DOĞAL DİL İŞLEMEYE GİRİŞ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BURSA TEKNİK ÜNİVERSİTESİ

DR. HAYRI VOLKAN AGUN

Özet

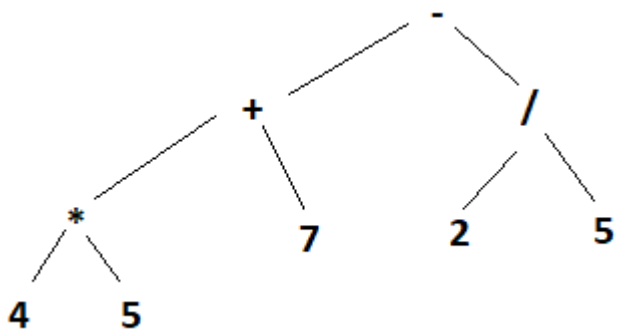
- Söz-dizimsel ayrıştırma

Özet

- Söz dizimsel ayıştırma nedir?
- Ayrıştırmada kullanılan yöntemler nelerdir?
- Gramer nedir? Kaç farklı gramer vardır?
- İstatistiksel ayrıştırmada kullanılan yöntemler nelerdir?

Ayrıştırma

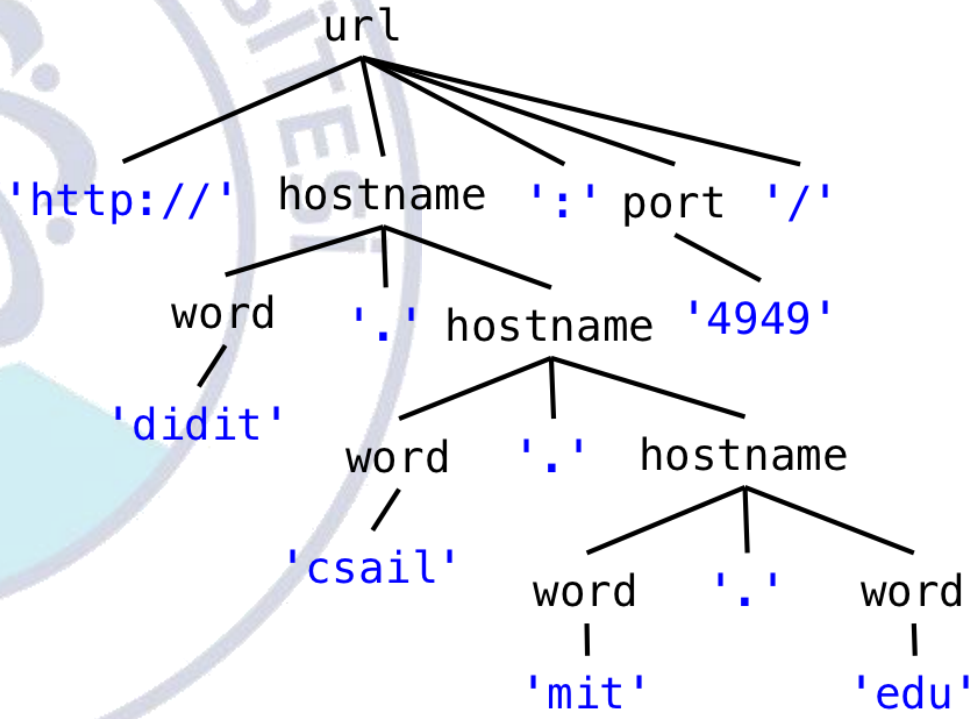
- Ayrıştırma farklı etiketlere sahip kelime/sembol gruplarının kural tabanlı olarak birleştirilmesi işlemidir.
- Ayrıştırma kod editörlerinde, XML/JSON kütüphanelerinde, HTML işleyen doküman nesne modeli (DOM) oluşturan Web tarayıcılarında, SQL dilini yorumlayan veri tabanlarında ve her hangi bir dil derleyicisinde kullanılan bir işleme yöntemidir.
- Ayrıştırma genel olarak ardışık ilişkili bir veriyi ağaç yapısına çevirir. Bu işlem ile birleşim oluşturan öbekler ağaç yapısına sahip forma dönüştürülür.
- Örneğin:
- 4 * 5 + 7 – 2 / 5 işlemi yandaki gibi ağaç yapısına dönüşür.



Ayrıştırma

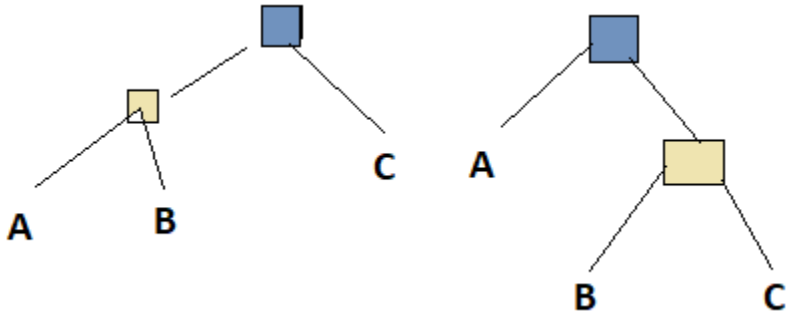
Düzenli ifadeler uzun ardışık formda geçen tekrarlı ifadeleri her zaman doğru bir şekilde yakalayamazlar. Bunun en önemli nedeni ayrıştırılmak istenen dilin karmaşıklık düzeyidir.
Dilin karmaşıklık düzeyi dili üreten gramer ile anlaşılabilir. Örneğin; XML veya HTML metinlerini ayrıştırmak için kullanılan dil düzenli dildir. Bu düzenli ifadeler ile ayrıştırılabilen bir dile karşılık gelir.

Nesneye Yönelik Programlamada kullanılan Java ve .NET dilleri bağlam bağımsız dillerdir. Bağlam bağımsız dillerde temel ayırt edici nokta bağlamdan bağımsız olan iç içe tekrarlı yapıların bulunmasıdır. Örneğin bir sıfat tamlamasında; NP -> ADJ NP tamlama iç içe tekrarlıdır.
Sıfat tamlaması: «İri buz kristallerinin süslediği görkemli şatoda»
Sıfat tamlaması: NP -> [İri] ADJ NP



Ayrıştırma

- Bir dilde tüm ikili kelime/sembol grupları düşünüldüğünde oluşabilecek toplam birleşim sayısı ve toplam farklı ağaç yapısı en fazla kaç olabilir?
- Örneğin:
- A B C sembollerinden oluşan her bir ikili kelimeye karşılık gelen bir etiket olsun.



- Bu durumda A ve B'nin birleştiği ve B ve C nin birleştiği durumlardan oluşan iki farklı ağaç şeması oluşturulabilir.

- Toplam kaç adet ağaç şeması oluşur?

- $\binom{N}{2} - 1$ adet farklı ağaç oluşabilir.

Ayrıştırma

Type-0	**Yinelemeli sıralı**	**Turing makinesi**	$\gamma \to \alpha$ (no constraints)	$L = \{w \mid w\ \}$
Type-1	**Bağlam duyarlı**		$\alpha A \beta \to \alpha \gamma \beta$	$L = \{a^n b^n c^n \mid n > 0\}$
Type-2	**Bağımsız**		$A \to \alpha$	$L = \{a^n b^n \mid n > 0\}$
Type-3	**Düzenli**	**Sonlu durum otomatı**	$A \to \mathrm{a}$ and $A \to \mathrm{a}B$	$L = \{a^n \mid n \geq 0\}$

Terminal olmayan sembol

Terminal olan sembol

Ayrıştırma

- Ayrıştırma algoritmaları ağaç yapısının oluşma şekline göre aşağıdan yukarı, ve birleşme sırasına göre soldan sağa gibi farklı kategorilerde ifade edilir.
- LR Ayrıştırıcısı (LeftRight Parser) aşağıdan yukarı ve soldan sağa yönlü birleştirme yapan bir ayrıştırıcıdır.

Ayrıştırma

- Ayrıştırmada kullanılan ayrıştırma kurallarına gramer denir. Her bir dilin bir adet grameri vardır.
- Diller için bu gramere söz dizim (syntax) denmektedir.
- Gramer öz yinelemeli ise kurallar kendi içinde başka kuralları barındıran sonlu durum olmayan sembollerle ifade edilir.

- Örneğin aşağıdaki gramer (recursive) öz yinelemeli değildir.

A -> a a b
A -> a a
B -> a b

- Yukarıda 3 kuraldan oluşan gramer ile aşağıdaki ifade ayrıştırılsaydı, ağaç (birleşim) yapısı nasıl olurdur?

Gramer Kuralları

Gramerler bir sonlu olmayan sembolün açılımı şeklinde yazılır. Örneğin:
A -> a A b
A -> c
şeklinde yazılan bir gramer bir karakter dizisi türetseydi. Aşağıdaki şekilde olurdu
A -> a **a A b** b

A -> a **a** a c b **b** b

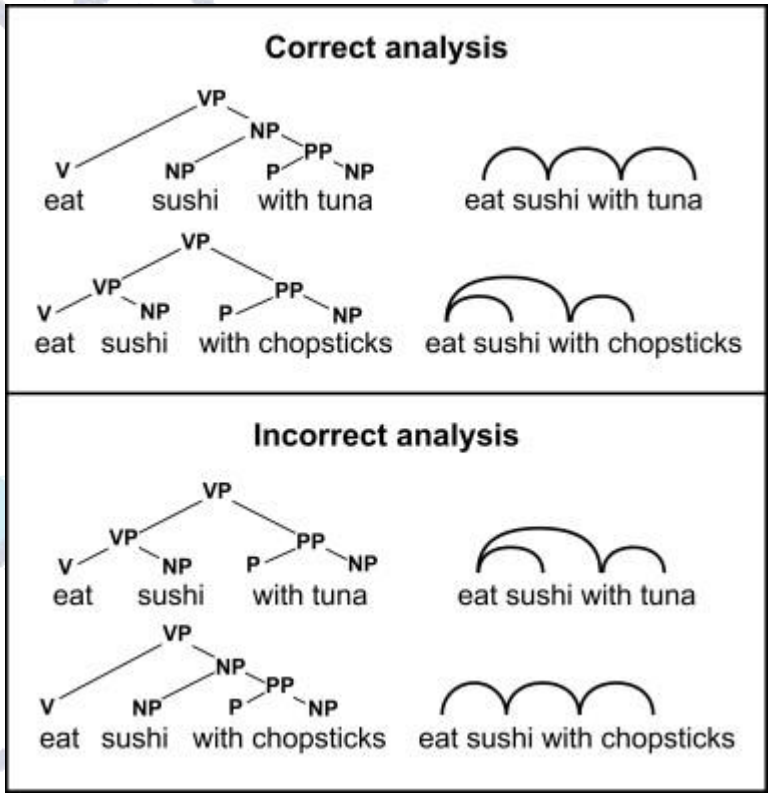
Gramer

Bazen gramer sonucu oluşan ağaç yapısı sayısı çok olabilir veya bu ağaç yapılarından küçük bir kısmı doğru olabilir. Bu durumda gramerin belirsizliği yüksektir denir.
Yandaki örnekte doğru ağaç sonucu ve yanlış ağaç sonucu alt alta verilmiştir.
Gramer her zaman doğru sonuç elde edecek düzgün bir kural yapısına sahip olmayabilir.
Bu tür gramerlere belirsizliği yüksek gramer denir. Örneğin aşağıdaki gramer hem soldan hem de sağdan öz yinelemelidir. Bu belirsizliği yüksek bir gramerdir.

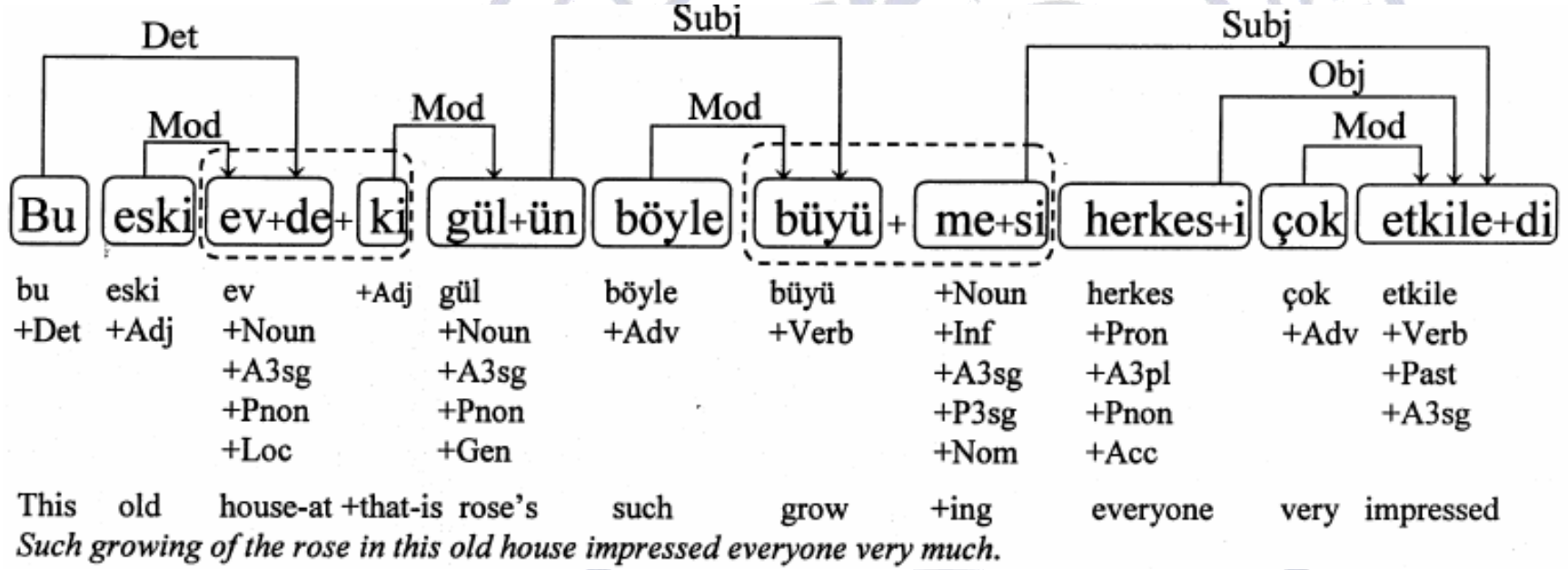
A -> a A
A -> A b
A -> a
A -> b

Aşağıdaki karakter dizileri bu gramer ile farklı ağaç yapıları oluşturabilir.

abab
baaa
bbbb



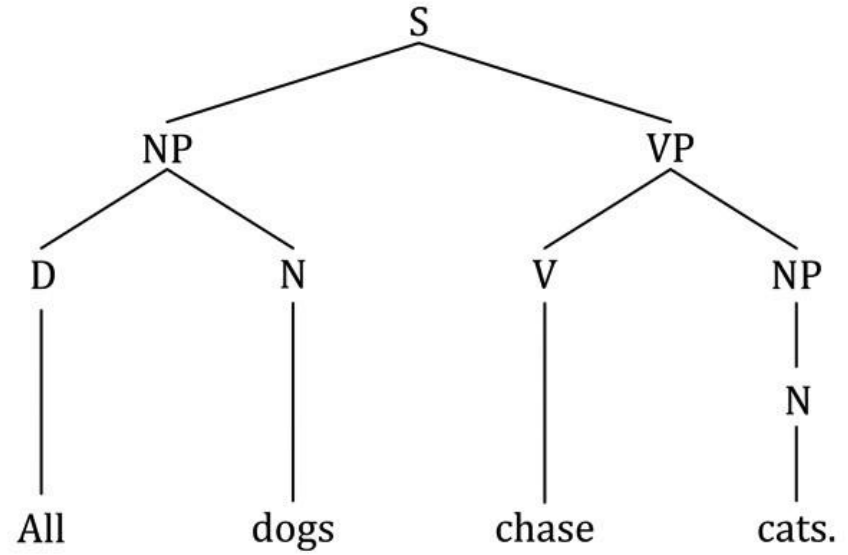
Bağımlı Gramer / Ayrıştırma



Ayrıştırma

- Ayrıştırma işlemi 2 ana bileşene sahiptir. Bunlar;
- Ayrıştırma metodu: Ayrıştırma algoritmasını barındıran yöntem yada yaklaşım.
- Gramer: Ayrıştırmadan kullanılan kurallar bütünü.
- Ayrıştırma metodu gramer kurallarını kullanarak bir ağaç yapısı oluşturur.
- Eğer gramer bağımlı (dependency) gramer ise bağımlı yapı oluşur ve eğer gramer bağımlı yapı değilse ağaç yapısı (tree) oluşur.

- Yandaki şekilde ayrıştırma işlemi sonucu oluşan ağaç yapısı gösterilmektedir. Bu ağaç yapısını oluşturmada kullanılan gramer kuralları neler olabilir?



Ayrıştırma

Arama şekli : Aşağıdan yukarı (öz yinelemeli)
Arama şekli: Yukarıdan aşağıya (öz yinelemeli)
Tablo kullanıp kullanmamasına göre (chart)

Shift Reduce Ayrıştırma

Gramer kuralları

Assign ← *id* = Sums
Sums ← Sums + Products
Sums ← Products
Products ← Products * Value
Products ← Value
Value ← *int*
Value ← *id*

A = B + C * 2

- Shift reduce ayrıştırma işleminde ayrıştır ikili bir karar mekanizması şeklinde yapılır.
- Ayrıştırıcı ya girdi metini içerisinde bir sembol sağa kayar yada var olan yığın içindekileri birleştirerek yığına ekler.

Adım	Ayrıştırma Yığını	Bir sonraki adım	İşlenmemiş	Ayrıştırma Olayı
0	empty	id	= B + C*2	Shift
1	id	=	B + C*2	Shift
2	id =	id	+ C*2	Shift
3	id = id	+	C*2	Reduce by Value ← id
4	id = Value	+	C*2	Reduce by Products ← Value
5	id = Products	+	C*2	Reduce by Sums ← Products
6	id = Sums	+	C*2	Shift
7	id = Sums +	id	*2	Shift
8	id = Sums + id	*	2	Reduce by Value ← id
9	id = Sums + Value	*	2	Reduce by Products ← Value
10	id = Sums + Products	*	2	Shift
11	id = Sums + Products *	int	eof	Shift
12	id = Sums + Products * int	eof		Reduce by Value ← int
13	id = Sums + Products * Value	eof		Reduce by Products ← Products * Value
14	id = Sums + Products	eof		Reduce by Sums ← Sums + Products
15	id = Sums	eof		Reduce by Assign ← id = Sums
16	Assign	eof		Done

Shift Reduce Ayrıştırma

Gramer kuralları

- S –> S + S
- S –> S * S
- S –> id

İfade: id + id + id

Stack	Input Buffer	Parsing Action
$	id+id+id$	Shift
$id	+id+id$	Reduce S->id
$S	+id+id$	Shift
$S+	id+id$	Shift
$S+id	+id$	Reduce S->id
$S+S	+id$	Reduce S->S+S
$S	+id$	Shift
$S+	id$	Shift
$S+id	$	Reduce S->id
$S+S	$	Reduce S->S+S
$S	$	Accept

- Shift reduce ayrıştırma işleminde ayrıştır ikili bir karar mekanizması şeklinde yapılır.
- Ayrıştırıcı ya girdi metini içerisinde bir sembol sağa kayar yada var olan yığın içindekileri birleştirerek yığına ekler.

Shift Reduce Ayrıştırma

Gramer kuralları

- S –> (L) | a
- L –> L , S | S

İfade: (a, (a, a))

- Shift reduce ayrıştırma işleminde ayrıştır ikili bir karar mekanizması şeklinde yapılır.
- Ayrıştırıcı ya girdi metini içerisinde bir sembol sağa kayar yada var olan yığın içindekileri birleştirerek yığına ekler.

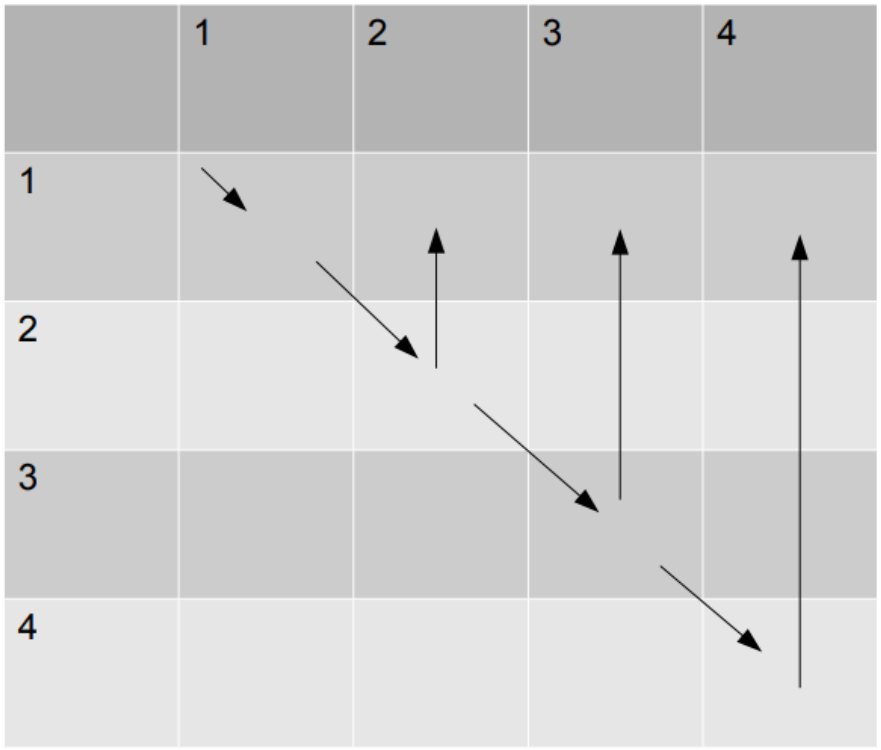
Stack	Input Buffer	Parsing Action
$	(a , (a , a)) $	Shift
$ (	a , (a , a)) $	Shift
$ (a	, (a , a)) $	Reduce S → a
$ (S	, (a , a)) $	Reduce L → S
$ (L	, (a , a)) $	Shift
$ (L ,	(a , a)) $	Shift
$ (L , (	a , a)) $	Shift

Chart Parsing

Gramer içerisinde her zaman sembole karşılık bir kural çalışmaz. Çoğu zaman gramer belirsizlik gösterir. Bu ayrıştırma sonucu oluşan ağaç yapısının birden çok olacağı anlamına gelir.

Bu belirsizliğin ayrıştırma işlemi içerisinde kısmen giderilmesi için tablo tabanlı bir veri yapısı kullanılır. Bu yapıda oluşan tüm kombinasyonlar test edilerek birleşimler oluşturulur.

CYK Algoritması
(John Cocke, Daniel H. Younger, and Tadao Kasami)



CYK Algoritması

0 Mary 1 feeds 2 the 3 otter 4

	1	2	3	4
1	N NP			
2		V		
3			DET	
4				N NP

S → NP VP

NP → N

NP → DET N

VP → V NP

N → Mary | otter

V → feeds

DET → the

0 Mary 1 feeds 2 the 3 otter 4

	1	2	3	4
1	N NP			S
2		V		VP
3			DET	NP
4				N NP

S → NP VP

NP → N

NP → DET N

VP → V NP

N → Mary | otter

V → feeds

DET → the

Referanslar

https://en.wikipedia.org/wiki/Shift-reduce_parser
https://en.wikipedia.org/wiki/CYK_algorithm
https://www.coli.uni-saarland.de/~yzhang/rapt-ws1112/slides/schmidt.pdf