بسم‌الله‌الرحمن‌الرحیم

تجزیهٔ نحوی با استفاده از دستور وابستگی

پژوهش و گردآوری:

محمد صادق رسولی

نظارت و بازبینی:

امید کاشفی

پژوهشکدهٔ مرکز تحقیقات کامپیوتری علوم اسلامی، واحد تهران

فهرست مطالب

عنوان | شماره صفحه

۱. مقدمه

تجزیهٔ وابستگی[۱] رهیافتی برای تجزیهٔ نحوی زبان طبیعی به صورت خودکار است. این رهیافت از زبان‌شناسی سنتی مبتنی بر دستور وابستگی[۲] اقتباس شده است. در سال‌های اخیر این روش بیش از پیش مورد توجه قرار گرفته است. چند دلیل عمده برای این اقبال عمومی وجود دارد. نخست این که این گونه از نمایش ساختار نحوی زبان طبیعی، کاربردهای بسیاری در برنامه‌های مربوط به فهم زبان طبیعی از جمله ترجمهٔ خودکار[۳] و استخراج اطلاعات[۴] دارد. دومین دلیل این است که این نوع از دستور زبان (و تجزیهٔ بر مبنای آن)، در مقایسه با دستور زبان مبتنی بر عبارات[۵]، سازگاری بیش‌تری با طبیعت زبان‌های بی‌ترتیب[۶] دارد. امّا مهم‌ترین دلیل، نتایج رضایت‌بخش حاصل از اعمال این روش در برخی از زبان‌ها با استفاده از روش‌های یادگیری خودکار[۷] بوده است [2].

۱-۱. دستور وابستگی

نظریهٔ دستور وابستگی یکی از نظریه‌های ساخت‌گرا[۸] و صورت‌گراست[۹] که اساساً در آن از طریق بررسی روابط وابستگی بین عناصر هسته و وابسته در زبان، به توصیف ساخت‌های نحوی در زبان‌های گوناگون پرداخته می‌شود [۳]. شاید آغاز رویکرد زبان وابستگی مربوط به اندیشه‌های زبان‌شناسی پانینی[۱۰] [4] در مورد زبان سانسکریت باشد؛ امّا کار تنی‌یر[۱۱] آغازی بر استفاده از این رویکرد در زبان‌شناسی نوین است. او نخستین بار در کتاب کم‌حجمی با عنوان گفتارهایی در نحو ساختاری [5] این دیدگاه را مطرح کرد که

---

[۱] معادل فارسی عبارت انگلیسی *Dependency Parsing*

[۲] معادل فارسی عبارت انگلیسی *Dependency Grammar*

[۳] معادل فارسی عبارت انگلیسی *Machine Translation*

[۴] معادل فارسی عبارت انگلیسی *Information Extraction*

[۵] معادل فارسی عبارت انگلیسی *Phrase-Based*

[۶] معادل فارسی عبارت انگلیسی *Free Order*: در چنین زبانهایی قابلیت جابه‌جایی اجزای جمله وجود دارد؛ مثال: «من در مدرسه کتاب را به علی دادم»؛ «من در مدرسه به علی کتاب را دادم»؛ «من به علی در مدرسه کتاب را دادم»؛ و «من کتاب را به علی در مدرسه دادم». علاوه بر این در این زبان‌ها امکان حذف اسم‌ها و ضمایر به قرینهٔ حضوری وجود دارد و یک واحد معنایی یا نحوی واحد (مانند گروه اسمی) قابلیت تکه‌تکه شدن و پخش در سطح جمله دارد *[1]*.

[۷] معادل فارسی عبارت انگلیسی *Machine Learning*

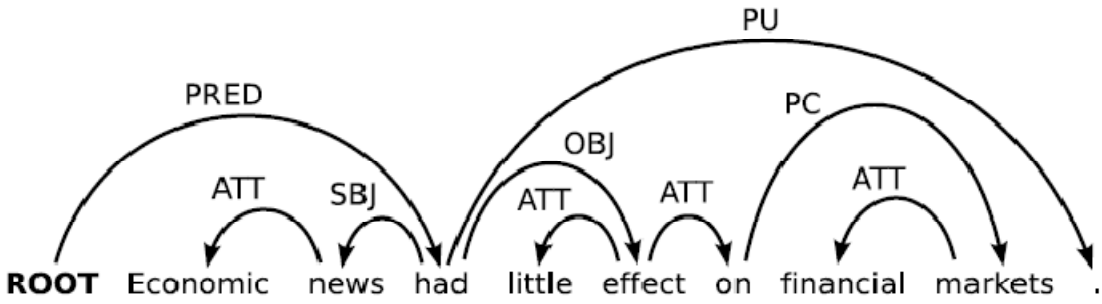
[۸] معادل فارسی واژهٔ انگلیسی *Structuralist*

[۹] معادل فارسی واژهٔ انگلیسی *Formalist*

[۱۰] نوشتار فارسی نام *Panini*

[۱۱] نوشتار فارسی نام فرانسوی *Tesnière*

شرح مبسوط آن پس از مرگش در کتاب مبانی نحو ساختاری [6] منتشر شد. پس از تنی‌یر، زبان‌شناسان مختلف روش‌های مختلفی را برای ارائهٔ دستور زبان وابستگی پیشنهاد داده‌اند. در همهٔ این دستورها یک فرض پایه وجود دارد. در همهٔ انواع دستور زبان وابستگی فرض بر این است که ساختار نحوی شامل واژه‌هایی است که این واژه‌ها با روابط دودویی نامتقارن با هم در ارتباط هستند. به این روابط، ارتباط وابستگی یا وابستگی گفته می‌شود [2]. دو فرض اساسی در نظریهٔ دستور وابستگی وجود دارد. نخست این که هر جمله یک فعل مرکزی دارد و دوم این که بر اساس نوع و تعداد متمم‌های اجباری و اختیاری، می‌توان ساخت بنیادین جمله‌هایی را که فعل در آن‌ها به کار رفته است، تعیین کرد [۳]. در همهٔ این رابطه‌ها یک واژه وابسته[۱] و واژهٔ دیگر سر[۲] است[۳]. در شکل ۱ نمونه‌ای از یک درخت وابستگی نشان داده شده است.



شکل ۱ نمونه‌ای از یک درخت وابستگی [2]

شایان ذکر است که اطلاعات موجود در ساختار وابستگی با اطلاعات موجود در ساختار مبتنی بر عبارات متفاوت است. برای مقایسه می‌توان ساختار موجود در شکل ۲ را با ساختار شکل ۱ مورد بررسی قرار داد. در دستور وابستگی، جمله به دو بخش نهاد و گزاره تقسیم نمی‌شود. در این نظریه این که در تجزیهٔ جمله به دو گروه اسمی و فعلی تقسیم می‌شود، رد می‌شود. به اعتقاد انگل[۴] [7]، تجزیهٔ جمله به دو بخش نهاد و گزاره برای تحلیل ساخت اطلاعاتی جمله مفید است ولی در تحلیل نحوی مرکز ثقل

---

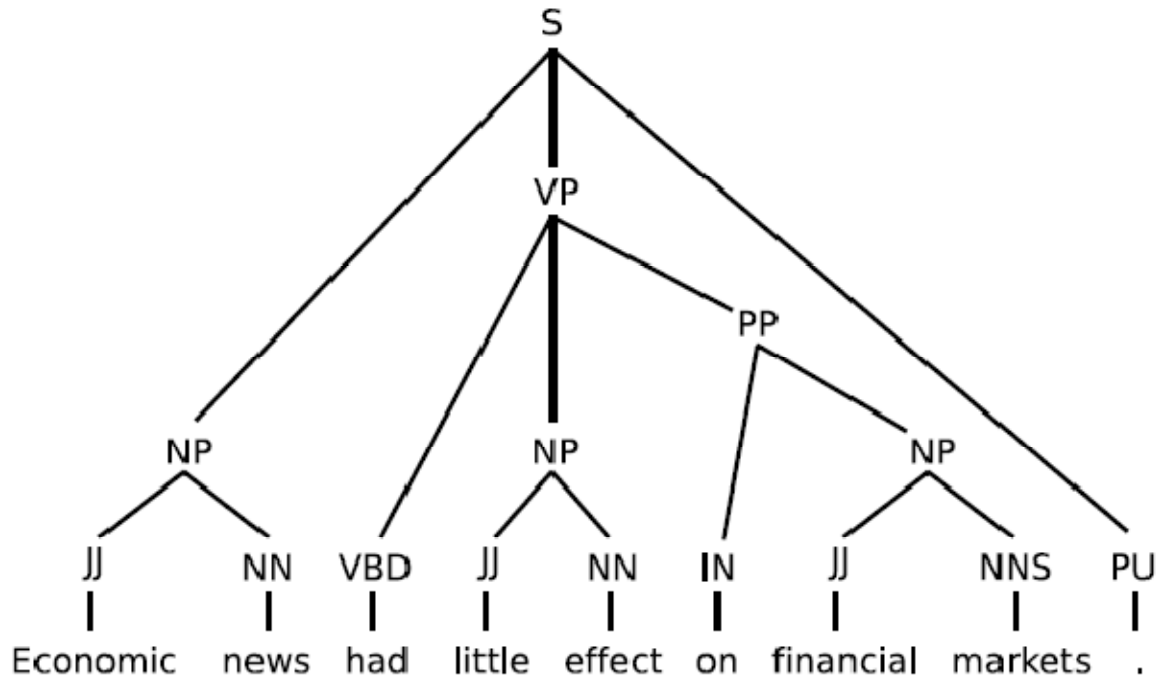
[۱] معادل فارسی واژهٔ انگلیسی *Dependent*

[۲] معادل فارسی واژهٔ انگلیسی *Head*

[۳] اصطلاحات دیگری نیز به جای این دو اصطلاح به کار می‌رود. اصطلاح بچه (*Child*) و پیرایند (*Modifier*) به جای وابسته؛ و اصطلاح حاکم (*Governor*)، رئیس (*Regent*) و والدین (*Parent*) به جای سر به کار می‌رود.

[۴] نوشتار فارسی نام *Engel*

ساختاری جمله فعل است [۳]. البته تبدیل اطلاعات موجود هر کدام از این دو ساختار به هم، امکان‌پذیر است ولی برای سهولت فرض بر این است که رویکرد مبتنی بر ساختار وابستگی و رویکرد مبتنی بر عبارات، دو رهیافت مختلف و متفاوت هستند [2].



شکل ۲ نمونه‌ای از یک ساختار نحوی مبتنی بر عبارات [2]

با این فرض که ساختار وابستگی نمایندهٔ خوبی برای نشان دادن ساختار نحوی زبان طبیعی است، نیاز به اعمال ضابطه‌هایی بر این ساختار وجود دارد. بنابراین با وجود ساختار زبانی C و واژهٔ وابستهٔ D و واژهٔ سَرِ H شش ضابطهٔ زیر را خواهیم داشت [2]:

۱) با استفاده از H مقولهٔ نحوی C قابل تعیین است و H می‌تواند به عنوان نمایندهٔ کل ساختار C، جای C را پر کند.

۲) با استفاده از H مقولهٔ معنایی C قابل تعیین است و D به این ساختار ویژگی‌های معنایی می‌افزاید.

۳) وجود H لازم است ولی وجود D ممکن است اختیاری باشد.

۴) H، D را انتخاب و اختیاری یا اجباری بودن آن را تعیین می‌کند.

۵) شکل D وابسته به H است.

۶) موقعیت خطی D با ارجاع به H مشخص می‌شود.

همان‌طور که از ضابطه‌ها پیداست، برخی از این ساختارها مربوط به ساخت‌واژه، برخی مربوط به نحو و برخی مربوط به معناست. البته برخی بر این اعتقادند که باید برای دو نوع ساختار درون‌مرکز[۱] و برون‌مرکز[۲] ضابطه‌های متفاوتی را در نظر داشت. در ساختار درون‌مرکز ممکن است واژهٔ سر نمایندهٔ مقولهٔ نحوی کل گروه باشد ولی در ساختار برون‌مرکز همهٔ واژه‌ها با هم مقولهٔ نحوی را می‌سازند. به عنوان مثال عبارت «میز کهنه» یک ساختار درون‌مرکز است ولی عبارت «روی آن میز» یک عبارت برون‌مرکز است [8]. به وضوح مشخص است که در ساختار درون‌مرکز هر شش ضابطهٔ ذکرشده صدق می‌کند؛ البته در این ساختار ضابطهٔ ۴ کمی نامناسب به نظر می‌رسد. در ساختار برون‌مرکز ضابطهٔ ۱ درست نخواهد بود [2].

این دو ساختار قابل مقایسه با دو ساختار پیرایندهٔ واژهٔ سر[۳] (شبیه به ساختار درون‌مرکز) و مکمل واژهٔ سر[۴] (شبیه به ساختار برون‌مرکز) است. تفاوت بین مکمل و پیراینده در مفهوم ظرفیت[۵] نهفته است. در تعریف سنتی دستور وابستگی، ظرفیت مفهومی اساسی دارد. دو زبان‌شناس آلمانی با نام‌های هلبیگ[۶] و شنکل[۷] [9] با رویکرد نظریهٔ زبان زایشی[۸] در مورد ظرفیت فعل در زبان آلمانی پژوهش‌هایی را صورت دادند. تا این که انگل [7] این مفهوم را مخصوص نظریهٔ دستور وابستگی مطرح کرد [۳]. بر اساس این مفهوم، هر فعل به ساختار جمله، حالات خاصی از وابسته‌ها را تحمیل می‌کند که بر این اساس به دو نوع مقید به ظرفیت[۹] و آزاد از ظرفیت[۱۰] تقسیم‌بندی می‌شوند [2].

در زمینهٔ قواعد موجود در ساختار وابستگی بین زبان‌شناسان مختلف اختلاف نظرهایی وجود دارد. یکی از این اختلاف نظرها در مورد هم‌پایگی‌ها[۱۱] است. از دیدگاه ساختارگراهای سنتی، ساختار هم‌پایگی یک ساختار درون‌مرکز است. دلیل این ادعا، امکان جایگزینی یک یا حتی چند واژه به جای کل عبارت نحوی

---

[۱] معادل فارسی واژهٔ انگلیسی *Endocentric*

[۲] معادل فارسی واژهٔ انگلیسی *Exocentric*

[۳] معادل فارسی عبارت انگلیسی *Head-Modifier*

[۴] معادل فارسی عبارت انگلیسی *Head-Complement*

[۵] معادل فارسی واژهٔ انگلیسی *Valency*

[۶] نوشتار فارسی نام آلمانی *Helbig*

[۷] نوشتار فارسی نام آلمانی *Schenkel*

[۸] معادل فارسی واژهٔ انگلیسی *Generative*

[۹] معادل فارسی عبارت انگلیسی *Valency-Bound*

[۱۰] معادل فارسی عبارت انگلیسی *Valency-Free*

[۱۱] معادل فارسی واژهٔ انگلیسی *Coordination*

مورد نظر است. مثلاً در جملهٔ «او سیبی را خرید و خورد»، واژهٔ «سیبی» مفعول ساختار هم‌پایگی فعلی «خرید و خورد» است.

۱-۱-۱. ظرفیت

مهم‌ترین مبحث در دستور وابستگی، عبارت است از مسألهٔ ظرفیت نحوی که در آن به بحث در مورد وابسته‌های فعل، اسم و صفت پرداخته می‌شود. بر اساس این نظریه، مرکز ثقل ساختاری جمله فعل است [۳]. تنی‌یر [10] مفهوم ظرفیت را از شیمی اقتباس کرده بود. ظرفیت در شیمی عبارت است از توانایی یک عنصر در ترکیب با تعداد خاصی از اتم‌های عناصر دیگر. این که ساخت بنیادین جمله حول فعل مرکزی آن صورت می‌گیرد، مبین این واقعیت است که هر فعل پیش از آن که وارد جمله بشود، خود مشخص‌کنندهٔ نوع ساخت بنیادین است [۳].

۱-۱-۲. ساخت بنیادین جمله

ساخت‌های بنیادین جمله به ساخت‌هایی اطلاق می‌شود که از بسط و تعریف آن‌ها یا از ترکیب آن‌ها با هم یا از تبدیل آن‌ها به هم یا به ساخت‌های مشتق یا فرعی دیگر، یا از آمیزه‌ای از دو تا یا چند تا از روش‌های گفته‌شده، بتوان تمام جمله‌های محتمل موجود در زبان را تولید کرد. استخراج چنین ساختارهایی از زمرهٔ مهم‌ترین و ابتدایی‌ترین وظایف تحلیل نحوی است؛ زیرا این ساخت‌های محدود و تکرارشونده تمام ساخت‌های نحوی هر زبان را تشکیل می‌دهند [11]. ظرفیت فعل مفهومی انتزاعی و متعلق به واژه‌هاست ولی ساخت بنیادین مفهومی متعلق به جمله است..

۱-۲. تجزیهٔ وابستگی

به تجزیه و تحلیل خودکار ساختار وابستگی جملات، تجزیهٔ وابستگی گویند. در مجموع می‌توان گفت که در تجزیهٔ وابستگی برای هر جملهٔ ورودی یک گراف وابستگی ساخته می‌شود و دو رهیافت عمومی برای آن وجود دارد: ۱) مبتنی بر داده[۱]؛ و ۲) مبتنی بر دستور زبان[۲]. در رهیافت مبتنی بر داده، از روش‌های یادگیری خودکار و در روش‌های مبتنی بر دستور زبان از دستور زبان‌های صوری[۳] استفاده می‌شود. البته بدیهی است که می‌توان از هر دو رهیافت به صورت تلفیقی برای تجزیهٔ وابستگی استفاده کرد [2].

---

[۱] معادل فارسی عبارت انگلیسی *Data-Driven*

[۲] معادل فارسی عبارت انگلیسی *Grammar-Based*

[۳] معادل فارسی واژهٔ انگلیسی *Formal*

در روش یادگیری باناظر[۱] دو مرحلهٔ اصلی در ساخت یک سامانهٔ تجزیهٔ وابستگی وجود دارد. در مرحلهٔ نخست با استفاده از یک پیکرهٔ آموزشی، دستور زبان وابستگی به دست می‌آید. به عمل به دست آوردن دستور زبان، استنتاج دستور زبان[۲] گویند. با به دست آمدن دستور زبان وابستگی، الگوی تجزیه[۳] به دست خواهد آمد. در مرحلهٔ بعدی بر اساس الگوی به دست آمده در مرحلهٔ قبل، برای هر جملهٔ ورودی، گراف وابستگی تولید خواهد شد. به مرحلهٔ اوّل یادگیری[۴] و به مرحلهٔ دوم تجزیه گفته می‌شود [2].

انواع مختلفی از روش‌های مبتنی بر داده وجود دارد که از مهم‌ترین آن‌ها می‌توان به روش مبتنی بر گذار[۵] و روش مبتنی بر گراف اشاره کرد. در همهٔ روش‌های مبتنی بر داده فرض اولیه بر این است که داده‌های ورودی حتماً دارای ساختار نحوی درست هستند ولی در روش‌های مبتنی بر دستور زبان فرض بر این است که اگر ساختاری در قالب هیچ یک از قواعد موجود در پایگاه قوانین نگنجد، آن جمله از نظر دستوری نادرست است. روش‌های مبتنی بر دستور زبان به دو نوع اصلی مستقل از متن[۶] و مبتنی بر محدودیت[۷] تقسیم می‌شوند. در روش مستقل از متن، ساختار وابستگی تبدیل به عبارات مستقل از متن می‌شود و بر اساس دستور زبان مستقل از متن تجزیه صورت می‌گیرد و در روش مبتنی بر محدودیت، صورت مسأله تبدیل به مسألهٔ ارضای محدودیت[۸] می‌شود [2].

۲. تجزیهٔ وابستگی

در این بخش پس از ارائهٔ تعریف‌های اولیه در اصطلاحات موجود در نظریهٔ وابستگی به دسته‌بندی روش‌های مختلف تجزیهٔ وابستگی پرداخته می‌شود.

---

[۱] معادل فارسی واژهٔ انگلیسی *Supervised*

[۲] معادل فارسی عبارت انگلیسی *Grammar Induction*

[۳] معادل فارسی عبارت انگلیسی *Parsing Model*

[۴] معادل فارسی واژهٔ انگلیسی *Learning*

[۵] معادل فارسی عبارت انگلیسی *Transition-Based*

[۶] معادل فارسی عبارت انگلیسی *Context-Free*

[۷] معادل فارسی عبارت انگلیسی *Constraint-Based*

[۸] معادل فارسی عبارت انگلیسی *Constraint Satisfaction*

۲-۱. درخت‌ها و گراف‌های وابستگی

همان‌طور که در بخش ۱-۲ اشاره شد، یک گراف وابستگی، نمایشی از ساختار نحوی ایجاد شده از جمله به صورت گراف است.

تعریف (۲-۱) جمله دنباله‌ای از نمادهای[۱] پشت سر هم است [2]:

$$S = w_0 w_1 \dots w_n$$

در این روند فرض اولیه بر این است که فرایند نمادیابی[۲] به صورت درست و کامل انجام شده است و هیچ دغدغه‌ای در این مورد وجود نخواهد داشت. در تعریف (۲-۱) منظور از w_0 واژهٔ ساختگی ریشه است که نشان‌دهندهٔ هیچ واژهٔ واقعی‌ای در این زبان نیست. تعریف نماد وابسته به زبان است و ممکن است شامل علائم نگارشی نیز باشد.

تعریف (۲-۲) اگر $R = \{r_1, r_2, \dots, r_m\}$ مجموعهٔ محدودی از گونه‌های ممکن از ارتباط‌های وابستگی باشد و قابلیت وجود این ارتباط بین دو واژه در جمله وجود داشته باشد، گونهٔ ارتباط $r \in R$ را برچسب *یال*[۳] گویند [2].

بر اساس تعریف (۲-۱) و تعریف (۲-۲) می‌توان گراف وابستگی را تعریف کرد.

تعریف (۲-۳) یک گراف وابستگی $G = (V, A)$ یک گراف جهت‌دار با برچسب بر اساس معیارهای نظریهٔ گراف است که شامل تعدادی گرهِ V و یال‌های A است؛ به طوری که برای جملهٔ $s = w_0 w_1 \dots w_n$ و مجموعه برچسب R قواعد ذیل صدق می‌کند [2]:

۱- $V \subseteq w_0 w_1 \dots w_n$

---

[۱] در تعریفی که در مرجع قید شده است عبارت انگلیسی *Token* نوشته شده است که آن را به معنای فارسی «نماد» برگردانده‌ایم. این برگردان بر اساس فرهنگ باطنی [۱۲] صورت گرفته است.

[۲] معادل فارسی واژهٔ انگلیسی *Tokenization*

[۳] معادل فارسی واژهٔ انگلیسی *Arc*

$A \subseteq V \times R \times V$ -۲

۳- اگر $(w_i, r, w_j) \in A$ باشد، آن‌گاه برای همهٔ $(w_i, \acute{r}, w_j) \notin A$، خواهیم داشت: $r \neq \acute{r}$.

در تعریف (۲-۳) منظور از (w_i, r, w_j) رابطهٔ r بین w_i به عنوان واژهٔ سر و w_j به عنوان واژهٔ وابسته است. یک مجموعه گره معیار، مجموعه‌ای از گره‌های پوشا است که همهٔ واژه‌های یک جمله در آن وجود داشته باشد. محدودیتی که بند ۳ از تعریف (۲-۳) ایجاد می‌شود، باعث خواهد شد که بین دو واژه بیش از یک وابستگی وجود نداشته باشد. این محدودیت مرتبط با نظریهٔ وابستگی نحوی تک‌قشری[۱] است [2]. برای مثال می‌توان شکل ۱ را به صورت عبارات شکل ۳ خلاصه کرد:

1. *G= (V, A)*

2. *V= V_s= {root, Economic, news, had, little, effect, on, financial, markets, .}*

3. A= *{(root, PRED, had), (had, SBJ, news), (had, OBJ, effect),(had, PU, .), (news, ATT, Economic), (effect, ATT, little), (effect, ATT, on), (on, PC, markets), (markets, ATT, financial)}*

شکل ۳ خلاصهٔ درخت وابستگی در شکل ۱ [2]

تعریف (۲-۴) یک *گراف وابستگی خوش‌ساخت*[۲] $G = (V, A)$ برای جملهٔ S و مجموعه رابطهٔ وابستگی R، درختی است که ریشه‌اش w_0 و دارای مجموعهٔ گره‌های پوشای $V = V_S$ باشد. به چنین گرافی، *درخت وابستگی* می‌گویند [2].

نشانه‌گذاری (۲-۱) برای جملهٔ ورودی S و مجموعه رابطهٔ وابستگی R، فضای حالت همهٔ گراف‌های وابستگی خوش‌ساخت با $\mathcal{G}_S$ نشان داده می‌شود [2].

۲-۲. ویژگی‌های درخت وابستگی

در این بخش به بررسی برخی از ویژگی‌های اصلی درخت‌های وابستگی پرداخته خواهد شد.

---

[۱] معادل فارسی واژهٔ انگلیسی *Monostratal*

[۲] معادل فارسی عبارت انگلیسی *Well-formed*

نشانه‌گذاری (۲-۲) نماد $w_i \rightarrow w_j$ به معنای رابطهٔ وابستگی بدون برچسب در درخت $G = (V, A)$ است. بنابراین $w_i \rightarrow w_j$ است، اگر و تنها اگر $(w_i, r, w_j) \in A$ و $r \in R$ باشد [2].

نشانه‌گذاری (۲-۳) نماد $w_i \rightarrow^* w_j$ به معنای بستار متعدی بازتابی[۱] رابطهٔ وابستگی در درخت $G = (V, A)$ است. بنابراین $w_i \rightarrow^* w_j$ اگر و تنها اگر $i = j$ (بازتابی) یا هم $w_i \rightarrow^* w_i$ و $w_i \rightarrow^* w_j$ (برای $w_i \in V$) برقرار باشد [2].

نشانه‌گذاری (۲-۴) نماد $w_i \leftrightarrow w_j$ به معنای رابطهٔ وابستگی بدون جهت در درخت $G = (V, A)$ است. بنابراین $w_i \leftrightarrow w_j$ است، اگر و تنها اگر یکی از دو حالت $w_i \rightarrow w_j$ یا $w_j \rightarrow w_i$ برقرار باشد [2].

نشانه‌گذاری (۲-۱) نماد $w_i \leftrightarrow^* w_j$ به معنای بستار متعدی بازتابی[۲] رابطهٔ وابستگی بدون جهت در درخت $G = (V, A)$ است. بنابراین $w_i \leftrightarrow^* w_j$ است، اگر و تنها اگر $i = j$ (بازتابی) یا هم $w_i \leftrightarrow^* w_i$ و $w_i \leftrightarrow^* w_j$ (برای $w_i \in V$) برقرار باشد [2].

حال با توجه به نشانه‌گذاری‌های ذکرشده، برخی از ویژگی‌های درخت‌های وابستگی مورد بررسی قرار خواهند گرفت.

ویژگی (۲-۱) درخت وابستگی $G = (V, A)$ همیشه *ویژگی ریشه* را ارضا می‌کند. طبق ویژگی ریشه، هیچ $w_i \in V$ وجود ندارد که $w_i \rightarrow w_0$ برقرار باشد [2].

---

[۱] معادل فارسی عبارت انگلیسی *Reflexive transitive closure*

[۲] معادل فارسی عبارت انگلیسی *Reflexive transitive closure*

لازم به ذکر است که ویژگی (۲-۱) یک ویژگی مصنوعی در نظریهٔ وابستگی است. زیرا در واقع واژهٔ ریشه وجود ندارد و یک نماد ساختگی برای ریشهٔ درخت وابستگی است.

ویژگی (۲-۲) درخت وابستگی $G = (V, A)$ همیشه *ویژگی پوشایی درخت* را ارضا می‌کند. طبق این ویژگی، $V = V_s$ خواهد بود [2].

ویژگی (۲-۲) را عمدهٔ نظریه‌پردازان ساختار وابستگی تأیید می‌کنند. البته ممکن است در مواردی مانند علائم نگارشی این ویژگی صدق نکند. این ویژگی در زبان برای واژهٔ ریشه، مصنوعی است [2].

ویژگی (۲-۳) درخت وابستگی $G = (V, A)$ همیشه *ویژگی هم‌بندی*[1] را ارضا می‌کند، یعنی برای همهٔ $w_i, w_j \in V$، شرط $w_i \leftrightarrow^* w_j$ برقرار است. با توجه به این ویژگی بین همهٔ واژه‌ها بدون در نظر داشتن جهت وابستگی‌ها، مسیری از طریق درخت وابستگی وجود دارد. در نظریهٔ گراف‌ها به گراف با این ویژگی، *گراف جهت‌دار هم‌بند ضعیف* گویند [2].

ویژگی (۲-۳) به طور عمومی پذیرفته نیست. ولی با در نظر گرفتن واژهٔ ساختگی ریشه، این ویژگی برقرار خواهد بود [2].

ویژگی (۲-۴) درخت وابستگی $G = (V, A)$ همیشه *ویژگی تک‌سَری*[2] را ارضا می‌کند. یعنی برای همهٔ $w_i, w_j \in V$، اگر $w_i \rightarrow w_j$ باشد، آن‌گاه $w_i \in V$ است به طوری که $i \neq j$ و $w_i \rightarrow w_j$ وجود نخواهد داشت. در نتیجه، در ساختار درخت وابستگی، هر واژه حداکثر یک سر دارد [2].

[1] معادل فارسی واژهٔ انگلیسی *Connectedness*

[2] معادل فارسی عبارت انگلیسی *Single-headed*

ویژگی (۲-۵) درخت وابستگی $G = (V, A)$ همیشه *ویژگی بدون دور بودن*[۱] را داراست. یعنی برای همهٔ $w_i, w_j \in V$، اگر $w_i \to w_j$، آن‌گاه حالت $w_j \to^* w_i$ برقرار نخواهد بود. در این صورت یک درخت وابستگی دور نخواهد داشت [2].

ویژگی (۲-۶) درخت وابستگی $G = (V, A)$ همیشه *ویژگی اندازه*[۲] را ارضا می‌کند. یعنی $|A| = |V| - 1$ [2].

در ویژگی (۲-۶) ویژگی‌های تک‌سری (ویژگی (۲-۴)) و بدون دور بودن (ویژگی (۲-۵)) مورد شمول قرار گرفته است [2].

۲-۲-۲. درخت‌های وابستگی افکنشی

یک ویژگی علاوه بر ویژگی‌های ذکر شده در بخش ۲-۲ وجود دارد که محدودیت بیش‌تری را در درخت‌های وابستگی ایجاد می‌نماید. به این ویژگی افکنشی بودن[۳] (انعکاسی بودن) گفته می‌شود.

تعریف (۲-۱) یال $(w_i, r, w_j) \in A$ در درخت وابستگی $G = (V, A)$، *افکنشی*[۴] است اگر و تنها اگر برای همهٔ $i < k < j$ وقتی که $i < j$ یا همهٔ $j < k < i$ وقتی که $j < i$، $w_i \to^* w_k$ [2].

با توجه به تعریف (۲-۱) یک یال در درخت وابستگی افکنشی است، اگر مسیری جهت‌دار از واژهٔ سر (w_i) به همهٔ واژه‌های بین نقاط ابتدای و انتهای دو یال وجود داشته باشد [2].

تعریف (۲-۲) درخت وابستگی $G = (V, A)$، یک *درخت وابستگی افکنشی* است اگر: ۱) این ساختار درخت وابستگی باشد (تعریف (۲-۴))، و ۲) همهٔ یال‌های $(w_i, r, w_j) \in A$ افکنشی باشند [2].

---

[۱] معادل فارسی عبارت انگلیسی *Acyclicity*

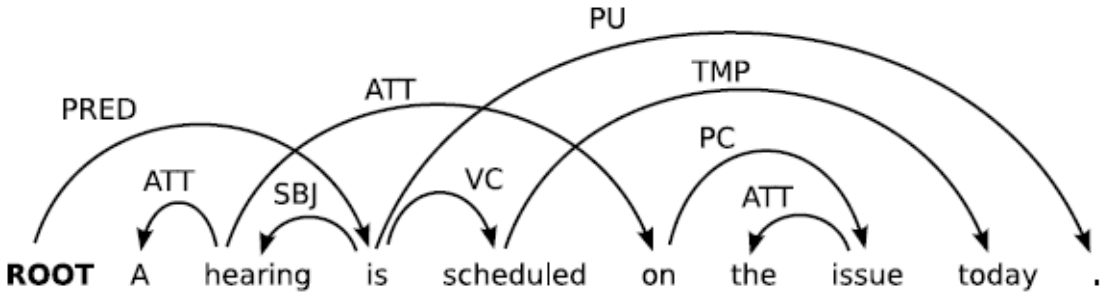
[۲] معادل فارسی عبارت انگلیسی *Size Property*

[۳] معادل فارسی عبارت انگلیسی *Projectiveness*

[۴] معادل فارسی عبارت انگلیسی *Projective*

تعریف (۲-۳) درخت وابستگی $G = (V, A)$، یک *درخت وابستگی غیرافکنشی*[۱] است اگر: ۱) این ساختار درخت وابستگی باشد (تعریف (۲-۴))، و ۲) درخت افکنشی نباشد [2].

درخت موجود در شکل ۱ نمونه‌ای از یک درخت وابستگی افکنشی است. در زبان انگلیسی تعداد جملاتی که درخت وابستگی آن‌ها غیرافکنشی است، بسیار کم است ولی در زبان‌هایی مانند چکی، آلمانی و ترکی فراوانی جملات با درخت وابستگی غیرافکنشی بیش‌تر از حدی است که قابل چشم‌پوشی باشد. عمدهٔ نظریه‌های زبان‌شناسی ساختارهای وابستگی، با این پیش‌فرض هستند که زبان طبیعی دارای ساختار درختی غیرافکنشی است [2]. در زبان‌های بی‌ترتیب تعداد حالت‌هایی که باعث ساخت درخت غیرافکنشی می‌شود، زیاد است[۲] [14, 15]. درشکل ۴ نمونه‌ای از یک درخت وابستگی غیرافکنشی به نمایش گذاشته شده است.



شکل ۴ نمونه‌ای از یک درخت وابستگی غیرافکنشی [2]

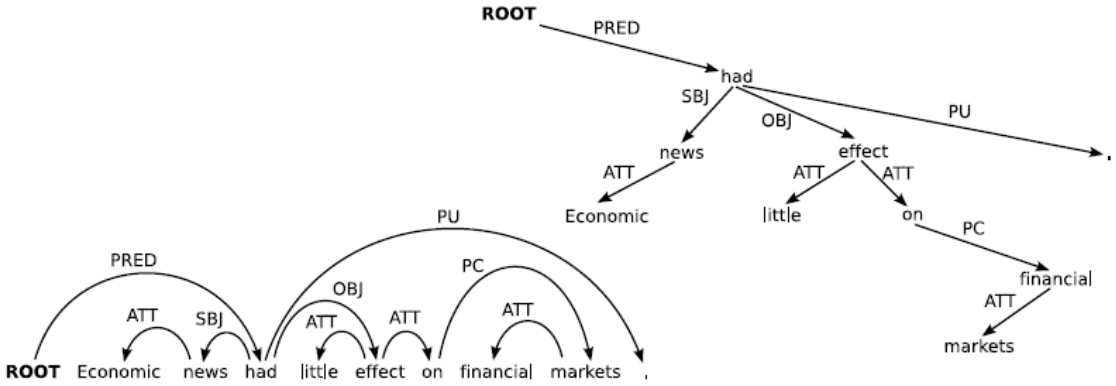
نشانه‌گذاری (۲-۲) برای جملهٔ ورودی S و مجموعه رابطهٔ وابستگی R، فضای همهٔ درخت‌های وابستگی افکنشی $\mathcal{G}_s^p$ با نشان داده می‌شود [2].

[۱] معادل فارسی عبارت انگلیسی *Non-projective*

[۲] با توجه به شمس‌فرد [۱۳]، زبان فارسی جزء زبان‌های بی‌ترتیب است.

ویژگی (۲-۷) یک درخت وابستگی افکنشی $G = (V, A)$ *مسطح*[۱] است. در این صورت کل گراف را می‌توان بر روی کاغذ طوری کشید به طوری که هیچ یالی دیگری را قطع نکند [2].

در شکل ۵، ویژگی (۲-۷) به نمایش گذاشته شده است. همان طور که از این شکل پیداست، می‌توان یک درخت وابستگی افکنشی را طوری روی کاغذ ترسیم کرد که هیچ یالی دیگری را قطع نکند.



شکل ۵ درخت وابستگی افکنشی (سمت چپ) و شکل مسطح آن (سمت راست) [2]

ویژگی (۲-۸) یک درخت وابستگی افکنشی $G = (V, A)$ دارای *تو در تو*[۲] است. بنابراین برای همهٔ گره‌های $w_i \in V$، مجموعه واژه‌های $\{w_j | w_i \rightarrow^* w_j\}$ یک زیردنبالهٔ هم‌جوار[۳] در جملهٔ S است [2].

معمولاً به $\{w_j | w_i \rightarrow^* w_j\}$ ثمرهٔ[۴] w_i در G می‌گویند. به دلیل ساده شدن پیچیدگی محاسباتی با وجود ویژگی (۲-۸)، حتی برای زبان‌هایی که ذاتاً دارای ساختار غیرافکنشی هستند، زبان را به صورت افکنشی تبدیل می‌کنند و با وجود از دست رفتن اطلاعات زبانی و پایین آمدن دقت، ساده شدن محاسبات را ترجیح می‌دهند [2].

---

[۱] معادل فارسی واژهٔ انگلیسی *Planar*

[۲] معادل فارسی واژهٔ انگلیسی *Nested*

[۳] معادل فارسی عبارت انگلیسی *Contiguous subsequence*

[۴] معادل فارسی واژهٔ انگلیسی *Yield*

۲-۳. تعریف صوری تجزیهٔ وابستگی

در مورد روش‌های مبتنی بر داده، نیاز به تعریف فرآیند یادگیری و برای روش‌های مبتنی بر دستور زبان و مبتنی بر داده، نیاز به تعریف فرآیند تجزیه است. البته ممکن است روش‌های مبتنی بر دستور زبان هم مبتنی بر داده باشند. در این صورت دستور زبان از روی داده‌های آموزشی موجود در پیکرهٔ نشانه‌گذاری‌شده استخراج خواهد شد. علاوه بر تعریف‌های گفته‌شده در بخش ۲-۲، تعریف ویژه‌ای از تابع ویژگی[۱] وجود دارد. $f(x): X \to Y$ ورودی‌های x را به نمایش ویژگی‌ها در فضای Y نگاشت می‌کند [2].

تعریف (۲-۴) الگوی تجزیهٔ وابستگی[۲] شامل مجموعه‌ای از محدودیت‌ها $\mathbb{\Gamma}$، مجموعه‌ای از شاخص‌ها[۳] λ (که ممکن است تهی باشد) و الگوریتم معین تجزیه h است. $\mathbb{\Gamma}$ مجموعه‌ای از فضای درخت‌های وابستگی قابل قبول برای جملهٔ S است. این الگو با $M = (\mathbb{\Gamma}, \lambda, \mathrm{h})$ نشان داده می‌شود [2].

در مرحلهٔ یادگیری، مجموعه شاخص‌های λ بر اساس داده‌های آموزشی ساخته خواهد شد. مجموعهٔ آموزشی D شامل جملات S و درخت وابستگی متناظر با آن است [2]:

$$D = \{(S_d, G_d)\}_{d=0}^{|D|} \quad (۱-۲)$$

این شاخص‌ها معمولاً بر اساس بهینه‌سازی بر روی شاخص‌های از پیش‌تعریف‌شدهٔ $\mathbf{\Lambda}$ است. روش‌های بهینه‌سازی متفاوتی بر روی این شاخص‌ها قابل تعریف است که یکی از معروف‌ترین این روش‌ها بر اساس کاهش خطای تجزیه بر روی داده‌های آموزشی است. پس از آن‌که محدودیت‌ها، شاخص‌ها و الگوی یادگیری معین و تثبیت شد، برای هر جملهٔ ورودی S، این الگوریتم معین باید بهترین درخت وابستگی ممکن را بازگرداند [2]:

$$G = h(S, \mathbb{\Gamma}, \lambda) \quad (۲-۲)$$

تابع h یک فرآیند جست و جو در تمام گراف‌های وابستگی خوش‌ساختِ $\mathcal{G}_S$ برای جملهٔ S است. در این صورت تنها یک درخت به عنوان درخت وابستگیِ تشخیص‌داده‌شده شناخته می‌شود. در صورتی که با توجه به $\mathbb{\Gamma}$، دستور زبان طوری تعریف شود که در آن جملهٔ S عضویت نداشته باشد، ممکن در برخی مواقع خروجی فرآیند جست و جو تهی باشد [2].

---

[۱] معادل فارسی عبارت انگلیسی *Feature Function*

[۲] معادل فارسی عبارت انگلیسی *Dependency Parsing Model*

[۳] معادل فارسی واژهٔ انگلیسی *Parameter*

۳. تجزیهٔ مبتنی بر گذار

روش‌های مبتنی بر گذار، بر اساس یادگیری از روی پیکره‌های آموزشی شکل گرفته‌اند. به دلیل این که در این نوع از تجزیه، از دستور زبان صوری[۱] استفاده نمی‌شود، $\mathbb{\Gamma}$ تنها محدود به مجموعه درخت‌های ممکن برای جملهٔ S است.

۳-۱. سامانهٔ گذار

سامانهٔ گذار[۲] یک *دستگاه انتزاعی*[۳]، شامل تعدادی *پیکربندی*[۴] (حالت[۵]) و گذار (انتقال) بین این پیکربندی‌هاست [2]. ساده‌ترین مثال از یک سامانهٔ گذار، آتاماتون حالت محدود[۶] است. اساس تفکر سامانهٔ گذار بر این نگره است که با آغاز از پیکربندی آغازین و طی چند گذار معتبر، سامانه به یک پیکربندی نهایی خواهد رسید که در آن یک درخت وابستگی معتبر در خروجی ظاهر خواهد شد. یکی از راه‌های مرسوم برای نمایش پیکربندی استفاده از داده‌ساختار پشته[۷] است. در این حالت می‌توان یک پیکربندی را با یک پشته، حافظهٔ میان‌گیر ورودی[۸] و مجموعه‌ای از یال‌های وابستگی نشان داد [2].

تعریف (۳-۱) اگر مجموعهٔ گونه‌های وابستگی R را داشته باشیم، یک *پیکربندی* برای جملهٔ $S = w_0 w_1 \dots w_n$ به صورت سه‌تایی مرتب $c = (\sigma, \beta, A)$ خواهد بود؛ به طوری که [2]:

۱. σ پشته‌ای از واژه‌های $w_i \in V_s$؛

۲. β حافظهٔ میان‌گیر از واژه‌های $w_i \in V_s$؛ و

۳. A مجموعه‌ای از یال‌های وابستگی $(w_i, r, w_j) \in V_s \times R \times V_s$ است.

---

[۱] معادل فارسی عبارت انگلیسی *Formal Grammar*

[۲] معادل فارسی عبارت انگلیسی *Transition System*

[۳] معادل فارسی عبارت انگلیسی *Abstract Machine*

[۴] معادل فارسی واژهٔ انگلیسی *Configuration*

[۵] معادل فارسی واژهٔ انگلیسی *State*

[۶] معادل فارسی عبارت انگلیسی *Finite State Automaton*

[۷] معادل فارسی واژهٔ انگلیسی *Stack*

[۸] معادل فارسی واژهٔ انگلیسی *Input Buffer*

در تعریف (۳-۱) پشته شامل واژه‌های پردازش‌شده، حافظهٔ میان‌گیر شامل واژه‌های پردازش‌نشده و باقی‌مانده از جمله و A مجموعهٔ یال‌هایی است که تاکنون از جمله استخراج شده‌اند.

برای جملهٔ $S = w_0 w_1 \dots w_n$: ۱) پیکربندی *آغازین* $c_0(S)$ به صورت $([w_0]_\sigma, [w_1, \dots, w_n]_\beta, \emptyset)$، و ۲) پیکربندی *پایانی* به صورت $(\sigma, [\,]_\beta, A)$ است [2].

بر این اساس، گذار به صورت انتقال از یک پیکربندی به پیکربندی دیگر است. برای تجزیهٔ وابستگی جابه‌جایی-کاهش[۱] سه نوع گذار وجود دارد[۲]. در جدول ۱ این سه گونه گذار به نمایش گذاشته شده است[۳].

تعریف (۳-۲) دنبالهٔ گذار برای $S = w_0 w_1 \dots w_n$ دنباله‌ای از پیکربندی‌های $C_{0,m} = (c_0, c_1, \dots, c_m)$ به طوری که [2]:

۱. c_0 پیکربندی آغازین $c_0(S)$ برای S است؛

۲. c_m پیکربندی پایانی است؛ و

۳. برای هر i به طوری که $1 \le i \le m$، گذار $t \in T$ وجود دارد به طوری که $c_i = t(c_{i-1})$.

جدول ۱ سه نوع گذار در سامانهٔ گذار جابه‌جایی-کاهش [2]

نام گذار	نوع گذار	شرایط پیشین
تشکیل یال چپ $(Left\text{-}Arc_r)$	$(\sigma \vert w_i, w_j \vert \beta, A) \Longrightarrow (\sigma, w_j \vert \beta, A \cup \{(w_j, r, w_i)\})$	$i \neq 0$
تشکیل یال راست $(Right\text{-}Arc_r)$	$(\sigma \vert w_i, w_j \vert \beta, A) \Longrightarrow (\sigma, w_j \vert \beta, A \cup \{(w_i, r, w_j)\})$	
جابه‌جایی $(Shift)$	$(\sigma \vert w_i, w_j \vert \beta, A) \Longrightarrow (\sigma \vert w_i, \beta, A)$	

[۱] معادل فارسی عبارت انگلیسی *Shift-Reduce*

[۲] در *[16]* به این الگوریتم، روش یال معیار (*Arc-Standard*) گفته شده است.

[۳] از این پس از نماد T برای نشان دادن گذارهای مجاز استفاده خواهیم کرد.

لازم به ذکر است که با هر گذار ویژگی‌های پوشایی (ویژگی (۲-۲))، ریشه (ویژگی (۲-۱)) و تک‌سری (ویژگی (۲-۴)) ارضا می‌شود ولی ویژگی هم‌بندی (ویژگی (۲-۳)) لزوماً رعایت نخواهد شد. به خاطر این که ویژگی بدون دور بودن (ویژگی (۲-۵)) ارضا شود و هم‌بندی هم رعایت شود، برای ریشه‌های هر یک از درخت‌های پوشای وابستگی یک یال از واژهٔ ریشهٔ جمله به آن‌ها متصل خواهد شد.

Transition	Configuration		
	([ROOT],	[Economic, . . . , .],	Ø)
SH ⇒	([ROOT, Economic],	[news, . . . , .],	Ø)
LA_{ATT} ⇒	([ROOT],	[news, . . . , .],	$A_1 = \{(news, ATT, Economic)\}$)
SH ⇒	([ROOT, news],	[had, . . . , .],	A_1)
LA_{SBJ} ⇒	([ROOT],	[had, . . . , .],	$A_2 = A_1 \cup \{(had, SBJ, news)\}$)
SH ⇒	([ROOT, had],	[little, . . . , .],	A_2)
SH ⇒	([ROOT, had, little],	[effect, . . . , .],	A_2)
LA_{ATT} ⇒	([ROOT, had],	[effect, . . . , .],	$A_3 = A_2 \cup \{(effect, ATT, little)\}$)
SH ⇒	([ROOT, had, effect],	[on, . . . , .],	A_3)
SH ⇒	([ROOT, . . . on],	[financial, markets, .],	A_3)
SH ⇒	([ROOT, . . . , financial],	[markets, .],	A_3)
LA_{ATT} ⇒	([ROOT, . . . on],	[markets, .],	$A_4 = A_3 \cup \{(markets, ATT, financial)\}$)
RA_{PC} ⇒	([ROOT, had, effect],	[on, .],	$A_5 = A_4 \cup \{(on, PC, markets)\}$)
RA_{ATT} ⇒	([ROOT, had],	[effect, .],	$A_6 = A_5 \cup \{(effect, ATT, on)\}$)
RA_{OBJ} ⇒	([ROOT],	[had, .],	$A_7 = A_6 \cup \{(had, OBJ, effect)\}$)
SH ⇒	([ROOT, had],	[.],	A_7)
RA_{PU} ⇒	([ROOT],	[had],	$A_8 = A_7 \cup \{(had, PU, .)\}$)
RA_{PRED} ⇒	([ ],	[ROOT],	$A_9 = A_8 \cup \{(ROOT, PRED, had)\}$)
SH ⇒	([ROOT],	[ ],	A_9)

شکل ۶ نمونه‌ای از دنبالهٔ گذار در جملهٔ انگلیسی شکل ۱ [2]

به مجموعه درخت‌های پوشا با ویژگی‌های پوشایی، ریشه، تک‌سری و بدون دور بودن، جنگل وابستگی[۱] گویند. یک جنگل وابستگی، افکنشی است اگر و تنها اگر همهٔ درخت‌های آن افکنشی باشد. ویژگی دیگر سامانهٔ گذار جابه‌جایی-کاهش این است که هر کدام از درخت‌های جنگل وابستگی، افکنشی است [2].

نیوْر[۲] [16] سه تعریف برای ارزیابی سامانه‌های گذار تعریف کرده است:

[۱] معادل فارسی عبارت انگلیسی *Dependency Forest*

[۲] نوشتار فارسی نام سوئدی *Nivre*

تعریف (۳-۳) فرض کنید $S = (C, T, c_s, C_t)$ یک سامانهٔ گذار برای تجزیهٔ وابستگی باشد[۱]:

۱- S برای ردهٔ **G** از گراف‌های وابستگی **استوار**[۲] است، اگر و تنها اگر برای همهٔ جمله‌های x و برای همهٔ دنباله‌های گذار $C_{0,m}$ برای x در S، $G_{c_m} \in G$ باشد.

۲- S برای ردهٔ G از گراف‌های وابستگی **کامل**[۳] است، اگر و تنها اگر برای همهٔ جمله‌های x و برای همهٔ گراف‌های وابستگی G_x در x موجود در **G** و برای همهٔ دنباله‌های گذار $C_{0,m}$ برای x در S، $G_{c_m} = G_x$ باشد.

۳- S برای ردهٔ **G** از گراف‌های وابستگی **صحیح**[۴] است، اگر و تنها اگر هم **کامل** و هم **استوار** باشد.

۳-۲. الگوریتم تجزیه

سامانه‌های گذار به صورت عادی قطعی[۵] نیستند. معمولاً برای یک جمله چندین حالت ممکن به عنوان خروجی درخت وابستگی وجود دارد. در این صورت نیاز به یک تابع *پیش‌گو*[۶] برای تعیین درخت وابستگی درست وجود دارد. اگر این تابع پیش‌گو در دسترس باشد، کار تجزیهٔ وابستگی ساده خواهد شد. در این صورت با داشتن پیکربندی اولیهٔ $C_0(S)$، با تابع پیش‌گوی $o(c)$، گذار t تعیین می‌شود و پیکربندی مرحلهٔ بعد بر اساس گذار انجام شده در پیکربندی کنونی تعیین می‌شود. شبه‌برنامه ۱ نشان‌دهندهٔ روند استفاده از تابع پیش‌گو برای قطعی کردن سامانهٔ گذار است [2].

پیچیدگی زمانی روش استفاده از تابع پیش‌گو برای یک جمله با n واژه در حالت تجزیهٔ درخت افکنشی برابر با $O(n)$ است. مسأله‌ای که باقی‌ست این است که پیدا کردن تابع پیش‌گو در عمل کاری بسیار دشوار است. به همین دلیل به جای پیدا کردن تابع پیش‌گو، تقریبی از این تابع استفاده می‌شود. به همین منظور راهبردهای بسیاری مورد آزمون قرار گرفته است که موفق‌ترین راهبرد استفاده از رده‌بندهای آموزش‌دیده[۷] از پیکره‌های آموزشی بوده است. بر این اساس، تجزیهٔ مبتنی بر رده‌بند[۱]، یکی از

---

[۱] نمادگذاری‌های این تعریف بر اساس مرجع و بدون دخل و تصرف انجام شده است.

[۲] معادل فارسی واژهٔ انگلیسی *Correct*

[۳] معادل فارسی واژهٔ انگلیسی *Complete*

[۴] معادل فارسی واژهٔ انگلیسی *Correct*

[۵] معادل فارسی واژهٔ انگلیسی *Deterministic*

[۶] معادل فارسی واژهٔ انگلیسی *Oracle*

[۷] معادل فارسی عبارت انگلیسی *Trained Classifier*

مؤلفه‌های اصلی تجزیه بر مبنای گذار خواهد بود [2]. در بخش ۳-۳ به این نوع از تجزیه پرداخته شده است.

```
h(S, Γ, o)
1    c ← c_0(S)
2    while c in not termial
3        t ← o(c)
4        c ← c(t)
5    return G_c
```

شبه‌برنامه ۱ تجزیهٔ مبتنی بر گذار قطعی با استفاده از تابع پیش‌گو [2]

۳-۳. تجزیهٔ مبتنی بر رده‌بند

در مسألهٔ تجزیهٔ مبتنی بر رده‌بند، بر اساس روش‌های یادگیری خودکار از روی پیکرهٔ نشانه‌گذاری‌شدهٔ آموزشی، الگوی تجزیه استخراج می‌شود. به همین خاطر، نیاز به ایجاد نمایشی انتزاعی از فضای پیکربندی و ویژگی‌ها وجود دارد. در نتیجه یک تابع ویژگی $f(x): X \to Y$ وجود دارد که در دامنهٔ X مجموعهٔ C از پیکربندی‌های ممکن و Y حاصل ضرب مجموعه‌های مقدار ویژگی m-تایی است. بنابراین در تابع ویژگی $f(c): C \to Y$ هر پیکربندی به یک بردار ویژگی m-بُعدی نگاشت می‌شود. بر اساس این نمایش، می‌خواهیم رده‌بند $g: Y \to T$ را یاد بگیریم؛ به طوری که T مجموعهٔ همهٔ گذارهای ممکن است. در نتیجه برای رده‌بند g شرط $g(f(c)) = (o(c))$ در همهٔ پیکربندی‌های c صدق می‌کند. یعنی می‌خواهیم رده‌بندی را از پیکرهٔ درختی[۲] آموزشی یاد بگیریم که بتواند تقریب بسیار خوبی از تابع پیش‌گو باشد. در این‌جا سه سؤال اصلی وجود دارد [2]:

۱) چگونه پیکربندی‌ها را با استفاده از بردارهای ویژگی نشان دهیم؟

۲) از پیکره‌های درختی چگونه داده‌های آموزشی را استخراج کنیم؟

---

[۱] معادل فارسی عبارت انگلیسی *Classifier-Based Parsing*

[۲] معادل فارسی عبارت انگلیسی *Treebank*

۳) چگونه رده‌بندها را آموزش دهیم؟

در بخش‌های ۳-۳-۱، ۳-۳-۲ و ۳-۳-۳ به بررسی این پرسش‌ها پرداخته شده است.

۳-۳-۱. نمایش ویژگی

نمایش ویژگی $f(c)$ از پیکربندی c یک بردار m-بُعدی از ویژگی‌های سادهٔ $f_i(c)$ برای $1 \leq i \leq m$ است. نحوهٔ تعیین ویژگی‌ها بستگی به نوع روش یادگیری دارد. مثلاً در برخی از روش‌های یادگیری ویژگی‌ها باید عددی باشد. معمول‌ترین نوع انتخاب ویژگی، انتخاب ویژگی‌های مربوط به واژه است. به همین ترتیب، برای هر واژه دو نوع تابع ویژگی وجود دارد. یک *تابع نشانی*[۱] و یک *تابع صفت*[۲]. در تابع نشانی شناسهٔ یک واژه در پیکربندی (مثلاً واژهٔ بالای پشته) و در تابع صفت، صفت مخصوص به واژه (مثلاً برچسب اجزای سخن) مشخص می‌شود. به این ویژگی‌ها، *ویژگی‌های پیکربندی واژه*[۳] گفته می‌شود [2].

تعریف (۳-۴) در جملهٔ ورودی $S = w_0 w_1 \dots w_n$ با مجموعهٔ گره‌های V_s، تابع $(v\ o\ a)(c): C \to Y$ متشکل از:

۱) تابع نشانی $a(c): C \to V_s$، و

۲) تابع صفت $v(w): V_s \to Y$

یک *ویژگی پیکربندی واژه* است [2].

تابع نشانی قابل تقسیم به توابع ساده‌تری است که هر کدام از این توابع روی برخی از مؤلفه‌های پیکربندی c کار می‌کنند؛ مثلاً واژهٔ k-ام در پشته. البته توابع نشانی جزئی[۴] است، یعنی اگر پشته خالی باشد، واژهٔ k-ام تهی خواهد بود. در توابع صفت، دو نوع صفت وجود دارد: ایستا[۵] و پویا[۶]. به عنوان مثال

---

[۱] معادل فارسی عبارت انگلیسی *Address Function*

[۲] معادل فارسی عبارت انگلیسی *Attribute Function*

[۳] معادل فارسی عبارت انگلیسی *Configurational Word Feature*

[۴] معادل فارسی واژهٔ انگلیسی *Partial*

[۵] معادل فارسی واژهٔ انگلیسی *Static*

[۶] معادل فارسی واژهٔ انگلیسی *Dynamic*

صفت ریشهٔ واژه[1] ایستاست. زیرا این صفت ثابت است و بستگی به محتوا و بافت جمله ندارد. از طرفی دیگر، صفت‌هایی مانند برچسب واژهٔ سمت راست، به دلیل این که به محتوای جمله ربط مستقیم دارند، پویا هستند [2].

جدول ۲ الگوی ویژگی‌ها برای تجزیهٔ مبتنی بر گذار [2]

	نشانی	صفت
1	STK [0]	FORM
2	BUF [0]	FORM
3	BUF [1]	FORM
4	LDEP [STK [0]]	DEPREL
5	RDEP [STK [0]]	DEPREL
6	LDEP [BUF [0]]	DEPREL
7	RDEP [BUF [0]]	DEPREL

در جدول ۲ نمونه‌ای از بردار ویژگی‌ها، برای تجزیه مبتنی بر گذار نشان داده شده است. در این جدول، BUF و STK نشان‌دهندهٔ واژه‌های حافظهٔ میان‌گیر و پشته و LDEP و RDEP واژه‌های سمت چپ و سمت راست هستند. توابع صفت مورد استفاده در این جدول، FORM (به معنای شکل سادهٔ واژه) و DEPREL به معنای برچسب وابستگی است. با وجود این که نوع بردار ویژگی نشان‌داده‌شده در جدول ۲ مفید است ولی تجربهٔ حاصل از اجرای این برنامه نشان داده است که این اطلاعات برای به دست آوردن دقت بالا، ناکافی هستند. علاوه بر این ویژگی‌ها می‌شود ویژگی‌های دیگری مانند ریشهٔ واژه و ویژگی‌های نحو-ساخت‌واژی[2] را اضافه کرد.

[1] معادل فارسی واژهٔ انگلیسی *Lemma*

[2] معادل فارسی عبارت انگلیسی *Morphosyntactic*

$\mathbf{f}(c_0)$	=	(ROOT	Economic	news	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_1)$	=	(Economic	news	had	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_2)$	=	(ROOT	news	had	NULL	NULL	ATT	NULL)
$\mathbf{f}(c_3)$	=	(news	had	little	ATT	NULL	NULL	NULL)
$\mathbf{f}(c_4)$	=	(ROOT	had	little	NULL	NULL	SBJ	NULL)
$\mathbf{f}(c_5)$	=	(had	little	effect	SBJ	NULL	NULL	NULL)
$\mathbf{f}(c_6)$	=	(little	effect	on	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_7)$	=	(had	effect	on	SBJ	NULL	ATT	NULL)
$\mathbf{f}(c_8)$	=	(effect	on	financial	ATT	NULL	NULL	NULL)
$\mathbf{f}(c_9)$	=	(on	financial	markets	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_{10})$	=	(financial	markets	.	NULL	NULL	NULL	NULL)
$\mathbf{f}(c_{11})$	=	(on	markets	.	NULL	NULL	ATT	NULL)
$\mathbf{f}(c_{12})$	=	(effect	on	.	ATT	NULL	NULL	ATT)
$\mathbf{f}(c_{13})$	=	(had	effect	.	SBJ	NULL	ATT	ATT)
$\mathbf{f}(c_{14})$	=	(ROOT	had	.	NULL	NULL	SBJ	OBJ)
$\mathbf{f}(c_{15})$	=	(had	.	NULL	SBJ	OBJ	NULL	NULL)
$\mathbf{f}(c_{16})$	=	(ROOT	had	NULL	NULL	NULL	SBJ	PU)
$\mathbf{f}(c_{17})$	=	(NULL	ROOT	NULL	NULL	NULL	NULL	PRED)
$\mathbf{f}(c_{18})$	=	(ROOT	NULL	NULL	NULL	PRED	NULL	NULL)

شکل ۷ بردارهای ویژگی بر اساس شکل ۶ [2]

در شکل ۸ نمونه‌ای از اطلاعات مورد نیاز برای داده‌های آموزشی به نمایش گذاشته شده است. در این شکل LEMMA ریشهٔ واژه و FEATS ویژگی‌های نحو-ساخت‌واژی است.

Address	Attributes				
	FORM	LEMMA	POSTAG	FEATS	DEPREL
STK[0]	+	+	+	+	
STK[1]			+		
LDEP(STK[0])					+
RDEP(STK[0])					+
BUF[0]	+	+	+	+	
BUF[1]	+		+		
BUF[2]			+		
BUF[3]			+		
LDEP(BUF[0])					+
RDEP(BUF[0])					+

شکل ۸ نمونه‌ای از بردار ویژگی‌ها برای تجزیهٔ مبتنی بر گذار [2]

۳-۳-۲. داده‌های آموزشی

در پیکره‌ای که به عنوان مجموعه داده‌های آموزشی مورد استفاده قرار می‌گیرد، باید اطلاعات در مورد جملات، بردار ویژگی‌ها و برچسب‌های وابستگی درست باشد، تا رده‌بند یادگیرنده بتواند با تقریب مناسبی تابع پیش‌گو را تقریب بزند. همان‌طور که در بخش ۳-۲ اشاره شد، یک مجموعهٔ آموزشی شامل تعدادی جمله و تجزیهٔ نحوی درست آن است:

$$D = \{(S_d, G_d)\}_{d=0}^{|D|} \quad (۱-۳)$$

برای کار با روش مبتنی بر گذار، نیاز به تبدیل پیکرهٔ آموزشی D به پیکرهٔ جدید $\acute{D}$ داریم که بر اساس قواعد پیکربندی بتوان در آن سامانهٔ گذار را تعریف کرد [2].

$$\acute{D} = \{(\boldsymbol{f}(c_d), t_d)\}_{d=0}^{|\acute{D}|} \quad (۲-۳)$$

روند تبدیل به شرح ذیل است [2]:

- برای هر نمونهٔ $(S_d, G_d) \in D$ نخست دنبالهٔ گذار $C_{0,m}^d = (c_0, c_1, \dots, c_m)$ ساخته می‌شود؛ به طوری که

۱) $c_0 = c_0(S_d)$، و

۲) $G_d = (V_d, A_{c_m})$.

- برای پیکربندی‌های غیرپایانی $c_i^d \in C_{0,m}^d$ تابع $\left(\boldsymbol{f}(c_i^d), t_d^i\right)$ را به $\acute{D}$ اضافه می‌کنیم، به طوری که $t_d^i(c_i^d) = c_i^{d+1}$.

در این حال برای هر جملهٔ S_d با درخت وابستگی G_d، می‌توان دنبالهٔ گذار را برای G_d ساخت. با این فرض که همهٔ درخت‌ها افکنشی هستند، می‌توان از رابطهٔ (۳-۳) استفاده کرد [2].

$$o\big(c = (\sigma, \beta, A)\big) = \begin{cases} \boldsymbol{LEFT - ARC_r} & if(\beta[0], r, \sigma[0]) \in A_d \\ \boldsymbol{RIGHT - ARC_r} & if(\sigma[0], r, \beta[0]) \in A_d \; and, \\ for\; all\; w, \acute{r}, & if(\beta[0], \acute{r}, w) \in A_d \; then(\beta[0], \acute{r}, w) \in A \\ \boldsymbol{SHIFT_r} & otherwise \end{cases} \qquad (۳-۳)$$

نخستین حالت در رابطهٔ (۳-۳) زمانی است که اولین واژهٔ در حافظهٔ میان‌گیر و اولین واژهٔ پشته با هم دارای رابطه باشند و واژهٔ حافظهٔ میان‌گیر، واژهٔ سر باشد. دومین حالت برای زمانی است که مانند حالت نخست است با این تفاوت واژهٔ سر متفاوت است. یک شرط در این زمینه وجود دارد و آن این است که همهٔ یال‌های خارج‌شونده از $\beta[0]$ از قبل به A اضافه شده باشند.

۳-۳-۳. رده‌بندها

روش‌های مختلفی برای رده‌بندی بر اساس روش گذار وجود دارد. از بین این روش‌ها دو روش بیش‌ترین استفاده را داشته است: یادگیری مبتنی بر حافظه[۱] و دستگاه بردار پشتیبان (اس.وی.ام.)[۲] [2]. در یادگیری و رده‌بندی مبتنی بر حافظه همهٔ نمونه‌های آموزشی باید در حافظه نگهداری شود و رده‌بندی بر اساس شباهت همهٔ داده‌ها با هم انجام می‌شود. به همین دلیل این روش به روش یادگیری تنبل[۳] معروف است [17]. بر همین اساس یادگیری مبتنی بر حافظه در حین آموزش بهینه است. زیرا تنها پیش‌نیاز آوردن نمونه‌ها در حافظه است. ولی فرآیند رده‌بندی این روش مخصوصاً برای پیکره‌های حجیم بسیار کند است [2, 18]. در روش دستگاه بردار پشتیبان، سعی بر این است که بین داده‌های آموزشی پهن‌ترین مرز ممکن به وجود بیاید یا عباراتی دارای بیشینه حاشیهٔ[۴] ممکن باشند [19]. در حالت عادی

---

[۱] معادل فارسی عبارت انگلیسی *Memory-Based Learning*

[۲] معادل فارسی عبارت انگلیسی *Support Vector Machine (SVM)*

[۳] معادل فارسی عبارت انگلیسی *Lazy Learning Method*

[۴] معادل فارسی عبارت انگلیسی *Max-Margin*

این روش برای مسائل با دو رده[۱] است و ویژگی‌های داده‌های آموزشی باید عددی باشند. لذا ویژگی‌های مقوله‌ای[۲] (مانند برچسب اجزای سخن) باید تبدیل به ویژگی‌های عددی شود. برای استفاده از رده‌بند برای حل مسائل غیرخطی، باید تابع شالودهٔ[۳] آن را به صورت غیرخطی تبدیل کرد. مشکلی که در یادگیری با تابع شالودهٔ غیرخطی وجود دارد، فرآیند طولانی پردازش یادگیری است. یکی دیگر از مشکلات این روش‌ها دودویی بودن مسئله است. به همین دلیل نیاز به روش‌هایی برای تبدیل این روش‌ها برای حل مسائل چندرده‌ای است. برای مسائل چندرده‌ای، راه حل‌های مختلفی وجود دارد که در دستگاه بردار پشتیبان که هم‌اکنون پیشرفته‌ترین روش رده‌بندی برای تجزیهٔ وابستگی مبتنی بر گذار، دستگاه بردار پشتیبان است، مورد استفاده قرار می‌گیرد [2]. یامادا و ماتسوموتو [20] برای نخستین بار از دستگاه بردار پشتیبان برای تجزیهٔ وابستگی استفاده کرده‌اند.

۳-۴. روش‌های دیگر تجزیهٔ مبتنی بر گذار

علاوه بر روش‌های اشاره‌شده در بخش‌های قبل، روش‌های دیگری در سامانهٔ گذار وجود دارد. روشی که تاکنون اشاره شد، روش یال معیار[۴] بوده است که روش مشتاق به یال[۵] تعمیمی بر این روش است. روش مبتنی بر فهرست[۶] نیز روشی است که با آن می‌توان هم درخت‌های افکنشی و هم غیرافکنشی را تجزیه کرد.

۳-۴-۱. تجزیهٔ مشتاق به یال

یکی از مشکلات روش یال معیار این است که تا زمانی که در یک واژهٔ وابستهٔ راست، وابسته‌هایش تعیین نشود، این واژه قابل اتصال به واژهٔ سرش نیست. در نتیجه برای از بین بردن این مشکل، نیاز به عمل بیرون انداختن[۷] واژهٔ بالای پشته وجود دارد. در این صورت، یک گذار جدید به عنوان کاهش[۸] اضافه

---

[۱] معادل فارسی واژهٔ انگلیسی *Class*

[۲] معادل فارسی واژهٔ انگلیسی *Categorical*

[۳] معادل فارسی عبارت انگلیسی *Kernel Function*

[۴] معادل فارسی عبارت انگلیسی *Arc- Standard*

[۵] معادل فارسی عبارت انگلیسی *Arc-Eager* دلیل استفاده از این ترجمه، نوع خود الگوریتم است که در این الگوریتم ترجیح بر این است که هر چه سریع‌تر یال چپ یا راست تولید شود.

[۶] معادل فارسی عبارت انگلیسی *List-Based*

[۷] معادل فارسی واژهٔ انگلیسی *Pop*

[۸] معادل فارسی واژهٔ انگلیسی *Reduce*

خواهد شد. شرط عمل کاهش این است که واژهٔ سر پشته، خود دارای واژهٔ سر (w_k) با رابطهٔ $\acute{r}$ باشد. رابطهٔ $\acute{r}$ ممکن است هر نوع رابطه‌ای باشد. این روش به روش مشتاق به یال معروف است که نخستین نیوْر [21] آن را ارائه کرده است. فهرست چهار گذار اصلی در این روش در جدول ۳ نشان داده شده است [2, 16].

جدول ۳ چهار نوع گذار در سامانهٔ گذار جابه‌جایی-کاهش مشتاق به یال [2]

نام گذار	نوع گذار	شرایط پیشین
تشکیل یال چپ ($Left\text{-}Arc_r$)	(	
تشکیل یال راست ($Right\text{-}Arc_r$)		
کاهش (Reduce)	(	
جابه‌جایی (Shift)		

هر دو روش یال معیار و روش مشتاق به یال برای درخت‌های افکنشی استوار، کامل و صحیح هستند. روش مشتاق به یال در بدترین حالت دارای پیچیدگی زمانی و فضایی $O(n)$ است [16]. برای این که بتوان این روش را برای درخت‌های غیرافکنشی قابل پیاده‌سازی کرد، در گذارهای تشکیل یال چپ و تشکیل یال راست تغییراتی جزئی ایجاد خواهد شد. در این تغییر واژهٔ دوم در پشته نیز مورد بررسی قرار خواهد گرفت. در این صورت بسیاری از درخت‌های غیرافکنشی با این روش قابل حل خواهند بود [22]. این دو گذار در جدول ۴ نشان داده شده است.

۳-۴-۲. تجزیهٔ مبتنی بر فهرست

برای این که بتوان همهٔ حالت‌های ممکن را برای درخت‌های غیرافکنشی مورد تجزیه قرار داد، یک پشته کافی نخواهد بود. به همین علت کاوینگتون [23] روشی را بر مبنای دو پشته پیشنهاد کرد. این روش که به روش مبتنی بر فهرست معروف است، قابلیت تجزیهٔ ساختارهای افکنشی و غیرافکنشی را داراست.

جدول ۴ دو گذار اضافه شده به گذارهای جدول ۳ برای تجزیهٔ درخت‌های غیرافکنشی [2]

نام گذار	نوع گذار	شرایط پیشین
$NP\text{-}LEFT_r$		
$NP\text{-}RIGHT_r$		

تعریف (۳-۵) پیکربندی مبتنی بر فهرست برای جملهٔ $x = (w_0, w_1, \dots, w_n)$ یک چهارتایی مرتب $c = (\lambda_1, \lambda_2, \beta, A)$ است؛ به طوری که [16]:

۱) λ_1 فهرستی از نمادهای $i_1 \leq k_1$ (برای $k_1 \leq n$) است؛

۲) λ_2 فهرستی از نمادهای $i_2 \leq k_2$ (برای $k_2 \leq n$) است ؛

۳) β حافظهٔ میان‌گیر از نمادها $j > k$ است ؛ و

۴) A مجموعه‌ای از یال‌های وابستگی به طوری که $G = (\{0,1, \dots, n\}, A)$ گراف وابستگی از x است.

در تعریف (۳-۵)، λ_1 به صورت نزولی و λ_2 به صورت صعودی مرتب شده است. در نتیجه $\lambda_1|i$ و $j|\lambda_2$ نشان‌دهندهٔ این است که i سر λ_1 و j سرِ λ_2 است. بنابراین می‌توان وضعیت پشته‌های جملهٔ [1 2 3 4 5] را به صورت [5 4 3] .[2 1] نوشت [16].

تعریف (۳-۶) یک *سامانهٔ گذار مبتنی بر فهرست* چهارتایی مرتب $S = (C, T, c_S, C_t)$ است؛ به طوری که [16]:

۱) C مجموعهٔ همهٔ پیکربندی‌های مبتنی بر فهرست است؛

۲) $c_S(x = (w_0, w_1, w_n)) = ([0], [], [1, \dots, n], \emptyset)$؛

۳) T مجموعهٔ همهٔ گذارها به صورت تابعی به شکل $t: C \to C$ است؛ و

۴) $C_t = \{c \in C | c = (\lambda_1, \lambda_2, [], A)\}$ است.

روش مبتنی بر فهرست هم برای درخت‌های افکنشی و هم برای درخت‌های غیرافکنشی کارساز است. برای تجزیهٔ مبتنی بر فهرست، در آغاز در λ_1 ریشهٔ مصنوعی گره صفر و λ_2 خالی است. بقیهٔ واژه‌ها در حافظهٔ میان‌گیر هستند و A خالی است. روند تجزیه با به انتها رسیدن حافظهٔ میان‌گیر به پایان می‌رسد [16].

- **تجزیهٔ مبتنی بر فهرست غیرافکنشی**

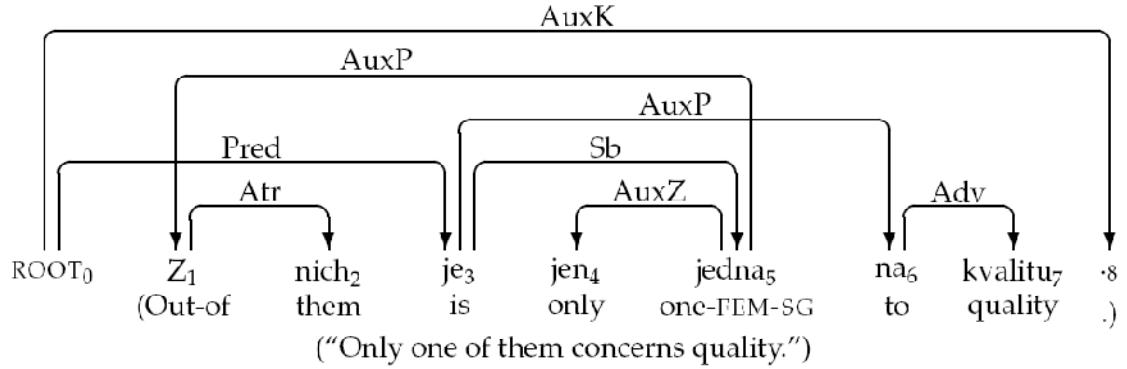
در تجزیهٔ مبتنی بر فهرست برای درخت‌های افکنشی، چهار نوع گذار تعریف می‌شود. در جدول ۵ گذارهای موجود در سامانهٔ گذار مبتنی بر فهرست غیرافکنشی به نمایش گذاشته شده است.

جدول ۵ گذارهای سامانهٔ گذار مبتنی بر فهرست برای ساختار غیرافکنشی [16]

نام گذار	نوع گذار	شرایط پیشین
تشکیل یال چپ ($Left\text{-}Arc_r^l$)		
تشکیل یال راست ($Right\text{-}Arc_r^l$)		
عدم تشکیل یال ($No\ Arc^n$)		
جابه‌جایی ($Shift^\lambda$)		

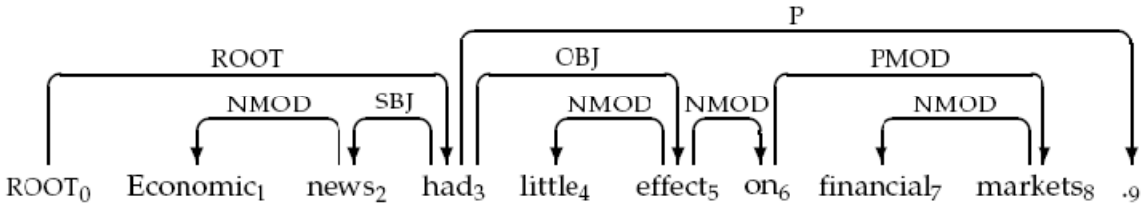
به این دلیل که برای یک رابطهٔ وابستگی، هم واژهٔ سر و واژهٔ وابسته یا در β یا در λ_2 وجود دارد و این که با گذار عدم تشکیل یال[1]، قابلیت ساخته شدن گراف وابستگی غیرافکنشی وجود دارد، این روش برای ردهٔ جنگل افکنشی همیشه صحیح است. پیچیدگی زمان این الگوریتم در بدترین حالت $O(n^2)$ و پیچیدگی فضایی در بدترین حالت برابر با $O(n)$ است [16].

[1] معادل فارسی عبارت انگلیسی *No Arc*



شکل ۹ نمونه‌ای از یک درخت غیرافکنشی جملهٔ چکی از پیکرهٔ درختی پراگ [16]

در شکل ۹ نمونه‌ای یک درخت وابستگی غیرافکنشی برای یک جمله به زبان چکی از پیکرهٔ درختی پراگ [24] به نمایش گذاشته شده است. در شکل ۱۱ فرآیند تجزیهٔ این جمله به نمایش گذاشته شده است.



شکل ۱۰ نمونه‌ای از یک درخت وابستگی افکنشی برای یک جملهٔ انگلیسی [16]

Transition	Configuration				
	([0],	[ ],	$[1,\ldots,8]$,	$\emptyset$	)
$\text{SHIFT}^{\lambda} \Longrightarrow$	($[0,1]$,	[ ],	$[2,\ldots,8]$,	$\emptyset$	)
$\text{RIGHT-ARC}^{n}_{\text{Atr}} \Longrightarrow$	([0],	[**1**],	$[2,\ldots,8]$,	$A_1 = \{(1,\text{Atr},2)\}$	)
$\text{SHIFT}^{\lambda} \Longrightarrow$	($[0,1,2]$,	[ ],	$[3,\ldots,8]$,	A_1	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,1]$,	[2],	$[3,\ldots,8]$,	A_1	)
$\text{NO-ARC}^{n} \Longrightarrow$	([0],	[**1**,2],	$[3,\ldots,8]$,	A_1	)
$\text{RIGHT-ARC}^{n}_{\text{Pred}} \Longrightarrow$	([ ],	$[0,1,2]$,	$[3,\ldots,8]$,	$A_2 = A_1 \cup \{(0,\text{Pred},3)\}$	)
$\text{SHIFT}^{\lambda} \Longrightarrow$	($[0,\ldots,3]$,	[ ],	$[4,\ldots,8]$,	A_2	)
$\text{SHIFT}^{\lambda} \Longrightarrow$	($[0,\ldots,4]$,	[ ],	$[5,\ldots,8]$,	A_2	)
$\text{LEFT-ARC}^{n}_{\text{AuxZ}} \Longrightarrow$	($[0,\ldots,3]$,	[4],	$[5,\ldots,8]$,	$A_3 = A_2 \cup \{(5,\text{AuxZ},4)\}$	)
$\text{RIGHT-ARC}^{n}_{\text{Sb}} \Longrightarrow$	($[0,1,2]$,	$[3,4]$,	$[5,\ldots,8]$,	$A_4 = A_3 \cup \{(3,\text{Sb},5)\}$	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,1]$,	$[2,3,4]$,	$[5,\ldots,8]$,	A_4	)
$\text{LEFT-ARC}^{n}_{\text{AuxP}} \Longrightarrow$	([0],	[**1**,...,4],	$[5,\ldots,8]$,	$A_5 = A_4 \cup \{(5,\text{AuxP},1)\}$	)
$\text{SHIFT}^{\lambda} \Longrightarrow$	($[0,\ldots,5]$,	[ ],	$[6,7,8]$,	A_5	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,\ldots,4]$,	[5],	$[6,7,8]$,	A_5	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,\ldots,3]$,	$[4,5]$,	$[6,7,8]$,	A_5	)
$\text{RIGHT-ARC}^{n}_{\text{AuxP}} \Longrightarrow$	($[0,1,2]$,	$[3,4,5]$,	$[6,7,8]$,	$A_6 = A_5 \cup \{(3,\text{AuxP},6)\}$	)
$\text{SHIFT}^{\lambda} \Longrightarrow$	($[0,\ldots,6]$,	[ ],	$[7,8]$,	A_6	)
$\text{RIGHT-ARC}^{n}_{\text{Adv}} \Longrightarrow$	($[0,\ldots,5]$,	[6],	$[7,8]$,	$A_7 = A_6 \cup \{(6,\text{Adv},7)\}$	)
$\text{SHIFT}^{\lambda} \Longrightarrow$	($[0,\ldots,7]$,	[ ],	$[8]$,	A_7	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,\ldots,6]$,	[7],	$[8]$,	A_7	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,\ldots,5]$,	$[6,7]$,	$[8]$,	A_7	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,\ldots,4]$,	$[5,6,7]$,	$[8]$,	A_7	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,\ldots,3]$,	$[4,\ldots,7]$,	$[8]$,	A_7	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,1,2]$,	$[3,\ldots,7]$,	$[8]$,	A_7	)
$\text{NO-ARC}^{n} \Longrightarrow$	($[0,1]$,	$[2,\ldots,7]$,	$[8]$,	A_7	)
$\text{NO-ARC}^{n} \Longrightarrow$	([0],	[**1**,...,7],	$[8]$,	A_7	)
$\text{RIGHT-ARC}^{n}_{\text{AuxK}} \Longrightarrow$	([ ],	$[0,\ldots,7]$,	$[8]$,	$A_8 = A_7 \cup \{(0,\text{AuxK},8)\}$	)
$\text{SHIFT}^{\lambda} \Longrightarrow$	($[0,\ldots,8]$,	[ ],	[ ],	A_8	)

شکل ۱۱ فرآیند تجزیهٔ مبتنی بر فهرست غیرافکنشی برای جملهٔ شکل ۹ [16]

- **تجزیهٔ مبتنی بر فهرست افکنشی**

در روش مبتنی بر فهرست افکنشی مانند روش غیرافکنشی چهار نوع گذار وجود دارد. در جدول ۶ گذارهای این سامانه نشان داده شده است.

جدول ۶ گذارهای سامانهٔ گذار مبتنی بر فهرست برای ساختار افکنشی [16]

نام گذار	نوع گذار	شرایط پیشین
تشکیل یال چپ ($Left\text{-}Arc_r^l$)		
تشکیل یال راست ($Right\text{-}Arc_r^l$)		
عدم تشکیل یال ($No\ Arc^n$)		
جابه‌جایی ($Shift^\lambda$)		

در شکل ۱۲ نمونه‌ای از فرآیند تجزیهٔ مبتنی بر فهرست به صورت افکنشی برای جملهٔ شکل ۱۰ به نمایش گذاشته شده است. پیچیدگی زمان این الگوریتم در بدترین حالت $O(n^2)$ و پیچیدگی فضایی در بدترین حالت برابر با $O(n)$ است [16].

Transition	Configuration				
	(	[0],	[ ],	[1, . . . , 9],	∅)
SHIFT$^{\lambda}$ ⟹	(	[0, 1],	[ ],	[2, . . . , 9],	∅)
LEFT-ARC$^{p}_{\text{NMOD}}$ ⟹	(	[0],	[ ],	[2, . . . , 9],	$A_1 = \{(2, \text{NMOD}, 1)\}$)
SHIFT$^{\lambda}$ ⟹	(	[0, 2],	[ ],	[3, . . . , 9],	A_1)
LEFT-ARC$^{p}_{\text{SBJ}}$ ⟹	(	[0],	[ ],	[3, . . . , 9],	$A_2 = A_1 \cup \{(3, \text{SBJ}, 2)\}$)
RIGHT-ARC$^{p}_{\text{ROOT}}$ ⟹	(	[0, 3],	[ ],	[4, . . . , 9],	$A_3 = A_2 \cup \{(0, \text{ROOT}, 3)\}$)
SHIFT$^{\lambda}$ ⟹	(	[0, 3, 4],	[ ],	[5, . . . , 9],	A_3)
LEFT-ARC$^{p}_{\text{NMOD}}$ ⟹	(	[0, 3],	[ ],	[5, . . . , 9],	$A_4 = A_3 \cup \{(5, \text{NMOD}, 4)\}$)
RIGHT-ARC$^{p}_{\text{OBJ}}$ ⟹	(	[0, 3, 5],	[ ],	[6, . . . , 9],	$A_5 = A_4 \cup \{(3, \text{OBJ}, 5)\}$)
RIGHT-ARC$^{p}_{\text{NMOD}}$ ⟹	(	[0, . . . , 6],	[ ],	[7, 8, 9],	$A_6 = A_5 \cup \{(5, \text{NMOD}, 6)\}$)
SHIFT$^{\lambda}$ ⟹	(	[0, . . . , 7],	[ ],	[8, 9],	A_6)
LEFT-ARC$_{\text{NMOD}}$ ⟹	(	[0, . . . 6],	[ ],	[8, 9],	$A_7 = A_6 \cup \{(8, \text{NMOD}, 7)\}$)
RIGHT-ARC$^{p}_{\text{PMOD}}$ ⟹	(	[0, . . . , 8],	[ ],	[9],	$A_8 = A_7 \cup \{(6, \text{PMOD}, 8)\}$)
NO-ARC^{p} ⟹	(	[0, . . . , 6],	[8],	[9],	A_8)
NO-ARC^{p} ⟹	(	[0, 3, 5],	[6, 8],	[9],	A_8)
NO-ARC^{p} ⟹	(	[0, 3],	[5, 6, 8],	[9],	A_8)
RIGHT-ARC$^{p}_{\text{P}}$ ⟹	(	[0, 3, 9],	[ ],	[ ],	$A_9 = A_8 \cup \{(3, \text{P}, 9)\}$)

شکل ۱۲ فرآیند تجزیهٔ مبتنی بر فهرست افکنشی برای جملهٔ شکل ۱۰ [۷]

۳-۴-۳. تجزیهٔ شبه‌افکنشی

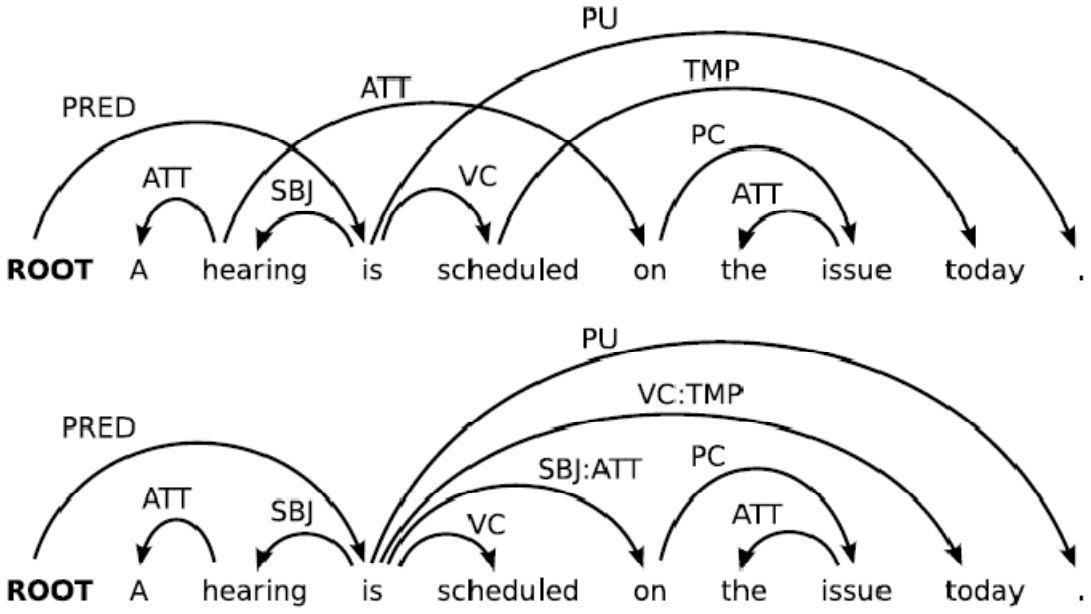
یکی از مشکلاتی که در عمدهٔ روش‌های موجود در سامانه‌های گذار در تجزیهٔ مبتنی بر وابستگی وجود دارد، این است که اصل این روش بر تجزیهٔ ساختارهای افکنشی بنا شده است و بسیاری از درخت‌های غیرافکنشی در این روش‌ها قابل تجزیه نیستند. یکی از راه‌حل‌هایی که در این مورد پیشنهاد شده است، استفاده از روشی به نام روش شبه‌افکنشی[۱] است. مراحل این روش به شرح ذیل است [2]:

۱- درخت‌های غیرافکنشی با اعمال تغییراتی در یال‌ها، به درخت‌های افکنشی تبدیل شود. در این تغییرات به یال‌های تغییریافته اطلاعات اضافی منتقل شود.

۲- بر روی درخت جدید، مرحلهٔ آموزش روش تجزیهٔ افکنشی اجرا شود.

۳- بر روی متون آزمایشی بر اساس روش افکنشی، تجزیه صورت بگیرد.

۴- خروجی تجزیه‌گر بر اساس یال‌های با اطلاعات اضافی به ساختار غیرافکنشی تبدیل شود.



شکل ۱۳ نمونه‌ای از یک درخت غیرافکنشی (بالا) و تبدیل‌یافتهٔ افکنشی آن (پایین) [2]

به دلیل این که همیشه برای یک یال غیرافکنشی گره والدینی وجود دارد که با تغییر مبدأ یال از گره اصلی به والد، یال افکنشی به یال غیرافکنشی تبدیل می‌شود، این روش پیشنهاد شده است. در این

[۱] معادل فارسی عبارت انگلیسی *Pseudo-Projective*

صورت به جای یال غیرافکنشی $(ANC(w_i), r, w_j)$ یال جدید (w_i, r, w_j) تشکیل می‌شود که $ANC(w_i)$ نزدیک‌ترین گره والد به w_i است؛ به طوری که این یال جدید افکنشی باشد. به عنوان قرارداد برای این که نشان دهیم که یال تشکیل شده مصنوعی است برچسبی به آن اضافه می‌کنیم. مثلاً در شکل ۱۳ به جای برچسب *TMP* برچسب *VC: TMP* گذاشته شده است.

برای تبدیل درخت شبه‌افکنشی به درخت افکنشی باید بین گره‌های فرزند جست و جو صورت گیرد. طبق گفتهٔ نیوْر و نیلسون [25] بیش از ۹۰٪ از جمله‌های با ساختار غیرافکنشی در زبان طبیعی با این روش قابل تجزیه است. مزیت اصلی این روش در تجزیهٔ جملات با ساختار غیرافکنشی در زمان خطی است و عیب این روش در این است که تعداد یال‌های متمایز و جدا افزایش می‌یابد که البته این عیب تأثیر منفی در زمان آموزش و تجزیه خواهد داشت [16].

۴. تجزیهٔ مبتنی بر گراف

با توجه به تعاریفی که از گراف وابستگی گفته شد، بدیهی است که چنین ساختاری را می‌توان با نمایش گراف مورد بررسی قرار داد. بدین صورت مانند تجزیهٔ مبتنی بر گذار، با الگوی تجزیه $M = (\mathbb{\Gamma}, \lambda, \mathrm{h})$ روبه‌رو خواهیم بود که $\mathbb{\Gamma}$ مجموعه‌ای از محدودیت‌ها روی مجموعه‌ای از ساختارهای قابل قبول، λ مجموعه‌ای از شاخص‌ها و h الگوریتم تجزیه است. در نمایش مبتنی بر گراف برای هر گراف وابستگی یک امتیاز[۱] در نظر گرفته می‌شود. در نتیجه، امتیاز نشان‌دهندهٔ این است که یک گراف برای جملهٔ ورودی S تا چه اندازه صحیح است [2].

$$Score\ (G) = Score\ (V, A) \in \mathbb{R} \quad (۱-۴)$$

در نتیجه برای زیرگراف‌های ψ و مجموعهٔ زیرگراف‌های Ψ_G مربوط به گراف G تابع f تعریف می‌شود:

$$Score\ (G) = f(\psi_1, \psi_2, \dots, \psi_q);\ \psi_i \in \Psi_G \quad (۲-۴)$$

معمولاً فرض می‌شود که این تابع برابر با جمع شاخص‌های زیرگراف‌هاست.

$$Score(g) = \sum_{\psi \in \Psi_G} \lambda_\psi \quad (۳-۴)$$

در نتیجه بر اساس رابطهٔ (۴-۳) نیاز به تعریف چهار مؤلفه خواهد بود [2]:

۱- تعریف Ψ_{G} برای گراف G.

۲- تعریف $\lambda = \{\lambda_\psi | for\ all\ \psi \in \Psi_{\mathrm{G}}, for\ all\ G \in \mathcal{G}_s, for\ all\ S\}$.

[۱] معادل فارسی عبارت انگلیسی *Score*

۳- تعریف تابع برای یادگیری λ از روی داده‌های برچسب‌دار.

۴- الگوریتم تجزیهٔ $h\left(\mathbb{\Gamma}, \lambda, \mathrm{h}\right) = \mathrm{argmax}_{\mathrm{G}\in\mathcal{G}_s}\mathrm{Score}\ (\mathrm{G})$.

ساده‌ترین نوع تعریف برای تجزیهٔ مبتنی بر گراف، الگوی مبتنی بر یال[1] است [2].

۴-۲. الگوی مبتنی بر یال

برای گراف وابستگی $G = (V, A)$ به این صورت تعریف می‌شود [2]:

- $\Psi_G = A$
- برای هر $(w_i, r, w_j) \in A$ خواهیم داشت: $\lambda_\psi = \lambda_{(w_i,r,w_j)} \in \mathbb{R}$

در این صورت $\boldsymbol{\lambda}$ مجموعهٔ همهٔ وزن‌های یال‌هاست. در نتیجه تعریف امتیاز یک گراف وابستگی به صورت رابطهٔ (۴-۴) خواهد بود.

$$Score(g) = \sum_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)} \quad (\mathbf{۴-۴})$$

با فرض این که همهٔ وزن‌های موجود در $\boldsymbol{\lambda}$ معلوم است، دغدغهٔ اصلی در تجزیه، پیدا کردن بهترین حالت گراف است.

$$(\mathbb{\Gamma}, \lambda, h) = argmax_{G\in\mathcal{G}_s} Score\ (G) = argmax_{G\in\mathcal{G}_s} \sum_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)} \quad (\mathbf{۵-۴})$$

برای پیدا کردن بهترین درخت تجزیهٔ ممکن باید درختی پیدا شود که جمع شاخص مؤلفهٔ یال‌هایش بیشینه باشد. البته می‌توان برای سهولت حالت جمع را به ضرب تبدیل کرد [2].

$$h(\mathbb{\Gamma}, \lambda, h) = \quad (\mathbf{۶-۴})$$

$$argmax_{G\in\mathcal{G}_s} \prod_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)} = argmax_{G\in\mathcal{G}_s} log\ [\prod_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)}] = argmax_{G\in\mathcal{G}_s} \sum_{(w_i,r,w_j)\in A} log\ [\lambda_{(w_i,r,w_j)}]$$

۴-۳. الگوریتم تجزیهٔ مبتنی بر یال

الگوریتم مبتنی بر یال حاصل از ادغام نظریه‌های گراف و تجزیهٔ وابستگی است. به همین صورت برای پیدا کردن بهترین تجزیه، باید درخت پوشای بیشینه[2] پیدا شود.

---

[1] معادل فارسی عبارت انگلیسی *Arc-Factored*

[2] معادل فارسی عبارت انگلیسی *Maximum Spanning Tree (MST)*

تعریف (۴-۱) درخت پوشای بیشینه برای گراف جهت‌دار G، زیرگراف Ǵ با بیش‌ترین امتیاز است که شرایط درخت پوشا را ارضا کند [2]:

- $V \doteq V$ یعنی همهٔ گره‌های G در Ǵ وجود داشته باشد؛ و
- Ǵ درختی جهت‌دار باشد.

گزاره (۴-۱) مجموعهٔ گراف‌های وابستگی خوش‌ساخت برای جملهٔ S، $\mathcal{G}_s$، و مجموعهٔ درخت‌های پوشای G_s منحصر به فرد هستند [2].

گزاره (۴-۱) با استفاده از قضایای موجود در نظریهٔ گراف قابل اثبات است (اثبات این گزاره در [2] آمده است). با وجود این گزاره می‌توان یک نتیجهٔ فرعی نیز گرفت. در نتیجه می‌توان مسألهٔ تجزیه را به مسألهٔ یافتن درخت پوشای بیشینه نگاشت کرد. مسألهٔ حاضر چالش اصلی در الگوریتم مبتنی بر یال است. در این الگوریتم برای ساختارهای افکنشی بیش از آن که نظریهٔ گراف‌ها مطرح باشد با دغدغه‌های تجزیهٔ مبتنی بر نمودار[۱] روبه‌رو هستیم [2].

۴-۳-۲. کاهش مسألهٔ برچسب‌دار به مسألهٔ بدون برچسب

می‌توان مسألهٔ موجود را از ساختار برچسب‌دار به ساختار بدون برچسب کاهش داد. این کاهش با پیچیدگی زمانی $O(|R|n^2)$ انجام می‌شود که گراف جدید $\acute{G}$ با $O(n^2)$ یال حاصل خواهد شد که برای هر یال شاخص بر اساس $|R|$ حالت ممکن تخمین زده می‌شود [2].

گزاره (۴-۲) اگر $G = (V, A)$ درخت پوشای بیشینه برای G_s و $\acute{G} = (V, A)$ درخت پوشای بیشینه برای $\acute{G}_s$ باشد؛ حالت‌های ذیل به وجود خواهد آمد [2]:

۱) $\sum_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)} = \sum_{(w_i,w_j)\in \acute{A}} \lambda_{(w_i,w_j)}$

۲) $(w_i, r, w_j) \in A$ اگر و تنها اگر $(w_i, w_j) \in \acute{A}$ و $r = argmax_r \lambda_{(w_i,r,w_j)}$.

[۱] معادل فارسی عبارت انگلیسی *Chart-Parsing*

با توجه به گزاره (۴-۲) می‌توان یک مسألهٔ تجزیهٔ برچسب‌دار را به صورت بدون برچسب حل و سپس با استفاده از نگاشت معکوس، نتیجه را به صورت برچسب‌دار تبدیل کرد.

۴-۳-۳. الگوریتم تجزیهٔ غیرافکنشی

برای حل مسأله به صورت غیرافکنشی می‌توان از الگوریتم چو-لیو-ادموندز[۱] [26, 27] استفاده کرد. شبه‌برنامهٔ این الگوریتم در شکل ۱۴ نشان داده شده است. در حالت عادی این الگوریتم دارای پیچیدگی زمانی $O(n^3)$ است که تارجان [28] با اعمال بهبودی بر روی این الگوریتم برای گراف‌های انبوه و کامل این پیچیدگی را به $O(n^2)$ کاهش داد. از آن‌جایی که الگوریتم ارائه شدهٔ تارجان بسیار پیچیده است و کم‌تر اتفاق می‌افتد که تعداد فراخوانی‌های بازگشتی در الگوریتم اصلی به عدد n برسد، در نتیجه از استفاده از این الگوریتم صرف نظر می‌شود. بنابراین با در نظر گرفتن پیچیدگی کاهش درخت برچسب‌دار به بدون برچسب، پیچیدگی یافتن بهترین تجزیه در این روش برابر با $O(|R|n^2 + n^2)$ یا همان $O(|R|n^2)$ خواهد بود [2].

[۱] معادل فارسی عبارت انگلیسی *Chu-Liu-Edmonds*

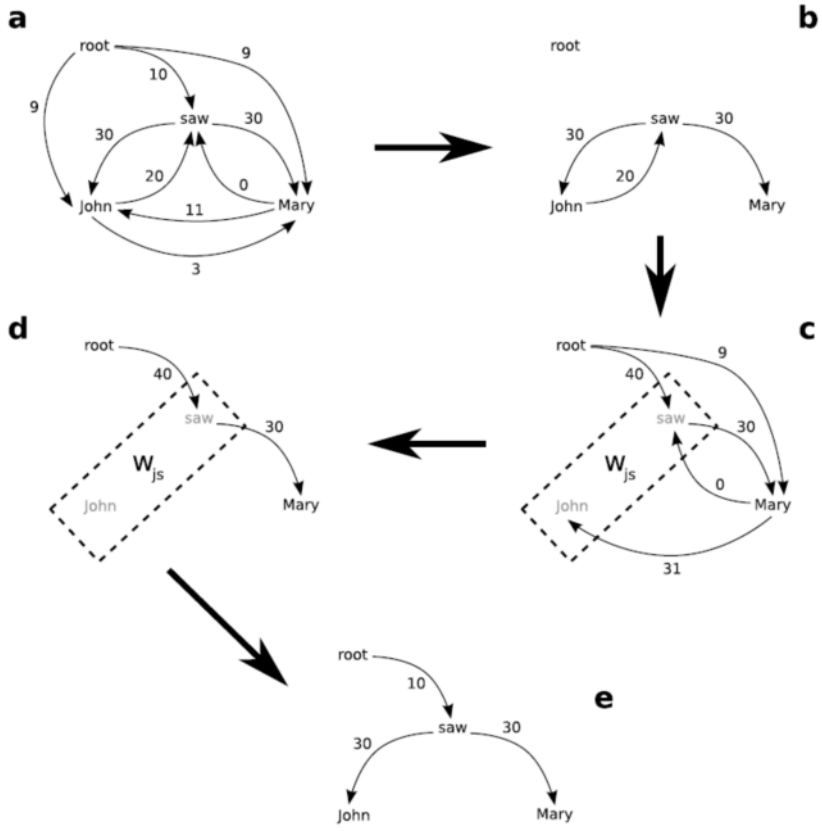
h(S, Γ, λ) – non-projective
Sentence $S = w_0 w_1 \ldots w_n$
Arc parameters $\lambda_{(w_i, w_j)} \in \lambda$
1 Construct $G_S = (V_S, A_S)$
$V_S = \{w_0, w_1, \ldots, w_n\}$
$A_S = \{(w_i, w_j) \mid \text{for all } w_i, w_j \in V_S\}$
2 return Chu-Liu-Edmonds(G_S, λ)

Chu-Liu-Edmonds(G, λ)
Graph $G = (V, A)$
Arc parameters $\lambda_{(w_i, w_j)} \in \lambda$
1 $A' = \{(w_i, w_j) \mid w_j \in V, w_i = \text{argmax}_{w_i} \lambda_{(w_i, w_j)}\}$
2 $G' = (V, A')$
3 If G' has no cycles, then return G' and quit
4 Find any arc set A_C that is a cycle in G'
5 $< G_C, w_c, ep >= \text{contract}(G', A_C, \lambda)$
6 $G = (A, V) = \text{Chu-Liu-Edmonds}(G_C, \lambda)$
7 For the arc $(w_i, w_c) \in A$ where $ep(w_i, w_c) = w_j$,
identify the arc $(w_k, w_j) \in C$ for some w_k
8 Find all arcs $(w_c, w_l) \in A$
9 $A = A \cup \{(ep(w_c, w_l), w_l)\}_{\text{for all } (w_c, w_l) \in A}$
$\cup A_C \cup \{(w_i, w_j)\} - \{(w_k, w_j)\}$
10 $V = V$
11 return G

contract$(G = (V, A), C, \lambda)$
1 Let G_C be the subgraph of G excluding nodes in C
2 Add a node w_c to G_C representing cycle C
3 For $w_j \in V - C : \exists_{w_i \in C}(w_i, w_j) \in A$
Add arc (w_c, w_j) to G_C with
$ep(w_c, w_j) = \text{argmax}_{w_i \in C} \lambda_{(w_i, w_j)}$
$w_i = ep(w_c, w_j)$
$\lambda_{(w_c, w_j)} = \lambda_{(w_i, w_j)}$
4 For $w_i \in V - C : \exists_{w_j \in C}(w_i, w_j) \in A$
Add arc (w_i, w_c) to G_C with
$ep(w_i, w_c) = \text{argmax}_{w_j \in C} \left[\lambda_{(w_i, w_j)} - \lambda_{(a(w_j), w_j)}\right]$
$w_j = ep(w_i, w_c)$
$\lambda_{(w_i, w_c)} = \left[\lambda_{(w_i, w_j)} - \lambda_{(a(w_j), w_j)} + \text{score}(C)\right]$
where $a(w)$ is the predecessor of w in C
and $\text{score}(C) = \sum_{w \in C} \lambda_{(a(w), w)}$
5 return $< G_C, w_c, ep >$

شکل ۱۴ الگوریتم چو-لیو-ادموندز برای یافتن درخت پوشای بیشینه [2]

در شکل ۱۵ نمونه‌ای از اجرای الگوریتم چو-لیو-ادموندز به نمایش گذاشته شده است. نخستین قدم در اجرای این الگوریتم پیدا کردن یال‌های وارد شونده به هر گره با بیش‌ترین وزن است. اگر پس از انتخاب یال‌های با وزن بیشینه درخت تشکیل شد، الگوریتم به پایان می‌رسد. در غیر این صورت، تا زمانی که نتیجهٔ حاصل از الگوریتم درخت نباشد، فراخوانی‌های بازگشتی به الگوریتم ادامه می‌یابد [2].



شکل ۱۵ نمونه‌ای از اجرای الگوریتم چو-لیو-ادموندز بر روی یک جملهٔ انگلیسی [2]

۴-۳-۴. الگوریتم تجزیهٔ افکنشی

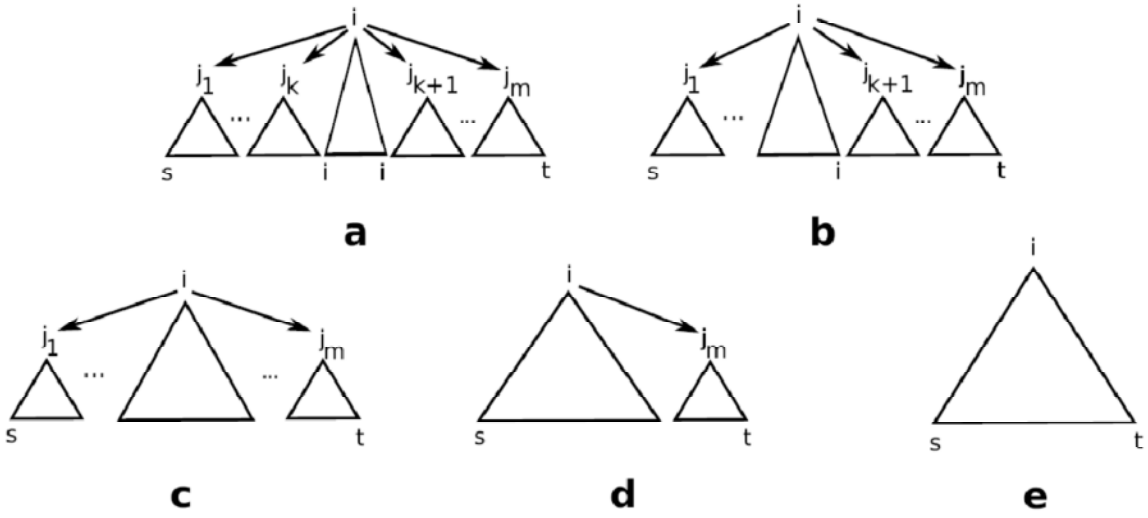
ساختار افکنشی ارتباط زیادی به دستور زبان مستقل از متن دارد. لذا برای تجزیهٔ افکنشی می‌توان از الگوریتم‌های تجزیهٔ زبان‌های مستقل از متن استفاده کرد [2]. یکی از معروف‌ترین الگوریتم‌های تجزیهٔ زبان‌های مستقل از متن الگوریتم کوک-کوزامی-یانگر (سی کِی وای)[۱] [29] است. در این الگوریتم از روش‌های برنامه‌ریزی پویا استفاده می‌شود. در نتیجه $C[s][t][i]$ نشان‌دهندهٔ پرامتیازترین زیردرخت پوشایی است که گره‌های w_s تا w_t را پوشش می‌دهد و دارای ریشهٔ w_i است به طوری که $s \leq i \leq t$ باشد. در نتیجه برای همهٔ iها $C[i][i][i] = 0$ خواهد بود. با دو فرض می‌توان این مسأله را حل کرد. نخست این که هر گراف پوشا از زیرگراف‌های پوشای مجاور هم تشکیل شده است و درخت‌های بزرگ‌تر حاصل از اتصال پشت سر هم زیردرخت‌های مجاور هستند. نمونه‌ای از این حالت در شکل ۱۶ به نمایش گذاشته شده است. در نتیجه برای ساختن زیردرختی که w_s تا w_t را می‌پوشاند، نیاز به پیدا کردن دو

[۱] معادل فارسی عبارت انگلیسی *Cocke-Kasami-Younger (CKY)*

زیردرخت مجاور با بیش‌ترین امتیاز ممکن است [2]. در رابطهٔ (۴-۷) این مسأله به صورت رابطهٔ ریاضی بیان شده است.

$$C[s][t][i] = max_{s \le q < t, s \le j \le t} \begin{cases} C[s][q][i] + C[q+1][t][j] + \lambda_{(w_i, w_j)} & if\ j > i \\ C[s][q][j] + C[q+1][t][i] + \lambda_{(w_i, w_j)} & if\ i > j \end{cases} \quad (۴-۷)$$

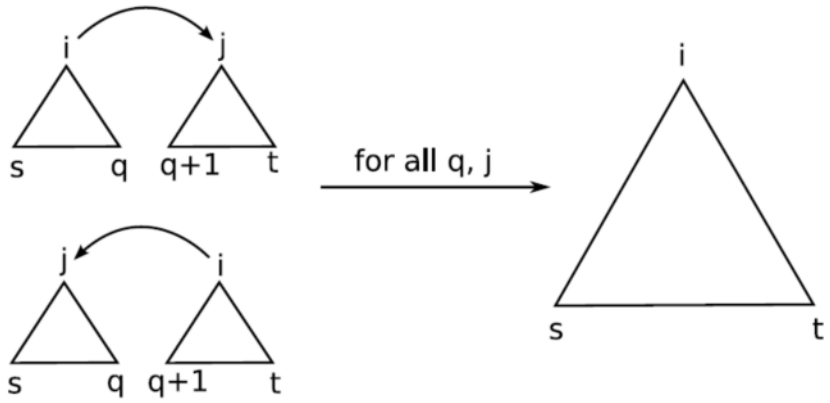
از آن‌جایی که در این الگوریتم برای جدول داده‌ها $O(n^3)$ ورودی وجود دارد و برای هر ورودی $O(n^2)$ امکان وجود دارد، پیچیدگی زمانی این الگوریتم برابر با $O(n^5)$ خواهد بود که با نظر داشتن فرآیند کاهش برچسب‌ها برابر با $O(|R|n^2 + n^5)$ می‌شود [2].



شکل ۱۶ نمونه‌ای از ادغام زیرگراف‌های مجاور برای ساخت درخت وابستگی [2]

در این الگوریتم تنها اشاره به روش پیدا کردن پاسخ بهینه اشاره شده و چیزی در مورد استخراج درخت بهینه گفته نشده است. شاید راحت‌ترین راه برای این کار استفاده از جدول کمکی برای نگهداری اطلاعات یال‌ها است. در (۴-۸) نحوهٔ پر کردن این جدول به صورت ریاضی بیان شده است [2].

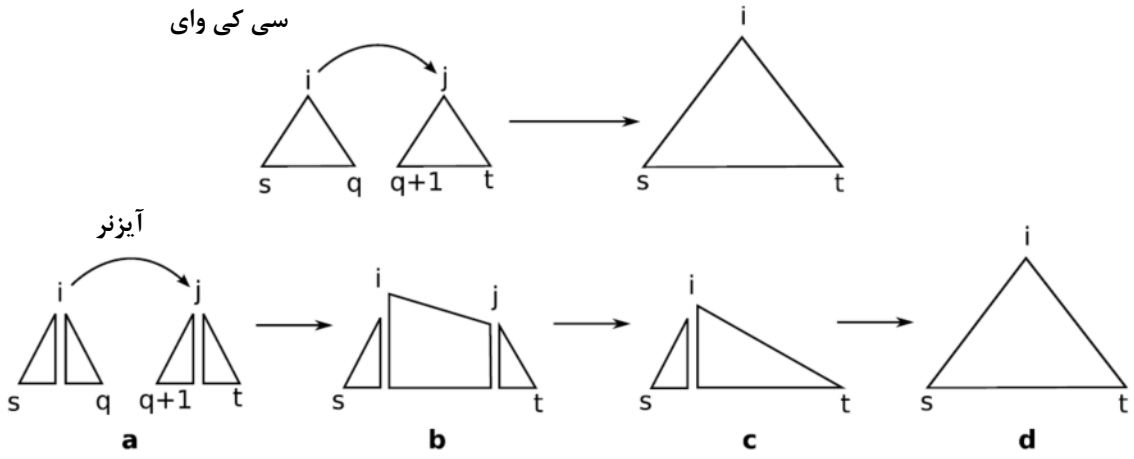
$$A[s][t][i] = \begin{cases} A[s][q][i] \cup A[q+1][t][j] \cup (w_i, w_j) & if\ j > i \\ A[s][q][j] \cup A[q+1][t][i] \cup (w_i, w_j) & if\ i > j \end{cases} \quad (۴-۸)$$



شکل ۱۷ فرآیند الگوریتم سی کی وای [2]

- **الگوریتم آیزنر**

یکی از راه‌های کم‌کردن پیچیدگی این الگوریتم استفاده از اشاره‌گر انتها و ابتدای زیرگراف‌ها با استفاده از الگوریتم ویتربی[۱] [30] است. در این صورت پیچیدگی مسأله به $O(n^3)$ کاهش می‌یابد. آیزنر[۲] [31] راهی ساده‌تر برای کم کردن پیچیدگی الگوریتم ارائه کرد. این الگوریتم بر این اساس است که روند انتخاب وابسته‌های راست یک واژه مستقل از روند انتخاب وابسته‌های راست واژه است. در شکل ۱۸ این الگوریتم و الگوریتم سی کی وای نشان داده شده است.

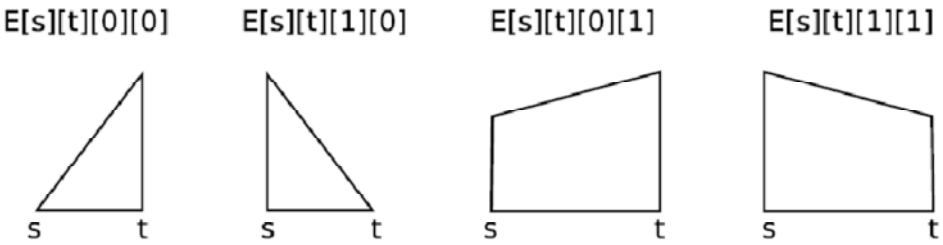


شکل ۱۸ مقایسهٔ روند الگوریتم سی کی وای و آیزنر [2]

[۱] معادل فارسی عبارت انگلیسی *Viterbi*

[۲] معادل فارسی نام *Eisner*

در شکل ۱۹ انواع زیرگراف آیزنر را به صورت $E[s][t][d][c]$ به نمایش گذاشته شده است. اگر d برابر با ۱ باشد، بدین معنی است که واژهٔ سر w_t و اگر برابر با صفر باشد، واژهٔ سر w_s است. اگر c برابر با ۱ باشد بدین معنی خواهد بود که زیرگراف تنها محدود به زیرگراف‌های چپ یا راست وابسته به یال قبلی خواهد بود؛ در غیر این صورت (یعنی حالتی که زیرگراف مربوط به زیرگراف چپ و راست به طور توأمان باشد)، c برابر صفر است [2].



شکل ۱۹ انواع زیرگراف در الگوریتم آیزنر [2]

پیچیدگی زمانی الگوریتم آیزنر برابر با $O(n^3)$ است. با یک اثبات سادهٔ ساختاری می‌توان نشان داد که در این الگوریتم همهٔ زیرگراف‌های ممکن مورد بررسی قرار خواهد گرفت [2].

۴-۴. یادگیری الگوهای مبتنی بر یال

همان‌گونه که در روش‌های مبتنی بر داده نیاز به تعریف شاخص‌های الگوی یادگیری است، در روش‌های مبتنی بر یال نیز این نیاز وجود دارد. در این بخش به بررسی روش‌های یادگیری با استفاده از روش مبتنی بر یال از روی پیکرهٔ وابستگی نشانه‌گذاری‌شده پرداخته شده است.

۴-۴-۱. نمایش شاخص‌ها و ویژگی‌ها

امروزه در روش‌های مبتنی بر گراف، فرض بر این است که شاخص یال‌ها با استفاده از رده‌بندها خطی از هم جداپذیر هستند. یعنی بر هر یال با بردار ویژگی‌های از پیش‌تعریف‌شده، تابع خطی به صورت $\mathbf{f}(w_i, r, w_j) \in \mathbb{R}^m$ و وزن مربوط به آن $\mathbf{w} \in \mathbb{R}^m$ تعریف می‌شود [2]:

$$\boldsymbol{\lambda}_{(w_i,r,w_j)} = \boldsymbol{w}.\boldsymbol{f}(w_i, r, w_j) \quad (۹-۴)$$

Eisner$(S, \mathbb{\Gamma}, o)$

Sentence S= $w_0 w_1 \dots w_n$

Arc weight Parameters $\lambda_{(w_i, w_j)} \in \boldsymbol{\lambda}$

1 *Instantiate* $E[n][n][2][2] \in \mathbb{R}$

2 *Initialization* $E[s][s][d][c] = 0.0 \ \ for\ all\ s, d, c$

3 *for m=1:n*

4 *for s=1:n*

5 *t=s+m;*

6 *if t>n then break;*

% Create subgraphs with c = 1 by adding arcs

7 $E[s][t][0][1] = \max_{s \le q < t}(E[s][q][1][0] + E[q+1][t][0][0] + \lambda_{(w_t, w_s)})$

8 $E[s][t][1][1] = \max_{s \le q < t}\big(E[s][q][1][0] + E[q+1][t][0][0] + \lambda_{(w_s, w_t)}\big)$

% Add corresponding left/right subgraphs

9 $E[s][t][0][0] = \max_{s \le q < t}(E[s][q][0][0] + E[q+1][t][0][1])$

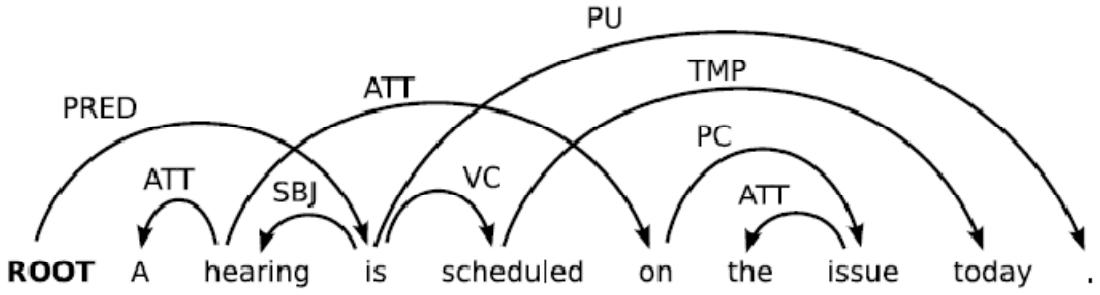
10 $E[s][t][1][0] = \max_{s < q \le t}(E[s][q][1][1] + E[q+1][t][1][0])$

شبه‌برنامه ۲ الگوریتم آیزنر [2]

به عنوان مثال در شکل ۲۰ برای یال $(hearing, ATT, on)$ ویژگی‌های ذیل قابل نمایش است [2]:

- شناسهٔ w_i: $hearing$
- شناسهٔ w_j: on
- شناسهٔ برچسب اجزای سخن w_i: NN
- شناسهٔ برچسب اجزای سخن w_j: IN
- شناسهٔ r: ATT
- شناسهٔ برچسب اجزای سخن بین w_j و w_j: VBZ، VBN
- شناسهٔ برچسب اجزای سخن w_{i-1}: DT
- شناسهٔ برچسب اجزای سخن w_{i+1}: VBZ
- شناسهٔ برچسب اجزای سخن w_{j-1}: VBN
- شناسهٔ برچسب اجزای سخن w_{j+1}: DT
- فاصلهٔ واژه‌ای بین w_j و w_j: ۲

- سمت یال: راست



شکل ۲۰ درخت غیرافکنشی بر یک جملهٔ انگلیسی [2]

۴-۴-۲. داده‌های آموزشی

در همهٔ روش‌ها دادهٔ آموزشی اولیه به صورت رابطهٔ (۴-۱۰) است:

$$D = \{(S_d, G_d)\}_{d=0}^{|D|} \quad (۱۰-۴)$$

(۴-۱۰)از آن‌جایی که در روش‌های مبتنی بر یال نیازی به اطلاعات به صورت دیگر نیست، داده‌های آموزشی به همین صورت برای یادگیری قابل استفاده هستند [2].

۴-۴-۳. یادگیری شاخص‌ها

معمول‌ترین روش برای یادگیری شاخص‌ها در تجزیهٔ مبتنی بر یال، استفاده از یادگیری مبتنی بر استنتاج[۱] است. با این فرض که رده‌بند مورد نظر به صورت خطی جداپذیر است، می‌توان مسألهٔ استنتاج را به صورت رابطهٔ (۴-۱۱) نوشت.

$$h\left(S, \mathbb{\Gamma}, \boldsymbol{\lambda}\right) = \quad (۱۱-۴)$$

$$argmax_{G=(V,A)\in\mathcal{G}_S} \sum_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)} = argmax_{G=(V,A)\in\mathcal{G}_S} \sum_{(w_i,r,w_j)\in A} \boldsymbol{w}.\boldsymbol{f}(w_i, r, w_j)$$

در این صورت در $\boldsymbol{\lambda}$ علاوه بر تابع ویژگی اطلاعات، ضریب وزن $\boldsymbol{w}$ وجود دارد.

ساده‌ترین الگوریتم رده‌بندی خطی الگوریتم شبکهٔ عصبی پرسپترون[۲] است. اگر واقعاً مسأله به صورت خطی جداپذیر باشد، با استفاده از این الگوریتم جواب همیشه درست خواهد بود. البته الگوریتم‌های

---

[۱] معادل فارسی عبارت انگلیسی *Inference-Based Learning*

[۲] معادل فارسی واژهٔ انگلیسی *Perceptron*

دیگری نیز وجود دارد که در این الگوریتم‌ها برای جمع کردن امتیاز زیردرخت‌ها زمان محاسباتی بسیار زیادی صرف می‌شود که نسبت به الگوریتم چو-لیو-ادموندز بسیار کند هستند [2].

Perceptron(D)

$Training\ Data: D = \{(S_d, G_d)\}_{d=0}^{|D|}$

1 $\mathbf{w} = 0$

2 $for\ n = 1:N$

3 $for\ d = 1:|D|$

4 $Let\ \acute{G} = h\left(S_d, \mathbb{\Gamma}, \boldsymbol{\lambda}\right) = argmax_{\acute{G} \in \mathcal{G}_{s_d}} \sum_{(w_i, r, w_j) \in \acute{A}} \boldsymbol{w}.\boldsymbol{f}\left(w_i, r, w_j\right)$

5 $If\ (G \neq \acute{G}_d)$

6 $\mathbf{w} = \mathbf{w} + \sum_{(w_i, r, w_j) \in A_d} \boldsymbol{f}\left(w_i, r, w_j\right) - \sum_{(w_i, r, w_j) \in \acute{A}} \boldsymbol{f}\left(w_i, r, w_j\right)$

7 ***return*** **w**

شبه‌برنامه ۳ الگوریتم پرسپترون در تجزیهٔ مبتنی بر گراف [2]

۴-۵. روش‌های دیگر

روش مبتنی بر یال از نظر سادگی و استقلال از زبان، بهترین روش ممکن در روش‌های مبتنی بر گراف است. ولی مشکلی که وجود دارد این است که معیار استقلال به هیچ وجه در دنیای واقعی در زبان طبیعی وجود ندارد. در این بخش بر روش مبتنی بر یال دو نوع تعمیم ساده معرفی می‌شود تا با استفاده از این دو تعمیم بتوان از روش مبتنی بر یال استفاده کرد.

۴-۵-۱. استفاده از مرتبهٔ وابستگی واژه‌ها

منظور از مرتبهٔ وابستگی[۱] برای واژهٔ w_i در جمله، تعداد وابسته‌های واژه در درخت وابستگی صحیح است [2]. با استفاده از مرتبهٔ وابستگی می‌توان اطلاعات جالبی را به دست آورد. به عنوان مثال، فعل حداقل یک وابسته دارد و علائم نگارشی وابسته‌ای ندارند. در نتیجه در رابطهٔ (۴-۴) احتمال داشتن مرتبهٔ a_{w_i} برای هر واژهٔ به صورت $\lambda_{a_{w_i}}$ اضافه شده، به صورت رابطهٔ (۴-۱۲) خواهد بود [2].

[۱] معادل فارسی واژهٔ انگلیسی *Arity*

$$Score(g) = \sum_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)} + \sum_{w_i\in V} \lambda_{a_{w_i}} \quad (۱۲-۴)$$

در این صورت در ساختن درخت علاوه بر در نظر داشتن امتیاز ساخته شدن هر یال، باید هم‌زمان احتمال هر مرتبهٔ وابستگی را برای هر واژه در نظر داشت. در این صورت برای درخت‌های غیرافکنشی این مسأله به صورت نمایی غیرقطعی کامل[۱] خواهد شد. در این صورت نیاز به پیدا کردن حداقل یک مسیر جهت‌دار که همهٔ گره‌ها را می‌پوشاند، وجود دارد. این مسأله همان مسألهٔ مسیر همیلتونی[۲] است. البته برای یک جملهٔ ورودی $S = w_0 v_1 \dots v_n$ و همهٔ $v_i \in V$ و واژهٔ جدید w_0، مسأله قابل کاهش است. در این صورت خواهیم داشت $\lambda_{(w_0,-,v_i)} = 0$ و $\lambda_{(v_i,-,v_j)} = 0$ اگر و تنها اگر $(v_i, v_j) \in A$، $\lambda_{a_{w_0=1}} =$ 0، $\lambda_{a_{v_i=1}} = 0$ و $\lambda_{a_{v_i=0}} = 0$ باشد. بقیهٔ شاخص‌ها به $-\infty$ مقداردهی می‌شوند. اگر و تنها اگر مسیری همیلتونی برای گراف G وجود داشته باشد، بالاترین امتیاز درخت وابستگی برابر صفر خواهد بود. به دلیل این که هر واژهٔ یا احتمال مرتبهٔ حداکثر برابر یک یا $-\infty$ دارد، چنین اتفاقی می‌افتد. از آن‌جایی حتماً واژهٔ w_0 ریشهٔ درخت خواهد بود، در فرآیند یافتن درخت وابستگی این گره حذف و مسیر بازیابی می‌شود. در ضمن، امتیاز هر مسیر همیلتونی صفر است. در این صورت می‌توان مسألهٔ پیدا کردن درخت وابستگی صحیح را ساده‌تر کرد. البته مسائل مطرح‌شده در بدترین حالت هستند و در این بخش در حالت متوسط مورد بررسی قرار نگرفته است [2].

۴-۵-۲. مارکوف‌سازی

تعمیم دیگری که بر روش مبتنی بر یال پیشنهاد شده است، روش مارکوف‌سازی[۳] است. در این روش فرض عمومی بر این است که هر یال وابستگی وابسته به همهٔ یال‌های وابستگی در یک درخت وابستگی است. با این فرض بدیهی است که بررسی چنین وابستگی از دیدگاه محاسباتی مناسب نیست. در این صورت از فرض مارکوف[۴] بر روی یال‌ها استفاده می‌شود. در این صورت برای یال‌های وابستگی دو نوع مارکوف‌سازی وجود خواهد داشت: عمودی[۵] و افقی[۶] و به این وابستگی‌ها همسایه‌های[۷] عمودی و افقی یال (w_i, r, w_j) می‌گویند. همسایهٔ عمودی شامل همهٔ مسیرها از ریشه به برگ است که از یال

---

[۱] معادل فارسی عبارت انگلیسی *NP-Complete*

[۲] معادل فارسی عبارت انگلیسی *Hamiltonian Path Problem*

[۳] معادل فارسی واژهٔ انگلیسی *Markovization*

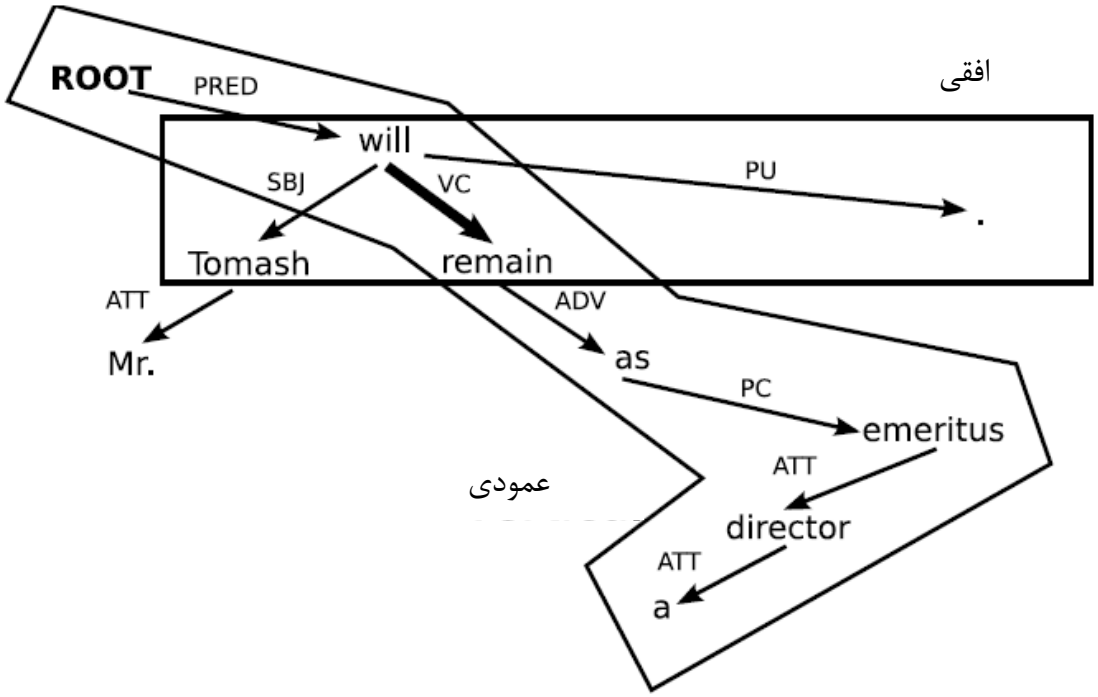
[۴] معادل فارسی عبارت انگلیسی *Markov Assumption*

[۵] معادل فارسی واژهٔ انگلیسی *Vertical*

[۶] معادل فارسی واژهٔ انگلیسی *Horizontal*

[۷] معادل فارسی واژهٔ انگلیسی *Neighborhood*

(w_i, r, w_j) می‌گذرد و همسایهٔ افقی شامل همهٔ یال‌های $(w_i, \acute{r}, \acute{w_j})$ می‌شود [2]. در شکل ۲۱ نمونه‌ای از این همسایگی برای یک یال به نمایش گذاشته شده است.



شکل ۲۱ همسایگی‌های چپ و راست مارکوفی برای درخت وابستگی برای یال (will, VC, remain) [2]

با استفاده از مارکوف‌سازی عمودی و افقی، قابلیت امتیازدهی با در نظر داشتن حوزهٔ وسیعی از یال‌ها وجود خواهد داشت. یک مارکوف‌سازی مرتبهٔ d برای یک یال برابر یک است اگر شامل آن یال و d-1 یال همسایه‌اش باشد. با این وجود نیز حتی مارکوف‌سازی مرتبهٔ دوم برای تجزیهٔ غیرافکنشی بسیار مشکل خواهد بود. به همین دلیل معمولاً در عمل به مارکوف‌سازی با مرتبهٔ بیش‌تر از مرتبهٔ دوم، پرداخته نمی‌شود [2].

در مارکوف‌سازی افقی، اگر $\lambda_{(w_i,r,w_j)(w_i,\acute{r},\acute{w_j})}$ امتیاز دو یال همسایهٔ افقی باشد؛ امتیاز به صورت رابطهٔ (۴-۱۳) تعریف می‌شود [2].

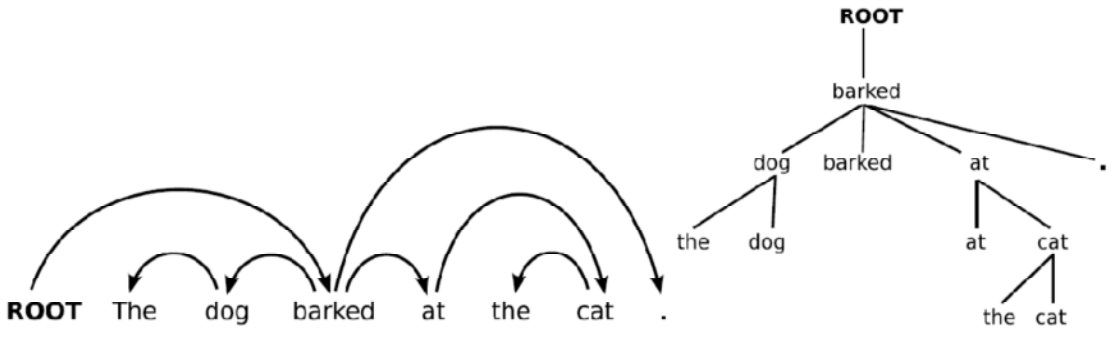
$$Score(g) = \sum_{(w_i,r,w_j),(w_i,\acute{r},\acute{w_j})\in A} \lambda_{(w_i,r,w_j)(w_i,\acute{r},\acute{w_j})} + \sum_{(w_i,r,w_j)\in A} \lambda_{(w_i,r,w_j)} \quad \textbf{(۱۳-۴)}$$

۵. تجزیهٔ مبتنی بر دستور زبان

در تجزیهٔ مبتنی بر دستور زبان وابستگی، در الگوی تجزیهٔ $M = (\mathbb{\Gamma}, \lambda, \mathrm{h})$ برای $\mathbb{\Gamma}$ به جای تعریف مجموعه‌ای از محدودیت‌ها، یک دستور زبان تعریف‌شده استفاده می‌شود. در این صورت عمل تجزیه به معنای تحلیل یک جمله با توجه به دستور زبان موجود و شاخص‌های موجود در $h\left(S, \mathbb{\Gamma}, \boldsymbol{\lambda}\right)$ است. اگر روش به صورت محض مبتنی بر دستور زبان باشد، $\boldsymbol{\lambda}$ مجموعه‌ای تهی خواهد بود. مگر این که برای هر قاعده در دستور زبان احتمالات استخراج شود. در صورتی که تجزیه‌گر نتواند خروجی تولید کند، بدین‌معنا خواهد بود که جملهٔ مورد تجزیه عضو زبان تعریف‌شده نیست [2]. رویکرد دیگری که در تجزیهٔ مبتنی بر دستور زبان وجود دارد، استفاده از روش ارضای محدودیت‌ها است. در این صورت دستور زبان، شامل تعدادی محدودیت است که با استفاده از این محدودیت‌ها تجزیه صورت می‌گیرد [2].

تعریف (۵-۱) دستور زبان $\mathbb{\Gamma}$ از زبان L در تجزیهٔ وابستگی، هر تعریف کامل، محدود و صوری از زبان ممکن است باشد [2].

شاید مهم‌ترین تفاوت روش‌های مبتنی بر داده‌ها با روش مبتنی بر دستور زبان این است که در روش مبتنی بر داده‌ها در بسیاری از موارد برای جملات غیرمعیار و نادرست در فرآیند تجزیه خروجی تولید می‌شود ولی در روش مبتنی بر دستور زبان اگر جمله‌ای عضو زبان نباشد، این امکان وجود نخواهد داشت که درخت تجزیه‌ای برای این جمله تولید شود [2].



شکل ۲۲ نمونه‌ای از یک درخت وابستگی افکنشی و معادل مستقل از متن آن [2]

۵-۲. دستور زبان وابستگی مستقل از متن

این نظریه نخستین بار از سوی گِیف‌من [32] و هِیز [33] در دههٔ شصت میلادی مطرح شده است. برای دستور زبان افکنشی می‌توان ساختار را به صورت دستور زبان مستقل از متن نشان داد. در این صورت درخت حاصل از دستور زبان مستقل از متن یک درخت حاصل بود که نمادهای غیرپایانی‌اش[۱] شامل واژه‌ها می‌شود [2]. در شکل ۲۲ نمونه‌ای از یک درخت وابستگی افکنشی و معادل درخت تجزیهٔ مستقل از متن آن به نمایش گذاشته شده است.

تعریف (۵-۱) یک دستور زبان مستقل از متن $\mathbb{\Gamma}$ به صورت چهارتایی مرتب $(N, \sum, \prod, start)$ است؛ به طوری که [2]:

N مجموعه‌ای محدود از نمادهای غیرپایانی است؛

$\sum$ مجموعه‌ای محدود از نمادهای پایانی است؛

$\prod$ مجموعه‌ای قواعد تولید زبان به شکل $X \to \alpha$ است، به طوری که $X \in N$ یک نماد غیرپایانی و α رشته‌ای از نمادهای پایانی و غیرپایانی است؛ و

$start \in N$ نماد آغازین است.

یکی از حسن‌های استفاده از دستور زبان مستقل از متن، وجود الگوریتم‌های معروف تجزیه مانند سی کی وای [29] و ارلی[۲] [34] است.

تعریف (۵-۲) دستور زبان مستقل از متن دوسویه[۳] $\mathbb{\Gamma}_B$، دستور مستقل از متنی است که $\prod$ شامل مجموعهٔ L از وابسته‌های چپ به صورت $H \to NH\ H$ و مجموعهٔ R از وابسته‌های راست به صورت $H \to H\ NH$ باشد [2].

---

[۱] معادل فارسی عبارت انگلیسی *Non-terminal symbols*

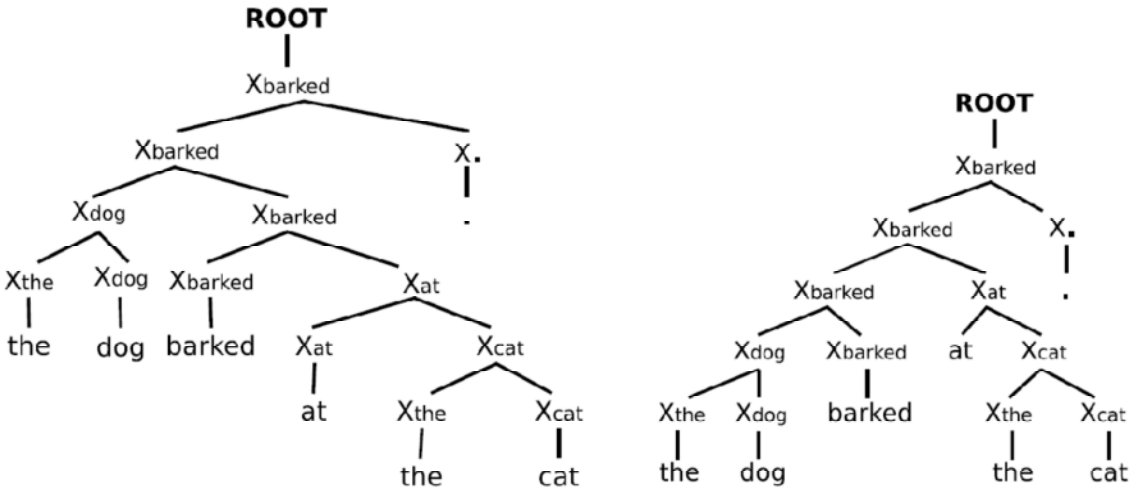
[۲] نوشتار فارسی واژهٔ انگلیسی *Early*

[۳] معادل فارسی واژهٔ انگلیسی *Bilexical*

مهم‌ترین تفاوت موجود بین دستور زبان مستقل از متن دوسویه با دستور زبان مستقل از متن متعارف این است که در دستور دوسویه وابسته‌های چپ مستقل از وابسته‌های راست انتخاب می‌شوند [35].

۵-۲-۲. تجزیه با دستور دوسویه

یکی از مشکلاتی که در تجزیهٔ مبتنی بر نمودار با استفاده از دستور دوسویه وجود دارد، پیچیدگی زمانی بالا برای تبدیل داده‌ها از قالب دستور زبان وابستگی به دستور دوسویه است. دلیل بالا بودن این پیچیدگی این است که در هر واحد برای هر واژه باید سر بودن واژه مورد بررسی قرار بگیرد. به همین دلیل پیچیدگی زمانی به صورت $O(n^3.\min(|\prod|,n^2)$ خواهد بود. از آن‌جایی که $|\prod|$ از n^2 بیش‌تر است، پیچیدگی زمانی به اندازهٔ $O(n^5)$ است. در عین حال در حین تغییر ممکن است چند نوع درخت متفاوت تولید شود [2]. به عنوان مثال در شکل ۲۳ دو نوع درخت وابستگی دوسویه برای یک جمله قابل تولید است.



شکل ۲۳ دو نوع درخت وابستگی دوسویهٔ ممکن برای یک جملهٔ انگلیسی [2]

برای اجتناب از پیچیدگی زمانی بالا و امکان ساخت چند درخت برای یک جمله، از نمایش انشعابِ سر[۱] استفاده می‌شود. در این نمایش هر واژهٔ پایانی w_i از درخت وابستگی به صورت دو واژهٔ پایانی w_i^l و w_i^r نمایش داده می‌شود. در این صورت همهٔ وابسته‌های چپ واژهٔ w_i به w_i^l و همهٔ وابسته‌های راست آن به w_i^r وصل خواهند شد [2]:

[۱] معادل فارسی عبارت انگلیسی *Split-Head*

$$R_i^r \rightarrow w_i^r, L_i^l \rightarrow w_i^l \quad (۲-۵)$$

در این صورت برای ساخت یک واژهٔ پایانی قانون $X_i \rightarrow L_i^l R_i^r$ اضافه خواهد شد. علاوه بر این به دو قانون دیگر برای نشان دادن وابستگی چپ و راست نیاز داریم:

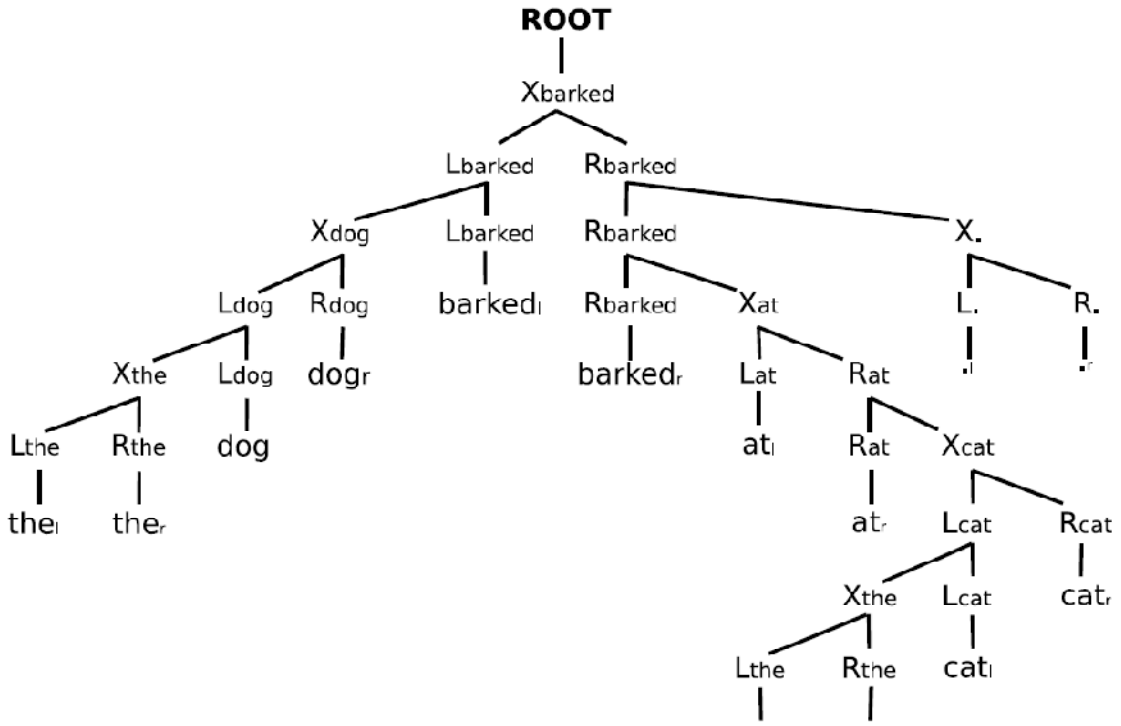
$$L_i \rightarrow X_j L_i \quad (۳-۵)$$

$$R_i \rightarrow R_i X_j \quad (۴-۵)$$

در نهایت نیز نیاز به قانونی برای ساخت ریشه وجود دارد:

$$ROOT \rightarrow X_i \quad (۵-۵)$$

در نتیجه جملهٔ نشان‌داده‌شده در شکل ۲۲ را می‌توان با استفاده از نمایش انشعاب سر، به صورت شکل ۲۴ نشان داد.



شکل ۲۴ درخت وابستگی با نمایش انشعاب سر برای جملهٔ شکل ۲۲ [2]

با استفاده از نمایش انشعاب سر پیچیدگی تبدیل داده‌ها به $O(n^4)$ کاهش می‌یابد که باز هم پیچیدگی زمانی بسیار زیادی است. به همین خاطر از تبدیل گشودن- تا کردن[۱] استفاده می‌شود. این روش از

[۱] معادل فارسی عبارت انگلیسی *Unfold-Fold Transformation*

برنامه‌ریزی کارکردی[1] اقتباس شده است [36, 37]. در این روش به جای X_i معادلش یعنی $L_j^l R_j^r$ جای‌گذاری می‌شود:

$$L_i \rightarrow L_j^l R_j^r L_i \quad (۶-۵)$$

$$R_i \rightarrow R_i L_j^l R_j^r \quad (۷-۵)$$

حال با تعریف نماد $M_{i,j}$ قواعد سه‌گانه‌ای برای هر واژهٔ غیرپایانی تعریف می‌شود:

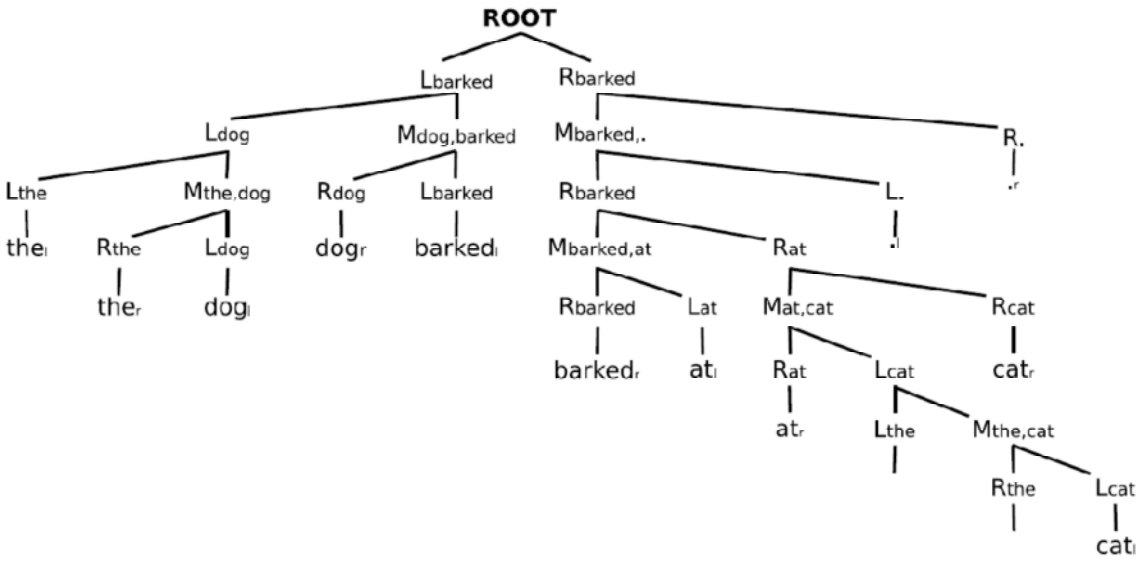
$$L_i \rightarrow L_j M_{i,j} \quad (۸-۵)$$

$$R_i \rightarrow M_{i,j} R_j \quad (۹-۵)$$

$$M_{i,j} \rightarrow R_i L_j \quad (۱۰-۵)$$

در این صورت قاعدهٔ ریشه به صورت رابطهٔ (۵-۱۱) خواهد شد.

$$ROOT \rightarrow L_i R_i \quad (۱۱-۵)$$



شکل ۲۵ درخت وابستگی با روش روش گشودن-تا کردن [2]

با تبدیلاتی که آیزنر [38] پیشنهاد داده بود، پیچیدگی زمانی به $O(n^3)$ کاهش می‌یابد [2]. خلاصهٔ این تبدیلات در جدول ۷ نشان داده شده است.

[1] معادل فارسی عبارت انگلیسی *Functional Programming*

جدول ۷ تبدیلات روش گشودن-تا کردن [2]

i	for $ROOT \rightarrow w_i$
	for $w_j \leftarrow w_i$
	for $w_i \rightarrow w_j$
	for all $i, j \in \sum$

با توجه به این که می‌توان هر دستور زبان وابستگی افکنشی را به صورت دستور زبان مستقل از متن نشان داد، در نتیجه امکان تلفیق این روش با روش‌های مبتنی بر داده وجود دارد. به عنوان مثال می‌توان از دستور زبان مستقل از متن احتمالی[۱] استفاده کرد [2].

۵-۳. دستور وابستگی محدودیت

دستور وابستگی محدودیت[۲] سه‌تایی $\mathbb{\Gamma} = (\sum, R, C)$ است که $\sum$ مجموعه‌ای از نمادهای پایانی (واژه‌ها)، R مجموعهٔ برچسب‌ها و C مجموعهٔ محدودیت‌هاست. محدودیت‌ها وابسته به زبان هستند. در این صورت با یک مسألهٔ ارضای محدودیت روبه‌رو خواهیم بود. در این مسأله با سه نوع محدودیت مواجه هستیم [2]:

۱- مجموعه‌ای از متغیرهای $S = w_0 w_1 \dots w_n$ که نشان‌دهندهٔ مجموعه واژه‌های جمله است؛

۲- دامنهٔ متغیرهای w_i که مجموعهٔ $\{w_j | i \neq j \ \& \ 1 \leq j < n\}$ سرهای ممکن واژه است؛ و

۳- مجموعهٔ C از محدودیت‌ها که با متغیرهای قابل قبول تعریف می‌شود.

از آن‌جایی که مجموعه مسائل ارضای محدودیت جزء مسائل نمایی غیرقطعی کامل هستند، نیاز به استفاده روش‌های ابتکاری در حل آن‌هاست [2].

---

[۱] معادل فارسی عبارت انگلیسی *Probabilistic Context-Free Grammar (PCFG)*

[۲] معادل فارسی عبارت انگلیسی *Constraint Dependency Grammar*

۵-۳-۱. دستور وابستگی محدودیت وزن‌دار

استفاده از محدودیت‌های قطعی و خدشه‌ناپذیر موجب ایجاد اشکال‌هایی در تجزیهٔ صحیح جملات می‌شود. دلیل این امر مسائلی مانند استثنائات دستوری فراوان در زبان‌های طبیعی مخصوصاً در زبان‌های بی‌ترتیب است. در بسیاری از متن‌ها، حتی در متن‌های تصحیح‌شده، خطاهای دستوری ناخواسته وجود دارد. در دستور زبان‌های محدودیت این مسائل در نظر گرفته نشده و تنها دستور زبان‌های صحیح قید شده است. حتی اگر طراح دستور زبان بخواهد در روش‌های مبتنی بر دستور زبان قواعد نادرست را به عنوان دستورهای نادرست قید کند با مشکل مواجه خواهد شد. به همین علت محدودیت‌های *نرم*[۱] یا *فسخ‌شدنی*[۲] به مجموعهٔ قواعد اضافه می‌شود. بنابراین برای هر محدودیت c یک وزن λ_c (وزن صفر سخت‌ترین محدودیت و وزن یک، نرم‌ترین محدودیت است) در نظر گرفته می‌شود. در نتیجه برای هر تجزیه، مقبولیتی به صورت ضرب وزن‌ها در نظر گرفته می‌شود که هر چه بیش‌تر باشد به معنای بهتر بودن مقبولیت است [2].

$$weight\ (G) = \prod_{c \in C} \lambda_c \quad (۱۲-۵)$$

۵-۳-۲. تجزیهٔ وابستگی محدودیت مبتنی بر تبدیل

نتایج حاصل از روش‌های انتشار محدودیت[۳] [39, 40] و محدودیت وزن‌دار جالب است ولی مشکلاتی در این روش‌ها وجود دارد. مشکل اصلی این روش‌ها این است که به دلیل استفاده از جست و جوهای ابتکاری در فضای حالت محدودیت‌ها، پاسخ درست از دست برود. ساز و کاری برای حل این مشکل در نظر گرفته شده است. در این روش در هر مرحله از تجزیهٔ یک جمله با استفاده از وزن محدودیت‌ها فرآیند تجزیه بهبود می‌یابد. در این روش از کم‌ترین وزن (سخت‌ترین محدودیت) اصلاح آغاز می‌شود. در شکل ۲۶ نمونه‌ای از این اصلاحات را برای یک جملهٔ آلمانی وجود دارد. نکتهٔ جالب دربارهٔ این روش *ویژگی هرزمانی*[۴] است؛ یعنی در هر زمان دل‌خواه قابل پایان است [2].

برای طراحی چنین روشی نیاز به پاسخ سه پرسش اساسی است [2]:

۱) کدام گراف وابستگی به عنوان گراف اولیه مشخص شود؟

۲) کدام متغیر بعد از آن تغییر یابد؟

---

[۱] معادل فارسی عبارت انگلیسی *Soft Constraint*

[۲] معادل فارسی عبارت انگلیسی *Defeasible*

[۳] معادل فارسی عبارت انگلیسی *Constraint Propagation*

[۴] معادل فارسی عبارت انگلیسی *Anytime Property*

۳) اگر متغیر w باید تغییر کند، چه مقداری باید بشود؟

تعریف (۵-۳) یک *محدودیت یگانی*[1] محدودیتی است که به واسطهٔ آن دامنهٔ یک متغیر محدود می‌شود [2].

با وجودی که در این روش دقت روش جست و جوی کامل وجود ندارد، ولی فوث [41] نشان داد که ۸۰٪ موارد در زبان آلمانی (به عنوان یک زبان بی‌ترتیب) با استفاده از این روش درست تشخیص داده شده و در ۱۰٪ موارد با اصلاحات به جواب رسیده و در بقیهٔ ۱۰٪ موارد این روش کارآیی لازم را نداشته است.

۶. ارزیابی تجزیهٔ وابستگی

به دلیل کمبود پیکره‌های متنی مخصوص تجزیهٔ وابستگی، یکی از دغدغه‌های اصلی در فرآیند طراحی و ساخت تجزیه‌گرهای وابستگی، تبدیل پیکره‌های متنی موجود (که بر اساس جداسازی واحدهای زبانی هستند) به پیکره‌های وابستگی است. در این بخش به بررسی روش‌های ارزیابی تجزیه‌های وابستگی و تبدیل پیکره‌ها پرداخته می‌شود.

۶-۱. روش‌های ارزیابی

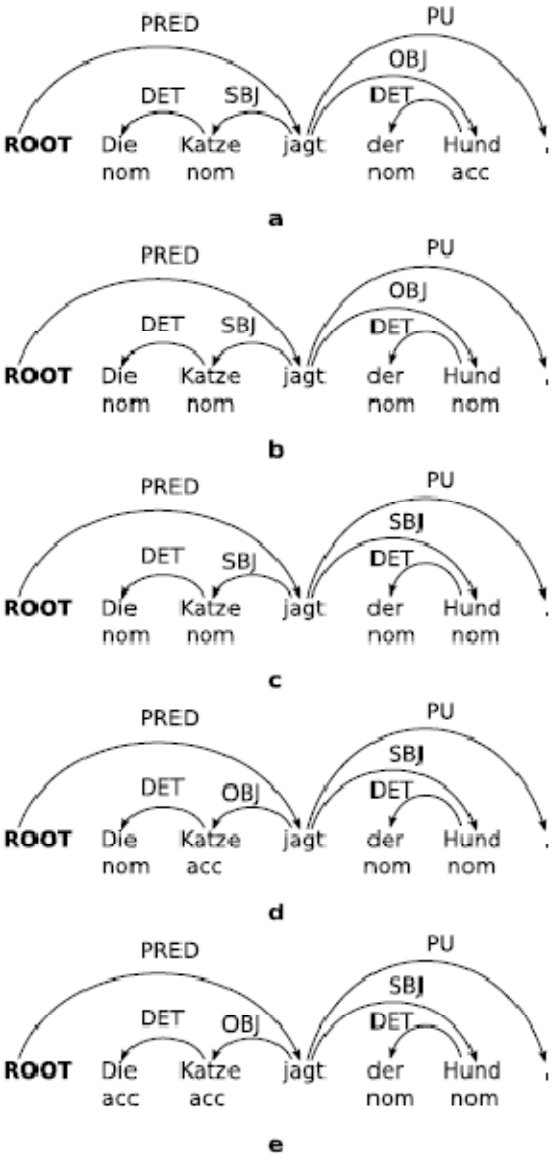
سه نوع امتیاز معروف برای ارزیابی تجزیه‌های وابستگی وجود دارد [2]:

- تطابق کامل: در این روش در صورتی یک تجزیه صحیح است که درخت تجزیه و درخت واقعی دقیقاً به یک شکل باشند.
- امتیاز اتصال[2]: در این روش هر برچسب یا یال به عنوان یک معیار در نظر گرفته می‌شود و بر اساس آن تعداد یال‌های صحیح با برچسب درست معیار صحت خواهد بود. در این صورت این روش شباهت بسیار زیادی به معیارهای برچسب‌زنی اجزای سخن پیدا می‌کند.
- فرخوانی و دقت: در این معیار از سه امتیاز دقت، فراخوانی و سنجهٔ اف[1] استفاده می‌شود.

---

[1] معادل فارسی عبارت انگلیسی *Unary Constraint*

[2] معادل فارسی عبارت انگلیسی *Attachment Score*



شکل ۲۶ فرآیند رسیدن از کمینهٔ محلی (بالاترین درخت) به کمینهٔ سراسری (پایین‌ترین درخت) برای یک جملهٔ آلمانی [41]

پرکاربردترین معیارها امتیاز اتصال برچسب‌دار[۲] و بدون برچسب[۳] هستند [2]. این امتیازها خود بر دو نوع مبتنی بر واژه و مبتنی بر جمله هستند. در روش مبتنی بر واژه تعداد برچسب‌ها درست تقسیم بر تعداد

---

[۱] معادل فارسی عبارت انگلیسی *F-Measure*

[۲] معادل فارسی عبارت انگلیسی *Labeled Attachment Score (LAS)*

[۳] معادل فارسی عبارت انگلیسی *Unlabeled Attachment Score (UAS)*

کل واژه‌های دادهٔ آموزشی می‌شود ولی در روش مبتنی بر جمله نخست در هر جمله، امتیاز مبتنی بر واژهٔ استخراج شده، سپس میانگین این امتیازها به عنوان امتیاز نهایی محسوب می‌شود [2].

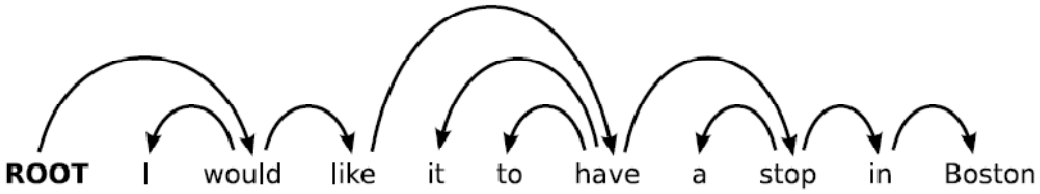
۶-۲. استفاده از وابستگی در روش‌های ارزیابی تجزیهٔ نحوی

چند مشکل اساسی در روش‌های ارزیابی تجزیهٔ نحوی وجود دارد. یک مشکل در مقایسهٔ روش‌ها با هم است. به عنوان مثال، مفهوم دقت در روش‌های مبتنی بر واحدها با روش تجزیهٔ وابستگی متفاوت است. مشکل دیگر در مقایسه‌های معمول، در نظر نگرفتن سطح خطا در روش‌های تجزیهٔ عمیق است. یعنی باید گره‌های بالایی درخت تجزیه از امتیاز بیش‌تری برخوردار باشند [42]. یکی از راه‌های ممکن برای ارزیابی این روش‌ها و حل مشکلات روش‌های مرسوم استفاده از نمایش تجزیه‌ها به صورت ساختار وابستگی است [2, 42]. کروچ و همکارانش [43] نشان دادند که فرآیند تبدیل خروجی تجزیه‌های مختلف به ساختار وابستگی با خطای ۲٪ صورت می‌گیرد.

۶-۳. تبدیل پیکره‌های مبتنی بر واحدها به پیکره‌های وابستگی

یکی از مشکلاتی که در تبدیل پیکره‌های مبتنی بر واحدها وجود دارد، عدم صراحت در برخی از رابطه‌ها است [2]. در شکل ۲۷ نمونه‌ای از این تبدیل نشان داده شده است.

```
( (S
    (NP-SBJ I/PRP-HD )
    (VP-HD would/MD-HD
      (VP like/VB-HD
         (S
            (NP-SBJ it/PRP )
            (VP-HD to/TO
               (VP-HD have/VB-HD
                 (NP
                    (NP-HD a/DT stop/NN-HD )
                    (PP-LOC in/IN-HD
                       (NP Boston/NNP-HD ))))))))))
```



شکل ۲۷ نمونه‌ای از تبدیل درخت مبتنی بر واحد به درخت وابستگی [2]

۶-۴. نتایج ارزیابی

در سال‌های ۲۰۰۶ و ۲۰۰۷ کار مشترکی[۱] در همایش سالانهٔ یادگیری رایانه‌ای زبان طبیعی[۲] زیر نظر انجمن زبان‌شناسی رایانه‌ای[۳] بر روی تجزیهٔ وابستگی بر روی پیکره‌های متنی زبان‌های مختلف انجام شده است. بهترین نتایج به دست آمده در این همایش در جدول ۸ به نمایش گذاشته شده است.

[۱] معادل فارسی عبارت انگلیسی *Shared Task*

[۲] معادل فارسی عبارت انگلیسی *Conference on Computational Natural Language Learning (CoNLL)*

[۳] معادل فارسی عبارت انگلیسی *Association of Computational Linguistics (ACL)*

جدول ۸ بهترین نتایج کار مشترک همایش یادگیری رایانه‌ای زبان طبیعی ۲۰۰۷ [44]

زبان	عربی	باسکی	کاتالان	چینی	چکی	انگلیسی	یونانی	مجاری	ایتالیایی	ترکی
امتیاز اتصال برچسب‌دار	۷۶/۵۲	۷۶/۹۴	۸۸/۷۰	۸۴/۶۹	۸۰/۱۹	۸۹/۶۱	۷۶/۳۱	۸۰/۲۷	۸۴/۴۰	۷۹/۸۱
امتیاز اتصال بدون برچسب	۸۶/۰۹	۸۲/۸۴	۹۳/۴۰	۸۸/۹۴	۸۶/۲۸	۹۰/۶۳	۸۴/۰۸	۸۳/۵۵	۸۷/۹۱	۸۶/۲۲

۷. مقایسهٔ روش‌های تجزیهٔ وابستگی

در حالت کلی دو نوع روش تجزیهٔ وابستگی وجود دارد: مبتنی بر داده و مبتنی بر دستور زبان. در روش‌های مبتنی بر داده نیز دو نوع روش کلی وجود دارد: مبتنی بر گذار و مبتنی بر گراف. در این بخش نخست دو روش مبتنی بر گذار و مبتنی بر گراف با هم مقایسه شده، ارتباط‌ها و تفاوت‌های‌شان مورد بررسی اجمالی قرار می‌گیرد و سپس روش‌های مبتنی بر داده و مبتنی بر دستور زبان با هم مقایسه می‌شوند.

۷-۱. مقایسهٔ روش‌های مبتنی بر گذار و مبتنی بر گراف

در مقایسهٔ این دو روش باید تعادلی بین دقت تجزیه و نحوهٔ نمایش ویژگی‌ها وجود داشته باشد. در روش مبتنی بر گذار دقت تجزیه کم‌تر ولی نمایش ویژگی‌ها دقیق‌تر است و هم‌چنین در روش مبتنی بر گذار فرآیند جست و جو با وجود ساختار حریصانه ممکن است به بهینهٔ سراسری همگرا نشود [2].

مک‌دونالد[۱] و نیوِر [46] نتایج حاصل دو روش مبتنی بر گراف و مبتنی بر گذار از کار مشترک همایش یادگیری رایانه‌ای زبان طبیعی ۲۰۰۶ را با هم مقایسه کردند و به نتایج جالبی رسیدند. این نتایج در جدول ۹ آمده است. از نظر آن‌ها با وجود نتایج به ظاهر برابر این دو روش در سطوح مختلف، این دو روش دارای دقت‌های مختلفی هستند. در سطح ریشه، روش مبتنی بر گذار دارای دقت بالاتری است. از دیدگاه برچسب، روش مبتنی بر گذار برای برچسب اسم و ضمیر و روش مبتنی بر گراف در بقیهٔ برچسب‌ها دارای دقت بالاتری است. در مجموع دقت و فراخوانی در روش مبتنی بر گراف بیش‌تر است ولی در روش مبتنی بر گذار دقت بازیابی مقادیر اسمی مانند فاعل بالاتر است.

[۱] نوشتار فارسی نام *McDonald*

جدول ۹ مقایسهٔ روش‌های مبتنی بر گذار و مبتنی بر گراف در کار مشترک همایش یادگیری رایانه‌ای زبان طبیعی ۲۰۰۶ [45]

زبان	عربی	بلغاری	چینی	چکی	دانمارکی	هلندی	آلمانی	ژاپنی	پرتغالی	اسلوونیایی
روش مبتنی بر گذار	۶۶/۷۱	۸۷/۴۱	۸۶/۹۲	۷۸/۴۲	۸۴/۷۷	۷۸/۵۹	۸۵/۸۲	۹۱/۶۵	۸۷/۶۰	۷۰/۳۰
روش مبتنی بر گراف	۶۶/۹۱	۸۷/۵۷	۸۵/۹۰	۸۰/۱۸	۸۴/۷۹	۷۹/۱۹	۸۷/۳۴	۹۰/۷۱	۸۶/۸۲	۷۳/۴۴

زبان	اسپانیایی	سوئدی	ترکی	میانگین
روش مبتنی بر گذار	۸۱/۲۹	۸۴/۵۸	۶۵/۶۸	۸۰/۷۵
روش مبتنی بر گراف	۸۲/۲۵	۸۲/۵۵	۶۳/۱۹	۸۰/۸۳

یکی از نگره‌های جدید برای توفق بر اشکالات این دو روش، تلفیق این دو روش است [47-50]. یکی از روش‌ها، استفاده از خروجی روش تجزیهٔ مبتنی بر گذار به عنوان ورودی روش مبتنی بر گراف است. به چنین روشی تجزیه‌گر پشته‌ای[۱] یا رده‌بند پشته‌ای[۲] گویند [48].

۷-۲. مقایسهٔ روش‌های مبتنی بر داده و مبتنی بر دستور زبان

همان طور که در بخش‌های قبل در مورد روش‌های مبتنی بر داده و مبتنی بر دستور زبان توضیح داده شد، این روش‌ها از جنبه‌های مختلف شباهت‌های بسیاری با هم دارند. به عنوان مثال روش مبتنی بر گراف و روش محدودیت وزن‌دار از جهاتی با هم شبیه هستند. همان‌طور که الگوریتم تجزیهٔ سی کی وای در تجزیهٔ مبتنی بر داده قابل استفاده است، در روش مبتنی بر دستور زبان نیز استفاده می‌شود. از این دست مثال‌های بسیار زیادی را می‌توان ذکر کرد که گویای شباهت‌های روش‌های مبتنی بر داده و مبتنی بر دستور زبان باشد.

[۱] معادل فارسی عبارت انگلیسی *Stacked Parser*

[۲] معادل فارسی عبارت انگلیسی *Stacked Classifier*

۸. منابع و تجزیه‌گرهای موجود

در این بخش معرفی اجمالی از تجزیه‌گرهای موجود و منابع در دسترس ارائه خواهد شد و به روش‌هایی که آن‌ها استفاده کرده‌اند به صورت کوتاه پرداخته می‌شود.

۸-۱. تجزیه‌گرها

برخی از تجزیه‌گرهایی که قابل آموزش هستند، به شرح ذیل است[۱]:

- **تجزیه‌گر وابستگی احتمالی جیسون آیزنر**[۲]: این تجزیه‌گر بر مبنای تجزیهٔ مبتنی بر دستور زبان پایین به بالا که در بخش ۵-۲ معرفی شده است، کار می‌کند. این تجزیه‌گر به زبان لیسپ[۳] نوشته شده است.
- **تجزیه‌گر درخت پوشای بیشنیه**[۴]: این ابزار پیاده‌سازی روش مبتنی بر گراف است که به زبان جاوا نوشته شده است.
- **تجزیه‌گر مالت**[۵]: در این تجزیه‌گر که به زبان جاوا نوشته شده است، همهٔ روش‌های معروف مبتنی بر گذار پیاده‌سازی شده است.
- **تجزیه‌گر درخت پوشای بیشنیه بهترین k حالت**[۶]: این تجزیه‌گر بر مبنای تلفیق روش مبتنی بر یال و بیشینهٔ به‌هم‌ریختگی[۷] است.
- **تجزیه‌گر واین**[۸]: پیاده‌سازی روش تلفیقی وابستگی احتمالی با رتبه‌بندی تبعیضی دوبارهٔ کمینهٔ مخاطره[۹] و تجزیهٔ احتمالی [51] است که به پیاده‌سازی داینا[۱۰] و سی‌پلاس‌پلاس آن موجود است.

---

[۱] به دلیل متغیر بودن وب‌گاه‌های این ابزارها از گذاشتن نشانی وب آن‌ها صرف نظر شده است، ولی با جست و جوی ساده‌ای در شبکل اینترنت این ابزارها قابل دسترس هستند.

[۲] نوشتار فارسی نام *Jason Eisner*

[۳] نوشتار فارسی واژهٔ انگلیسی *LISP*

[۴] نوشتار فارسی عبارت انگلیسی *MST Parser*

[۵] نوشتار فارسی عبارت انگلیسی *MALT Parser*

[۶] معادل فارسی عبارت انگلیسی *K-Best Maximum Spanning Tree Dependency Parser*

[۷] معادل فارسی عبارت انگلیسی *Maximum Entropy*

[۸] نوشتار فارسی عبارت انگلیسی *Vine Parser*

[۹] معادل فارسی عبارت انگلیسی *Discriminative Minimum Risk Reranker*

[۱۰] نوشتار فارسی واژهٔ انگلیسی *Dyna*

- **تجزیه‌گر شبکهٔ باور حلقوی افزایشی**[1] [52, 53]: این تجزیه‌گر بر مبنای روش احتمالی با استفاده از شبکهٔ باور حلقوی افزایشی با پیاده‌سازی در زبان سی است.

علاوه بر تجزیه‌گرهای مورد اشاره، تجزیه‌گرهای خاص منظورهٔ دیگری هم وجود دارند که مستقل از زبان و قابل یادگیری برای زبان‌های دیگر نیستند، لذا از ذکر نام آن‌ها خودداری شده است.

۸-۲. دادگان درختی

پیکره‌های مختلفی برای زبان‌های مختلف وجود دارد که از میان آن‌ها می‌توان پیکرهٔ درخت وابستگی عربی پراگ[2] [54] اشاره کرد. برای زبان فارسی تاکنون هیچ پیکره‌ای که در دسترس و یا خصوصی و قابل خرید باشد، وجود ندارد.

۹. مراجع

[1] S. Steele, "Word order variation: a typological survey," *Universals of human language*, pp. 585-623: Stanford University Press, 1978.

[2] S. Kübler, R. McDonald, and J. Nivre, *Dependency Parsing*: Morgan & Claypool, 2009.

[۳] ا. طبیب‌زاده، ظرفیت فعل و ساختهای بنیادین جمله در فارسی امروز: نشر مرکز، ۱۳۸۵.

[4] A. Bharati, V. Chaitanya, and R. Sangal, *Natural Language Processing: A Paninian Perspective*, New Delhi: Prentice Hall of India, 1994.

[5] L. Tesnière, *Esquisse d'une Syntaxe structurale*, Paris: Klincksieck, 1953.

[6] L. Tesnière, *Éléments de syntaxe structurale*: Editions Klincksieck, 1959.

[7] U. Engel, *Kurze Grammatik der deutschen Sparche*, München: Iudicium Verlge, 2002.

[8] "The American Heritage® Dictionary of the English Language," Houghton Mifflin Company, 2009.

[9] G. Helbig, and W. Schenkel, *Wörterbuch zur Valenz und Distribution deutscher Verben*: Niemeyer, Tübingen, 1991.

[10] L. Tesnière, *Grundgzüge der Strukturalen Syntax*, Stuttgart: Klettcotta, 1980.

[11] D. J. Allerton, *Valency and the English Verb*, London: Academic Press, 1982.

[1] معادل فارسی عبارت انگلیسی *Incremental Sigmoid Belief Network (ISBN) Parser*

[2] معادل فارسی عبارت انگلیسی *Prague Arabic Dependency Treebank*

[۱۲] م. باطنی، ز. احمدی‌نیا، ف. محمدی و دیگران، "فرهنگ معاصر پویا انگلیسی - فارسی،" فرهنگ معاصر پویا انگلیسی - فارسی، فرهنگ معاصر، ۱۳۸۷.

[۱۳] م. شمس‌فرد، "پردازش متون فارسی: دستاوردهای گذشته، چالشهای پیش رو،" دومین کارگاه پژوهشی زبان فارسی و رایانه، ۱۳۸۴.

[14] I. Mel'čuk, *Dependency Syntax: Theory and Practice*: State University of New York Press, 1988.

[15] R. A. Hudson, *English Word Grammar*: Blackwell, 1990.

[16] J. Nivre, "Algorithms for deterministic incremental dependency parsing," *Computational Linguistics,* vol. 34, no. 4, pp. 513-553, 2008.

[17] W. Daelemans, and A. Van den Bosch, *Memory-Based Language Processing*: Cambridge University Press, 2005.

[18] J. Nivre, J. Hall, and J. Nilsson, "Memory-Based Dependency Parsing," in Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL), Boston, Massachusetts, 2004, pp. 49-56.

[19] V. N. Vapnik, *The Nature of Statistical Learning Theory*: Springer, 1995.

[20] H. Yamada, and Y. Matsumoto, "Statistical dependency analysis with support vector machines," in Proceedings of the 8th International Workshop on Parsing Technologies (IWPT), Nancy, France, 2003, pp. 195-206.

[21] J. Nivre, "An efficient algorithm for projective dependency parsing ",in Proceedings of the 8th InternationalWorkshop on Parsing Technologies (IWPT), Nancy, France, 2003, pp. 149-160.

[22] G. Attardi, "Experiments with a multilanguage non-projective dependency parser," in Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL), New York, USA, 2006, pp. 166-170.

[23] M. A. Covington, "A Fundamental Algorithm for Dependency Parsing," in Proceedings of the 39th Annual ACMSoutheast Conference, Athens, GA, 2001, pp. 95-102.

[24] A. B. J. H. E. H .B. Hladká, "The Prague Dependency Treebank: A three-level annotation scenario," *Treebanks: Building and Using Parsed Corpora*, pp. 103-127, 2003.

[25] J. Nivre, and J. Nilsson, "Pseudo-projective dependency parsing," in Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL), Ann Arbor, MI, 2005, pp. 99-106.

[26] Y. J. Chu, and T. H. Liu, "On the shortest arborescence of a directed graph," *Science Sinica,* vol. 14, pp. 1396-1400, 1965.

[27] J. Edmonds, "Optimum branchings," *Journal of Research of the National Bureau of Standards,* vol. 71, no. B, pp. 233-240, 1967.

[28] R. E. Tarjan, "Finding optimum branchings," *Networks,* vol. 7, pp. 25-37, 1977.

[29] D. H. Younger, "Recognition and parsing of context-free languages in time n^3," *Information and Control,* vol. 10, pp. 189-208, 1967.

[30] A. J. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE Transactions on Information Theory,* vol. 13, no. 2, pp. 260-269, 1967.

[31] J .M. Eisner, “Three New Probabilistic Models for Dependency Parsing: An Exploration,” in Proceedings of COLING 1996, 1996.

[32] H. Gaifman, “Dependency systems and phrase-structure systems,” *Information and Control,* vol. 8, pp. 304-337, 1965.

[33] D. G. Hays, “Dependency theory:Aformalism and some observations,” *Language,* vol. 40, pp. 511-525, 1964.

[34] J. Early, “An efficient context-free parsing algorithm,” *Communications of the ACM,* vol. 13, pp. 94-102, 1970.

[35] J. Nivre, *Two models of stochastic dependency grammar*, Växjö University, School of Mathematics and Systems Engineering, 2002.

[36] J. Eisner, and J. Blatz, “Program transformations for optimization of parsing algorithms and other weighted logic programs,” in Proceedings of the 11th Conference on Formal Grammar, Dublin, Ireland, 2007, pp. 45-85.

[37] M. Johnson, “Transforming projective bilexical dependency grammars into efficiently parseable CFGs with unfold-fold,” in Proceeding of the 45th Annual Meeting of the Association of Computational Linguistics, Prague, Czech Republic, 2007, pp. 168-175.

[38] J. Eisner, "Bilexical grammars and their cubic-time parsing algorithms," *Advances in Probabilistic and Other Parsing Technologies*, H. Bunt and A. Nijholt, eds., pp. 29-62: Kluwer, 2000.

[39] D. Duchier, “Axiomatizing dependency parsing using set constraints,” in Proceedings of the Sixth Meeting on Mathematics of Language, Orlando, FL, 1999, pp. 115-126.

[40] D. Duchier, and R. Debusmann, “Topological dependency trees: A constraint-based account of linear precedence,” in Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL) and the 10th Conference of the European Chapter of the ACL (EACL), Toulouse, France, 2001, pp. 180-187.

[41] K. Foth, “Tranformationsbasiertes Constraint-Parsing,” Diplomarbeit, Universität Hamburg., 1999.

[42] D. Lin, “A dependency-based method for evaluating broad-coverage parsers,” in Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI), Montreal, Canada, 1995, pp. 1420-1425.

[43] R. Crouch, R. M. Kaplan, T. H. King *et al.*, “A comparison of evaluation metrics for a broad coverage stochastic parser,” in Proceedings of the LREC Workshop on the Evaluation of Parsing Systems, Las Palmas, Gran Canaria, 2002, pp. 67-74.

[44] J. Nivre, J. Hall, S. Kübler *et al.*, “The CoNLL 2007 Shared Task on Dependency Parsing,” in Proceeding of CoNLL 2007, New York, USA, 2007.

[45] S. Buchholz, and E. Marsi, “CoNLL-X shared task on multilingual dependency parsing “,in Proceeding of the Tenth Conf. on Computational Natural Language Learning (CoNLL), 2006.

[46] R. McDonald, and J. Nivre, “Characterizing the errors of data-driven dependency parsing models,” in Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), Praque, Czech Republic 2007, pp. 122-131.

[47] K. Sagae, and A. Lavie, “Parser combination by reparsing,” in Proceedings of the Human Language Technology Conference of the NAACL, Main Conference, New York, NY, 2006, pp. 125-132.

[48] A. F. T. Martins, D. Das, N. A. Smith *et al.*, “Stacking dependency parsers,” in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Honolulu, Hawaii, 2008.

[49] R. McDonald, and J. Nivre, “Integrating graph-based and transition-based dependency parsers,” in Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL), Columbus, OH, 2008.

[50] Y. Zhang, and S. Clark, “A tale of two parsers: investigating and combining graph-based and transition-based dependency parsing using beam-search,” in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Honolulu, Hawaii, 2008.

[51] J. Eisner, and N. Smith, “Parsing with soft and hard constraints on dependency length,” in Proceedings of the 9th International Workshop on Parsing Technologies (IWPT), Vancouver, Canada, 2005, pp. 30-41.

[52] I. Titov, and J. Henderson, “Fast and robust multilingual dependency parsing with a generative latent variable model,” in Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007, Prague, Czech Republic, 2007, pp. 947-951.

[53] I. Titov, and J. Henderson, “A latent variable model for generative dependency parsing,” in Proceedings of the 10th International Conference on Parsing Technologies (IWPT), Prague, Czech Republic, 2007, pp. 145-155.

[54] J. Hajič, O. Smrž, P. Zemánek *et al.*, “Prague Arabic Dependency Treebank: Development in data and tools,” in Proceedings of the NEMLAR 2004 International Conference on Arabic Language Resources and Tools, Cairo, Egypt, 2004.