “ Π ” işaretidir ve gösterilecek olan (project) kolon ismi indis olarak yazılır.

Örneğin aşağıdaki tabloya T1 ismi verelim:

İsim	Soyisim	Yaş	Melsek
Şadi Evren	ŞEKER	30	Bilgisayar Müh.
Ali	Baba	50	Tüccar
Veli	Demir	20	Öğrenci
Cem	Yıldız	40	Programc

Yukarıdaki tablo üzerine isim kolonuna gösterim (projection) uygulanması durumunda aşağıdaki sonuç elde edilir:

$\Pi_{\text{isim}}(T1)$

İsim
Şadi Evren
Ali
Veli
Cem

SORU 42: Parçalama Ağacı (Parse Tree)

Parçalama işlemi (parsing) bilgisayar bilimlerinde çeşitli amaçlar için kullanılmaktadır. Özellikle de dil ile ilgili işlemlerin hemen hepsinde ihtiyaç duyulan bir işlemdir. Örneğin bir programlama dilinde yazılan komutların algılanması için öncelikle kelimelerin parçalanması (parse) gerekir. Benzer şekilde doğal dil işleme (natural language processing) işlemlerinde de doğal dilde bulunan kelimelerin algılanması bir parçalamadan (ek ve köklerin ayrılmasından) sonra gerçekleşmektedir.

Çeşitli sebeplerle kullanılan parçalama ağaçları basitçe verilen bir dilbilgisine (grammar) göre verilen cümlelerin (veya kelimelerin) nasıl parçalandığını şekilsel olarak gösteren ağaçlardır:

Örneğin aşağıda [BNF yapısında](#) verilmiş dili ele alalım:

$\langle \text{dil} \rangle ::= \langle \text{işlem} \rangle$

$\langle \text{işlem} \rangle ::= \langle \text{işlem} \rangle + \langle \text{terim} \rangle \mid \langle \text{işlem} \rangle - \langle \text{terim} \rangle \mid \langle \text{terim} \rangle$

$\langle \text{terim} \rangle ::= \langle \text{terim} \rangle * \langle \text{unsur} \rangle \mid \langle \text{terim} \rangle / \langle \text{unsur} \rangle \mid \langle \text{unsur} \rangle$

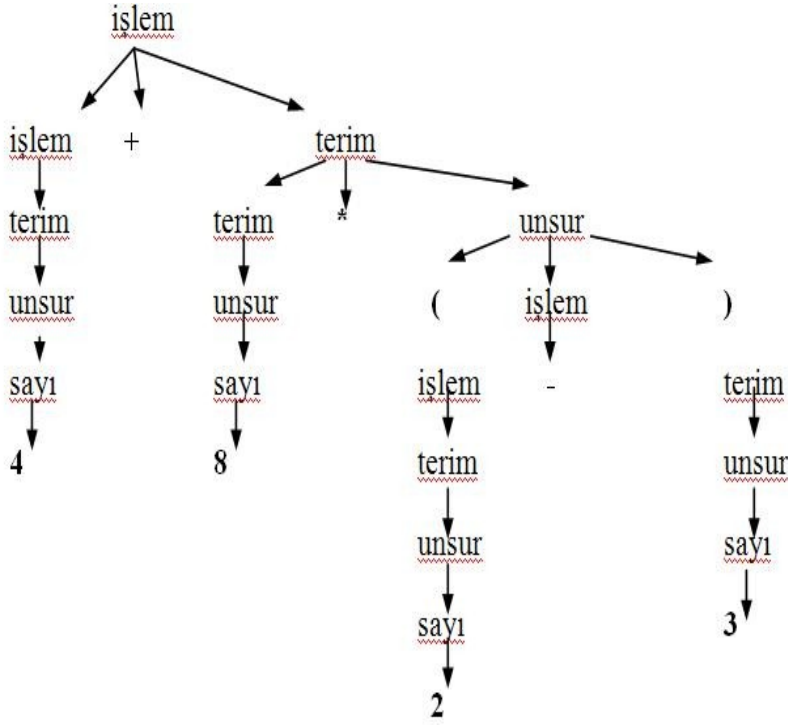
$\langle \text{unsur} \rangle ::= \text{sayı} \mid (\langle \text{işlem} \rangle)$

$\langle \text{sayı} \rangle ::= 1|2|3\dots|9|0$

Bu dilde aşağıdaki örneğin nasıl parçalandığını inceleyelim:

$4+8*(2-3)$

Bu dilde tanımlı olan yukarıdaki işlemin parçalama ağacı aşağıdaki şekildedir:



yukarıdaki şekilde devamlı (nonterminal) terimlerden sonuncu (terminal) terimlere kadar bir BNF dilinin nasıl açıldığı gösterilmiştir.

SORU 43: Türkçe için TimeML

TimeML İngilizce esas alınarak geliştirilmiş bir dildir ve ne yazık ki TimeML tam bir makine dili değildir bu yüzden TimeML içerisinde bir takım doğal dile ait olan özelliklerin Türkçeye tam olarak uygulanması mümkün değildir. Bu özelliklerin başında fiillerin zaman kavramı gelir. Örneğin İngilizcede kullanılan bakışların (aspect) Türkçe için uyarlanması oldukça zordur ve TimeML içerisinde bu dil özelliği neredeyse tamamen korunarak alınmıştır.

Bu dökümanda TimeML dilinin bir Türkçe için nasıl kullanılabileceği ve Türkçenin TimeML ile uyumsuz olan yanları ortaya çıkarılacak, bu noktalarda çözümler önerilecek ve nihayi olarak bir Türkçe metin için TimeML'in bu değişmiş hali uygulanacaktır.

Bu dökümanda izlenecek yol TimeML dökümanlarında izlendiği üzere TimeML dilinde tanımlı olan 3 seviyeyi sırasıyla açıklayacaktır. Bu seviyelerdeki etiketler (tags) sırasıyla incelenecek ve yukarıda açıklandığı üzere Türkçe için uygunluğu tartışılarak açıklanacaktır.

TimeML dilinde bulunan seviyeler:

1. Zaman içerikli olayların saklandığı ve [ağacın yapraklarını](#) oluşturan olaylar
2. Yaprakların üzerindeki ve zaman olayları üzerindeki, zaman belirleyici değerler. Bu değerlere dilde sinyal (işaret) anlamı verilmiştir.
3. Olaylar arasındaki bağlantılar (link)

TimeML dilinin en alt seviyesinde ve yukarıdaki listede 1. seviye diye adlandırılabilir seviyede fiiller (olaylar, eylemler, events) bulunur.

<EVENT> Etiketi

Bu etiketler basitçe olma veya vukûu halini anlatırlar. Aynı zamanda hal ve durum anlatan olaylarda birer eylem olarak kabul edilebilir. Ancak TimeML dili için her kaziye (önerme, statement) bir fiil olarak kabul edilmemiştir. Bu tip cümleler için daha sonra göreceğimiz “STATE” etiketi bulunmaktadır.

Bir <EVENT> etiketi 6 farklı ihtimal ihtiva edebilir:

zamanlı fiiller

zamansız fiiller

SORU 44: Cümle Zamanları (Tense) ve Bakış (aspect)

Bir cümlenin geçtiği zamanı belirten yapıdır. Genellikle fiil cümlelerinde fiilin taşıdığı zaman yapısına göre belirlenir. Örneğin “Ali uyudu” cümlesindeki “uyudu” fiili olayın geçmiş zamanda olduğunu -du eki ile belirtmektedir.

Cümle zamanlarının, bakışlardan (aspect) farkı ise, bakışın olayın ne zamanı etkilediğidir. Örneğin yukarıdaki “Ali uyudu” cümlesi ile aynı zamanda olan aşağıdaki örnekleri inceleyelim:

“Ali uyuyordu”

“Ali uyuyacaktı”

“Ali uyumuştı”

Yukarıdaki bütün örneklerde “uyuma” eyleminin geçmişte olduğunu biliyoruz. Ancak her cümlede farklı bir zamanın etkilenmesi söz konusudur. İşte bu etkilenen zaman o cümlenin bakışıdır (aspect).

Dillerin yüksek bir çoğunluğunda 3 farklı zaman vardır. Bu zamanlar geçmiş, şimdi ve gelecektir. Ancak bazı dillerde bu zamanlardan bir kısmı bulunmaz. Örneğin İngilizce için gelecek zaman yoktur. (bunun yerine bir modal bulunur)

Türkçedeki zamanlar:

- -Di li geçmiş zaman (bilinen geçmiş zaman, mazi malum). Fiili yapan kişi tarafından fiil malum ise ve fiil geçmişte ise bu zamana bilinen geçmiş zaman ismi verilir. “Ali uyudu” gibi
- -Miş li geçmiş zaman (öğrenilen geçmiş zaman, mazi meçhul) Fiilin kişi tarafından bilinmediği cümle yapısıdır. Fiil geçmiştedir ve olay sonradan öğrenilir veya fark edilir. “Ali uyumuş” gibi
- Şimdiki Zaman: Cümlenin ifâde edildiği anda olan fiili belirtir. “Ali uyuyor” gibi
- Gelecek Zaman: Cümlenin ifâde edildiği andan daha sonra olan fiili belirtir. “Ali uyuyacak” gibi
- Geniş Zaman: Fiilin cümle zamanından önce başlayıp, cümle anında devam ettiğini ve gelecekte de devam edeceğini gösterir. “Ali kitap okumayı sever” gibi

İngilizcedeki zamanlar:

- Past Tense (Geçmiş Zaman) , “He came”
- Present Tense (Şimdiki Zaman), “He is coming”
- İngilizcede gelecek zaman bulunmaz.

Yukarıdaki bu zamanların ayrıca bakışları (aspect) bulunur:

- Simple (Basit, zaman etkisi yapmayan bakış)
- Progressive (Devamlı, olayın geçtiği zamanda devam olduğunu belirten bakış)
- Perfect (Olayın tamamlandığını ifade eder, “Completed Aspect”)
- Perfect Progressive (Devamlı ve tamamlanmış bakıştır (continuous , completed aspect))

Yukarıdaki bu bakışların zamanlara uygulanmış hali ve örnekleri aşağıda verilmiştir:

Present Tense:

- Present Simple : “I eat”
- Present Progressive : “I am eating”
- Present Perfect : “I have eaten”
- Present Perfect Progressive : “I have been eating”

Past Tense:

- Past Simple : “I ate”
- Past Progressive : “I was eating”
- Past Perfect : “I had eaten”
- Past Perfect Progressive : “I had been eating”

SORU 45: Sıralama Algoritmaları (Sorting Algorithms)

Bilgisayar bilimlerinde verilmiş olan bir grup sayının küçükten büyüğe (veya tersi) sıralanması işlemini yapan algoritmalara verilen isimdir. Örneğin aşağıdaki düzensiz sayıları ele alalım:

5 9 2 3 7 11 -4 6

Bu sayıların sıralanmış hali

-4 2 3 5 6 7 11

olacaktır. Bu sıralama işlemini yapmanın çok farklı yolları vardır ancak bilgisayar mühendisliğinin temel olarak üzerine oturduğu iki performans kriteri buradaki sonuçları değerlendirmede önemli rol oynar.

- Hafıza verimliliği (memory efficiency)
- Zaman verimliliği (Time efficiency)

Temel olarak algoritma analizindeki iki önemli kriter bunlardır. Bir algoritmanın hızlı çalışması demek daha çok hafızaya ihtiyaç duyması demektir. Ters durumda da bir algoritmanın daha az yere ihtiyaç duyması daha yavaş çalışması demektir. Ancak bir

algoritma hem zaman hem de hafıza olarak verimliyse bu durumda diğer algoritmalarından başarılı sayılabilir.

Genellikle verinin hafızada saklanması sırasında veriyi tutan bir belirleyici özelliğinin olması istenir. Veritabanı teorisinde birincil anahtar (primary key) ismi de verilen bu özellik kullanılarak hafızada bulunan veriye erişilebilir. Bu erişme sırasında şayet belirleyici alan sıralı ise erişimin logaritmik zamanda olması mümkündür. Şayet veri sıralı değilse erişim süresi doğrusal (linear) zamanda olmaktadır.

Aşağıda bazı sıralama algoritmaları verilmiştir:

- [Seçerek Sıralama \(Selection Sort\)](#)
- [Hızlı Sıralama Algoritması \(Quick Sort Algorithm\)](#)
- [Birleştirme Sıralaması \(Merge Sort\)](#)
- [Yığınlama Sıralaması \(Heap Sort\)](#)
- [Sayarak Sıralama \(Counting Sort\)](#)
- [Kabarcık Sıralaması \(Baloncuk sıralaması, Bubble Sort\)](#)
- [Taban Sıralaması \(Radix Sort\)](#)
- [Sokma Sıralaması \(Insertion Sort\)](#)
- [Sallama Sıralaması \(Shaker Sort\)](#)
- [Kabuk Sıralaması \(Shell Sort\)](#)
- [Rastgele Sıralama \(Bogo Sort\)](#)
- [Şanslı Sıralama \(Lucky Sort\)](#)
- [Serseri Sıralaması \(Stooge Sort\)](#)
- [Şimşek Sıralaması \(Flahs Sort, Bora Sıralaması\)](#)
- [Tarak Sıralaması \(Comb Sort\)](#)
- [Gnome Sıralaması \(Gnome Sort\)](#)
- [Permütasyon Sıralaması \(Permutation Sort\)](#)
- [Strand Sort \(İplik Sıralaması\)](#)

Yukarıda verilen veya herhangi başka bir sıralama algoritması genelde küçükten büyüğe doğru (ascending) sıralama yapar. Ancak bunun tam tersine çevirmek (descending) genelde algoritma için oldukça basittir. Yapılması gereken çoğu zaman sadece kontrol işleminin yönünü değiştirmektir.

Ayrıca sıralama işleminin yapılması sırasında hafızanın kullanımına göre de sıralama algoritmaları :

- [Harici Sıralama \(External Sort\)](#)
- Dahili Sıralama (Internal Sort)

şeklinde iki grupta incelenebilir.

Algoritmaların karşılaştırılması için aşağıdaki tablo hazırlanmıştır:

Algoritma	İngilizcesi	Algoritma Analizi	Kararlılı	Yöntem	Açıklama
		En İyi	Ortalama	En Kötü	

Seerek Sıralama	Selection Sort	n^2	n^2	n^2	Kararsız	Seerek	
Hızlı Sıralama	Quick Sort	$n \log(n)$	$n \log(n)$	n^2	Kararsız	Parala Fethet	
Birleřtirme Sıralaması	Merge Srot	$n \log(n)$	$n \log(n)$	$n \log(n)$	Kararlı	Parala Fethet	
Yıęınlama Sıralaması	Heap Sort	$n \log(n)$	$n \log(n)$	$n \log(n)$	Kararsız	Seerek	
Sayarak Sıralama	Counting Sort	$n + 2^k$	$n + 2^k$	$n + 2^k$	Kararsız	Sayarak	k ikinci dizinin boyutu.
Kabarcık Sıralaması	Bubble Sort	n	n^2	n^2	Kararlı	Yer Deęiřtirme	
Kokteyl Sıralması	Coctail Sort	n	n^2	n^2	Kararlı	Yer Deęiřtirme	ift Yönlü kabarcık sıralaması (bidirectional bubble sort) olarak da bilinir ve dizinin iki ucundan iřleyen kabarcık sıralamasıdır.
Taban Sıralaması	Radix Sort	$n(k/t)$	$n(k/t)$	$n(k/t)$	Kararlı	Gruplama / Sayma	k, en büyük eleman deęeri, t ise tabandır
Sokma Sıralaması	Insertion Sort	n	$d+n$	n^2	Kararlı	Sokma	d yer deęiřtirme sayısıdır ve n^2 cinsindendir
Sallama Sıralaması	Shaker Sort	n^2	n^2	n^2	Kararsız	Seme	ift yönlü seme sıralaması (bidirectional selection sort) olarak da bilinir.
Kabuk Sıralaması	Shell Sort	$n^{3/2}$	$n^{3/2}$	$n^{3/2}$	Kararsız	Sokma	
Rastgele Sıralama	Bogo Sort	1	$n n!$	sonsuz	Kararsız	Rastgele	Algoritma olduęu tartıřmalıdır. Knuth karıřtırması

							(knuth shuffle) süresinde sonuca ulaşması beklenir.
Bozo Sıralaması	Bozo Sort	1	$n!$	sonsuz	Kararsız	Rastgele	Rastegele sıralamanın özel bir halidir. Rastgele olarak diziyi karıştırdıktan sonra, dizi sıralanmamışsa, yine rastgele iki sayının yeri değiştirilip denenir.
Goro Sıralaması	Goro Sort	$2^{\frac{\log(d)}{\log(2)}}$	$2^{\frac{\log(d)}{\log(2)}}$	$2^{\frac{\log(d)}{\log(2)}}$	Kararsız	Rastgele	2011 Google kod yarışı (google code jam) sırasında ortaya çıkmıştır. Sıralanana kadar her alt küme permüte edilir. Buradaki performans değeri ispatlanmamıştır ve d dereceyi ifade eder.
Şanslı Sıralama	Lucky Sort	1	1	1	Kararsız	Rastgele	Algoritma olarak kabul edilmemelidir.
Serseri Sıralama	Stooge Sort	n^e	n^e	n^e	Kararsız	Yer değiştirme	e, doğal logairtma sayısıdır (2,71)
Şimşek Sıralaması	Partial Flash Sort	n	$n + d$	$n + d$	Kararsız	Yer Değiştirme	d, kullanılan algoritmanın performansıdır, bu algoritma bu listedekilerden herhangi birisi olabilir.
Permütasyon Sıralması	Perm Sort	n	$n n!$	$n n!$	Kararsız	Yer Değiştirme	

Bazı Yegane Sıralaması	Several Unique Sort	n	n^2	n^2	Kararsız	Yer Değiştirme	Bir bilgisayar programı tarafından bulunmuştur.
Tarak Sıralaması	Comb Sort	$n \log(n)$	$n \log(n)$	$n \log(n)$	Kararsız	Yer Değiştirme	Kabarcık ve hızlı sıralama algoritmalarının birleşimi şeklinde düşünülebilir

Yukarıdaki yazıda geçen kararlılık kolonu ile, bir algoritmanın bitiş kontrolüne dayanmaktadır. Örneğin sıralı bir dizi verilse bile sıralama işlemi yapmaya çalışır mı?

SORU 46: TimeML

TimeML, olaylara bağlı zaman kavramlarını ve bu zamanlar ve olaylar arasındaki ilişkileri tutmak için ağırlıklı olarak James Pustejovsky tarafından 2003 yılından beri geliştirilen XML tabanlı bir işaretleme dilidir. Dilin web üzerindeki sayfasına <http://www.timeml.org> adresinden erişilebilir.

Dilde temel olarak 3 seviye öge bulunmaktadır bunlar:

1. Zaman içerikli olayların saklandığı ve ağacın yapraklarını oluşturan olaylar
2. Yaprakların üzerindeki ve zaman olayları üzerindeki, zaman belirleyici değerler. Bu değerlere dilde sinyal (işaret) anlamı verilmiştir.
3. Olaylar arasındaki bağlantılar (link)

1. Zamansal Kayıtlar (yaprak seviyesi):

<EVENT>: En alt seviyedeki etiketlerdir(tag) ve olayları tutmak için kullanılır. Genel olarak her olay bir fiildir ve DTD kaydı aşağıda verilmiştir:

```

attributes ::= eid class

eid ::= ID
{eid ::= EventID
EventID ::= e<integer>}
class ::= 'OCCURRENCE' | 'PERCEPTION'
| 'REPORTING' | 'ASPECTUAL'
| 'STATE' | 'I_STATE' | 'I_ACTION'

```

Yukarıdaki tanımdan da anlaşılacağı üzere bir olayın ayırt edici bir ID değeri bulunmalı ve bu değer bir tam sayı (integer) olmalıdır. Ayrıca bir olay yukarıda tanımlanan sınıflardan (Class) bir tanesine dahil olabilir.

<MAKEINSTANCE>: Bir olayın vücut bulduğu etikettir. Buna göre bir olay tanımı yapılmış ancak gerçekleşmemiş veya birden çok kere gerçekleşmiş olabilir. İşte her

gerçekleşme durumu bir <MAKEINSTANCE> etiketidir. Bu olay nesne yönelimli programlama'daki nesne ve sınıf (object , class) ayrımı olarak düşünülebilir. Bu etiketin DTD tanımı da aşağıda verilmiştir:

```
attributes ::= eiid eventID tense aspect
[polarity] [modality] [signalID] [cardinality]

eiid ::= ID
{eiid ::= EventInstanceID
EventInstanceID ::= ei<integer>}
eventID ::= IDREF
{eventID ::= EventID}
tense ::= 'PAST' | 'PRESENT'
| 'FUTURE' | 'NONE'
aspect ::= 'PROGRESSIVE' | 'PERFECTIVE'
| 'PERFECTIVE_PROGRESSIVE' | 'NONE'
polarity ::= 'NEG' | 'POS' {default, if absent, is "POS"}
modality ::= CDATA
signalID ::= IDREF
{signalID ::= SignalID}
cardinality ::= CDATA
```

Bir <MAKEINSTANCE> etiketi basitçe bir EventID alan ve bu Event'ten bir EventInstanceID döndüren mekanizma olarak düşünülebilir. Yukarıdaki dil tanımında tense ve aspect kavramlarının karşılıkları bu dilde ne yazık ki İngilizce dilinde bulunan zamanlara göre verilmiştir. Bu durum Türkçe gibi aynı zamanları içermeyen dillerin TimeML ile modellenmesini imkansız hale getirmektedir.

Örneğin İngilizce olarak aşağıda bir zaman (tense) ve ifadeler (aspect) verilmiştir:

tense="PAST"

Verb group	aspect=
was taught	"NONE"
was being taught	"PROGRESSIVE"
had been taught	"PERFECTIVE"
had been being taught (?)	"PERFECTIVE_PROGRESSIVE"

Bu etiket üzerindeki önemli bir nokta da SignalID ve Cardinality(sayısalılık) özellikleridir. Örneğin her, ençok gibi kelimeler cardinality özelliğinin değerleridir.

<TIMEX3>: Zaman ifadesidir. Setzer's (2001)'den alınan ifadeye TIMEX ve Ferro, et al. (2002)'den alınan ifadeye TIMEX2 isimleri verilmiş daha sonra yapılan güncellemeler ile bu iki ifadeden de farklı bir yapı kazandığı için TIMEX3 şeklinde isimlendirilmiştir. Kısaca bir zamanı ifade etmek için kullanılan etikettir.

```
attributes ::= tid type [functionInDocument] [beginPoint]
[endPoint] [quant] [freq] [temporalFunction]
(value | valueFromFunction) [mod] [anchorTimeID]
```

```

tid ::= ID
{tid ::= TimeID
TimeID ::= t<integer>}
type ::= 'DATE' | 'TIME' | 'DURATION' | 'SET'
beginPoint ::= IDREF
{beginPoint ::= TimeID}
endPoint ::= IDREF
{endPoint ::= TimeID}
quant ::= CDATA
freq ::= CDATA
{freq ::= duration}
functionInDocument ::= 'CREATION_TIME' | 'EXPIRATION_TIME'
| 'MODIFICATION_TIME' | 'PUBLICATION_TIME'
| 'RELEASE_TIME' | 'RECEPTION_TIME'
| 'NONE' {default, if absent, is 'NONE'}
temporalFunction ::= 'true' | 'false'
{default, if absent, is 'false'}
{temporalFunction ::= boolean}
value ::= CDATA
{value ::= duration | dateTime | time | date | gYearMonth
| gYear | gMonthDay | gDay | gMonth}
valueFromFunction ::= IDREF
{valueFromFunction ::= TemporalFunctionID
TemporalFunctionID ::= tf<integer>}
mod ::= 'BEFORE' | 'AFTER' | 'ON_OR_BEFORE' | 'ON_OR_AFTER'
| 'LESS_THAN' | 'MORE_THAN'
| 'EQUAL_OR_LESS' | 'EQUAL_OR_MORE' | 'START'
| 'MID' | 'END' | 'APPROX'
anchorTimeID ::= IDREF
{anchorTimeID ::= TimeID}

```

Yukarıdaki tanımlara uyan bazı örnekler aşağıda verilmiştir:

```

no more than 60 days (60 günden kısa)
<TIMEX3 tid="t1" type="DURATION" value="P60D" mod="EQUAL_OR_LESS">
no more than 60 days
</TIMEX3>
the dawn of 2000 (2000'in şafağı)
<TIMEX3 tid="t2" type="DATE" value="2000" mod="START">
the dawn of 2000
</TIMEX3>

twice a month (ayda iki kere)

<TIMEX3 tid="t3" type="SET" value="P1M" freq="2X">
twice a month
</TIMEX3>
three days every month (Her ay üç gün)
<TIMEX3 tid="t4" type="SET" value="P1M" quant="EVERY" freq="3D">
three days every month
</TIMEX3>
daily (Günlük veya hergün)
<TIMEX3 tid="t5" type="SET" value="P1D" quant="EVERY">
daily
</TIMEX3>
two weeks from June 7, 2003 (7 temmuz 2003'den itibaren 2 hafta)

```

```

<TIMEX3      tid="t6"      type="DURATION"      value="P2W"      beginPoint="t61"
endPoint="t62">
two weeks
</TIMEX3>
<SIGNAL sid="s1">
from
</SIGNAL>
<TIMEX3 tid="t61" type="DATE" value="2003-06-07">
June 7, 2003
</TIMEX3>
<TIMEX3 tid="t62" type="DATE" value="2003-06-21" temporalFunction="true"
anchorTimeID="t6"/>
1992 through 1995 (1992'den 1995'e kadar veya 1992 ile 1995 arası)
<TIMEX3 tid="t71" type="DATE" value="1992">
1992
</TIMEX3>
<SIGNAL sid="s1">
through
</SIGNAL>
<TIMEX3 tid="t72" type="DATE" value="1995">
1995
</TIMEX3>
<TIMEX3      tid="t7"      type="DURATION"      value="P4Y"      beginPoint="t71"
endPoint="t72" temporalFunction="true"/>

```

2. Sinyaller (işaretler)

<SIGNAL> etiketinin amacı zamanlar üzerindeki etkilerin belirlenmesidir. Örneğin tekrarlı olaylar veya sürekli olaylar gibi zaman kavramları sinyal belirterek tanımlanabilir. DTD tanımı aşağıda verilmiştir:

```

attributes ::= sid

sid ::= ID
{sid ::= SignalID
SignalID ::= s<integer>}

```

3. Bağlantılar (Links)

Birden fazla olay üzerinde bir bağlantı tanımlamaya yarar. 3 ana grupta toparlanabilirler:

- TLINK olayların bağlanmasını sağlar (örneğin “depremde sonra televizyonlar canlı yayına geçtiler”)
- SLINK ikincil veya yan cümleden gelen bağlantılardır. Örneğin “Ali eve geç geliyor, akşam yemeklerinde evde olmuyor”
- ALINK bakış ifade eden bağlantılardır. Örneğin “gemi batmaya başladı”, “kurtarma ekibi kurtulanları aramayı bıraktı”

Bu bağlantı etiketlerinin (tags) DTD tanımları aşağıda listelenmiştir:

<TLINK>

```

attributes ::= [lid] [origin] (eventInstanceID | timeID) [signalID]
(relatedToEventInstance | relatedToTime) relType

```

```

lid ::= ID
{lid ::= LinkID
LinkID ::= 1<integer>}
origin ::= CDATA
eventInstanceID ::= IDREF
{eventInstanceID ::= EventInstanceID}
timeID ::= IDREF
{timeID ::= TimeID}
signalID ::= IDREF
{signalID ::= SignalID}
relatedToEventInstance ::= IDREF
{relatedToEventInstance ::= EventInstanceID}
relatedToTime ::= IDREF
{relatedToTime ::= TimeID}
relType ::= 'BEFORE' | 'AFTER' | 'INCLUDES' | 'IS_INCLUDED' | 'DURING' |
'SIMULTANEOUS' | 'IAFTER' | 'IBEFORE' | 'IDENTITY' |
'BEGINS' | 'ENDS' | 'BEGUN_BY' | 'ENDED_BY'

```

<SLINK>

```

attributes      ::=      [lid]      [origin]      [eventInstanceID]      [signalID]
subordinatedEventInstance relType

```

```

lid ::= ID
{lid ::= LinkID
LinkID ::= 1<integer>}
origin ::= CDATA
eventInstanceID ::= IDREF
{eventInstanceID ::= EventInstanceID}
subordinatedEventInstance ::= IDREF
{subordinatedEventInstance ::= EventInstanceID}
signalID ::= IDREF
{signalID ::= SignalID}
relType ::= 'MODAL' | 'EVIDENTIAL' | 'NEG_EVIDENTIAL'
| 'FACTIVE' | 'COUNTER_FACTIVE'

```

<ALINK>

```

attributes      ::=      [lid]      [origin]      eventInstanceID      [signalID]
relatedToEventInstance relType

```

```

lid ::= ID
{lid ::= LinkID
LinkID ::= 1<integer>}
origin ::= CDATA
eventInstanceID ::= ID
{eventInstanceID ::= EventInstanceID}
signalID ::= IDREF
{signalID ::= SignalID}
relatedToEventInstance ::= IDREF
{relatedToEventInstance ::= EventInstanceID}
relType ::= 'INITIATES' | 'CULMINATES' | 'TERMINATES' | 'CONTINUES' |
'REINITIATES'

```

SORU 47: HTML+TIME

HTML -> Hyper Text Markup Language (Hiper metin işaretleme dili)
TIME -> Timed Interactive Multimedia Extensions (Zaman etkileşimli çoklu ortam uzantıları)

Microsoft, Compac/DEC ve Macromedia firmaları tarafından W3C'a gönderilen bir dil önerisidir. Dil, XML üzerine kurulu SMIL (Synchronized Multimedia Integration Language , Eş güdümlü çoklu ortam uyarlama dili) üzerine kuruludur ve bu dildeki amaç ses, görüntü veya filim gibi çoklu ortam uygulamalarının web üzerinde bir standart olan HTML dilinin üzerine zamanlama ile birlikte gömülmesidir. Buna göre örneğin Internet Gezgini 5.5 sonrasında microsoft basit bir ses kaydının sayfada istenildiği zamanda çalınmasına imkan vererebilmektedir.

Örneğin	aşağıdaki	kodu	inceleyelim:
<HTML>			
<HEAD>			
<STYLE>			
.time	{behavior:	url(#default#time2);}	
</STYLE>			
</HEAD>			
<BODY>			
<H1 CLASS="time" BEGIN="0" DUR="11" TIMEACTION="style" STYLE="Color:Red;">timeAction</H1>			
<P>Bu satır derhal görüntülenir ve altına yeni satırlar eklenir</P>			
<P CLASS="time" BEGIN="2" DUR="5" TIMEACTION="display">2 saniye sonra görüntülenir.</P>			
<P CLASS="time" BEGIN="4" DUR="5" TIMEACTION="display">4 saniye sonra görüntülenir.</P>			
<P CLASS="time" BEGIN="6" DUR="5" TIMEACTION="display">6 saniye sonra görüntülenir.</P>			
<P>Son			satır.</P>
</BODY>			
</HTML>			

Yukarıdaki kodda, bulunan 5 satırdan ilk ve son satırlar sayfanın yüklenmesinde görüntülenirken, 2. 3. ve 4. satırlar 2şer saniye ara ile görüntülenir. Bunu sağlayan P taginin içindeki sınıf stillerine (class style) ilgili zaman kodlarının yüklenmiş olmasıdır. İlginç olan diğer bir özellik ise bu yazıların sadece 5şer saniye ekranda görüntülenmeleridir.

SORU 48: OWL Time (OWL Zaman, Web Varlıkbilim Dili Zaman)

Gelişen zamanlama ihtiyaçları ile birlikte zamanın gösterimi ve formüllemesi de bir ihtiyaç haline gelmiştir. Örneğin yapılan her siparişte, siparişin zamanının tutulması, basir bir kiralama işleminde veya bilet satış işleminde yapılan işlemin hangi tarih ve saatler için yapıldığının tutulması artık sıradan birer gereksinim haline gelmiştir. Bu amaçla doğmuş olan OWL Time , Web Ontology Language (Baş harflerinin sırası okunmasını kolaylaştırmak için değiştirilmiş ve OWL olarak kısaltılmıştır) altında zaman göstermek amacıyla XML üzerine kurulu bir dil olarak tasarlanmıştır. Projenin daha önceki ismi DAML-Time olarak geçmekteydi.

Dilin yapısında varlıklar TemporalEntity (zaman varlığı veya fânî varlık olarak isimlendirilmektedir) ve her TemporalEntity altında Instant (kesin, ânî) veya Interval (aralık, zaman aralığı) olarak iki farklı varlık bulunur.

```
:Instant
a      owl:Class ;
rdfs:subClassOf :TemporalEntity .
:Interval
a      owl:Class ;
rdfs:subClassOf :TemporalEntity .
:TemporalEntity
a      owl:Class ;
rdfs:subClassOf :TemporalThing ;
owl:equivalentClass
[ a      owl:Class ;
owl:unionOf (:Instant :Interval)
] .
```

Bu durum yukarıdaki RDF/XML gösterimi ile temsil edilmiştir. Dikkat edilirse Instant ve Interval sınıfları (Class) , TemporalEntity sınıfının (Class) birer alt sınıfıdır (subclass)

Sürelerin İfade edilmesi

OWL-Time kullanılarak sürelerin ifade edilmesi de ayrı bir problemdir. Örneğin bir eylemin süresi 1 gün ve 2 saat veya 26 saat veya 1560 dakika olabilir ve bütün bu zamanlar aynı süreye işaret etmektedir. Dolayısıyla sağlıklı bir zaman gösteriminde Yıl, Ay, Hafta, Gün, Saat, Dakika, Saniye olmak üzere 7 farklı zaman gösterimi gerekmektedir. Ayrıca bu zaman gösterimini belirli eden ilave bir gösterici ile toplam 8lik gösterici kullanılmaktadır. Bu durum aşağıdaki sınıf yapısında gösterilmiştir:

```
:DurationDescription
a      owl:Class ;
rdfs:subClassOf
[ a      owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :seconds
] ;
rdfs:subClassOf
[ a      owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :minutes
] ;
rdfs:subClassOf
[ a      owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :hours
] ;
rdfs:subClassOf
[ a      owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :days
] ;
rdfs:subClassOf
[ a      owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :weeks
] ;
```



```

rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :months
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :years
] .

```

Yukarıda gösterilen sınıf DurationDescription yani Süre Tanımlayıcı sınıfının yapısıdır. Bu sınıftan üretilen bir örnek aşağıda verilmiştir:

```

duration
a :DurationDescription ;
:seconds 20 ;
:hours 5 ;
:days 15 .

```

Yukarıdaki örnekte 15 gün 5 saat ve 20 saniyelik bir zaman dilimi temsil edilmiştir.

Yukarıda Süre Tanımlayıcısı (DurationDescription) sınıfının tanımına dikkat edilirse her zaman birimi için ayrıca bir de maxCardinality (azami sayısalılık) tanımı yapılmış ve bütün zaman birimleri için bu değer 1 atanmıştır. Bunun OWL dili açısından anlamı bu değerlerin sayısal olarak istedikleri gibi atanabilecekleridir. Ancak OWL dili bilindiği üzere varlıkbilimsel bir dildir ve her zamanın bütün bu birimleri ifade etmesi beklenemez. Örneğin üniversite eğitiminin saniyeler dakikalar cinsinden ifadesi beklenmedik bir durumdur, genelde üniversite eğitimi 2,3,4,5,6 gibi yıllar ile ifade edilir. Veya bir yemek süresi dakika ve saatler mertebesinde bu sürenin yıllar ile ifade edilmesi beklenmedik bir durumdur. İşte OWL-Time bu gibi durumlarda diğer zaman birimlerinin kapatılmasına imkan verir ve yapılması gereken basitçe bu zaman birimlerinin maxCardinality bilgisini 0 yapmaktır:

```

:Year
a owl:Class ;
rdfs:subClassOf :DurationDescription ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:cardinality 1 ;
owl:onProperty :years
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:cardinality 0 ;
owl:onProperty :months
] ;
...

rdfs:subClassOf
[ a owl:Restriction ;
owl:cardinality 0 ;
owl:onProperty :seconds
] .

```

Örneğin yukarıda bir yıl sınıfı tanımı yapılmıştır ve yıl dışındaki bütün zaman birimlerinin cardinality (sayısalılık) bilgisi 0 yapılmıştır.

OWL-Time kullanılarak belirli bir tarihin veya saatin ifade edilmesi de gerekmektedir. Örneğin 10:30 bir saat ifade etmektedir buna karşılık 26 mart 2008 ise bir tarih ifade etmektedir. Dolayısıyla herhangi bir zamanı ifade etmek için öncelikle hangi birimden ifadenin yapılacağına karar verilmelidir. Yani ifade edilecek olan zaman bir dakikayı mı bir günü mü yada bir saati mi ifade etmek amacındadır belirlenmelidir. (“saat 10” önermesi bir saat ifade ederken “26 mayıs” bir günü ifade etmektedir). Bu belirlenen birime unitType (birim tipi) denilirse geriye zaman diliminin ifade edilmesi sorunu kalmaktadır. Yani istenilen saat acaba hangi zaman dilimine göre (örneğin GMT, EST, PST) belirtilmektedir. Son olarak tarih ve saati ifade eden zaman birimleri aşağıdaki şekilde tanımlanabilir:

```

:DateTimeDescription
a owl:Class ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:cardinality 1 ;
owl:onProperty :unitType
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :second
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :minute
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :hour
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :day
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :dayOfWeek
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :dayOfYear
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :week
] ;
rdfs:subClassOf
[ a owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :month
] ;
rdfs:subClassOf

```

```
[ a          owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :year
] ;
rdfs:subClassOf
[ a          owl:Restriction ;
owl:maxCardinality 1 ;
owl:onProperty :timeZone
] .

:hasDateTimeDescription
a          owl:ObjectProperty ;
rdfs:domain :DateTimeInterval ;
rdfs:range :DateTimeDescription .
```

Görüldüğü üzere yukarıdaki gösterimde haftanın günleri veya yılın günü gibi kavramlar da ilave edilerek bir tarih/saat ifadesine yer verilmiştir. İlk eklenen alt sınıf (subclass) bir zaman birimini gösterirken son alt sınıf ise zaman dilimini (timeZone) göstermektedir.

Bu bilgiler paralelinde aşağıdaki toplantı varlığını inceleyelim:

```
:meetingStart
a          :Instant ;
:inDateTime
:meetingStartDescription ;
:inXSDDateTime
2006-01-01T10:30:00-5:00 .

:meetingStartDescription
a          :DateTimeDescription ;
:unitType :unitMinute ;
:minute 30 ;
:hour 10 ;
:day 1 ;
:dayOfWeek :Sunday ;
:dayOfYear 1 ;
:week 1 ;
:month 1 ;
:timeZone tz-us:EST ;
:year 2006 .
```

Yukarıda aynı toplantı varlığı için iki farklı gösterim bulunmaktadır. İlk gösterimde XSD zamanı denilen XML Scheme Definition kelimelerinin kısaltılmışı olan ve XML üzerinde tanımlama yapılan dilin biçimine uygun olarak yazılmış gösterime yer verilmiştir. Aynı tarih değeri altta bizim tanımladığımız tarih saat yapısına uygun olarak tekrar gösterilmiştir. Buna göre toplantı dakika cinsinden bir zamana işaret etmekte olup zaman birimi EST'dir ve toplantının zamanı: 1/1/2006 saat 10:30'dur.

OWL Time kullanarak zaman aralığı gösterimi
Bilindiği üzere zaman gösterimlerinde her zaman kesin ve belirli zamanları ifade etmek mümkün değildir. Örneğin bir kargo firmasının teslim süresi olarak 2-3 gün arasında demesi, teslimin en az 2 en çok 3 günde gerçekleşeceğini ifade etmektedir. Bu durum OWL Time kullanılarak aşağıdaki şekilde ifade edilebilir:

```
:DeliveryDuration
a          owl:Class ;
rdfs:subClassOf
```

```
[ a          owl:Restriction ;
owl:cardinality 1 ;
owl:onProperty :maxDeliveryDuration
] ;
rdfs:subClassOf
[ a          owl:Restriction ;
owl:cardinality 1 ;
owl:onProperty :minDeliveryDuration
] .
```

```
:maxDeliveryDuration
a          rdf:Property ;
rdfs:domain :DeliveryDuration ;
rdfs:range time:Interval .
```

```
:minDeliveryDuration
a          rdf:Property ;
rdfs:domain :DeliveryDuration ;
rdfs:range time:Interval .
```

Yukarıdaki gösterimde bir teslim tarihi için iki alt bilgi girilmiş bunlardan birisi minDelivery (asgari teslim) diğeri ise maxDelivery (azami teslim) süresi olarak belirlenmiştir. Bu belirlemenin yanında 2 günlük ve 3 günlük sürelerde aşağıdaki şekilde tanımlanabilir:

```
:Interval2Days
a          owl:Class ;
rdfs:subClassOf time:Interval ;
owl:subClassOf
[ a          owl:Restriction ;
owl:hasValue P2D ;
owl:onProperty time:durationDescriptionDataType
] .
```

```
:Interval3Days
a          owl:Class ;
rdfs:subClassOf time:Interval ;
owl:subClassOf
[ a          owl:Restriction ;
owl:hasValue P3D ;
owl:onProperty time:durationDescriptionDataType
] .
```

Artık bu tanımlamalardan sonra kargo firmasının teslim süresi tanımlanabilir:

```
:Cargo2-3dayDuration
a          owl:Class ;
rdfs:subClassOf :DeliveryDuration ;
rdfs:subClassOf
[ a          owl:Restriction ;
owl:allValuesFrom :Interval3Days ;
owl:onProperty :maxDeliveryDuration
] ;
rdfs:subClassOf
[ a          owl:Restriction ;
owl:allValuesFrom :Interval2Days ;
owl:onProperty :minDeliveryDuration
] .
```

yukarıda görüldüğü üzere kargo firmasının teslim süresi 2 günlük zaman diliminde asgari ve 3günlük zaman diliminde azami süreler olarak tanımlanmıştır.

OWL-Time kullanarak ne yazık ki belirsiz iki eylemin göreceli zamanlarını tanımlamak mümkün değildir. Örneğin A olayı B olayından sonra olmuştur önermesi OWL-Time kullanımıyla gösterilemez.

SORU 49: TTML (Time Tabling Markup Language, Zaman Çizelgeleme İşaretleme Dili)

Zaman çizelge işaretleme dili (Timetabling Markup Language (TTML)), XML üzerine kurulmuştur. MathML üzerine kurulu zaman çizelgeleme problemlerinin çözümünde kullanılmaktadır.

TTML üzerinde kullanılan formülleme dili küme teorisine dayandırılabilir. Örneğin MathML üzerinde kullanıcılara tamamen yeni semboller ile bu sembollerin ifade ettiği yeni fonksiyon ve formülleri tanımlama imkanı sağlar. Bu durum MathML kullanan TTML dilinin de küme teorisini desteklemesini sağlamaktadır. Bu sayede kullanıcı esnek bir şekilde kendi kıstaslarını belirleyebilmekte ve zaman çizelgelemesini özelleştirebilmektedir.

Bilindiği üzere zaman çizelgelemesi [NP Complete \(NP Tam\)](#) problem ailesindendir ve zaman çizelgelemesindeki amaç uygun zaman aralıklarına uygun olaylar kümesini yerleştirmektir. Gelişmekte olan şartlara göre zaman çizelgelemede aranan özellikler ve kurallar kümesi de gelişmekte dolayısıyla problemin karmaşıklığı artmaktadır. Bu konuda bir standart bulunmaması ayrıca araştırmacıların karşılaştırma imkanını kaldırmakta ve mukayeseli sonuçlar çıkarılmasını zorlaştırmaktadır. Bu konuda ilk çalışma Andrw Cumming tarafından ICPTAT'95 'de yapılmış ve çalışmanın sonucu olan dile SSTL ismi verilmiştir. Beklenildiği üzere SSTL dili de bir standartlaşmayı sağlayamamıştır çünkü elde bulunan problemlerin bu dile çevrilmesi tam anlamıyla mümkün olmamıştır. Günümüzde neredeyse her araştırmacı kendi gösterimini kullanmakta ve bu konuda bir standartlaşma henüz sağlanamamaktadır.

SORU 50: Hitabe (Nutuk, Söylev, Discourse)

Hitabe, dil bilim açısından bir konu üzerine yoğunlaşmış birden fazla cümleden meydana gelen açıklama, konuşma, diyalog ya da tartışmalara verilen isimdir. Anlambilimsel (semantic) çalışmalar açısından aynı konuya yönelmiş cümlelerin anlattığı konunun bulunması kolay hale gelmektedir.

Hitabe metinlerinin analizi doğal dil işleme açısından da büyük önem taşımaktadır çünkü metinler temel olarak bir kavramın dolayısıyla pramaların (durum ve koşullara bağlı gerçeklerin) çıkarılmasında kullanılabileceğine inanılan (ancak tam olarak hiç bir zaman gerçekleşmemiş) bilgiler barındırmaktadır. Örneğin Platon okulunun iddiasına göre “herşey” hitabet ile açıklanabilir. Dolayısıyla doğal dil işlemenin temelini oluşturan anlambilim ve [pragma çıkarılmasında](#) önemli bir alternatif niteliğindedir. Benzer bir çıkarım ise [külliyat \(corpus\)](#) üzerine yapılmaktadır.

SORU 51: Terminoloji Çıkarımı (Terminology Extraction)

Terminoloji çıkarımı doğal dil işlemenin uygulama konularından birisi olan bilgi çıkarımının alt konusudur. Amaç verilen bir [külliyattan \(corpus\)](#) otomatik olarak terimleri ve anlamlarını çıkarmaktır.

Günümüzde gelişen İnternet kaynaklarına bağlı olarak anlambilimsel ağlar (Semantic web) çalışmaları ve buna paralel olarak ağ uygulamaları (web applications), ağ hizmetleri (web services) , [ağ dolaşıcıları \(web crawlers\)](#) teknolojilerinin arkasında yatan amaç internet üzerindeki terimlerin çıkarılması, saklanması ve istenilen sorgulara göre işlenerek cevaplar üretilmesidir.

Terim çıkarma işlemi genellikle [isimlere](#) odaklanmış bir işlemdir ve amaç [isim kelime gruplarını \(noun phrase\)](#) çıkararak bu işlemi otomatize etmektir. Ayrıca başarılı bir çıkarım sonucu şayet isim kelime grubu bütün belirleyiciliği ile elde edilebilirse ifade edilmek istenen terimi açıklayan ve detaylandıran bir terime ulaşmak mümkün olacaktır. Bu durum belirsizliğin az olması sayesinde gerek [varoluş modellemelerinde \(ontology model\)](#) gerek de bilgi etki alanlarında (knowledge domain) oldukça kıymetli bilgiler elde edilmesini sağlar. Örneğin bu terimlerin elde edilmesi makine tercümesinde (machine translation) önemli bir role sahiptir.

SORU 52: Külliyat (corpus)

Dil biliminde ve günümüz doğal dil işleme çalışmalarında külliyat (corpus) kelimesi ile kastedilen, çok sayıdaki metnin düzenli ve yapısal olarak bir arada bulunması durumudur.

Külliyatlar, muhteviyatına göre tek dilli (tek bir dilden metinler içeren) veya çok dilli (birden fazla dilden metinler içeren) olarak sınıflandırılabilir.

Külliyatların, doğal dil işlemede kullanılmasının sağladığı bir avantaj da kelimelerin sınıflandırılması sırasında her kelime için yardımcı önermelerin (lemma) çıkarılmasıdır. Buna göre aynı yazılışa sahip (eş sesli) farklı anlamda kelimeler bulunması durumlarında bu kelimelerin anlam çözümlemelerinde kullanılabilecek bilgiler külliyata göre değişiklik göstermekle birlikte, mevcut külliyat bu anlamın tespitinde yardımcı rol oynayabilir. Bu konuda Kelime-İfade Belirsizliği (word-sense ambiguity) konusuna bakabilirsiniz.

SORU 53: Eş Atf (Coreference)

Bir isim zincirinde (birbirine atıfta bulunarak ilişkilendirilen kelime listesi) aynı kelimeye atıfta bulunan birden fazla kelimenin bulunması durumudur. Örneğin “Sen, o adamın, senin arkadaşın olduğunu söylemiştin” cümlesinde “Sen” ve “senin” kelimeleri aynı kişiye atıfta bulunmaktadır.

SORU 54: Bilgi Çıkarımı (Information Extraction)

Bilgi çıkarımı konusu, genellikle bir metin üzerinde doğal dil işleme kullanılarak belirli kriterdeki bilgileri elde etmeyi hedefler. Bu işlem sırasında örneğin bir kalıba uygun olan verilerin çıkarılması istenebilir. Amaç çok miktardaki veriyi otomatik olarak işleyen bir yazılım üreterek insan müdahalesini asgarî seviyeye indirmektir. Bilginin çıkarılacağı ortam genellikle yazılı metinlerdir ancak bu metinlerin bulunacağı ortamlar değişebilir örneğin veri tabanları, internet üzerindeki dökümanlar veya taranmış metinler bu verinin kaynağını oluşturabilir.

Bilgi kelime anlamı olarak verinin işlenmiş ve anlaşılabilir veriyi ifade etmektedir. Dolayısıyla veri kaynağı olan metinlerden işlenmiş bilginin çıkarılması işlemine bilgi çıkarımı denilmektedir. Örneğin gazete haberleri veri kaynağı olarak kabul edilsin. Bu veri kaynağından şirket birleşmeleri ile ilgili bir bilginin çıkarılması işlemine bilgi çıkarımı denilebilir. (a şirketi ile b şirketinin birleştiğinin gazete haberlerinden anlaşılması gibi)

Bilgi çıkarım işleminin en zor adımlarından birisi de veriyi işlerken belirli bir yapıya oturtmaktır. Örneğin internet üzerinde yayınlanan verilerin herhangi bir standart yapısı bulunmamakta, veriler dağınık halde istenildiği gibi yayınlanmaktadır. Bu verilerin düzenli bir hale getirilmesi için XML ve benzeri teknolojilerden faydalanarak bilgi çıkarımı işleminin basitleştirilmesi hedeflenmektedir. Bu konudaki güncel uygulamalardan birisi de internet üzerinde yayın yapan kurumların birer ağ hizmeti (web service) kurarak uygulamaların karşılıklı iletişimine izin vermeleridir.

Ayrıca çok sayıda yazılım bilgi çıkarımı aşamasına alt yapı hazırlamak amacıyla çeşitli ortamlardan (örneğin internet) veri toplayarak bunları düzenli bir halde yapılandırır.

Güncel uygulamalarda bilgi çıkarımı sırasında sık rastlana problemler şunlardır:

- Varlık isimlerinin algılanması (Named Entity Recognition): Bu konu, bir veri kaynağında geçen varlıkların ve isimlerinin algılanmasını hedefler. Bu varlıklar kişi, bina, yer, [soyut varlıklar](#) veya sayısal ifadeler olabilir.
- [Eş Atıf \(Coreference\)](#) : Aynı varlığa işaret eden birden fazla kelimenin tespiti. Örneğin zamirlerin isimlere atıfta bulunması gib. Bu bağlantılar sonucunda bir isim zinciri elde edilebilir.
- [Terim Çıkarımı \(Terminology Extraction\)](#): Verilen metne ait terimlerin çıkarılması

SORU 55: Geçişsiz Fiiller (intransitive verbs)

Geçişsiz fiiller, fiillerin nesne alamayanlarını içeren bir alt kategorisidir. Örneğin yemek fiili geçişli bir fiildir çünkü “ekmeği yedi” şeklinde bir nesneye atıfta bulunabilir ancak örneğin güldü kelimesi geçişsiz bir fiildir çünkü “birşeyi gülemeyiz”. Fiilin geçişli olup olmadığının anlaşılması için fiile “neyi ” sorusu sorulabilir ve fiil buna cevap verebiliyorsa geçişli kabul edilebilir. Daha fazla bilgi için geçişli fiiller (intransitive verbs) ve çift geçişli fiiller (bitransitive verbs) bakabilirsiniz.

SORU 56: Geçişli Fiiller (transitive verbs)

Geçişli fiiller, fiillerin nesne alabilenlerini içeren bir alt kategorisidir. Örneğin yemek fiili geçişli bir fiildir çünkü “ekmeği yedi” şeklinde bir nesneye atıfta bulunabilir ancak örneğin güldü kelimesi geçişsiz bir fiildir çünkü “birşeyi gülemeyiz”. Fiilin geçişli olup olmadığının anlaşılması için fiile “neyi,kimi,ne ” sorusu sorulabilir ve fiil buna cevap verebiliyorsa geçişli kabul edilebilir.

Geçişsiz fiillere “-(i)t, (i)r, tir” ekleri getirilerek geçişli fiil haline getirilebilir.

Örneğin yukarıdaki gülmek fiili geçişsiz bir fiildir. Güldürmek şeklinde -ür eki eklenerek geçişli olabilir ve kimi sorusuna cevap verebilir.

“Aliyi güldürdük”

cümlesinde olduğu gibi.

aynı zamanda zaten geçişli olan bir fiile “-(i)t, (i)r, tır” ekleri getirilerek geçişlilik derecesi de arttırılabilir.

Örneğin yukarıdaki örnekte yemek fiili zaten geçişlidir. Yedirmek fiili (ir) eki aldığında ettirgen özelliğe sahip olur.

“Ekmeği yedirdi”

cümlesinde olduğu gibi eylemi başka bir faile yaptırma fiili haline dönüşür.

Daha fazla bilgi için geçişsiz fiiller (intransitive verbs) ve çift geçişli fiiller (bitransitive verbs) bakabilirsiniz.

SORU 57: İkili dil (bigram)

Bir dilde kelimebilimsel olarak iki ihtimalin bulunması durumudur. Örneğin L1 ve L2 olarak iki farklı sözlüksel (lexical) grubumuzun olduğunu düşünelim. Ve yine örneğin bu iki grup N ve V (isim ve fiil) grupları olsun. Bir kelimenin dahil edileceği grubun belirsizliği (veya istatistiksel olarak belirliliği) bigram olarak ifade edilir. Örneğin “uç” kelimesi Fiil veya İsim grubuna dahil edilebilir. Doğal dil işleme dünyasında bigram diller istatistiksel diller olarak kabul edilirler ve bir olasılık değerlendirmesine tâbi tutulurlar. Örneğimiz için P(N|uç) veya P(V|uç) şekline iki adet farklı şartlı olasılık çıkarılabilir.

SORU 58: Kelime Bilim (lexicology, vocabulary)

Kelime bilimi, doğal dil için tanımlanmış ve kelime seviyesindeki çalışmaların bir araya geldiği bilimin ismidir. Basitçe bir dili meydana getiren en küçük anlamlı birim kelimedir. Her kelimenin ifade ettiği bir anlam bulunur.

Dil felsefesindeki zayıf bir akıma göre “ismi olan herşey vardır” denilebilir. Yani bir şeyin varlığını ispat için isminin olması yeterlidir. Bu yaklaşımın tersi ele alınırsa olmayan birşeyin de ismi olamaz.

Basit anlamda varlıklara(yaratıklara) isim verilmesinin sonucunda kelimeler ortaya çıkar. Burada belki, soyut varlıkların (mücerret isimlerin, (abstract noun)) anlaşılması, somut varlıkların (müşahhas isimlerin (concrete names)) varlığını anlamak kadar kolay olmayabilir.

Aslında her kelime varlıkları ifade eden birer atıftır. Yani kalem kelimesi bu kelimeyi okuyan herkes için geçmişindeki farklı bir bilgiye bir atıftır. Herkesin kalem kelimesi ile aklına gelen varlık farklı olabilir. Yine kelime olarak “kalemler, kalemde, kalemin” gibi ek almış halleri de aynı varlığa atıftır. Hepsine ifade edilen varlık aynıdır. İşte bu ifade edilen varlığa dil biliminde lexeme (zati) ismi verilir. Yani varlığın bizzatihi kendisi anlamındadır.

Yukarıdaki üç kelime (kalemler, kalemde, kalemin) anlatılan anlam aynıken şekilleri farklıdır. Bu şekillerin incelendiği bilime şekil bilim anlamında morphology ismi verilir. Dolayısıyla farklı şekillerde olan kelimeler (morpheme) aynı varlığı (lexeme) ifade için kullanılabilir.

SORU 59: Ekleme (apposition)

Cümle içerisinde bir kelimenin bir veya daha fazla kelime grubu ile ifade edilmesidir. İngilizce için genelde bir isim ile isim kelime grubu arasında kurulan ilişki türüdür. Örneğin “Alice, long haired girl, has followed the white rabbit” cümlesindeki “long haired girl” alice’i anlatan isim kelime grubudur. Benzeri durum Türkçe için isim ve alt cümle arasında kurulabilir. Örneğin “O yıllarda düşmanları tarafından da “muhteşem” olarak anılan Kânunî osmanlı devletinin padişahıydı” cümlesindeki “o sıralar düşmanları tarafından da “muhteşem” olarak anılan” cümlecisi Kânunî’nin belirleyicisi olarak kullanılır.

Ekleme olarak anılmasının sebebi bir kelime hakkında ilave bilgiler sunması ve cümleye eklenerek tek bir kelimeye atıfta bulunmasıdır.

SORU 60: Dönüşü (anaphora)

Doğal dil işleme açısından dönüşümler, bir kelimenin daha ileriki cümlelerde işaret edilmesi durumunda ortaya çıkar. Örneğin “Ali çantasını evde unuttu. O okulda gitti” cümlelerinde O kelimesi aliye, çantaya veya eve işaret edebilir. Bu kelimelerin hepsi isimdir ve bir zamir ile işaret edilebilecek kelimelerdir. Dönüşümler durumu bu ve benzeri cümlelerde ortaya çıkar. Her zaman bir zamir kullanılması gerekmez, benzer durumlar örneğin gizli özneyle de ortaya çıkabilir. Örneğin “O gün, Ali’nin çiftlikteki ilk günüydü. Bütün gün ata bindi.” cümlelerinden ikinci cümlede gizli özne Ali’dir. Benzer bir durum da fiil kelime gruplarını ifade etmek için kullanılabilir. Örneğin “Bugün Ali’yi başkanlıktan azlettim. Beklenmedik durumlarda bunu yapma yetkim bulunuyor” cümlelerindeki “bunu” kelimesi bir önceki cümledeki “başkanlıkta azletme” eylemine işaret etmektedir.

Bir kelimeye ileriki cümlelerde işaret edilmesinin belirlenmesi ve bu işaretin tam olarak hangi kelimeye yapıldığının ortaya konması dönüşümler çözümleme (anaphora resolution) olarak geçer. Örneğin yukarıdaki örneklerde zamir veya gizli öznenin taşıdığı anlam bir önceki cümledeki Ali’dir yargısı bu çözümlemenin sonucudur.

SORU 61: İlgili Belirsizliği (reference ambiguity)

Doğal dil işleme açısından bir kelimenin hangi kelime ile ilişkili olduğunun muğlak olduğu durumlardır. Genelde zamirler için bu problem yaşanır. Örneğin “Ali ile Veli kavga ettiler. O çok sinirliydi.” cümlelerindeki O zamirinin tam olarak Ali’ye mi Veli’ye mi işaret ettiği belirsizdir.

SORU 62: Yapısal Belirsizlik (Structural Ambiguity)

Bir kelimenin yapısal olarak ifade ettiği bilginin belirsiz olmasıdır. Tam bir yapısal belirsizliğe örnek “yaşlı kadın ve erkekler” örneğinde görülebilir. Burada yaşlı olan sadece kadın mıdır? Yoksa kadın ve erkekler midir? Elbette bu soruya bağlı olarak çoğul olan kimdir sorusu da sorulabilir (kadınlar mı kadın mı ?)

Yapısal belirsizlikler genelde bir kelime-ifade belirsizliği durumunun bağlaç ile birlikte kullanılmasında oluşurlar. Örneğin “bakan gözler” kelimeleri için, Aliye bakan gözler, Devletin menfaatlerini bir millet vekillerinden seçilen bir bakan gözler şeklinde üç farklı anlam kazandırılabilir. Burada bağlanmış kelimeler fiil-fiil, isim-fiil olmasına göre ayrılırlar.

SORU 63: Kelime-İfade Belirsizliği (word-sense ambiguity)

Bir kelimenin ifade ettiđi anlamdaki belirsizliktir. Anlam belirsizliđi de denilebilir. Genellikle bir dilde bulunan eşsesli kelimelerden kaynaklanan belirsizliklerdir. Örneđin Türkçede yüz kelimesi türkçede 4 farklı anlama gelmektedir (sayı, denizde yüzmek, deri yüzmek ve çehre anlamlarında). Bu kelimenin cümle içerisinde kullanımı da problem oluşturmakta ve cümlede tam olarak hangi anlama geldiđini belirlemek gerekmektedir.

SORU 64: Belirsizlik (ambiguity, muğlaklık, ikircimlik)

Doğal dil işleme açısından belirsizlik bir cümlede birden fazla anlamı bulunduran kelime, kelime grubu yada cümle sonucunun olmasıdır. Doğal dil işlemede üç tip belirsizlik bulunur:

1. [Kelime-anlam belirsizliđi \(word-sense ambiguity\)](#)
2. [Yapısal belirsizlik \(structural ambiguity\)](#)
3. [İlgi belirsizliđi \(referential ambiguity\)](#)

SORU 65: Bağlaç (conjunction)

İki farklı anlamsal kavramı birbirine bağlamak için kullanılan kelime tipleridir. Uyum bağlaçları (coordinate conjunction) ve Şart bağlaçları (Subordinate conjunction) (veya zarf bağlaçları veya yan cümlecik bağlaçları) olarak iki grupta incelenebilir.

Uyum bağlaçları anlamca birbirine yakın iki grubu bağlamak için kullanılır. Örneđin “ali ve veli okula gitti” cümlesindeki ve bağlacı anlamca uyumlu iki kelime olan ali ve veli fâillerini birbirine bağlamıştır.

Şart bağlaçlarında ise genelde iki cümle bir şart üzerinden birbirine bağlanır. Örneđin “yağmur yağar ise şemsiyemi açarım” cümlesindeki ise bağlacı iki cümleyi (yağmur yağmak ve şemsiyeyi açmak) birbirine bir şartla bağlamıştır. Başka bir örnekte “güneş doğduđuşu ile saldırı başlayacak” cümlesinde ile bağlacı yine iki cümleciciđi birbirine bağlamaktadır.

Bağlaçlar temel olarak kelime gruplarını birbirine bağlamakta kullanılırlar.

isimler: kediler ve köpekler [dolaşıyordu]

sıfatlar: siyah ve kırmızı [renkli elbise]

zarflar: hızlı ve akıcı [konuşuyordu]

fiiller: oynadı ve kazandı

SORU 66: Edat (preposition)

Bir isim kelime grubunun cümle içerisindeki rolünü belirten kelime çeşididir. Örneđin “ile” edadı genelde bir NP (noun phrase, isim kelime grubunun) kullandığı aracı ifade eder. Örneđin “ankaraya uçak ile gitti” cümlesindeki gitmek fiilinin uçak aracı ile olması gibi veya “tahtaya kalem ile resim çizdi” cümlesindeki çizmek fiilinin kalem aracı ile olması gibi. Bazı durumlarda da bir eylemin ikinci etmenini (co-agent) ifâde için kullanılır. Örneđin “ali ile tahtayra resim çizdi” cümlesindeki “ile” edadı ikinci bir etmeni ifade etmek için kullanılmıştır.

Edatlar, bazı durumlarda isim kelime gruplarını bağlamak için de kullanılabilirler. Ayrıca bir isim kelime grubunun yapabileceği eylemi belirtmek için de kullanılırlar (animate). Yani her eylem her isim kelime grubu tarafından yapılamaz. Örneğin “balta camı kırdı” ve “çocuk camı kırdı” cümlelerinden ilkinde balta kelimesi bir canlandıran (etken, fail, animate) değildir çünkü . Buna mukâbil ikinci cümlede çocuk kelimesi etken olarak kabul edilebilir.

SORU 67: zarf (adverb)

Zarflar bir fiili (verb) bir sıfatı (adjective) veya başka bir zarfı tanımlamak için kullanılan kelimelerdir. Kelimeler kök olarak isim veya sıfattan türetilebilir. Bu türetme işlemi sırasında genelde sıfat olarak kullanılan kelimeler ek almazken isimden gelen kelimelerin isimden sıfat yapım eki kullandıkları görülür. Örneğin “hızlı sürmek” kelime grubunda hızlı kelimesi sürmek fiilini , “sarı benekli kumaş” kelime grubunda ise sarı kelimesi benekli kelimesini tanımlamaktadır.

Zarf kelime grupları tanım itibariyle ya fiil, yada sıfat ile bitmelidir. Zarf kelime grupları genelde tek başlarına bir anlam ifade etmezler ancak betimledikleri sıfat veya fiil ile anlamlı hale gelirler ve genelde bu fiil veya sıfatın özelliği olarak algılanır ve sınıflandırılırlar.

SORU 68: Sıfat (adjective)

Genel anlamda isim olarak kabul edilebilen ancak başka bir ismin bir özelliğini belirtmeye yarayan kelimelerdir. Kısaca ADJ olarak kullanılırlar. Örneğin, kırmızı, güzel veya yavaş birer sıfattır. Bu örneklerin ortak yanı kelimelerin birer özellik ifade etmeleridir. Ayrıca anlam itibariyle birer özellik belirtmeyen isimlerde ek alarak bu gruba dahil olabilir. Örneğin cam kelimesi isimdir ancak “camlı ev” tamlamasındaki cam kelimesi -lı eki olarak sıfat haline gelmiştir.

Sıfatların diğer bir kullanımı ise isimlere benzer şekilde cümlede bir fiil ifade etmesidir. Bu işlem için örneğin olmak veya -dır eki kullanılabilir. “Araba yeşildir” cümlesinde olduğu gibi. Bu durum ingilizcede bulunan yardımcı fiillerin de fiil kabul edilmesinden kaynaklanmaktadır.

SORU 69: Fiil (verb)

Bir eylemi belirten kelimelerdir. Doğal dil işleme dünyasında daha çok V harfi olarak kısaltılırlar. Genelde bir fiilin ektisi altında kalan bir obje (mefûl) ve bu fiili yapan kişi (subje, Fâil) bulunur. Ayrıca fiilin yapıldığı zaman ve fiilin yapılış şeklini belirten belirleyicilerde bulunabilir.

Örneğin “Ali Baba kapıyı açtı.” cümlesindeki Ali Baba fâil, kapı mefûl ve açmak fiidir. Açmak fiiline eklenen -tı eki ise eylemin geçmiş bir zamanda olduğunu belirtmektedir.

Köklerine göre fiiller

Türkçede, [isim köklü kelimeler](#) de fiil olarak kullanılabilir. Bunun için kelimelerin sonuna isimden fiil yapma eki adı verilen yapım eklerinden en az birisinin gelmesi gerekir. Örneğin “Ali bahçeyi suladı” cümlesinde sulamak fiilinin kökü su kelimesidir ve su kelimesi bir isimdir.

Türkçede ayrıca [isimlerinde](#) fiil gibi kullanılması mümkündür. Örneğin “Her öğrenci bir insandır” cümlesindeki insan bir isimdir ancak -dır ekini alarak fiil gibi kullanılmıştır. Bu cümlelere isim cümleleri denilmez ancak İngilizce’de bu cümleler de birer fiil cümlesi olarak kabul edilmektedir çünkü İngilizcede bulunan “am is are” yardımcı fiilleri birer fiil olarak kabul edilmiştir. Doğal dil işleme sırasında bu farkın iki alternatifi de kullanılabilir.

Zamanlarına göre fiiller

Dilbilimde zaman olarak fiiller 3 zamanda incelenir:

- Geçmiş zaman
- Şimdiki zaman
- Gelecek zaman

Ancak Türkçe zaman bakımından biraz daha farklı zamanlar sunar. Buna göre Türkçede fiil zamanları basit ve bileşik zamanlı olarak iki grupta incelenebilir. Basit fiil zamanları:

- Rivayet Geçmiş Zaman (Gelmiş)
- Hikaye Geçmiş Zaman (Geldi)
- Geniş Zaman (Gelir)
- Şimdiki Zaman (Geliyor)
- Gelecek Zaman (Gelecek)

Yukarıda da görüldüğü üzere geçmiş zaman Türkçede iki farklı grupta bulunurken bir de geniş zaman eklenmiştir.

Ayrıca Türkçede birleşik fiil zamanlarının kullanılması da mümkündür. Bu zamanlar:

- Hikaye Bileşik Zaman (Geliyor-dum, Gelecek-dim, Gelmiş-dim, Gelir-dim, Geldiy-dim)
- Rivayet Bileşik Zaman (Geliyor-muşum, Gelecek-mişim, Gelmiş-mişim, Gelir-mişim, Geldiy-mişim)
- Şart Bileşik Zaman (Geliyor-sam, Gelecek-sem, Gelmiş-sem, Gelir-sem, Geldiy-sem)
- Katmerli Bileşik Zaman (Birden fazla birleşik zaman bulunduran fiiller, Gelecek idiysem gibi)

olarak sıralanabilir. Yukarıda göstermek için mekanik olarak üretilen bu fiillerde, Hikaye geçmiş zamanın rivayeti Türkçede bulunmaz. Yani geldiymiş fiili hatalıdır.

Ayrıca Türkçede:

- Emir kipinin hikayesi
- Emir kipinin rivayeti
- Dilek-şart kipinin şartı
- İstek kipinin şartı
- Emir kipinin şartı
- Hikaye geçmiş zamanın rivayeti

şeklinde sıralayabileceğimiz birleşik zamanlı fiiller kullanılmaz ve bunların üretilmesi bir dilbilgisi hatası olarak kabul edilebilir.

HİKÂYE KİPİ	
Hikâye geçmiş zamanın hikâyesi	Geldiydim
Rivayet geçmiş zamanın hikâyesi	Gelmiştim
Şimdiki zamanın hikâyesi	Geliyordum
Gelecek zamanın hikâyesi	Gelecektim
Geniş zamanın hikâyesi	Gelirdim
Dilek-şart kipinin hikâyesi	Gelseydim
Gereklilik kipinin hikâyesi	Gelmeliydim
İstek kipinin hikâyesi	Geleydim
Emir kipinin hikâyesi	YOK
RİVAYET KİPİ	
Hikâye geçmiş zamanın hikâyesi	YOK
Rivayet geçmiş zamanın hikâyesi	Gelmişmişim
Şimdiki zamanın rivayeti	Geliyormuşum
Gelecek zamanın rivayeti	Gelecekmişim
Geniş zamanın rivayeti	Gelirmişim
Dilek-şart kipinin rivayeti	Gelseymişim
Gereklilik kipinin rivayeti	Gelmeliymişim
İstek kipinin rivayeti	Geleymişim
Emir kipinin rivayeti	YOK
ŞART KİPİ	
Hikâye geçmiş zamanın hikâyesi	Geldiysem
Rivayet geçmiş zamanın hikâyesi	Gelmişsem
Şimdiki zamanın şartı	Geliyorsam
Gelecek zamanın şartı	Geleceksem
Geniş zamanın şartı	Gelirsem
Dilek-şart kipinin şartı	YOK
Gereklilik kipinin şartı	Gelmeliysem
İstek kipinin şartı	YOK
Emir kipinin şartı	YOK

Yukarıdaki tabloda bütün zamanların ve kiplerin gelmek fiilini birinci tekil şahıs için çekimini görmek mümkündür.

Fiil Kiplerinde Anlam Kayması

Fiil kipleri eklendikleri fiillerin anlamlarını değiştirmektedirler ve kipler sınıflandırıldıkları grup içerisindeki anlamı getirmekle mes'uldür. Ancak bazı durumlarda bir fiil kipi sorumlu olduğu anlam dışında kullanılarak farklı bir anlama gelebilir. Bu şekilde fiillerin kiplerinin anlamlarının değişmesine fiil kipinde anlam kayması ismi verilir.

Aşağıda fiil kiplerindeki kayma ve kaydıktan sonraki anlamları listelenmiştir.

- Şimdiki zaman kipinde kayma
 - Gelecek zamana kayma (Ali yarın okula gidiyor (gidecek))
 - Rivayet geçmiş zamana kayma (Ali on yıl önce Ahmet'e ziyarete gidiyor (gitmiş))
 - Geniş zamana kayma (Ali her gün okula gidiyor (gider))
- Geniş zaman kipinde kayma
 - Rivayet geçmiş zamana kayma (Ertesi günü Ali okula gider (gitmiş))
 - Gelecek zamana kayma (Yarın haber gelir (gelecek))
 - Şimdiki zamana kayma (Ali peşimden gelir (geliyor))
 - Emir kipine kayma (Ali çantamı getirirsin (getir))
- Gelecek zaman kipinde kayma
 - Emir kipine kayma (Yarın kitabım gelecek (getirilsin))
 - Gereklik kipine kayma (İnsanda biraz utanma olacak (olmalı))

Yukarıdaki bu listede kiplerin anlam kaymaları gösterilmiştir.

SORU 70: Somut (müşahhas) isim (concrete noun)

Fiziksel olarak var olan nesnelere verilen isimdir. Örneğin elma, ağaç, araba veya insan gibi. Soyut (mücerret) isimlerin tersi niteliğindedir. Daha fazla bilgi için isim tanımına bakabilirsiniz.

SORU 71: Özel isim (hususî isim, proper noun)

Bir varlığa has, sadece o varlığı işaret eden isimlere verilen isimdir. Örneğin Şadi Evren ŞEKER bir insan ismidir ve sadece bir insanı ifade eder. Bir anlamda cins ismin tersi niteliğinde olarak bir çeşit veya tür değil de sadece bir bireyi temsil eden isimlerdir. Örneğin kedi cins isim olurken tekir hususi bir isimdir ve kedi cinsinden bir bireye işaret eder. Daha fazla bilgi için isim tanımına bakabilirsiniz.

SORU 72: Cins İsim (common noun)

Bir tür, çeşit yada cinsi belirtmek için kullanılan isimlerdir. Cins isimler, isimler alanında tanımlı küme isimleri gibi düşünülebilir ve birer sınıf belirtirler. Örneğin insan, kedi, ders veya üniversite birer cins isim örnekleridir. Cins ismin özel haline özel isim (proper noun) denilir. Daha fazla bilgi için isim tanımına bakabilirsiniz.

SORU 73: Soyut İsim (mücerret, abstract noun)

Maddî bir varlığa işaret etmeye isimlerdir. Mücerret isim olarak da nitelenen bu isimler somut isimlerin (müşahhas isimlerin) tersi olarak kabul edilir. Fiziksel olarak vâ' olmayan nesneleri ifade etmek için kullanılır. Örneğin rüya, felsefe veya oyun gibi kelimeler birer soyut isimdirler. Daha fazla bilgi için isim tanımına bakabilirsiniz.

SORU 74: isim (noun)

Bir nesneyi veya soyut bir kavramı niteleyen kelimedir. Doğal dil işleme dünyasında kısaca N harfi ile gösterilir.

Daha fazla bilgi için aşağıdaki kavramlara bakabilirsiniz:

- [Grup \(kütle\) ismi \(mass noun\)](#)

- [Sayılabilir isimler \(count noun\)](#)
- [Soyut \(mücerret\) isimler \(abstract noun\)](#)
- [Somut \(muşahhas\) isimler \(concret noun\)](#)
- [Cins isimler \(common noun\)](#)
- [Özel \(hususî\) isimler \(proper noun\)](#)

Ayrıca diğerkelime tipleri için:

- [Fiil \(verb\)](#)
- [Sıfat \(adjective\)](#)
- [Zarf \(adverb\)](#)
- [Bağlaç \(conjunction\)](#)

SORU 75: Sayılabilir İsimler (count noun)

Sayılması mümkün isimlerdir örneğinkalem, bardak, at veya ağaç gibi isimler sayılabilir. Daha fazla bilgi için isim tanımına ve sayılamaz isimlere bakabilirsiniz.

SORU 76: kütle adı (mass noun)

Grup ismi yada sürü ismi adı da verilmiştir. Sayılamaz varlıklara verilen isimlerdir. Örneğinkum, su, orman veya sürü gibi. İsimlerin sayılamaz veya sayı belirtmez olmaları itibariyle bu gruba dahil olmaları gerekir. İsimlerin nitelediği varlıkların sayılamaz olması gerekmez. Örneğinağaçlar sayılabilir ve dolayısıyla bir ormandaki ağaç sayısı bilinebilir ancak orman yine de grup ismidir çünkü sayılan orman değil ağaçlardır.