دانشکده مهندسی کامپیوتر

بررسی روش‌های تجزیه در دستور وابستگی

سیمنار کارشناسی ارشد

در رشته مهندسی کامپیوتر - گرایش هوش مصنوعی و رباتیک

نام دانشجو

مجتبی خلاش

استاد راهنما:

دکتر بهروز مینایی بیدگلی

آبان ماه ۱۳۹۰

دانشکده مهندسی کامپیوتر

بررسی روش‌های تجزیه در دستور وابستگی

سیمنار کارشناسی ارشد

در رشته مهندسی کامپیوتر - گرایش هوش مصنوعی و رباتیک

نام دانشجو

مجتبی خلاش

استاد راهنما:

دکتر بهروز مینایی بیدگلی

آبان ماه ۱۳۹۰


بسم الله الرحمن الرحيم

تشکر و قدردانی

در این جا لازم می‌دانم که از زحمات و حمایت‌های همیشگی آقای محمد صادق رسولی که ساعت‌ها از وقت با ارزش خود را صرف کمک و راهنمایی بنده نمودند، تشکر نمایم و برای ایشان آرزوی موفقیت در تمام مراحل زندگی را دارم.

چکیده

روش‌های مبتنی بر وابستگی برای تجزیه نحوی در سال‌های اخیر در حوزه پردازش زبان طبیعی بسیار رایج شده‌اند. در این سمینار مقدمه‌ای بر روش‌های رایج خواهیم پرداخت. بعد از معرفی دستور وابستگی و تجزیه وابستگی، که به بررسی خصوصیات رسمی مسئله تجزیه وابستگی پرداخته می‌شود، به سه دسته اصلی از الگوهای تجزیه جاری یعنی الگوهای مبتنی بر گذار، مبتنی بر گراف و مبتنی بر دستور می‌پردازیم. در ادامه به مقایسه روش‌های مختلف تجزیه وابستگی پرداخته خواهد شد.

واژه‌های کلیدی: تجزیهٔ وابستگی، درخت وابستگی، تجزیهٔ مبتنی بر گذار، تجزیهٔ مبتنی بر گراف

فهرست مطالب

عنوان **صفحه**

فهرست شکل‌ها

عنوان **صفحه**

فهرست جدول‌ها

عنوان **صفحه**

فصل ۱:

مقدمه

۱-۱- شرح مسأله

زبان‌شناسی[1] علمی است که به مطالعه و بررسی نظام‌مند زبان می‌پردازد. از دیدگاه چامسکی اصلی‌ترین مسئله در زبان‌شناسی فهم زبان است. یکی از پیچیدگی‌های فهم زبان، وابستگی اجزای زبانی به بافت[2] است.

با فراگیر شدن استفاده از رایانه، تمایل به استفاده از رایانه برای فهم زبان‌های طبیعی انسانی به وجود آمد که این امر منجر به ایجاد زمینهٔ تازه‌ای تحت عنوان زبان‌شناسی رایانه‌ای[3] یا پردازش زبان طبیعی[4] شد. در این حوزه فهم زبان معادل تجزیهٔ[5] زبان است. در تجزیه نحوی ارتباط بین واژه‌های جمله یافت می‌شود.

دستور وابستگی یکی از نظریه‌های زبان‌شناختی است که در بررسی نحو و دستور زبان به کار می‌رود. این نظریه در اروپا و به ویژه در فرانسه، آلمان و روسیه مورد استقبال زبان‌شناسان قرار گرفت و به تدریج مبدل به یکی از مهم‌ترین نظریه‌های نحو در کار تدوین دستور زبان‌های گوناگون و نیز آموزش زبان به خارجی‌ها شد. تجزیهٔ وابستگی نیز یک الگوی زبانی مبتنی بر دستور وابستگی برای تجزیه و تحلیل خودکار جملات است. تجزیهٔ وابستگی در حوزه‌های مختلفی مانند «استخراج روابط»، «ترجمهٔ ماشینی»، «تولید واژه‌های مترادف»، «تقویت منابع واژگانی» کاربرد دارد.

۱-۲- ساختار فصل‌های آینده

در فصل دوم ابتدا به تعاریف موجود در این حوزه مانند دستور وابستگی، تجزیهٔ وابستگی، درخت وابستگی می‌پردازیم، سپس مقدمه‌ای بر انواع روش‌های موجود در تجزیهٔ وابستگی ارائه خواهد شد و بررسی جزئیات این روش‌های به فصل سوم موکول خواهد شد. در فصل سوم ضمن بررسی دقیق‌تر روش‌های موجود، ابزارهای موجود برای تجزیهٔ وابستگی نیز معرفی می‌شوند. در فصل چهارم ضمن جمع‌بندی مباحث مطرح شده در فصول گذشته، نقاط ضعف این روش‌ها مطرح شده، تا کارهای آینده خود را بر این اساس جهت‌دهی نماییم.

---

[1] Linquistic
[2] Context
[3] Computational linguistics
[4] Natural Language Processing (NLP)
[5] Parsing

فصل ۲:

تعاریف و مفاهیم مبنایی

۲-۱- مقدمه

در این فصل سعی شده با تعاریف موجود در حوزهٔ تجزیهٔ وابستگی آشنایی پیدا کرده و پس از آشنایی اولیه مقدمات تجزیهٔ وابستگی مطرح شود اما بررسی دقیق و همراه با جزئیات به فصل آینده موکول شده است.

در ابتدای این فصل تلاش شده جایگاه تجزیهٔ وابستگی را در تجزیهٔ نحوی مشخص کنیم. برای این منظور به معرفی مکاتب دستور نویسی پرداخته و تفاوت‌های آن‌ها ذکر شده است. سپس وارد موضوع تجزیهٔ وابستگی شده و مفاهیم مقدماتی که در بخش‌های بعدی مورد نیاز است، شرح داده شده است. در ادامه فصل، دسته‌بندی‌های موجود برای روش‌های تجزیهٔ وابستگی بیان شده و به بررسی اجمالی آن‌ها از جنبه الگوها و الگوریتم‌های موجود پرداخته شده است. در پایان این فصل روش‌های مختلف ارزیابی نتایج مطرح شده است.

۲-۲- مکتب‌های دستورنویسی

دو مکتب «دستور وابستگی»[1] و «دستور زایشی»[2] وجود دارد. این دو دستور به رغم شباهت‌های بسیاری که با هم دارند، غالباً به عنوان دو نظریهٔ رقیب و اساساً متفاوت در نظر گرفته می‌شوند. در ادامه به برخی تفاوت‌های این دو نظریه می‌پردازیم.

- جایگاه فعل: جایگاه یا ارزشی که هر کدام از این دو نظریه برای فاعل قائل هستند، متفاوت است:
  - دستور زایشی: از آغاز پیدایش به تبعیت از منطق ارسطویی که جمله را به دو قسمت نهاد (موضوع) و گزاره (محمول) تقسیم می‌کرد، فاعل را در کنار فعل به عنوان یکی از دو رکن اصلی جمله در نظر گرفته و در اولین گام تحلیل نحوی، جمله را به دو قسمت اصلی گروه اسمی یا همان نهاد و گروه فعلی یا همان گزاره تقسیم کرده است. (جمله = گروه اسمی + گروه فعلی)
  - دستور وابستگی: تجزیهٔ جمله به دو قسمت نهاد و گزاره پیش از آنکه مبین ساخت نحوی جمله باشد، ساخت اطلاعاتی و توزیع اطلاعات کهنه و نو را در جمله بازنمایی می‌کند. تجزیهٔ جمله به دو قسمت نهاد و گزاره شیوه مناسبی برای بررسی ساخت اطلاعاتی جمله

[1] Dependency grammar
[2] Generative grammar

است اما برای تجزیه، ساخت نحوی جمله باید تحلیل خود را از فعل (یعنی مرکز ثقل جمله) آغاز کرد. برای اولین بار دستور وابستگی دو مفهوم کهن نهاد و گزاره را که از دیرباز حاکم بر غالب دیدگاه‌ها و نظریه‌های نحوی بود را کاملاً کنار گذاشته است که صرفاً بر اساس قرارداد شکل گرفته‌اند و ربطی به واقعیت ساختاری زبان ندارند.

- تحلیل سلسله مراتبی جمله: هر دو نظریه می‌کوشند ساخت سلسله مراتبی جمله را بر اساس پیوندهای گوناگونی که میان اجزاء جمله وجود دارند، نمایش دهند.
  - دستور زایشی: تحلیل نحوی جمله را از کل (یعنی خود جمله) آغاز می‌کند. گروه اسمی یا نهاد از نظر ارزش سلسله مراتبی، همسطح گروه فعل یا گزاره در نظر گرفته می‌شوند. تمام پیوندها در دستور زایشی مبین رابطه کل و جزء است.
  - دستور وابستگی: تحلیل نحوی جمله را از جزء (یعنی از فعل) آغاز می‌کند. مفاهیم نهاد و گزاره یکسره کنار گذاشته می‌شود و فعل در ساختی بالاتر از فاعل قرار می‌گیرد. فعل تعیین می‌کند که در هر جمله چه وابسته‌هایی مثلاً فاعل یا انواع مفعول و غیره می‌تواند یا باید وجود داشته باشد. تمام روابط در دستور وابستگی به شکل رابطه حاکم (عنصر بالایی) و وابسته (عنصر پایینی) در می‌آید.
- هسته فعلی (فعل مرکزی): یکی از تفاوت‌های اصلی این دو مکتب نحوه تلقی و تعریف هسته فعلی جملات است.
  - دستور زایشی: فعلی که در مطابقت با فاعل به سر می‌برد.
  - دستور وابستگی: فعلی که نوع و تعداد وابسته‌های جمله را تعیین می‌کند.
- انواع و تعداد وابسته‌های فعل:
  - دستور زایشی: در مجموع تنها دو وابسته (متمم – افزوده) را در نظر می‌گیرد.
  - دستور وابستگی: تقسیم بندی دوگانه دستور زایشی را کافی نمی‌داند و وابسته‌های فعل را به سه دسته تقسیم می‌کند. ابتدا به دو دسته کلی متمم‌ها و افزوده‌ها تقسیم کرده و سپس متمم‌ها را به دو دسته اختیاری و اجباری تقسیم می‌کند.

دستور وابستگی مانند دستور زایشی دارای مکاتب و شاخه‌های گوناگونی است اما اولاً تعداد شاخه‌های دستور وابستگی بسیار کم‌تر از انشعابات متعدد دستور زایشی است، ثانیاً اختلافات میان آن‌ها بسیار ناچیز و غالباً محدود به مسائل ریز و جزئی می‌باشد [۱].

۲-۳- دستور وابستگی

دستور وابستگی[1] مکتب دستور نویسی است که یک نظریه ساخت‌گرا[2] و صورت‌گرا[3] می‌باشد. این دستور در اساس از طریق بررسی روابط وابستگی بین عناصر هسته و وابسته زبان به توصیف ساخت‌های نحوی در زبان‌های گوناگون می‌پردازد. نخستین بار لوسین تنی‌یر[4] فرانسوی مبانی این نظریه را در کتاب کم حجمی با عنوان «گفتارهایی در نحو ساختاری» (۱۹۵۳) معرفی کرد اما شرح مبسوط و مفصل این نظریه در سال ۱۹۵۹ و پس از مرگ وی در کتاب دیگر او با عنوان«مبانی نظریه ساختاری» منتشر شد.

طبیب‌زاده [۱] دو فرض زیر را برای دستور وابستگی بیان می‌کند:

۱) هر جمله یک فعل مرکزی دارد.

۲) بر اساس نوع و تعداد متمم‌های اجباری و اختیاری فعل مرکزی، می‌توان ساخت‌های بنیادین جمله‌هایی که این فعل در آن‌ها بکار رفته است را تعیین نمود.

زبان شناسی سنتی دستور وابستگی شامل خانواده بزرگ و نسبتاً متمایزی از تئوری‌های دستوری است که مفروضات پایه‌ای خاصی را در مورد ساختار نحوی به اشتراک می‌گذارند. به طور خاص این فرض که ساختار نحوی شامل عناصر واژگانی[5] است که توسط روابط دودویی نامتقارن به نام «رابطه وابستگی» یا «وابستگی» متصل شدند [۲].

هرگاه دو واژه توسط رابطهٔ وابستگی به یکدیگر متصل شوند، به یکی از آن‌ها واژهٔ سَر[6] (H) و به دیگری وابسته[7] (D) گویند که با یک ارتباط آن‌ها را به هم متصل می‌کند. اصطلاحات دیگری نیز به جای این دو اصطلاح به کار می‌رود. اصطلاح حاکم[8]، رئیس[9] و والد[10] به جای سر و اصطلاح بچه[11] و پیراینده[12] به جای وابسته به کار می‌رود. واژهٔ سَر نقش بزرگ‌تری در تعیین رفتار جفت بازی می‌کند. واژهٔ وابسته از قبل وجود سَر را مفروض دارد اما واژهٔ سَر می‌تواند نیازمند وجود وابسته باشد.

---

[1] Dependency Grammar (DG)
[2] Structuralist
[3] Formalist
[4] Tesnière
[5] Lexical element
[6] Head
[7] Dependent
[8] Governor
[9] Regent
[10] Parent
[11] Child
[12] Modifier

وابسته‌ای که مقدم بر سَر باشد را «وابستهٔ پیشین»[1] و وابسته‌ای که پس از سَر باشد را «وابستهٔ پسین»[2] گویند.

۲-۴- تجزیهٔ وابستگی

تجزیهٔ وابستگی[3] راهی برای تجزیهٔ نحوی زبان طبیعی است که به صورت خودکار به تجزیه و تحلیل ساختار وابستگی جملات پرداخته و برای هر جملهٔ ورودی یک گراف وابستگی ایجاد می‌کند.

۲-۴-۱- ساختار وابستگی

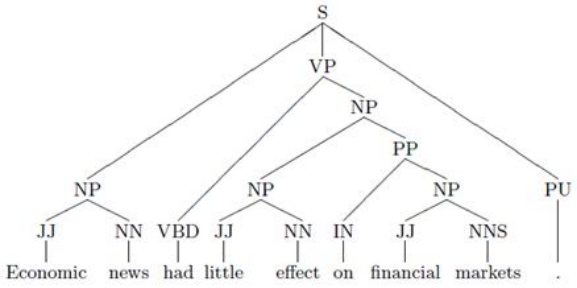
تجزیهٔ جملات طبیعی به انواع خاصی از ساختارها، یکی از موضوعات تحقیقاتی اصلی در پردازش زبان طبیعی است. برای این منظور ساختارهای مختلفی برای بازنمایی نحوی ارائه شدند:

- **ساختار عبارت یا مبتنی بر سازه**

پر مصرف‌ترین ساختار، «ساختار عبارت»[4] یا «مبتنی بر سازه»[5] است که در پیکرهٔ درختی پِن[6] استفاده شده است. این ساختار با یک سازه سطح بالا[7] (نماد S نشان دهنده یک جمله) شروع می‌شود. سپس آن را به سازه‌های کوچک‌تر (نمادهای NP، VP و غیره) می‌شکنند. در پایان این سازه‌ها را دوباره به سازه‌های کوچک‌تر (واژه‌ها) می‌شکنند. به عبارت دیگر، یک جمله به صورت بازگشتی به قسمت‌های کوچک‌تر که به آن عبارت یا سازه گویند، تجزیه می‌شود. شکل (۲-۱) نمونه‌ای از این ساختار برای جمله انتخاب شده از پیکرهٔ درختی پِن است. ساختار عبارت اطلاعات نحوی[8] فراهم می‌آورد که برای وظایف مختلف پردازش نحوی زبان مفید است.

---

[1] Predependent
[2] Postdependent
[3] Dependency parsing
[4] Phrase structure
[5] Constituency-based
[6] Penn Treebank
[7] Top level constituent
[8] Syntactic information



شکل (۲-۱) ساختار عبارت برای یک جملهٔ انگلیسی [۳]

این ساختار از چندین جهت دارای محدودیت است:

۱) این ساختار نسبت به ترتیب واژه‌ها انعطاف‌پذیر نیست. این امر در زبان‌هایی مثل چک و فنلاندی که ترتیب واژه در آن‌ها آزاد[۱] است، مشکل ساز خواهد بود زیرا جملات تنها در ترتیب واژه‌ها فرق دارند تا اینکه در ساختار عبارت متفاوت باشد.

۲) ساختار عبارت وابسته به زبان است، یعنی برای تجزیه جملات در زبان‌های مختلف باید مجموعهٔ قوانین جدیدی ارائه شود.

۳) بسیار مبتنی بر نحو[۲] است، یعنی شامل اطلاعات مفهومی[۳] مهم مانند نقش‌های مفهومی[۴] نیست (البته انواعی از ساختارهای عبارت ارائه شدند که شامل اطلاعات مفهومی نیز هستند).

◘ ساختار وابستگی

برای غلبه بر مشکلات ساختار عبارت، تمرکز روی ساختار دیگری بنام «ساختار وابستگی»[۵] متمایل شده است. در این ساختار بر خلاف ساختار عبارت، گره‌های عبارتی[۶] وجود ندارد. هر گره در این ساختار (به جز گرهٔ ریشه) نشان دهندهٔ واژه‌های جمله است و وابستگی‌ها روابط نحوی و مفهومی گره‌ها را بازنمایی می‌کنند. شکل (۲-۲) ساختار وابستگی برای جمله شکل (۲-۱) است.

---

[۱] 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. مثال) من در مدرسه کتاب را به علی دادم/ من در مدرسه کتاب به علی دادم/ من به علی در مدرسه کتاب را دادم/ من کتاب را به علی در مدرسه دادم.

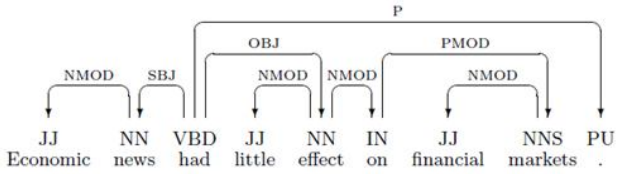
[2] Syntax oriented

[3] Semantic information

[4] Semantic role

[5] Dependency structure

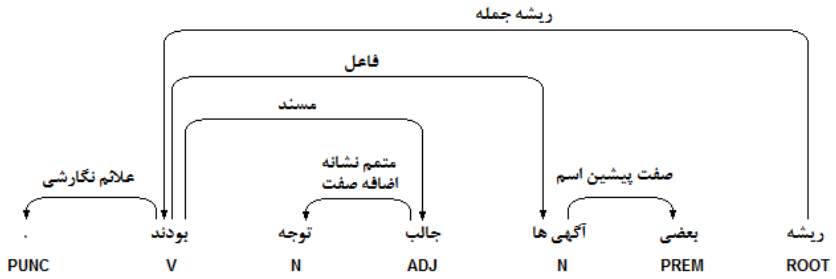
[6] Phrasal node



شکل (۲-۲) ساختار وابستگی برای یک جملهٔ انگلیسی [۳]

ساختار وابستگی ساده‌تر از ساختار عبارت است. این سادگی آن را برای یادگیری ساختارها توسط ماشین و انسان مناسب می‌سازد. علاوه بر این روابط، وابسته به ترتیب واژه‌ها نیست (اگرچه گاهی اوقات ترجیح داده می‌شود که ترتیب واژه‌ها حفظ شود). اکثر روابط وابستگی مثل SBJ و OBJ می‌تواند مستقل از زبان باشند.

برای زبان فارسی هیچ پیکره‌ای که در دسترس و یا خصوصی و قابل خرید باشد، وجود نداشت. اولین پیکرهٔ درختی برای تجزیه وابستگی در زبان فارسی [۴] نیز در حال طراحی است[1]. جملات این پیکره در سبک‌های مختلف (خبری، ادبی، علمی، فرهنگی، تحلیلی و انواع دیگر متن) هستند. در شکل (۲-۳) نمونه‌ای از جملات این پیکره آمده است.



شکل (۲-۳) ساختار وابستگی برای یک جملهٔ فارسی [۴]

۲-۴-۲- گراف وابستگی

در ساختار وابستگی، هر واژه حداکثر وابسته به یک واژهٔ دیگر می‌تواند باشد. این بدان معنی است که این ساختار را می‌توان به صورت گراف جهت‌دار بازنمایی که در آن گره‌ها معادل واژه‌ها و یال‌ها معادل روابط وابستگی هستند. یال‌ها می‌توانند با انواع وابستگی خاصی برچسب‌دار شوند یا بدون برچسب ایجاد شوند. نمادهای زیر را در نظر بگیرید:

[1] http://www.dadegan.ir

- $x = w_1 w_2 \dots w_n$: جملهٔ ورودی که در آن w_i معادل واژهٔ i در جمله است و مرتب شده با رابطهٔ تقدم خطی $<$ است. بعبارت دیگر می‌توان نوشت $w_i < w_j$ تا بیان کنیم که w_i مقدم بر w_j در رشتهٔ x است (یعنی $i<j$).
- $R_x = \{r_1, \dots, r_m\}$: مجموعهٔ تمام روابط وابستگی مجاز در x است.
- (w_i, r, w_j): یال جهت‌دار از واژهٔ سَر w_i به w_j با رابطهٔ وابستگی $r \in R_x$ است. برای هر $w_i \in W$ حداکثر یک یال (w_i, r, w_j) وجود دارد.
- $w_i \rightarrow w_j$: رابطهٔ وابستگی بدون برچسب جهت‌دار از واژهٔ سَر w_i به w_j است.
- $w_i \xrightarrow{r} w_j$: بازنمایی دیگری از (w_i, r, w_j) است.
- $w_i \rightarrow^* w_j$: بستار متعدی بازتابی[1] رابطهٔ وابستگی بدون برچسب که در آن واژهٔ w_i سَر غیر مستقیم واژهٔ w_j است. بنابراین $w_i \rightarrow^* w_j$ است 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.
- $w_i \leftrightarrow w_j$: رابطهٔ وابستگی بدون جهت است. بعبارت دیگر مسیری از w_i به w_j یا برعکس باید وجود باشد. 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.
- $w_i \leftrightarrow^* w_j$: بستار متعدی بازتابی رابطهٔ وابستگی بدون جهت در درخت وابستگی 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.
- V_x: مجموعهٔ کلیهٔ گره‌ها (واژه‌ها) در گراف وابستگی است. معمولاً از یک واژهٔ مصنوعی «ریشه»[2] قبل از اولین حرف جملهٔ استفاده می‌شود که دلیل تکنیکی آن ساده‌سازی محاسبات است. به این صورت هر کلمهٔ واقعی در جملهٔ سَر نحوی دارد.

$$V_x = \{w_0 = \text{root}, w_1, \dots, w_n\}$$

- E_x: مجموعهٔ کلیهٔ یال‌ها (روابط وابستگی) در گراف وابستگی است.

$G_x = (V_x, E_x)$: گراف وابستگی برای جملهٔ ورودی است.

◘ انواع گراف‌های وابستگی

گراف‌های وابستگی را از نظر نوع روابط وابستگی به دو دسته تقسیم می‌کنند:

---

[1] Reflexive transitive closure
[2] ROOT

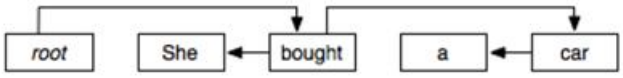
۱) گراف‌های وابستگی افکنشی[1]: اگر تمام واژه‌های جمله در یک خط چیده شوند و گراف وابستگی بدون یال با تقاطع[2] (یال با همپوشانی) باشد، گراف وابستگی را افکنشی یا انعکاسی می‌نامند. تعریف کاوینگتون[3] [۵] از افکنشی به صورت زیر است:

- یک درخت افکنشی است اگر و تنها اگر هر واژه در آن شامل یک زیر رشتهٔ پیوسته باشد.
- یک واژه شامل زیر رشتهٔ پیوسته است اگر و تنها اگر هر دو واژه که شامل آن می‌شود، شامل تمام واژه‌های بینشان باشد.

تعریف نیور[4] [۶] از افکنشی به صورت زیر است:

- یک یال $w_i \rightarrow w_k$ افکنشی است اگر و تنها اگر برای هر واژهٔ w_j رخ داده بین w_i و w_k در رشته ($w_i < w_j < w_k$ یا $w_i > w_j > w_k$) وجود داشته باشد، که رابطهٔ $w_i \rightarrow^* w_j$ موجود باشد.
- یک گراف وابستگی $G_x = (V_x, E_x)$ افکنشی است اگر و تنها اگر هر یال در E_x افکنشی باشد.

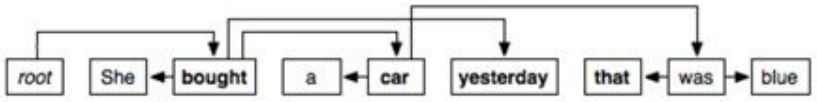
نمونه‌ای از درخت وابستگی افکنشی در شکل (۲-۴) نشان داده شده است.



شکل (۲-۴) مثالی از درخت وابستگی افکنشی [۷]

۲) گراف‌های وابستگی غیرافکنشی[5]: در این گراف‌ها حداقل دو یال که با یکدیگر همپوشانی داشته باشند، وجود دارد. نمونه‌ای از درخت وابستگی غیرافکنشی در شکل (۲-۵) نشان داده شده است. در زبان‌های با ترتیب آزاد واژه‌ها، ویژگی غیرافکنشی پدیدهٔ رایجی است، زیرا محدودیت‌های بالقوه نسبی روی وابستگی‌ها خیلی کمتر انعطاف‌پذیر هستند.

[1] Projective dependency graph
[2] Crossing edge
[3] Covington
[4] Nivre
[5] Non-projective dependency graph



شکل (۲-۵) مثالی از درخت وابستگی غیرافکنشی [۷]

◘ گراف وابستگی خوش‌ساخت

گراف وابستگی $G_x = (V_x, E_x)$ خوش‌ساخت است اگر و تنها اگر شرایط زیر ادا شود:

- ریشهٔ یکتا[۱]: رأس یکتا (ریشه) باید موجود باشد که یال ورودی نداشته باشد (وابسته به رأس دیگر نباشد) و تنها یک یال خروجی داشته باشد.
- برچسب یکتا[۲]: به هر یال در درخت وابستگی باید تنها یک برچسب نسبت داده شود.

$$\left(w_i \xrightarrow{r} w_j \wedge w_i \xrightarrow{r'} w_j\right) \Rightarrow r = r'$$

- تک‌سَری[۳]: کلیهٔ رأس‌ها (بجز ریشه) باید یک یال ورودی داشته باشد. اکثر دستورهای وابستگی یکتایی را در نظر می‌گیرند اما دستورهایی نیز هستند که وجود چندین سَر[۴] را نیز در نظر می‌گیرند.

$$(w_i \rightarrow w_j \wedge w_k \rightarrow w_j) \Rightarrow w_i = w_k$$

- بدون دور بودن[۵]: یک مسیر جهت‌دار بین هر دو رأس نباید یک دور ایجاد کند.

$$\neg(w_i \rightarrow w_j \wedge w_j \rightarrow^* w_i)$$

- همبندی[۶]: برای هر دو رأس باید یک مسیر (صرف نظر از جهت[۷]) موجود باشد.

$$w_i \leftrightarrow^* w_j$$

گاهی اوقات، شرط افکنشی بودن به این شروط اضافه می‌شود.

- افکنشی بودن یا مجاورت[۸]: اگر واژهٔ A وابسته به واژهٔ B باشد، آنگاه تمام واژه‌های بین A و B تبعی[۹] B هستند.

$$(w_i \leftrightarrow w_k \wedge w_i < w_j < w_k) \Rightarrow (w_i \rightarrow^* w_j \vee w_k \rightarrow^* w_j)$$

برخی دستورهای وابستگی، شرط افکنشی بودن را در نظر می‌گیرند و برخی دیگر در نظر نمی‌گیرند.

---

[1] Unique root
[2] Unique label
[3] Single head
[4] Multiple head
[5] Acyclic
[6] Connected
[7] Unidirectional
[8] Adjacency
[9] Subordinate

شرط افکنشی بودن ویژگی گراف وابستگی نیست اما مرتبط با ترتیب خطی واژه‌ها در سطح رشته است. حفظ افکنشی بودن می‌تواند مفید باشد زیرا در این صورت تولید دوباره[1] جملهٔ اصلی از روی گراف وابستگی با حفظ ترتیب واژه‌ها امکان‌پذیر است. در عین حال گاهی اوقات روابط وابستگی غیرافکنشی برای ساخت گراف وابستگی حتی در زبان‌هایی که ترتیب واژه‌ها در آن انعطاف‌ناپذیر[2] است (مثل انگلیسی)، ضروری است. به عنوان مثال در شکل (۲-۵) روابط وابستگی $\text{bought} \rightarrow \text{yesterday}$ و $\text{car} \rightarrow \text{that}$ بدون همپوشانی یال‌ها امکان پذیر نیست که در این حالت شرط افکنشی بودن به عنوان یکی از شروط خوش‌ساخت بودن گراف وابستگی نقض می‌شود.

گرافی که دو شرط «بدون دور بودن» و «همبندی» را داشته باشد، یک درخت است. بنابراین به گراف وابستگی، «درخت وابستگی» نیز می‌گویند.

وظیفهٔ نگاشت رشتهٔ x به یک گراف وابستگی که شرایط گفته شده را ادا کند را تجزیهٔ وابستگی گویند.

۲-۴-۳- انواع روش‌های تجزیه

دسته‌بندی‌های مختلف برای روش‌های تجزیهٔ وابستگی ارائه شدند که در ادامه به برخی از مهم‌ترین این دسته‌بندی‌ها می‌پردازیم.

در کتاب «تجزیهٔ وابستگی» [۸] این روش‌ها را به دو دستهٔ اصلی و هر کدام را به دو زیر بخش تقسیم شده است که این دسته‌بندی در شکل (۲-۶) نشان داده شده است.

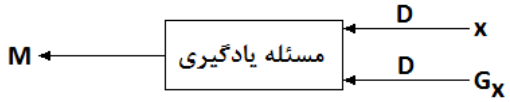
شکل (۲-۶) طبقه‌بندی روش‌های تجزیهٔ وابستگی

[1] Regeneration
[2] Rigid word-orders

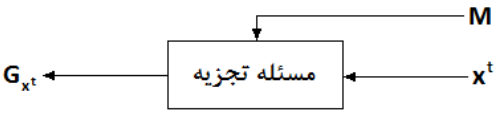
۱) راهکارهای مبتنی بر داده[1]: روش‌های این دسته از یادگیری ماشینی و به کمک داده‌های زبان شناختی برای تجزیهٔ جملات جدید بهره می‌برد. در همهٔ این روش‌ها فرض اولیه این است که داده‌های ورودی حتماً دارای ساختار نحوی درست است. به دلیل استفاده از یادگیری ماشینی دارای الگوریتم یادگیری هستند که الگوریتم یادگیری در آن‌ها می‌تواند باناظر، بی‌ناظر، نیمه‌ناظر یا تقویتی باشد.

در تجزیهٔ وابستگی باناظر دو مسئله مختلف باید حل شود:

اول) مسئلهٔ یادگیری: یادگیری معادل ساخت الگوی تجزیه[2] از جملات و ساختارهای وابستهٔ آموزشی است. در این مسئله، ورودی D جمله (x) آموزشی همراه با گراف‌های وابستگی (G) متناظر با آن است و خروجی الگوی آموزشی (M) است.



دوم) مسئلهٔ تجزیه: اعمال الگوی آموخته شده در مرحلهٔ قبل، برای تحلیل جملات جدید است. در این مسئله، ورودی الگوی تجزیه (M) و جملهٔ تست (x^t) است و خروجی گراف وابستگی (G_{x^t}) برای جملهٔ تست است.



راهکارهای مبتنی بر داده به دو دسته تقسیم می‌شوند:

۱) راهکارهای مبتنی بر گذار[3]: این راهکار با تعریف یک سامانهٔ گذار[4] یا ماشین حالت[5] برای نگاشت جمله به گراف وابستگی شروع می‌شود. مسئلهٔ یادگیری معادل استنتاج الگو برای پیش‌بینی گذار بعدی بر اساس تاریخچهٔ گذار است. مسئلهٔ تجزیه معادل ساخت رشتهٔ گذار بهینه برای جملهٔ ورودی توسط الگو است. به روش‌های این دسته اصطلاحاً «تجزیهٔ وابستگی جابه‌جایی-کاهش»[6] می‌گویند.

۲) راهکارهای مبتنی بر گراف[7]: در این راهکار فضایی از گراف‌های وابستگی کاندید برای جمله تعریف می‌شود. مسئلهٔ یادگیری معادل ارائه الگو برای انتساب امتیاز به گراف‌های وابستگی

---

[1] Data-driven approach
[2] Parsing model
[3] Transition-based approach
[4] Transition system
[5] State machine
[6] Shift-reduce dependency parse
[7] Graph-based approach

کاندید یک جمله است. مسئلهٔ تجزیه معادل یافت گراف وابستگی با بیش‌ترین امتیاز برای جملهٔ ورودی توسط الگو است. به روش‌های این دسته اصطلاحاً «تجزیهٔ درخت پوشای بیشینه»[1] گویند.

۲) راهکارهای مبتنی بر دستور[2]: این راهکار با تعریف دستور زبان‌های صوری[3] شروع می‌شود. در همهٔ این روش‌ها فرض بر این است که اگر ساختاری در قالب هیچ یک از قواعد موجود در پایگاه قوانین نگنجد، این جمله از نظر دستوری نادرست است. کاربرد دستور صوری بررسی این است که جملهٔ داده شده از قواعد دستور زبان پیروی می‌کند.

راهکارهای مبتنی بر دستور به دو دسته تقسیم می‌شوند:

۱) راهکارهای مستقل از متن[4]: در این راهکار ابتدا ساختار وابستگی به عبارات مستقل از متن تبدیل می‌شود و سپس تجزیه بر اساس دستور زبان مستقل از متن[5] انجام می‌شود. نگاشتی از ساختارهای وابستگی به بازنمایی ساختار عبارت مستقل از متن اعمال می‌کند و برای توسعه الگوریتم‌های تجزیه ، از گرامر مستقل از متن دوباره استفاده می‌کند.

۲) راهکارهای مبتنی بر محدودیت[6]: در این راهکار صورت مسئله، تبدیل به مسئلهٔ ارضای محدودیت[7] تبدیل می‌شود. دستور به صورت مجموعه‌ای از محدودیت‌های سازندهٔ گراف وابستگی تعریف می‌شود. مسئلهٔ تجزیه معادل یافتن گراف وابستگی برای یک جمله است که تمام محدودیت‌های دستور را ارضا کند.

شاید مهم‌ترین تفاوت روش‌های «مبتنی بر داده» با روش‌های «مبتنی بر دستور» این است که در روش مبتنی بر داده‌ها در بسیاری از موارد برای جملات غیر معیار 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.

در کنار این دو دسته راهکارهای تلفیقی نیز ارائه شدند که سعی می‌کنند از مزایای دو دسته استفاده کنند.

دسته‌بندی دیگر را نیور [۹] انجام داده است. در این دسته‌بندی روش‌های تجزیهٔ وابستگی را چهار گروه

---

[1] Maximum spanning tree parser
[2] Grammar-based approach
[3] Formal
[4] Context-free approach
[5] Context-free grammar (CFG)
[6] Constraint-based approach
[7] Constraint satisfaction problem

تقسیم کرده است.

۱) تجزیهٔ مبتنی بر نمودار[1]: روش سر راست برای تجزیهٔ وابستگی است که در آن با مسئله به صورت حالت محدود شدهٔ تجزیهٔ مستقل از متن برخورد می‌شود و از الگوریتم‌های تجزیهٔ مبتنی بر نمودار مثل سی‌کی‌وای[2] و اِرلی[3] استفاده می‌کند. این تکنیک‌ها دارای صحت تجزیهٔ خوبی برای درخت‌های افکنشی دارند اما از محدودیت‌های این روش می‌توان به عدم توسعهٔ سادهٔ آن به درختان غیرافکنشی اشاره کرد.

۲) تجزیهٔ مبتنی بر محدودیت: در این روش، مسئلهٔ تجزیهٔ وابستگی را به دید مسئلهٔ ارضای محدودیت نگاه می‌کند. نقطهٔ شروع این روش‌ها، بازنمایی فشردهٔ تمام گراف‌های وابستگی سازگار با ورودی است و پس از آن به صورت متوالی گراف‌های غیر معتبر را از طریق انتشار محدودیت‌های گرامری حذف می‌کند. با افزودن وزن‌های عددی به محدودیت‌ها و تعریف امتیاز[4] گراف به عنوان تابعی از وزن محدودیت‌های نقض، تجزیهٔ وابستگی تبدیل به یک مسئلهٔ بهینه‌سازی می‌شود که هدفش یافتن گراف وابستگی با بیش‌ترین امتیاز است. از محدودیت‌های این روش می‌توان به موارد زیر اشاره کرد:
   - این روش‌ها گرچه محدودیت‌های ذاتی تجزیهٔ مبتنی بر نمودار را ندارد، اما روش جستجوی دقیق را نمی‌توان جز در حالت‌های خاص اعمال کرد.
   - هر گونه تلاش برای توسعهٔ دامنهٔ محدودیت‌های وزن‌دار به فراتر از یک یال، مسئله را تبدیل به غیرقطعی کامل[5] می‌کند.

۳) تجزیهٔ مبتنی بر گذار: در این روش‌ها مسئلهٔ تجزیهٔ وابستگی به دید جستجوی قطعی توسط یک سامانهٔ گذار یا ماشین حالت نگاه می‌شود که توسط یک الگوی احتمالی[6] هدایت می‌شود تا گذار بعدی را پیش‌بینی کند. از محدودیت‌های این روش می‌توان مورد زیر اشاره کرد:
   - تجزیه‌کننده‌های مبتنی بر گذار می‌توانند تصمیماتشان را بر اساس بازنمایی بسیار غنی از تاریخچهٔ اشتقاق‌ها[7] (از جمله گراف وابستگی نیمه ساخته) بگیرد اما از انتشار خطا به دلیل خطاهای جستجو رنج می‌برد (به طور خاص زمانی که الگوی آماری طوری آموزش ببیند که بجای گذارهای کل جملات، صحت گذارهای محلی را بیشینه کند).

---

[1] Chart parsing
[2] CKY
[3] Early
[4] Score
[5] NP Complete
[6] Statistical model
[7] Derivation

۴) روش‌های ترکیبی[1]: مانند هر حوزهٔ دیگری برای بهبود صحت، می‌توان توانایی روش‌های مختلف را ترکیب کرد.

۲-۴-۴- الگوی تجزیه

الگوی تجزیهٔ وابستگی شامل مجموعه‌ای از محدودیت‌ها $\mathbb{\Gamma}$، مجموعه‌ای از شاخص‌ها[2] λ (که ممکن است تهی باشد) و الگوریتم معین تجزیه h است. $\mathbb{\Gamma}$ مجموعه‌ای از فضای درخت‌های وابستگی قابل قبول برای جملهٔ x است. این الگو را به صورت $M = (\mathbb{\Gamma}, \lambda, \mathrm{h})$ نشان می‌دهند.

در روش‌های مبتنی بر داده، نیاز به تعریف فرآیند یادگیری و برای روش‌های مبتنی بر دستور زبان و مبتنی بر داده، نیاز به تعریف فرایند 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 نشانه‌گذاری شده استخراج خواهد شد. در این الگوریتم‌ها از تابع خصوصیت[3] $f(x): X \to Y$ استفاده می‌شود که ورودی‌های x را به بازنمایی خصوصیات در فضای Y نگاشت می‌کند. در مرحلهٔ یادگیری، مجموعهٔ شاخص‌های λ بر اساس داده‌های آموزشی ساخته خواهد شد. مجموعهٔ آموزشی D شامل جملات x و درخت وابستگی متناظر با آن است [۸].

$$D = \{(x_d, G_d)\}_{d=0}^{|D|}$$

بر اساس تحقیقات انجام شده در کار مشترک همایش سالانهٔ یادگیری رایانه‌ای زبان طبیعی[4] سال ۲۰۰۷ [۱۰] راهکارهای ارائه شده در دستهٔ تجزیه‌کننده‌های مبتنی بر داده شامل سه بخش «الگوی تجزیه»، «الگوریتم استنتاج» و «الگوریتم یادگیری» است. در این بخش به بررسی الگوهای تجزیهٔ رایج در تجزیه‌کننده‌های مبتنی بر داده می‌پردازیم و در بخش‌های بعد به الگوریتم‌های استنتاج و یادگیری خواهیم پرداخت.

- تجزیه‌کننده‌های مبتنی بر گذار: در این دسته از تجزیه‌کننده‌ها الگوی تجزیه، گراف‌های وابستگی را با اعمال رشته‌ای از اعمال[5] یا گذارها تولید می‌کند. هر دو الگوریتم یادگیری و استنتاج بر پایهٔ پیش‌بینی

[1] Hybrid methods
[2] Parameter
[3] Feature Function
[4] CoNLL shared task
[5] Action

گذار درست مبتنی بر حالت جاری و/یا تاریخچه است. در ادامه برخی از رایج‌ترین الگوهای استفاده شده در این دسته بیان می‌شود.

- رایج‌ترین الگو برای تجزیه‌کننده‌های مبتنی بر گذار، الگوی الهام گرفته شده از «تجزیهٔ جابه‌جایی-کاهش»[1] است که در آن یک حالت تجزیه‌کننده شامل پشتهٔ واژه‌های پردازش شده جزئی و یک صف از واژه‌های باقیماندهٔ ورودی و جایی برای افزودن یال‌های وابستگی است. این الگو توسط اکثر تجزیه‌کننده‌های مبتنی بر گذار استفاده شده است.
- گاهی اوقات به صورت صریح الگوی قبلی با الگوی احتمالی برای رشتهٔ گذار ترکیب می‌شود. این الگو می‌تواند شرطی یا زایشی[2] باشد.
- الگوی دیگر «تجزیهٔ مبتنی بر لیست»[3] است که اولین بار توسط کاوینگتون [۵] ارائه شده است. این الگو روی واژه‌های ورودی با الگوی ترتیبی[4] تکرار انجام شده و برای هر واژه امکان اتصال واژهٔ قبلی به واژهٔ جاری، ارزیابی می‌شود. از این الگو در مقالات مختلف [۱۱، ۱۲] استفاده شده است.
- در برخی مقالات، از الگوی «مبتنی بر تجزیهٔ LR» استفاده شده است.

- تجزیه‌کننده‌های مبتنی بر گراف: در این دسته از تجزیه‌کننده‌ها الگوی تجزیه، تابع امتیاز[5] یا احتمال[6] روی مجموعهٔ یال‌های ممکن و کل گراف وابستگی تعریف می‌کند. بر همین اساس در «زمان یادگیری» پارامترهای این تابع تخمین زده شده و در «زمان تجزیه» به دنبال گرافی می‌گردد که مقدار این تابع را بیشینه کند. تفاوت اصلی این تجزیه‌کننده‌ها در موارد زیر است:

۱) نوع و ساختار تابع امتیازدهی (الگو)

۲) الگوریتم جستجو که بهترین تجزیه را پیدا می‌کند (استنتاج)

۳) روش تخمین پارامترهای تابع (یادگیری) است.

در ادامه برخی از رایج‌ترین الگوهای استفاده شده در این دسته بیان می‌شود.

- ساده‌ترین الگو مبتنی بر «مجموع امتیازهای اتصال محلی»[7] است که بر اساس ضرب

---

[1] Shift-reduce parsing
[2] Generative
[3] List-based parsing
[4] Sequential
[5] Scoring
[6] Probability
[7] Local attachment scores

نقطه‌ای[1] بردار وزن در بردار خصوصیت اتصال محاسبه می‌شود. به این نوع تابع امتیازدهی که اغلب به عنوان الگوی مرتبهٔ اول[2] یاد می‌شود، الگوی « مبتنی بر یال»[3] گویند زیرا امتیازها محدود به تک یال در درخت وابستگی است.

- در مقالهٔ [۱۳] توسعهٔ الگوی مرتبهٔ اول برای یکپارچه کردن مجموع امتیاز جفت یال‌های مجاور در درخت ارائه شده است که منجر به «الگوی مرتبهٔ دوم»[4] شده است. بر خلاف کارهای گذشته الگوی مرتبهٔ دوم که ضرورت داشت که روابط وابستگی سیبلینگ[5] باشند[۱۴] در این الگو روابط سَر با فرزندان وابسته نیز به حساب می‌آید.

۲-۴-۵- الگوریتم‌های استنتاج

- تجزیه‌کننده‌های مبتنی بر گذار:
  - رایج‌ترین تکنیک استنتاج در تجزیهٔ وابستگی مبتنی بر گذار «جستجوی قطعی حریصانه»[6] است که این جستجو توسط یک رده‌بند برای پیش‌بینی گذار بعدی با دریافت حالت جاری و تاریخچهٔ تجزیه‌کننده هدایت می‌شود. در این الگوریتم، واژه‌های جمله را به ترتیب متوالی چپ به راست[7] پردازش می‌کند. چندین گذر به صورت اختیاری روی ورودی انجام خواهد کرد تا اینکه هیچ واژه‌ای در سمت چپ بدون اتصال نماند.
  - به عنوان جایگزین تجزیه‌کننده‌های قطعی، تجزیه‌کنندهٔ دیگری وجود دارند که از الگوهای احتمالی استفاده می‌کنند و یک heap یا beam از رشتهٔ گذارهای جزئی[8] به منظور انتخاب محتمل‌ترین گذار در انتهای رشته استفاده می‌کند.
- تجزیه‌کننده‌های مبتنی بر گراف:
  - رایج‌ترین تکنیک استنتاج در تجزیهٔ وابستگی مبتنی بر گراف الگوریتم «درخت پوشای

---

[1] Dot product
[2] First-order model
[3] Edge-factored
[4] Second-order model

[5] Sibling: واژه‌هایی با والد یکسانی در درخت دارند.

[6] Greedy deterministic search
[7] Sequential left-to-right order
[8] Partial transition sequence

بیشینه»[1] است. بسته به اینکه درخت وابستگی افکنشی یا غیرافکنشی الگوریتم‌های مختلفی استفاده می‌شود.

- نسخهٔ افکنشی: از الگوریتم برنامه‌سازی پویای ارائه شده توسط آیزنر[2] [۱۵] برای حل الگوی مرتبهٔ اول استفاده می‌شود. برای حالت الگوی مرتبهٔ دوم در مقالهٔ [۱۳] نسخهٔ خاص الگوریتم آیزنر ارائه شده است.
- نسخهٔ غیرافکنشی: از الگوریتم چو-لیو-ادموندز[3] [۱۶، ۱۷] برای حل الگوی مرتبهٔ اول استفاده می‌شود. متأسفانه تجزیهٔ درخت پوشای بیشینهٔ غیرافکنشی در الگوی مرتبهٔ دوم غیرقطعی سخت[4] است. در مقالهٔ [۱۴] ضمن اثبات غیرقطعی سخت بودن این مسئله، تقریبی از آن را بر اساس الگوریتم افکنشی مرتبهٔ دوم آیزنر با زمان $O(n^3)$ ارائه کرده است.
- الگوی درخت پوشای بیشینه بسیار کارآمد و همواره دقیق است، اما قابل اعمال به الگوهایی که خواص عمومی درخت را به حساب می‌آورند نیست.

۲-۴-۶- الگوریتم‌های یادگیری

در روش‌های مبتنی بر داده از الگوریتم‌های یادگیری استفاده می‌شود. الگوریتم‌ها یادگیری باناظر شامل دو مرحله هستند:

۱) مرحلهٔ یادگیری: با یک پیکرهٔ آموزشی، دستور زبان وابستگی بدست می‌آید (استنتاج دستور زبان[5]) و از روی آن الگوی تجزیه بدست می‌آید.

۲) مرحلهٔ تجزیه: بر اساس الگو برای هر جملهٔ ورودی، گراف وابستگی تولید می‌شود.

در ادامه به برخی از رایج‌ترین راهکارهای یادگیری باناظر در تجزیهٔ وابستگی می‌پردازد:

- تجزیه‌کننده‌های مبتنی بر گذار: الگوهای تجزیه به صورت عادی قطعی[6] نیستند. به عنوان مثال در الگوی تجزیهٔ جابجایی-کاهش معمولاً برای یک جمله چندین حالت ممکن به عنوان خروجی درخت

[1] Maximum Spanning Tree (MST)
[2] Eisner
[3] Chu-Liu-Edmonds
[4] NP-hard
[5] Grammar induce
[6] Deterministic

وابستگی وجود دارد. در این صورت نیاز به یک تابع پیش‌گو[1] 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 حالت اولیهٔ $C_0(S)$، با تابع پیش‌گوی $o(c)$، گذار t تعیین می‌شود و حالت مرحله بعد بر اساس گذار انجام شده در پیکربندی کنونی تعیین می‌شود. پیچیدگی زمانی روش استفاده از تابع پیش‌گو برای یک جمله با n واژه در حالت تجزیهٔ درخت افکنشی برابر با $O(n)$ است. 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[2] از پیکره‌های آموزشی بوده است. بر این اساس، تجزیهٔ مبتنی بر رده‌بند[3]، یکی از مؤلفه‌های اصلی تجزیه بر مبنای گذار خواهد بود.

تجزیه‌کننده‌های مبتنی بر گذار، خواه از یک رده‌بند برای پیش‌بینی گذار بعدی استفاده کنند یا از الگوی احتمالی سراسری[4] که یک گراف کامل را امتیازدهی می‌کنند، برای آموزش رده‌بندها یا الگوهای احتمالی از چندین راهکار استفاده می‌شود.

◘ دستگاه بردار پشتیبان[5]

در این روش بین داده‌های آموزشی پهن‌ترین مرز ممکن ایجاد می‌شود به صورتی که بیش‌ترین حاشیه[6] ممکن را داشته باشد. در حالت عادی برای حل مسائل دو کلاسه استفاده می‌شود و خصوصیات داده‌های یادگیری باید عددی باشد. ویژگی‌های مقوله‌ای[7] مثل برچسب اجزای سخن باید به ویژگی‌های عددی تبدیل شوند. برای حل مسائل غیر خطی از تابع شالوده[8] استفاده می‌شود. نخستین بار توسط یامادا و ماتسوموتو[9] [۱۸] از این روش یادگیری در تجزیهٔ وابستگی استفاده شده است.

مشکل اصلی این الگوریتم، فرایند یادگیری طولانی و دودویی بودن آن است. زمانی که l تعداد نمونه‌های آموزشی باشد، هزینهٔ محاسباتی متناظر با l^2 یا l^3 خواهد بود. وقتی تعداد جملات آموزشی زیاد باشد

---

[1] Oracle
[2] Trained Classifier
[3] Classifier-Based Parsing
[4] Global probabilistic model
[5] Support vector machine (SVM)
[6] Maximum margin
[7] Categorical
[8] Kernel function
[9] Yamada and Matsumoto

(معمولاً بیش ۱.۵ میلیون) آموزش روی تمام نمونه‌ها به صورت یکجا دشوار است. برای حل این مشکل نمونه‌های آموزشی به چندین گروه تقسیم می‌کنند. با وجود این مشکلات این روش یادگیری، یکی از پیشرفته‌ترین روش رده‌بندی برای «تجزیهٔ وابستگی مبتنی بر گذار» محسوب می‌شود.

در اکثر مقالات از بستهٔ نرم‌افزاری LIBSVM [۱۹] استفاده شده است. در این پیاده‌سازی از استراتژی یک در مقابل یک[1] برای رده‌بندی چند کلاسه استفاده می‌شود. تبدیل خصوصیات مقوله‌ای به خصوصیات عددی با استفاده از تکنیک‌های استاندارد دودویی کردن[2] انجام می‌شود. یامادا و ماتسوموتو [۱۸] نشان دادند دادند که تابع شالوده چند جمله‌ای درجهٔ دو[3] ($K(x_i, x_j) = (\gamma x_i^T x_j + r)^2$) صحت بیشتری از الگوی خطی و توابع شالودهٔ چند جمله‌ای با درجات بالاتر دارد.

◘ یادگیری مبتنی بر حافظه[4]

در این روش تمام نمونه‌های آموزشی باید در حافظه نگهداری شود و رده‌بندی بر اساس شباهت همهٔ داده‌ها با هم انجام می‌شود. این یادگیری مبتنی بر دو قاعدهٔ اساسی است:

۱) یادگیری معادل ذخیرهٔ تجربیات در حافظه است.

۲) حل مسائل جدید با استفادهٔ مجدد از مسائل حل شدهٔ مشابه قبلی انجام می‌شود.

این روش الهام گرفته شده از راهکار نزدیک‌ترین همسایه[5] در بازشناسی آماری الگو[6] و هوش مصنوعی است. در اصطلاحات یادگیری ماشینی این روش را می‌توان در دسته روش‌های یادگیری تنبل[7] دانست زیرا پردازش ورودی را تا زمانی که مورد نیاز باشد به تأخیر می‌اندازد و ورودی را با ترکیب داده‌های ذخیره شده، پردازش می‌کند. یادگیری مبتنی بر حافظه به صورت موفقیت آمیزی قابل اعمال به تعداد از مسائل در پردازش زبان طبیعی مثل برچسب‌گذاری اجزای سخن[8] است.

دلیل استفاده از این روش یادگیری، انعطاف پذیری آن به خاطر کاوش مبتنی بر شباهت هنگام رده‌بندی حالاتی که قبلاً دیده نشدند است. علاوه بر این، روش یادگیری مبتنی بر حافظه به سادگی می‌تواند رده‌بندی چند کلاسه را مدیریت کند، بر خلاف «دستگاه بردار پشتیبان» که ذاتاً دو کلاسه است. این روش‌ها در حین

---

[1] One-versus-one
[2] Binarization
[3] Quadratic polynomial kernel
[4] Memory-based learning یا MBL
[5] Nearest neighbor
[6] Statistical Pattern Recognition
[7] Lazy learning method
[8] POS tagging

آموزش بهینه هستند زیرا تنها پیش‌نیاز آمدن نمونه‌ها به حافظه است اما برای پیکره‌های حجیم بسیار کند است.[۲۰].

در اکثر مقالات از بستهٔ نرم‌افزاری TiMBL[۱] (پیاده‌سازی الگوریتم یادگیری مبتنی بر حافظه) استفاده شده است[۲۱]. این پیاده‌سازی شامل معیارها، الگوریتم‌ها و توابع اضافی روی رده‌بند K نزدیک‌ترین همسایه است. این در حالی است که پارامترهای TiMBL بجای K نزدیک‌ترین همسایه برای K نزدیک‌ترین فاصله[۲] تعریف شده است، یعنی حتی با K=1 می‌توان بیش از یک همسایه داشته باشیم.

◘ یادگیری پرسپترون[۳]

این الگوریتم شبکهٔ عصبی اولین بار توسط کولینز[۴] [۲۲] به عنوان الگوریتم یادگیری در حوزهٔ پردازش زبان طبیعی مطرح شد و نشان داده شده است که قابل رقابت با الگوریتم‌های یادگیری مدرن مثل «دستگاه بردار پشتیبان» است. از پرسپترون در تعداد زیادی از کارهای پردازش زبان طبیعی مثل پردازش جزئی[۵] استفاده شده است. پرسپترون یک الگوریتم یادگیری آن‌لاین[۶] است که آموزش آن با اصلاح اشتباهات انجام شده توسط تجزیه‌کننده، هنگام مشاهدهٔ جملات آموزشی صورت می‌گیرد.

تعدادی از تجزیه‌کننده‌های مبتنی بر گذار[۲۳] از این روش یادگیری استفاده می‌کنند. الگوریتم بسیار ساده‌ای است که هزینهٔ زمانی و حافظه آن مستقل از اندازهٔ پیکرهٔ آموزشی است. از منظر کارایی، الگوریتم تجزیه باید در هر جملهٔ آموزشی، الگوریتم یادگیری را اجرا شود.

- تجزیه‌کننده‌های مبتنی بر گراف:

اکثر تجزیه‌کننده‌های مبتنی بر گراف از روش‌های مبتنی بر استنتاج آن‌لاین[۷] استفاده می‌کنند که در ادامه برخی از الگوریتم‌های این دسته را معرفی می‌کنیم:

---

[۱] Tilburg Memory Based Learner؛ این بسته در دانشگاه تیلبرگ توسعه یافته است.

[2] K Nearest Distance

[3] Perceptron Learning

[4] Collins

[5] Partial parsing

[۶] Online learning؛ در علم کامپیوتر به فرایندی آنلاین گویند که در آن بتوان ورودی را قطعه به قطعه و طی یک الگوی سریال پردازش کرد. تمرکز این الگوریتم بر کیفیت تصمیم‌گیری است. در مقابل الگوریتم‌های آفلاین هستند که کل داده‌های مسئله را می‌گیرند و از آغاز به خواندن آن می‌پردازند. نمونه‌ای از الگوریتم آفلاین selection sort است که نیاز به کل لیست برای مرتب سازی است.

[7] Online inference-based method

یادگیری پرسپترون

ساده‌ترین الگوریتم رده‌بندی خطی الگوریتم شبکهٔ عصبی پرسپترون است. اگر واقعاً مسأله به صورت خطی تفکیک‌پذیر باشد، با استفاده از این الگوریتم جواب همیشه درست خواهد بود.

تعدادی از تجزیه‌کننده‌های مبتنی بر گراف [۱۳، ۲۴] از این روش یادگیری استفاده می‌کنند. الگوریتم بسیار ساده‌ای است که هزینهٔ زمانی و حافظه آن مستقل از اندازهٔ پیکرهٔ آموزشی است. از منظر کارایی، الگوریتم تجزیه باید در هر جمله آموزشی، الگوریتم یادگیری را اجرا شود.

۲-۴-۷- معیارهای ارزیابی

در مقالات مختلف از معیارهای مختلفی برای ارزیابی صحت تجزیه استفاده شده است که در این بخش به معرفی آن‌ها می‌پردازیم:

امتیاز اتصال: این معیار بر اساس گراف وابستگی ایجاد شده محاسبه می‌شود، بر حسب اینکه درخت وابستگی برچسب‌دار یا بدون برچسب باشد، به دو صورت زیر تعریف می‌شود. این امتیازها خود بر دو نوع «مبتنی بر واژه» و «مبتنی بر جمله» هستند که در روش مبتنی بر واژه، تعداد برچسب‌ها درست تقسیم بر تعداد کل واژه‌های دادهٔ آموزشی می‌شود ولی در روش مبتنی بر جملهٔ نخست در هر جمله، امتیاز مبتنی بر واژه استخراج شده، سپس میانگین این امتیازها به عنوان امتیاز نهایی محسوب می‌شود.

- امتیاز اتصال بدون برچسب[1] (UAS): این معیار توسط آیزنر [۱۵] پیشنهاد شده است. نسبت واژه‌هایی که سَر آن‌ها به درستی پیدا شده است (یا اگر واژهٔ ریشه هستند، سَر ندارند).
- امتیاز اتصال برچسب‌دار[2] (LAS): این معیار توسط نیور [۲۰] پیشنهاد شده است. نسبت واژه‌هایی که سَر و نوع وابستگی آن‌ها به درستی پیدا شده است (یا اگر واژه ریشه هستند، سَر ندارند).

[1] Unlabled Attachment Score
[2] Labled Attachment Score

- **صحت وابستگی[1] (DA):** این معیار توسط یامادا و ماتسوموتو [۱۸] پیشنهاد شده است. نسبت واژه‌های غیر ریشه که واژهٔ سَر آن‌ها درست پیدا شده است.

$$DA = \frac{\#correct\ parents}{\#total\ parents}$$

- **صحت ریشه[2] (RA):** این معیار توسط یامادا و ماتسوموتو [۱۸] پیشنهاد شده است. نسبت واژه‌های ریشه که درست تحلیل شدند.

$$RA = \frac{\#correct\ root\ nodes}{\#total\ of\ sentences}$$

تمام معیارهای فوق به صورت میانگین امتیاز هر واژه و با کنار گذاشتن واژه‌های نشان‌گذاری[3] محاسبه می‌شوند.

- **تطابق کامل[4] (CM) یا نرخ کامل[5]:** این معیار نیز توسط یامادا و ماتسوموتو [۱۸] پیشنهاد شده است. در این روش در صورتی یک تجزیهٔ صحیح است که درخت تجزیه و درخت واقعی دقیقاً به یک شکل باشند. به عبارت دیگر نسبت جملاتی که ساختار وابستگی بدون برچسب آن‌ها کاملاً درست است.

$$CM = \frac{\#complete\ parsed\ sentences}{\#total\ number\ of\ sentences}$$

- **فراخوانی[6] و دقت[7]:** در این معیار از سه امتیاز دقت، فراخوانی و سنجهٔ اف[8] استفاده می‌شود.

نیور [۲۰] الگوریتم یادگیری مبتنی بر حافظه را مطرح کرد که خروجی آن رده‌بندی است که گذار بعدی (شامل نوع وابستگی) را با دریافت حالت جاری تجزیه‌کننده پیش‌بینی می‌کند. برای ارزیابی کیفیت این رده‌بند دو معیار ارزیابی ارائه شده است:

---

[1] Dependency Accuracy
[2] Root Accuracy
[3] Punctuation
[4] Complete Match
[5] Complete Rate
[6] Recall
[7] Precision
[8] F-Measure

- **صحت پیش‌بینی**[1]: این معیار بیان می‌کند که رده‌بند با دریافت حالت درست تجزیه‌کننده، گذار بعدی را چقدر خوب پیش‌بینی می‌کند. برای محاسبهٔ این مقدار از صحت رده‌بندی روی داده‌های گذار دیده نشده (با تابع زیان[2] صفر و یک) استفاده می‌شود.
  - برای اندازه‌گیری اختلاف آماری[3] مقادیر صحت بدست آمده در این معیار از «تست مک‌نیمار»[4] استفاده می‌شود.

- **صحت تجزیه:**[5] این معیار بیان می‌کند که رده‌بند به عنوان راهنمای تجزیه‌کنندهٔ قطعی چه عملکردی داشته است. برای محاسبهٔ این مقدار از صحت بدست آمده هنگام تجزیهٔ داده‌های جملات دیده نشده استفاده می‌شود. به صورت دقیق‌تر صحت تجزیه، توسط «امتیاز اتصال» محاسبه می‌شود که معیار استاندارد استفاده شده در تجزیهٔ وابستگی است.
  - برای اندازه‌گیری اختلاف آماری مقادیر صحت بدست آمده در این معیار از «تست تی»[6] استفاده می‌شود.

۲-۴-۸- تغییرات پله‌ای در تجزیهٔ وابستگی

در این بخش به مفهوم «تغییرات پله‌ای»[7] در تجزیهٔ وابستگی می‌پردازیم. با داشتن تعریف گراف‌های وابستگی، حال می‌توان تعریف را توسعه داد تا این گراف‌ها را به صورت پله‌ای ساخت. در سخت‌گیرانه‌ترین حالت، تغییرات پله‌ای به این معنی است که هر لحظه در طول فرایند تجزیه، یک ساختار متصل[8] بازنمایی کنندهٔ تحلیل ورودی موجود باشد. در قالب گراف‌های وابستگی، این بدان معنی است که گراف ساخته شده طی تجزیه، در تمام زمان‌ها متصل باشد. این مفهوم در تجزیهٔ وابستگی حداقل به دو دلیل مختلف مورد توجه قرار گرفته است:

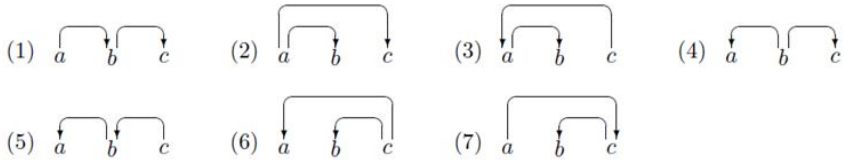
---

[1] Prediction Accuracy
[2] Loss
[3] Statistical significance
[4] McNemar's test
[5] Parsing Accuracy
[6] T-test
[7] Incrementality
[8] Single connected structure

۱) این نحوهٔ ساخت، عملی[1] است و می‌تواند در کاربردهای بلادرنگ[2] مثل تشخیص گفتار که نیازمند بروزرسانی پیوستهٔ تحلیل ورودی است، استفاده شود.

۲) مبانی تئوری بیش‌تری برای آن وجود دارد که تجزیه را به مدل‌سازی شناختی[3] متصل می‌کند. گواه‌های زبان‌شناسی وجود دارد که نشان می‌دهد تجزیهٔ انسانی ماهیت پله‌ای دارد.

برای درک مفهوم تغییرات پله‌ای شکل (۲-۷) را در نظر بگیرید. این شکل تمام ساختارهای افکنشی ممکن با سه گره را نشان می‌دهد. حالت‌های ۵ و ۶ و ۷ معکوس جهت یال حالت‌های ۱ و ۲ و ۳ دارند. معکوس حالت ۴ به دلیل نقض شرط تک‌سَری مجاز نمی‌باشد.



شکل (۲-۷) ساختارهای وابستگی افکنشی با سه گره

در مقالهٔ [۲۵] نشان داده شده است که رسیدن به «تغییرات پله‌ای اکید»[4] قابل دست‌یابی نیست. به عنوان مثال ساختار ۱ قابل تولید به صورت پله‌ای نیست. این مشکل ناشی از استراتژی اکیداً پایین به بالا است که نیازمند این است که هر واژه ابتدا تمام وابسته‌هایش قبل از ترکیب با واژهٔ سَر مشخص شده باشند. در ساختار ۱ تولید وابستگی‌ها از راست به چپ انجام می‌شود که تغییرات پله‌ای اکید را رد می‌کند. همین امر در مورد ساختارهای ۶ و ۷ نیز صادق است. این در حالی است که ساختارهای ۲ تا ۵ را می‌توان به صورت پله‌ای تولید کرد. ساختارهای ۶ و ۷ را نمی‌توان هرگز به صورت پله‌ای توسط چارچوب جاری پردازش کرد اما ساختار ۱ را می‌توان با تغییر استراتژی (ترکیب استراتژی پایین به بالا و بالا به پایین) ایجاد کرد. می‌توان بیان کرد که ایجاد ساختارهای ۶ و۷ حتی در چارچوب‌های دیگر تنها از طریق «شبه پله‌ای»[5] قابل انجام است.

---

[1] Practical
[2] Real-time
[3] Cognitive modeling
[4] Strict incrementality
[5] Pseudo-incremental

۲-۵- نتیجه‌گیری

این فصل با معرفی دو نظریهٔ زایشی و نظریهٔ وابستگی آغاز شد. همان‌طور که در این فصل بیان شد، تجزیهٔ وابستگی یکی از مکاتب موجود در نظریهٔ وابستگی است که زبان‌شناسان آن را برای توصیف ساخت‌های نحوی در زبان‌های گوناگون ارائه کردند. تجزیهٔ وابستگی نیز یک الگوی زبانی مبتنی بر دستور وابستگی برای تجزیه و تحلیل خودکار جملات است.

بعد معرفی جایگاه تجزیهٔ وابستگی، به تعریف ساختارها و گراف‌های وابستگی پرداختیم. در ادامه دو دسته‌بندی موجود برای روش‌های تجزیهٔ وابستگی مطرح شد. سپس الگوهای تجزیهٔ، الگوریتم‌های استنتاج و یادگیری هر کدام از این روش‌ها به تفکیک توضیح داده شدند. در پایان نیز انواع روش‌های ارزیابی صحت تجزیه اشاره شد.

فصل ۳:

مروری روش‌های تجزیهٔ وابستگی

۳-۱- مقدمه

در فصل قبل بیان شد که روش‌های تجزیهٔ وابستگی به دو دستهٔ تجزیهٔ مبتنی بر داده و مبتنی بر دستور تقسیم می‌شوند. روش‌های مبتنی بر داده در سال‌های اخیر توجهات زیادی به خود جلب کردند که دلیل آن در دسترس بودن پیکره‌های درختی در دامنه وسیعی از زبان‌هاست. به همین دلیل در این فصل تمرکز اصلی روی این روش‌های این خواهد بود و به مروری بر برخی از مهم‌ترین روش‌های این دسته خواهیم پرداخت. ابتدا به معرفی روش‌های مبتنی بر گذار و مبتنی بر گراف می‌پردازیم و در انتها مروری کلی بر روش‌های مبتنی بر دستور خواهیم پرداخت.

۳-۲- تجزیهٔ وابستگی مبتنی بر داده

تجزیهٔ وابستگی مبتنی بر داده شامل سیستم تجزیه است که تولید گراف‌های وابستگی برای جملات را از روی پیکره جملات نشانه‌گذاری شده با گراف‌های وابستگی، می‌آموزد. مزیت چنین الگوهایی در این است که آن‌ها به سادگی قابل تبدیل به هر دامنه یا زبانی هستند که منابع نشانه‌گذاری شده برای آن‌ها وجود داشته باشد. اکثر سیستم‌های تجزیهٔ مبتنی بر داده، بدون دستور[1] هستند و وجود دستوری که جملات مجاز زبان را تعریف کرده باشند، در نظر نمی‌گیرند. هدف اکثر سیستم‌های تجزیه مبتنی بر داده متمایز کردن تجزیه‌های خوب از بد برای جمله داده شده صرف‌نظر از دستور آن است.

۳-۲-۱- تجزیهٔ وابستگی 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[2] استفاده نمی‌شود، $\mathbb{T}$ تنها محدود به مجموعه درخت‌های ممکن برای جمله x است. راهکارهای مبتنی بر گذار رشته‌ای از مسائل محلی را حل می‌کنند و تضمین بهینه بودن به صورت عمومی و احتمالاً قدرت بیان را از دست می‌دهند.

[1] Grammar-less
[2] Formal Grammar

◘ الگوریتم کاوینگتون

کاوینگتون در مقالهٔ [۵] الگوریتم پایه‌ای برای تجزیهٔ جملات زبان طبیعی به درختان وابستگی ارائه کرده است. برای معرفی این الگوریتم از ساده‌ترین روش شروع کرده و به تدریج آن را بهبود می‌دهد.

- استراتژی اول: جستجوی ناشیانه[1] یا جستجوی فراگیر چپ به راست[2]

ساده‌ترین و ابتدایی‌ترین روش تجزیهٔ وابستگی است که در آن برای هر جفت واژه در کل جملات امکان اتصال به عنوان سَر به وابسته[3] یا وابسته به سَر[4] بر اساس دستور امتحان می‌شود. پیاده‌سازی این روش با یک گذر چپ به راست انجام می‌شود. برای n واژه، تمام $n(n-1)$ جفت باید امتحان شود که در این صورت پیچیدگی تجزیه $O(n^2)$ خواهد بود. اگر عملیات بازگشتی[5] اجازه داده شود این پیچیدگی $O(n^3)$ خواهد شد زیرا بعد از پذیرش n واژه، کل فرایند $O(n)$ ممکن است از ابتدا انجام شود. اینکه اول باید دنبال سَرها گشت و سپس دنبال وابسته‌ها یا بالعکس یا اینکه هر دو به صورت همزمان، هنوز تعیین نشده است. بسته به اینکه ابتدا دنبال واژهٔ سَر گشت یا وابسته، دو نسخه از این الگوریتم ارائه شده که در شکل (۳-۱) آمده است.

ESH[6]	**ESD**[7]
for i = 1 **to** n	**for** i = 1 **to** n
for j = i-1 **downto** 0	**for** j = i-1 **downto** 0
if HEAD?(j,i)	**if HEAD**?(i,j)
LINK(j,i)	**LINK**(i,j)
if HEAD?(i,j)	**if HEAD**?(j,i)
LINK(i,j)	**LINK**(j,i)

شکل (۳-۱) شبه کد جستجوی فراگیر چپ به راست کاوینگتون

در این الگوریتم دو تابع استفاده شده است که باید توضیح داده شوند.

- تابع $\text{HEAD?}(w_i, w_j)$: این تابع بررسی می‌کند که دستور اجازه می‌دهد واژهٔ w_i سَر واژهٔ w_j شود.
- تابع $LINK(w_i, w_j)$: این تابع رابطهٔ وابستگی را بین w_i و w_j در گراف وابستگی ایجاد می‌کند.

---

[1] Brute-force search
[2] Exhaustive left-to-right search
[3] heat-to-dependent
[4] Dependent-to-head
[5] Backtrack
[6] Exhaustive left-to-right search – heads first
[7] Exhaustive left-to-right search – dependent first

برای پیاده‌سازی این روش می‌توان از لیست‌های پیوندی[1] یا روش‌های استفاده کرد. واضح است که این الگوریتم ساده، کاملاً ناکارآمد است. با اعمال هر کدام از شروط گراف وابستگی به الگوریتم تعداد مقایسه‌ها کمتر شده و الگوریتم سریع‌تر خواهد شد.

- استراتژی دوم: اعمال شرط تک‌سَری

برای اعمال این شرط کافی است زمانی که یک واژه یک سر دارد، دیگر دنبال سر دیگری نگشت. پیاده‌سازی این ایده به صورت زیر خواهد بود:

- زمانی که دنبال وابسته‌های واژهٔ جاری می‌گردید، واژه‌هایی که در حال حاضر وابستهٔ واژهٔ دیگری باشند در نظر گرفته نشوند.
- زمانی که دنبال سَر واژهٔ جاری می‌گردید، بعد از یافتن یک سَر، نیازی به ادامه دادن و یافتن سَر دیگر نیست.

شبه کد این روش با نام‌های ESHU[2] و ESDU[3] دقیقاً معادل شکل (۳-۱) است با این تفاوت که تابع HEAD علاوه بر بررسی دستوری، اینکه واژهٔ مورد نظر خودش سَر نداشته باشد را نیز بررسی می‌کند.

در این الگوریتم مزیت بازنمایی مبتنی بر لیست پدیدار می‌شود. حذف عناصر غیر مطلوب و کار کردن با لیست‌ها ساده‌تر خواهد بود. برای این منظور الگوریتمی با دو لیست ارائه شده است.

- Wordlist: شامل تمام واژه‌ها مواجه شده که تا به حال دیده شده است.
- Headlist: تمام واژه‌هایی که سَر ندارند

هر دو لیست با افزودن عناصر در ابتدا ساخته می‌شود، بنابراین آن‌ها شامل واژه‌ها در ترتیب معکوس مواجه شدن است. این بار وابسته‌ها قبل از سرها دنبال می‌شوند. دلیل این امر این است که W (واژهٔ جاری) اگر سَر نداشته باشد، به Headlist اضافه شده است. شبه کد این الگوریتم در سمت چپ شکل (۳-۲) آمده است.

در مقالهٔ [۱۲] روش‌های کاوینگتون بررسی شده و تغییری در این الگوریتم داده است. در الگوریتم اصلی واژهٔ W به محض دیده شدن به wordlist اضافه می‌شود اما در این مقاله منتظر می‌ماند تا تمام تست‌ها کامل شود. شبه کد این الگوریتم در سمت راست شکل (۳-۲) آمده است.

---

[1] Linked lists

[2] Exhaustive search – head first with uniqueness

[3] Exhaustive search – dependent first with uniqueness

```
LSU[1]
Headlist=[]
Wordlist=[]
while (!end-of-sentence)
  W=next input word
  Wordlist=W+Wordlist
  foreach D in Headlist
    if HEAD?(W,D)
      LINK(W,D)
      delete D from Headlist
  end
  foreach H in Wordlist
    if HEAD?(H,W)
      LINK(H,W)
      terminate this foreach loop
  end
  if no head for W was found then
    Headlist=W+Headlist
  end
end
```

```
LSU*
Headlist=[]
Wordlist=[]
while (!end-of-sentence)
  W=next input word
  foreach D in Headlist
    if HEAD?(W,D)
      LINK(W,D)
      delete D from Headlist
  end
  foreach H in Wordlist
    if HEAD?(H,W)
      LINK(H,W)
      terminate this foreach loop
  end
  if no head for W was found then
    Headlist=W+Headlist
  end
  Wordlist=W+Wordlist
end
```

شکل (۳-۲) شبه کد الگوریتم مبتنی بر لیست با شرط یکتایی کاوینگتون

- استراتژی سوم: اعمال شرط افکنشی بودن

برای افزودن شرط افکنشی به این تجزیه‌کنندهٔ وابستگی پایین به بالا، باید محدودیت‌های زیر اعمال شود:

- عبور از روی وابسته‌های پیشین واژهٔ W که این امر می‌تواند همراه با اتصال هر واژهٔ قبلی متوالی که هنوز وابسته‌اند باشد یا با توقف جستجو انجام شود.
- زمانی که دنبال سَر W می‌گردیم تنها واژهٔ قبلی را به عنوان سَر در نظر بگیریم و به همین ترتیب تا ریشهٔ درخت ادامه دهیم.

شرط دوم به سادگی بیان می‌کند که سَر W (که به آن H گفته می‌شود) مقدم بر W باشد. این واژه می‌تواند شامل واژه‌ای باشد که بلافاصله قبل W است. به عبارت دیگر از تعریف افکنشی بیان می‌کند که زیر رشته H...W باید پیوسته باشد.

شرط اول بیان می‌کند که وابسته‌های W رشتهٔ پیوسته‌ای از واژه‌ها هستند که در زمان رویارویی با W هنوز مستقل هستند.

شبه کد این الگوریتم در شکل (۳-۳) آمده است.

[1] List-based search with uniqueness

```
LUSUP[1]
Headlist=[]
Wordlist=[]
while (!end-of-sentence)
  W=next input word
  Wordlist=W+Wordlist
  foreach D in Headlist
    if HEAD?(W,D)
      LINK(W,D)
      delete D from Headlist
    else
      terminate this foreach loop
  end
  H= word immediately preceding W
  loop
    if HEAD?(H,W)
      LINK(H,W)
      terminate this foreach loop
    if H is independent then terminate the loop
    H= the head of H
  end
  if no head for W was found then
    Headlist=W+Headlist
  end
end
```

شکل (۳-۳) شبه کد الگوریتم مبتنی بر لیست با شرط یکتایی و افکنشی کاوینگتون

◘ الگوریتم یامادا و ماتسوموتو

ایدهٔ تجزیه‌کننده‌های مبتنی بر گذار اولین بار توسط یامادا و ماتسوموتو ارائه شد. این تجزیه‌کننده روشی برای تحلیل وابستگی‌های واژه به واژه و تولید «درخت وابستگی افکنشی بدون برچسب» با استفاده از الگوی تجزیهٔ قطعی پایین به بالا مبتنی بر «جابجایی-کاهش»، همراه با الگوریتم یادگیری «دستگاه بردار پشتیبان» است. در علت استفاده از دستگاه بردار پشتیبان، دو مزیت آن برای تحلیل آماری وابستگی ذکر شده است:

۱) کارایی در فضای خصوصیات با ابعاد خیلی بالا با قابلیت تعمیم زیاد: در دستگاه بردار پشتیبان پارامتر w و b در ابر صفحه‌های مجزایی بر اساس استراتژی حاشیهٔ بیشینه، بهینه می‌شوند. این استراتژی از نظر تئوری تضمین می‌کند که تعمیم خطای اندکی برای نمونه‌های شناخته نشده در فضای خصوصیات با ابعاد بالا ارائه کند.

۲) یادگیری با ترکیب چندین خصوصیت توسط توابع چند جمله‌ای شالوده امکان پذیر است: در الگوریتم یادگیری می‌توان رده‌بندی غیر خطی را توسط توابع شالوده بدست آورد. به طور خاص استفاده از تابع چند جمله‌ای $(x'.x'' + 1)^d$ به عنوان شالوده توصیه می‌شود. با دارا بودن این مزایا می‌توان قواعد

[1] List-based search with uniqueness and projectivity

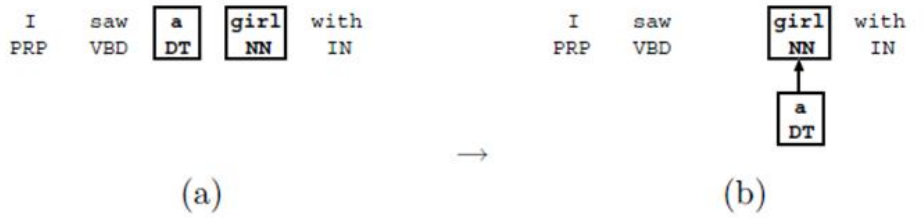
تحلیل ساختار وابستگی را با استفاده از خصوصیات زیاد انجام داد. به عبارت دیگر نـه تنهـا خصوصـیات معمول مانند تگ‌های برچسب اجزای سخن و واژه‌ها بلکه ترکیب آن‌ها را نیز می‌توان استفاده کرد.

این تجزیه‌کننده، درخت وابستگی را به صورت چپ به راست و با استفاده از سه عمل زیر ساخته می‌شـود که این اعمال می‌توانند به دو واژهٔ همسایه (که به آن گره‌های هدف گفته می‌شود) اعمال شوند.

۱) عمل جابه‌جایی: در این عمل هیچ ساخت وابستگی بین گره‌های هدف ایجاد نخواهد شد و نقطهٔ تمرکـز یکی به راست جلو خواهد رفت. شکل (۳-۴) مثالی از یک عمل جابجایی است کـه در آن بعـد از اعمـال گره‌های هدف از واژه‌های saw و a به واژه‌های a وgirl منتقل می‌شود.

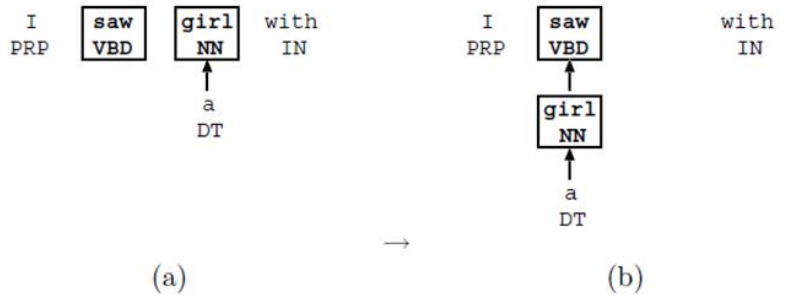
شکل (۳-۴) مثال عمل جابجایی [۱۸]

۲) عمل راست: در این عمل رابطهٔ وابستگی بین دو واژهٔ همسایه ایجاد می‌شود به صـورتی کـه گـرهٔ سـمت چپ از گره‌های هدف فرزند گرهٔ سمت راست خواهد شد. شکل (۳-۵) مثالی از یک عمل راست است که در آن بعد از اعمال این عمل واژهٔ a فرزند واژهٔ girl می‌شود. باید اشاره کرد که بعـد از پایـان ایـن عمـل گره‌های هدف بعدی واژه‌های saw و girl هستند. یعنی سـمت راسـت‌تـرین نقطـهٔ تمرکـز بـدون تغییـر می‌ماند.



شکل (۳-۵) مثال عمل راست [۱۸]

۳) عمل چپ: در این عمل رابطهٔ وابستگی بین دو واژهٔ همسایه ایجـاد مـی‌کنـد کـه گـرهٔ سـمت راسـت در گره‌های هدف، فرزند سمت چپ خواهد شد. شکل (۳-۶) مثالی از عمل چپ است.



شکل (۳-۶) مثال عمل چپ [۱۸]

پیچیدگی زمانی این الگوریتم در بدترین حالت درجهٔ دوم $O(n^2)$ است. یک قاعدهٔ مهم زبان‌شناسی بیان می‌کند که بدترین حالت رخ نخواهد داد یعنی انسان‌ها جملاتی به زبان نمی‌آورند که الگوریتم تجزیه را به شرایط بدترین حالت ببرد. درواقع بدترین حالت برای مقادیر کوچک n رخ می‌دهد.

- **تجزیه‌کنندهٔ یال معیار**[1]: الگوریتم یامادا و ماتسوموتو خاصیت «تغییرات پله‌ای» را نداشت. نیور [۲۵] الگوریتمی مبتنی بر پشته ارائه کرد که نسخهٔ پله‌ای الگوریتم یامادا و ماتسوموتو است و پیچیدگی زمانی آن را به زمان خطی $O(n)$ کاهش داده است. این الگوریتم تغییرات پله‌ای را تنها با یک گذر چپ به راست روی ورودی بدست می‌آورد در حالی که الگوریتم یامادا و ماتسوموتو نیاز به انجام چندین تکرار دارد تا گراف وابستگی پایین به بالا انجام شود.

هر حالت این تجزیه‌کننده توسط سه تایی $\langle S, I, A \rangle$ بازنمایی می‌شود. این سه تایی حاوی سه ساختمان داده زیر است:

- S: پشته برای نگهداری ورودی‌های پردازش شدهٔ قبلی است (معمولاً به صورت یک لیست بازنمایی می‌شود).
- I: لیست واژه‌های جملهٔ ورودی که هنوز پردازش نشدند.
- A: مجموعهٔ روابط وابستگی تولید شده برای ساخت درخت وابستگی است.

در ابتدای کار این تجزیه‌کننده، کلیهٔ واژه‌های ورودی در لیست قرار می‌گیرد و پشته و مجموعهٔ روابط وابستگی تهی در نظر گرفته می‌شوند (حالت شروع $\langle nil, W, \emptyset \rangle$). کار تجزیه‌کننده، زمانی به پایان می‌رسد که تمام واژه‌های لیست ورودی پردازش شده و لیست خالی شود (حالت نهایی $\langle S, nil, A \rangle$). در حالت نهایی رشتهٔ ورودی W را تنها زمانی پذیرفته می‌شود که گراف وابستگی تولید شده خوش‌ساخت

[1] Arc-standard

باشد.

این تجزیه‌کننده از همان سه عمل الگوریتم یامادا و ماتسوموتو استفاده می‌کند که تنها مفهوم پشته را وارد این عمل‌ها کرده است.

۱) عمل جابجایی: این عمل واژهٔ ورودی بعدی w_i را در پشته قرار می‌دهد. تنها شرط اجرا این است که لیست ورودی خالی نباشد.

$$\langle S, w_i | I, A \rangle \rightarrow \langle w_i | S, I, A \rangle$$

۲) عمل کاهش چپ[1]: دو واژهٔ بالای پشتهٔ w_i و w_j را توسط یال چپ به راست $w_j \rightarrow w_i$ با هم ترکیب می‌کند. تنها شرط اجرا این است که واژهٔ w_i سَر دیگری نداشته باشد.

$$\langle w_j w_i | S, I, A \rangle \rightarrow \langle w_j | S, I, A \cup \{(w_j, w_i)\} \rangle \qquad \neg \exists w_k : (w_k, w_i) \in A$$

۳) عمل کاهش راست[2]: دو واژهٔ بالای پشتهٔ w_i و w_j را توسط یال راست به چپ $w_i \rightarrow w_j$ با هم ترکیب می‌کند. تنها شرط اجرا این است که واژهٔ w_j سَر دیگری نداشته باشد.

$$\langle w_j w_i | S, I, A \rangle \rightarrow \langle w_i | S, I, A \cup \{(w_i, w_j)\} \rangle \qquad \neg \exists w_k : (w_k, w_j) \in A$$

این سامانهٔ گذار غیر قطعی است زیرا در هر حالت، چندین گذار می‌تواند اعمال شود. بنابراین برای بدست آوردن یک تجزیه‌کنندهٔ قطعی باید مکانیزمی برای حل تعارض[3] میان گذارها ارائه شود. صرف نظر از اینکه چه مکانیزمی استفاده شود، تجزیه‌کننده تضمین می‌کند بعد از حداکثر 2n گذار (با طول رشتهٔ ورودی n) پایان یابد (O(2n)=O(n)). علاوه بر این تجزیه‌کننده تضمین می‌کند گراف وابستگی بدون دور و افکنشی (و شرط تک‌سَری) است. این بدان معنی است که گراف وابستگی داده شده در حالت نهایی خوش‌ساخت است اگر و تنها اگر متصل باشد.

حال می‌توان به بررسی پله‌ای بودن تجزیه در این چارچوب پرداخت. نیاز است که گراف وابستگی در تمام زمان‌ها متصل باشد. با تعاریف «کاهش چپ» و «کاهش راست» ایجاد رابطهٔ وابستگی بدون اینکه واژهٔ جدید ابتدا داخل پشته منتقل شود، غیر ممکن است. بنابراین به نظر می‌رسد که شرط منطقی برای حفظ پله‌ای بودن این است که اندازه پشته نباید از ۲ تجاوز کند. به همین صورت نیاز است که هر واژه تا حد امکان زود به پشته منتقل شود.

◘ الگوریتم نیور

تجزیه‌کنندهٔ قطعی برای تولید «درخت‌های وابستگی افکنشی برچسب‌دار» با الگوی تجزیهٔ شبیه

[1] Left-reduce
[2] Right-reduce
[3] Conflict

تجزیه‌کنندهٔ «جابجایی-کاهش» در دستور مستقل از متن، همراه با الگوریتم یادگیری «مبتنی بر حافظه» است. کار کردن با درخت‌های وابستگی برچسب‌دار یکی از انگیزه‌های استفاده از یادگیری مبتنی بر حافظه نسبت به دستگاه بردار پشتیبان است، زیرا نیاز به رده‌بندی چند کلاسه وجود دارد. اگرچه با دستگاه بردار پشتیبان می‌توان مسائل چند کلاسه را حل کرد اما در حالت وجود تعداد کلاس‌های زیاد به مشکلاتی بر می‌خورد. در الگوریتم یامادا و ماتسوموتو تنها ۳ کلاس (معادل انتخاب یکی از سه عمل) مطرح بود[۲۶]. گراف وابستگی تولید شده توسط این تجزیه‌کننده، تضمین شده است که افکنشی و بدون دور است.

- **تجزیه‌کنندهٔ مشتاق با یال**[1]: یکی از مشکلات روش «یال معیار» این است که تا زمانی که یک واژهٔ وابستهٔ راست، تمام وابسته‌هایش معین نشود، قابل اتصال به واژهٔ سَر خود نیست. در نتیجه این الگوریتم قادر به تولید پله‌ای ساختارهای ۶ و ۷ در شکل (۲-۷) نیست زیرا ساختارهای ۶ و۷ دو واژهٔ اول با یک یال در گراف وابستگی نهایی به هم متصل نیستند و برای ایجاد رابطهٔ وابستگی بین a و c ابتدا باید رابطهٔ وابستگی بین b و c ایجاد شود. تجزیه‌کنندهٔ «مشتاق به یال» در واقع راهکاری برای افزایش قابلیت تغییرات پله‌ای است. برای حل این مشکل نیاز به عمل بیرون انداختن واژهٔ بالای پشته است. یک گذار جدید به عنوان کاهش اضافه می‌شود. به منظور افزایش پله‌ای بودن نیاز است پردازش بالا به پایین و پایین به بالا را ترکیب کرد. به صورت دقیق‌تر، نیاز است تا وابسته‌های سمت چپ را پایین به بالا، و وابسته‌های سمت راست را بالا به پایین پردازش کنیم. در این روش یال‌ها به محض اینکه واژهٔ سَر و وابسته موجود هستند به گراف وابستگی اضافه می‌شود. حتی اگر وابستگی از نظر وابسته‌های خودش کامل نشده باشد. در این صورت ساختارهای ۶ و ۷ نیز قابل تولید هستند که در آن ابتدا رابطهٔ وابستگی a و c تولید می‌شود.

در این تجزیه‌کننده، چهار عمل تعریف شده است:

۱) عمل جابجایی (SH): یک عنصر از پشته خارج می‌کند[2].

$$\langle S, w_i | I, A \rangle \rightarrow \langle w_i | S, I, A \rangle$$

۲) عمل کاهش[3] (RE): واژهٔ بعدی ورودی w_i را به پشته وارد می‌کند[4].

$$\langle w_i | S, I, A \rangle \rightarrow \langle S, I, A \rangle \qquad \neg w_j : (w_j, w_i) \in A$$

۳) عمل تشکیل یال چپ[5] (LA): با انجام این عمل یال $w_j \xrightarrow{r} w_i$ از واژهٔ بعدی ورودی w_j به واژهٔ

[1] Arc-eager
[2] Pop or Reduce
[3] Reduce
[4] Push or Shift
[5] Left-arc

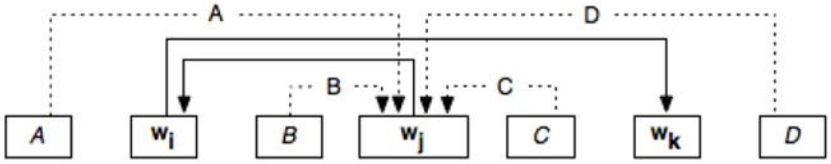
پشته w_i اضافه شده و w_i از پشته خارج می‌شود.

$$\langle w_i|S,w_j|I,A\rangle \rightarrow \langle S,w_j|I,A\cup\{(w_j,w_i)\}\rangle \qquad \neg\exists w_k:(w_k,w_i)\in A$$

۴) عمل تشکیل یال راست[1] (RA): با انجام این عمل یال $w_i \xrightarrow{r} w_j$ از واژهٔ بالای پشته w_i به واژهٔ بعدی ورودی w_j اضافه شده و w_j به پشته وارد.

$$\langle w_i|S,w_j|I,A\rangle \rightarrow \langle w_j|w_i|S,I,A\cup\{(w_i,w_j)\}\rangle \qquad \neg\exists w_k:(w_k,w_j)\in A$$

دو گذار اول برای تضمین شرط برچسب یکتا و تک‌سَری هستند و گذار سوم تنها زمانی اعمال می‌شود که واژهٔ بالای پشته در حال حاضر سَر داشته باشد. گذار جابجایی تنها زمانی قابل اعمال است که لیست ورودی غیر تهی باشد. در این تجزیه‌کننده، شرط افکنشی بودن را نیز به عنوان یکی از شروط خوش‌ساخت بودن گراف در نظر می‌گیرد. برای درک نحوهٔ حفظ افکنشی بودن در الگوریتم شکل (۳-۷) را در نظر بگیرید. در این شکل عملیات تشکیل یال چپ نشان داده شده است که در آن پیکان‌های خط‌چین نشان دهندهٔ سَرهای احتمالی واژهٔ w_j است. اگر A یا D به عنوان سَر برای w_j انتخاب شوند، رابطهٔ غیرافکنشی تولید می‌شود. در صورتیکه B یا C به عنوان سَر برای w_j انتخاب شوند، نمی‌توان سَری از بین این دو بدون نقض شرط بدون دور یا افکنشی بودن ساخت. بنابراین برای خوش‌ساخت بودن گراف، w_j باید از پشته خارج شود.



شکل (۳-۷) مثال عمل تشکیل یال چپ [۷]

گذارهای LA و RE یک اشتراک دارند که هر دو بدون اینکه روی طول لیست ورودی اثر بگذارند، اندازهٔ پشته را یکی کم می‌کند. که به این دو اصطلاحاً «گذار بیرون انداختن»[2] گویند.

گذارهای RA و SH نیز یک ویژگی مشترک دارند و آن اینکه طول لیست ورودی را یکی کاهش می‌دهد و اندازهٔ پشته را یکی زیاد می‌کند. به این دو اصطلاحاً «گذار وارد کردن»[3] گویند.

با مقایسهٔ دو الگوریتم «یال معیار» و «مشتاق به یال» مشاهده می‌شود که گذار «تشکیل یال چپ» در الگوریتم مشتاق به یال مستقیماً متناظر با گذار «کاهش چپ» در الگوریتم یال استاندارد است. تنها

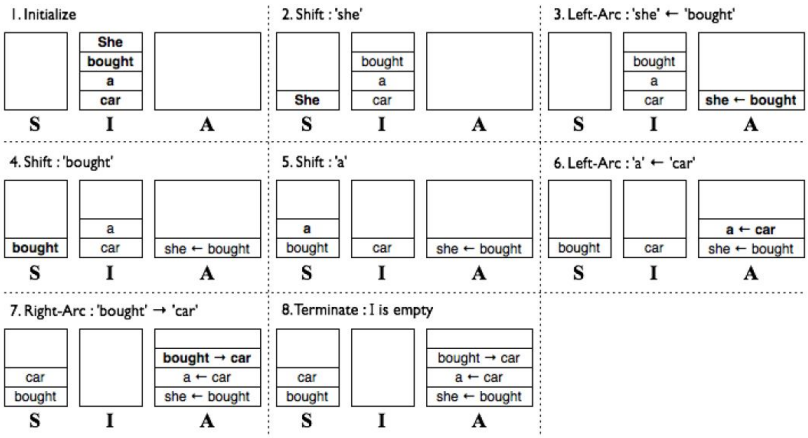
[1] Right-arc
[2] POP transition
[3] PUSH transition

تفاوت در این است که به دلیل تقارن، اولی روی واژهٔ بالای پشته و واژهٔ ورودی بعدی اعمال می‌شود در حالی که دومی روی دو واژهٔ بالای پشته اعمال می‌شود. در مقایسهٔ «تشکیل یال راست» با «کاهش راست» مشاهده می‌شود که اولی کاهش ندارد اما وابستهٔ راست را به پشته منتقل می‌کند. می‌توان گفت که عمل انجام شده توسط گذار «کاهش راست» در الگوریتم یال استاندارد، شامل گذار «تشکیل یال راست» و زیر رشته‌ای از گذارهای کاهش در الگوریتم مشتاق به یال است. از آنجایی که «تشکیل یال راست» و «کاهش» می‌توانند به صورت تعدادی گذار اختیاری مجزا اجرا شوند، این امر امکان تولید تجزیه‌کنندهٔ پله‌ای توسط زنجیره‌ای از وابسته‌های راست با طول اختیاری فراهم می‌آورد.

با استفاده از این الگوریتم نه تنها ساختارهای ۲ تا ۵ را بلکه ساختار ۱ در شکل (۲-۷) را نیز می‌توان به صورت پله‌ای، با ترتیب زیر تجزیه کرد. می‌توان نتیجه گرفت که الگوریتم مشتاق به یال نسبت به تغییرات پله‌ای در تجزیهٔ وابستگی بهینه است. اما هنوز ساختارهای ۶ و ۷ را نمی‌تواند به صورت پله‌ای تجزیه کند.

$$\langle nil, abc, \emptyset\rangle \xrightarrow{SH} \langle \mathrm{a, bc, \emptyset}\rangle \xrightarrow{\mathrm{RA}} \langle \mathrm{ba, c, \{(a, b)\}}\rangle \xrightarrow{\mathrm{RA}} \langle \mathrm{cba, nil, \{(a, b), (b, c)\}}\rangle$$

نحوهٔ تجزیه جمله «She bought a car» توسط این الگوریتم در شکل (۳-۸) نشان داده شده است.



شکل (۳-۸) تجزیهٔ جملهٔ She bought a car توسط الگوریتم نیور [۷]

این تجزیه‌کننده از الگوریتم یادگیری مبتنی بر حافظه استفاده می‌کند. برای این منظور از تابع تقریب f برای نگاشت از حالت به عمل تجزیه‌کننده استفاده می‌کند. در این نگاشت هر عمل (بجز جابجایی و کاهش) شامل یک گذار، همراه با یک نوع وابستگی است. در این تابع، Config مجموعه حالات در R_{x} است. برای حل مسئله f توسط تابع $\hat{f}$ تقریب زده می‌شود.

$$f: Config \rightarrow \{LA, RA, RE, SH\} * (R_x \cup \{nil\})$$

◘ الگوریتم‌های تولید درخت‌های غیرافکنشی

کلیهٔ الگوریتم‌هایی که تا به حال مطرح شد تنها قادر به تولید درخت‌های وابستگی افکنشی بودند. در این بخش به بررسی مهم‌ترین روش‌های ساخت تجزیه‌کننده‌های مبتنی بر گراف با قابلیت تولید درخت وابستگی غیرافکنشی می‌پردازیم.

- **پیاده‌سازی بهینهٔ الگوریتم کاوینگتون:** نیور [۲۷] در سال ۲۰۰۷ نسخهٔ بهینه شدهٔ الگوریتم کاوینگتون ارائه کرد که در آن امکان تولید درختان وابستگی غیرافکنشی وجود داشت.
- **تولید درخت شبه افکنشی:** نیور [۶] در سال ۲۰۰۵ روشی برای تولید درختان شبه‌افکنشی توسط الگوریتم افکنشی نیور ارائه کرد. نشان داده شده است که چطور تجزیهٔ وابستگی غیرافکنشی را می‌توان با ترکیب تجزیه‌کنندهٔ افکنشی مبتنی بر داده توسط تکنیک‌های تبدیلات خاص گراف تولید کرد. مراحل کار به صورت زیر است:
  - ابتدا داده‌های آموزشی برای تجزیه‌کننده با اعمال تعداد کمینه از عملیات لیفت[1] تبدیل به افکنشی می‌شود[2]. اطلاعات این لیفت‌ها به صورت رمزنگاری شده در برچسب‌های یال ذخیره می‌شود.
  - بعد از آموزش تجزیه‌کننده روی داده تبدیل شده، ایده آل این است که نه تنها ساختارهای وابستگی افکنشی را ایجاد کند بلکه برچسب‌های یال حاوی اطلاعات رمزنگاری شده لیفت منتصب شده را بیاموزد.
  - با اعمال تبدیل معکوس به خروجی تجزیه‌کننده، یال‌های با برچسب‌های غیر استاندارد را می‌توان به مکان مناسب در گراف وابستگی برگرداند که منجر به ساختارهای غیرافکنشی می‌شود.

نشان داده شده است که هر گراف وابستگی غیرافکنشی را تنها با یک عملیات لیفت می‌توان به یک گراف افکنشی تبدیل کرد که در آن هر یال غیرافکنشی $w_j \rightarrow w_k$ با یک یال افکنشی $w_i \rightarrow w_k$ جایگزین می‌شود که $w_i \rightarrow^* w_j$ در گراف اصلی موجود باشد. عملیات لیفت به صورت زیر تعریف می‌شود که روی هر یال اعمال شده و واژه سَر را در هر زمان یک گام بالا می‌برد.

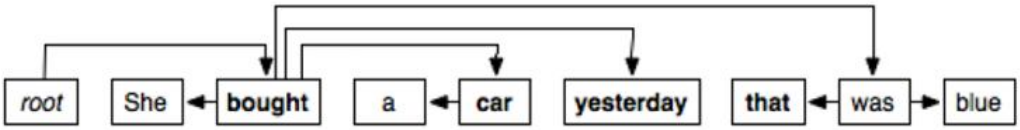
$$LIFT(w_j \rightarrow w_k) = \begin{cases} w_i \rightarrow w_k & if\ w_i \rightarrow w_j \\ undefined & otherwise \end{cases}$$

برای رابطهٔ $w_j \rightarrow w_k$ باید سَر جدید w_i برای w_k یافت شود که سَر مستقیم یا غیر مستقیم w_j باشد

[1] Lift
[2] Projectivized

(مثلاً $w_i \rightarrow w_j \rightarrow w_k$). در این رابطه، w_j سَر مستقیم w_k است و w_i سَر غیرمستقیم w_k است. رابطهٔ وابستگی جدید $w_i \rightarrow w_j$ باید افکنشی باشد. عملیات undefined تنها زمانی رخ می‌دهد که w_i وجود نداشته باشد که w_j را ریشهٔ جمله بسازد به طوری که رابطهٔ اصلی $w_j \rightarrow w_k$ لزوماً افکنشی باشد.

در شکل (۳-۹) که افکنشی شدهٔ درخت وابستگی غیرافکنشی شکل (۲-۵) است $w_j = \text{car}$ و $w_k = that$ و $w_i = bought$ است و رابطهٔ $car \rightarrow that$ بعد از عملیات لیفت به روابط $bought \rightarrow that$ و $bought \rightarrow car$ تبدیل می‌شود.



شکل (۳-۹) درخت وابستگی افکنشی تبدیل شده از شکل (۲-۵) [۷]

افکنشی کردن یک گراف وابستگی با انجام لیفت روی یال‌های غیرافکنشی در حالت کلی عملیات غیرقطعی است. چون می‌خواهیم تا حد امکان ساختار اصلی را حفظ کنیم. بنابراین مطلوب یافتن تبدیلی است که شامل تعداد کمینه‌ای از لیفت‌ها باشد. حتی این امر نیز در حالتی که گراف شامل چندین یال غیرافکنشی باشد که برای لیفت مطلوب باشند، می‌تواند غیرقطعی باشد. اما از الگوریتم شکل (۳-۱۰) برای ساخت یک تبدیل افکنشی کمینه $G' = (W, A')$ یک گراف وابستگی غیرافکنشی G=(W,A) استفاده می‌شود:

```
PROJECTIVIZE(W,A)
    A' = A
    while(W,A') is non-projective
        α =SMALLEST-NONP-ARC(A')
        A' = (A' − {α}) ∪ {LIFT(α)}
    end
    return (W, A')
end
```

شکل (۳-۱۰) شبه کد الگوریتم شبه افکنشی

تابع SMALLEST-NONP-ARC یال غیرافکنشی با کم‌ترین فاصله از سَر تا وابسته را برمی‌گرداند (حالت تساوی از با اولویت چپ به راست برطرف می‌شود). اما این بازنمایی افکنشی شده شکل (۳-۹)

هدف نهایی این فرایند تجزیه نیست بلکه باید تبدیل معکوس اعمال شود تا گراف وابستگی غیرافکنشی بدست آید. برای تسهیل این کار، اطلاعات عملیات لیفت در برچسب یال‌ها رمزگذاری می‌شود. به طور کلی امکان رمزنگاری کردن مکان دقیق سَر نحوی در برچسب یال از سَر خطی وجود دارد اما این امر منجر به مجموعهٔ نامحدود برچسب‌های یال می‌شود و آموزش تجزیه‌کننده را سخت می‌کند. در عمل یک مصالحه بین مقدار افزایش اطلاعات رمزگذاری شده در برچسب‌های یال (که منجر به افزایش صحت تبدیل معکوس می‌شود) و دشواری آموزش باید انجام شود. بر همین اساس روش‌های مختلفی برای رمزگذاری اطلاعات یال پیشنهاد می‌شود. در مقالهٔ [۶] سه الگوی رمزگذاری پیشنهاد شده که در جدول (۳-۱) توصیف شده است.

جدول (۳-۱) الگوهای رمزگذاری در تجزیهٔ شبه‌افکنشی

	برچسب یال لیفت شده	برچسب مسیر	تعداد برچسب‌ها
حالت پایه‌ای	d	p	n
Head	$d \uparrow h$	p	$n(n+1)$
Head+Path	$d \uparrow h$	$p \downarrow$	$2n(n+1)$
Path	$d \uparrow$	$p \downarrow$	$4n$

- حالت پایه‌ای: در این روش برچسب‌های اصلی تمام یال‌ها صرف‌نظر از اینکه لیفت شدند یا نشدند، حفظ می‌شود.
- Head: در این روش تنها یک برچسب جدید $d \uparrow h$ برای هر یال لیفت شده اضافه می‌شود.
  - d: رابطهٔ وابستگی بین سَر نحوی و وابسته در بازنمایی غیرافکنشی است.
  - h: رابطهٔ وابستگی که سَر نحوی با سَر خودش در ساختار زیرین دارد.
- Head+Path: در این روش علاوه بر تغییر برچسب هر یال، در مسیر عملیات لیفت از سَر نحوی به سَر خطی اضافه می‌کند که اگر برچسب اصلی p باشد برچسب جدید $p \downarrow$ خواهد بود.
- Path: در این روش تنها اطلاعات اضافی روی برچسب‌های مسیر حفظ می‌شود اما اطلاعات سَر نحوی یال لیفت شده را نگهداری نمی‌کند.

◘ تجزیه‌کننده‌های مبتنی بر گذار موجود

- MaltParser: یک تجزیه‌کنندهٔ وابستگی افکنشی که توسط «هال»[1]، «نیلسون»[2] و «نیور» در دانشگاه

[1] Hall
[2] Nilsson

واکسجوی[1] سوئد توسعه یافته است [۲۸] که به زبان جاوا نوشته شده و به صورت متن باز موجود است[2].

- برای یادگیری امکان انتخاب «دستگاه بردار پشتیبان» و «مبتنی بر حافظه» را دارد. به صورت پیش‌فرض از دستگاه بردار پشتیبان استفاده می‌کند زیرا بهترین کارایی را در این روش دارد.
- به منظور تجزیه امکان انتخاب الگوریتم «نیور» [۳] برای تجزیهٔ گراف‌های وابستگی افکنشی و الگوریتم «کاوینگتون» [۵] برای تجزیهٔ گراف‌های وابستگی غیرافکنشی را دارد. به صورت پیش‌فرض از الگوریتم نیور استفاده می‌کند. اگرچه الگوریتم کاوینگتون قادر به تجزیهٔ گراف‌های وابستگی افکنشی و غیرافکنشی است اما از الگوریتم نیور به عنوان الگوریتم تجزیهٔ پیش‌فرض استفاده شده زیرا این الگوریتم تجزیه با صحت بالاتر و پیچیدگی کمتر انجام می‌کند. برای تجزیهٔ گراف‌های وابستگی غیرافکنشی با الگوریتم نیور، امکان استفاده از الگوریتم تجزیهٔ وابستگی شبه افکنشی [۶] نیز وجود دارد.

۳-۲-۲- تجزیهٔ وابستگی مبتنی بر گراف

در این روش‌ها گراف وابستگی را با نمایش گراف مورد بررسی قرار می‌دهد. بدین صورت مانند تجزیهٔ مبتنی بر گذار، با الگوی تجزیهٔ $M = (\mathbb{\Gamma}, \lambda, h)$ 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 در نظر گرفته می‌شود. در نتیجه، امتیاز نشان‌دهندهٔ این است که یک گراف برای جملهٔ ورودی x تا چه اندازه صحیح است. روش‌های مبتنی بر گراف استنتاج سراسری انجام می‌دهند و سعی در یافتن جواب بهینه سراسری دارند.

- **الگوریتم درخت پوشای بیشینه:** برای جملهٔ ورودی x گراف وابستگی $G_x=(V_x,E_x)$ تعریف می‌شود. در طی مرحلهٔ تجزیه ابتدا یک گراف جهت‌دار کامل ساخته می‌شود که برای هر جفت (w_i, w_j) یال جهت‌داری از w_i به w_j وجود دارد که به صورت زیر امتیازدهی می‌شود. (بردار وزن توسط الگوریتم یادگیری بدست می‌آید و f بردار خصوصیت w_i و w_j است):

$$s(w_i, w_j) = w.f(w_i, w_j)$$

چون تجزیه‌کننده یک گراف جهت‌دار کامل ایجاد می‌کند، بیش از درخت پوشا را می‌توان از آن بدست

[1] Vaxjo
[2] http://w3.msi.vxu.se/~nivre/research/MaltParser.html

آورد. این درخت پوشا را می‌توان به صورت درختی تعریف کرد که شرایط زیر را ادا کند:

۱) درخت باید تمام رئوس گراف را طی کند.

۲) هر گرهٔ درخت بجز ریشه باید تنها و تنها یک یال ورودی داشته باشد.

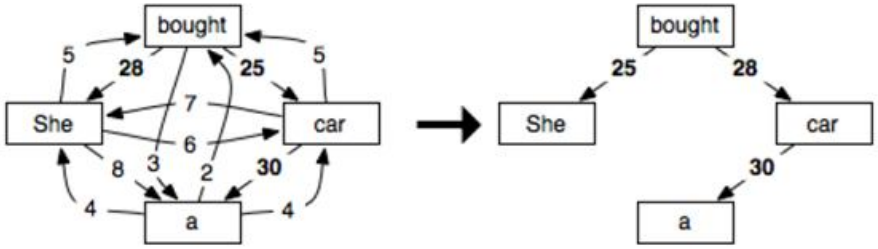
۳) گرهٔ ریشه باید تنها و تنها یک یال خروجی داشته باشد.

۴) هر مسیر جهت‌دار بین دو گره باید بدون دور باشد.

برای یافتن بهترین گراف وابستگی y برای جملهٔ داده شده x باید درختی پیدا شود که s(x,y) را بیشینه کند.

$$s(x,y) = \sum_{(w_i,w_j)\in y} s(w_i,w_j) = \sum_{(w_i,w_j)\in y} w.f(w_i,w_j)$$

شکل (۳-۱۱) نتیجهٔ حاصل از اعمال الگوریتم درخت پوشای بیشینه روی یک گراف جهت‌دار کامل را نشان می‌دهد.



شکل (۳-۱۱) درخت پوشای بیشینهٔ به دست آمده از گراف جهت‌دار کامل [۷]

در ادامه به معرفی دو الگوریتم برای محاسبهٔ درخت پوشای بیشینه می‌پردازیم:

- الگوریتم آیزنر: این الگوریتم که توسط آیزنر [۱۵] پیشنهاد شده است که تنها جملات با روابط افکنشی را کنترل می‌کند. یک الگوی زایشی[1] با الگوریتم تجزیهٔ مکعبی[2] (پیچیدگی زمانی $O(n^3)$) است. این الگوریتم از جدول پویا (C) برای نگهداری امتیاز تمام زیردرخت‌ها (یک زیردرخت با بیش‌ترین امتیاز) از مکان s تا t ($1 \le s < t \le n$) که n تعداد واژه‌ها در یک جمله است) استفاده می‌کند. این جدول، آرایهٔ چهار بعدی شامل $s,t \in \{1,\dots,n\}$ و $d \in \{\leftarrow,\rightarrow\}$ و $c \in \{0,1\}$ است که d برای نگهداری جهت یال و c برای نگهداری این که زیرگراف کامل است

[1] Generative model
[2] Cubic

یا خیر استفاده می‌شود. هدف از ساخت زیردرخت‌های ناکامل، یافتن بهترین زیردرخت است تا با یال جدید $w_s \rightarrow w_t$ آن‌ها را به هم متصل کند. بدون ایجاد زیردرخت‌های ناکامل، این یال جدید هرگز در الگوریتم اضافه نمی‌شود. بنابراین:

$C[s][t][\rightarrow][0]$: امتیاز بهترین زیردرخت از مکان s تا t شامل یک مسیر از $w_s \rightarrow w_t$ است.

$C[s][t][\rightarrow][1]$: امتیاز بهترین زیردرخت از مکان s تا t شامل یک مسیر از $w_s \rightarrow^* w_t$ است.

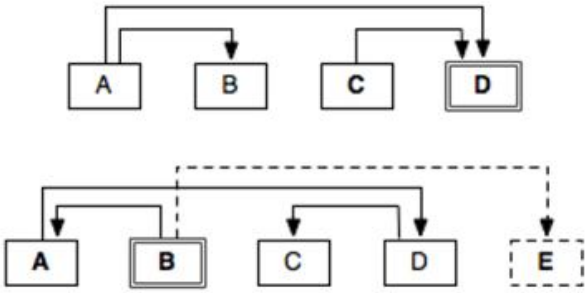
شبه کد الگوریتم آیزنر در شکل (۳-۱۲) آمده است:

```
1. #initialize the table
2. C[s][s][d][c]=0.9   ∀s ∈ {1, … , n}, d ∈ {←, →}, c ∈ {0,1}
3.
4. for k = 1 to n
5.   for s = 1 to n
6.     if (t = s + k) > n
7.       break
8.
9.     #create incomplete subtrees
10.    C[s][t][←][0] = max_{s≤u<t}(C[s][u][→][1] + C[u + 1][t][←][1] + s(t, s))
11.    C[s][t][→][0] = max_{s≤u<t}(C[s][u][→][1] + C[u + 1][t][←][1] + s(s, t))
12.
13.    #create complete subtrees
14.    C[s][t][←][1] = max_{s≤u<t}(C[s][u][←][1] + C[u][t][←][0])
15.    C[s][t][→][1] = max_{s≤u<t}(C[s][u][→][0] + C[u][t][→][1])
```

شکل (۳-۱۲) شبه کد الگوریتم آیزنر [۷]

الگوریتم با مقداردهی اولیه امتیازها به صفر (خط ۲) شروع شده، سپس برای هر s و t ابتدا بهترین زیر درخت ناکامل را با عرضه یال جهت‌دار جدید $s \rightarrow t$ و $s \leftarrow t$ (خطوط ۱۰ و ۱۱) پیدا می‌کند. زیر درخت ناکامل ترکیبی از دو درخت کامل «سمت راست‌ترین سَر»[1] و «سمت چپ‌ترین سَر»[2] است. هر ترکیب دیگر با جهت‌های مختلف به این دلیل که یک گره با چندین سَر یا یال غیرافکنشی تولید می‌کند، رد می‌شود. مثال شکل (۳-۱۳) ترکیبات اشتباه از زیر درخت ناکامل است.

[1] Right-most headed
[2] Left-most headed



شکل (۳-۱۳) ترکیبات اشتباه از زیر درخت‌های ناکامل [۷]

ایجاد یک زیر درخت ناکامل برای C[s][t][→][0] اگر C[u + 1][t][→][1] بجای C[u + 1][t][←][1] اضافه شود، گره D، انتخاب خواهد شد که دو سَر خواهد داشت (درخت بالایی در شکل (۳-۱۳)). از سوی دیگر اگر C[s][u][←][1] بجای C[s][u][→][1] اضافه شود، گره B نمی‌تواند سَرش را پیدا کند مگر اینکه شرط افکنشی را نقض کند (درخت پایینی در شکل (۳-۱۳)).

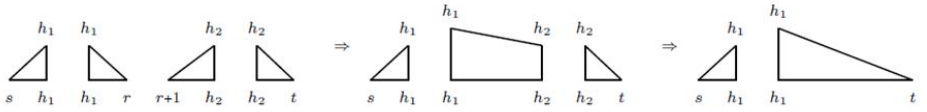
در خط ۱۰ و ۱۱ الگوریتم بهترین زیر درخت ناکامل بین s و t را با افزودن یک مسیر مستقیم بین s و t می‌یابد. این امر در درخت سمت چپ شکل (۳-۱۴) نشان داده شده است.

در خط ۱۴ و ۱۵ الگوریتم، بهترین زیر درخت ناکامل را با سایر درخت‌هایی که شامل مسیر مستقیم نیستند اما مسیر مستقیم بین s و t دارند، مقایسه می‌کند. این امر در درخت سمت راست شکل (۳-۱۴) نشان داده شده است. اگر زیر درختی باشد که مسیر مستقیمی نداشته باشد اما یک مسیر غیرمستقیم بین s و t موجود باشد امتیازش بالاتر از بهترین زیر درخت ناکامل خواهد بود. در نتیجه بهترین زیر درخت کامل بین s و t خواهد شد. در غیر این صورت بهترین درخت ناکامل، معادل بهترین درخت کامل بین s و t خواهد بود.

شکل (۳-۱۴) زیر درخت ناکامل (سمت چپ) و زیر درخت کامل (سمت راست) [۷]

در شکل (۳-۱۵) شمای کلی الگوریتم آیزنر را نشان می‌دهد. این شکل نشان می‌دهد که تنها نیاز به نگهداری ۳ اندیس در هر حالت دارد. تنها دو متغیر دودویی برای نگهداری جهت آیتم

(جهت وابستهٔ چپ یا راست) و یک متغیر نشان دهندهٔ کامل شدن زیر درخت (به این معنی که آمادهٔ جمع‌آوری وابسته‌های بیش‌تر است یا خیر) نیاز است.



شکل (۳-۱۵) شمای کلی الگوریتم آیزنر [۲۹]

- الگوریتم چو-لیو-ادموندز: این الگوریتم ابتدا توسط چو و لیو [۱۶] پیشنهاد شد و سپس توسط ادموندز [۱۷] بهبود یافت. جملات با روابط افکنشی و غیرافکنشی را کنترل می‌کند و پیچیدگی زمانی آن $O(n^3)$ است. این الگوریتم شبیهٔ الگوریتم آیزنر است که با ساخت یک گراف کامل جهت‌دار شروع می‌کند. برای هر راس در گراف جهت‌دار کامل الگوریتم تنها یال ورودی با بیش‌ترین امتیاز را نگهداری کرده و یک زیرگراف می‌سازد. اگر این زیرگراف دوری نداشته باشد، درخت پوشای بیشینه خواهد بود. اما اگر دور وجود داشته باشد، الگوریتم رأس‌های تشکیل دهندهٔ دور را با هم گروه‌بندی کرده[1] و امتیازات یال‌های ورودی به گروه را دوباره محاسبه می‌کند. الگوریتم بار دیگر بر روی زیرگراف جدید که شامل تنها یال‌های ورودی با بیش‌ترین امتیاز است اجرا می‌شود. شبه کد این الگوریتم در شکل (۳-۱۶) آمده است.

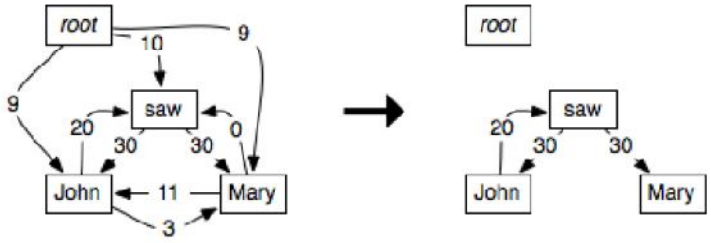
Chu-Liu-Edmonds(G, s)
Graph $G = (V, E)$
Edge weight function $s : E \rightarrow \mathbb{R}$
1. Let $M = \{(x^*, x) : x \in V, x^* = \arg\max_{x'} s(x', x)\}$
2. Let $G_M = (V, M)$
3. If G_M has no cycles, then it is an MST: return G_M
4. Otherwise, find a cycle C in G_M
5. Let $< G_C, c, ma >= \text{contract}(G, C, s)$
6. Let $\boldsymbol{y} = \text{Chu-Liu-Edmonds}(G_C, s)$
7. Find vertex $x \in C$
   such that $(x', c) \in \boldsymbol{y}$ and $ma(x', c) = x$
8. Find edge $(x'', x) \in C$
9. Find all edges $(c, x''') \in \boldsymbol{y}$
10. $\boldsymbol{y} = \boldsymbol{y} \cup \{(ma(c, x'''), x''')\}_{\forall (c,x''') \in \boldsymbol{y}}$
    $\cup\, C \cup \{(x', x)\} - \{(x'', x)\}$
11. Remove all vertices and edges in $\boldsymbol{y}$ containing c
12. return $\boldsymbol{y}$

contract$(G = (V, E), C, s)$
1. Let G_C be the subgraph of G excluding nodes in C
2. Add a node c to G_C representing cycle C
3. For $x \in V - C : \exists_{x' \in C}(x', x) \in E$
   Add edge (c, x) to G_C with
   $ma(c, x) = \arg\max_{x' \in C} s(x', x)$
   $x' = ma(c, x)$
   $s(c, x) = s(x', x)$
4. For $x \in V - C : \exists_{x' \in C}(x, x') \in E$
   Add edge (x, c) to G_C with
   $ma(x, c) = \arg\max_{x' \in C} [s(x, x') - s(a(x'), x')]$
   $x' = ma(x, c)$
   $s(x, c) = [s(x, x') - s(a(x'), x') + s(C)]$
   where $a(v)$ is the predecessor of v in C
   and $s(C) = \sum_{v \in C} s(a(v), v)$
5. return $< G_C, c, ma >$

شکل (۳-۱۶) شبه کد الگوریتم چو-لیو-ادموندز

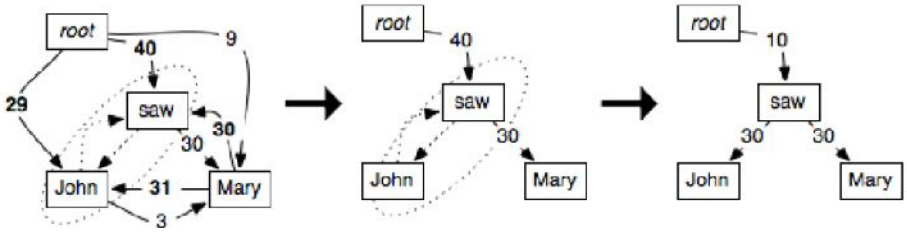
[1] Contract

در ادامه مراحل اجرای این الگوریتم آمده است. مثالی از اجرای الگوریتم چـو-لیـو-ادمونـدز روی جملهٔ انگلیسی در شکل (۳-۱۷) آمده است. اولین گام الگوریتم یافتن یال ورودی با بـیش‌تـرین امتیاز برای هر واژه است. همان‌طور که در شکل نشان داده شده است بعد از یک بار اجرای ایـن الگوریتم دور ایجاد شده است نیاز به ادامه اجرای الگوریتـم است.



شکل (۳-۱۷) مثالی از ایجاد دور بعد از اجرای یک مرحله از اجرای الگوریتم چو-لیو-ادموندز [۷]

در شکل (۳-۱۸) راه حل دور ایجاد شده و بدست آوردن درخت پوشای بیشینه نشان داده شـده اسـت. بـرای رفـع دور از ایـن گـراف، واژه‌هـای درگیـر دور یعنـی John و saw را در یـک گـره گروه‌بندی می‌کنیم. کافی است وزن یال‌های این رأس جدید را بروز کنیم. برای این منظـور وزن یال‌های root → saw و mary → saw و root → john و mary → john بروز مـی‌شـود. بـه عنوان مثال برای بروز کردن وزن root → saw امتیاز جدید برابـر بـا مجمـوع امتیـاز یـال‌هـای root → saw و saw → john اسـت کـه برابـر بـا 10+30 یعنـی ۴۰ اسـت. حـال الگـوریتم چو-لیو-ادموندز را با رئوس و وزن‌های جدید اجرا می‌کنیم. چون درخت حاصل بدون دور اسـت، درخت پوشای بیشینه بدسـت آمـده اسـت (گـراف وسـط در شـکل (۳-۱۸)). ویژگـی بنیـادی الگوریتم چو-لیو-ادموندز این است که درخت پوشای بیشینه در این گراف جدید را می‌تـوان بـه درخت پوشای بیشینه در گراف اصلی تبدیل کرد [۳۰]. برای ایـن منظـور بـه صـورت بازگشـتی الگوریتم را روی این گراف اجرا می‌کنیم. با اجرای الگـوریتم بایـد بهتـرین یـال ورودی بـه تمـام یال‌ها یافت. کافی است دو رأس حذف شده و وزن‌های تغییر کرده را به حالـت اولیـه برگردانـد. حال نیاز است تا یک سطح بالا برویم و گراف را بازسازی کنیم. یال از گرهٔ جدیـد بـه Marry در گراف اصل واژهٔ saw بوده است، بنابراین این رأس را وارد می‌کنیم. یال از ریشه به گرهٔ جدیـد در گراف اصلی واژهٔ saw بوده است. برای تولید یک درخت و عدم نقض تک‌سَـری، یـال از واژهٔ saw به john اضافه می‌شود.



شکل (۳-۱۸) رفع دور ایجاد شده در شکل (۳-۱۷) و بدست آوردن درخت پوشای بیشینه [۷]

کاهش مسئلهٔ تجزیه به یافت درخت پوشای بیشینه در گراف جهت‌دار ناشی از الگوی مبتنی بر یال است و می‌تواند در زبان‌های افکنشی با تجزیهٔ آیزنر و در زبان‌های غیرافکنشی با تجزیهٔ چو-لیو-ادموندز اعمال شود. تنها مسئلهٔ باقی‌مانده، نحوهٔ آموزش بردار وزن w است.

- **الگوریتم MIRA**[1]: این الگوریتم در مقالهٔ [۲۹] برای تجزیهٔ وابستگی معرفی شده. یک الگوریتم یادگیری کارا برای امتیازدهی خطی[2] تجزیه‌کنندهٔ وابستگی ارائه شده است که آموزش آن‌لاین چند کلاسه با بیش‌ترین حاشیه انجام می‌دهد. برخلاف الگوریتم یادگیری دستگاه بردار پشتیبان، این الگوریتم آموزش می‌بیند تا صحت سراسری درخت را بیشینه کند. در کنار سادگی، کارا و صحیح است. شبه کد الگوریتم MIRA در شکل (۳-۱۹) آمده است.

#Training Data: T = $\{(x_t, y_t)\}_{t=1}^{T}$
1. $w_0 = 0; v = 0; i = 0;$
2. **for** n = 1 **to** N
3. **for** t = 1 **to** T
4. $w^{(i+1)}$ = update $w^{(i)}$ according to instance (x_t, y_t)
5. $v = v + w^{(i+1)}$
6. i = i + 1
7. $w = v/(N * T)$

شکل (۳-۱۹) شبه کد الگوریتم یادگیری MIRA [۳۰]

هدف این الگوریتم حل مسئلهٔ رده‌بندی ساختار یافتهٔ شکل (۳-۲۰) است که در خط ۴ شبه کد MIRA قرار خواهد گرفت.

[1] Margin Infused Relaxed Algorithm
[2] Linearly-scored

$$\begin{aligned}&\textbf{min } \|w^{(i+1)} - w^{(i)}\| \\ &\textbf{s.t.} \quad s(x_t, y_t) - s(x_t, y') \geq L(y_t, y') \\ &\qquad \forall (x_t, y_t) \in T, y' \in dt(x_t)\end{aligned}$$

شکل (۳-۲۰) مسئلهٔ بهینه‌سازی ساختار یافتهٔ الگوریتم MIRA [۳۰]

در این رابطه که یک مسئلهٔ برنامه‌نویسی درجهٔ دو استاندارد[1] است و به سادگی قابل حل است، $L(y,y')$ مقدار حقیقی زیان[2] درخت y' نسبت به درخت صحیح y است. الگوریتم‌های یادگیری آنلاین یک نمونهٔ آموزشی را در هر بروزرسانی w در نظر می‌گیرد. در هر بروزرسانی MIRA تلاش می‌کند بردار وزن جدید را تا حد امکان نزدیک به بردار وزن قبلی نگهدارد (نرم تغییرات بردار وزن تا حد امکان کوچک بماند) که این کار با توجه به رده‌بندی درست نمونهٔ تحت نظر با یک حاشیهٔ تعیین شده توسط زیان ناشی از رده‌بندی اشتباه، انجام می‌شود. بصورت غیر رسمی، این بروزرسانی دنبال ایجاد یک حاشیه بین درخت وابستگی درست و هر درخت وابستگی نادرست است که حداقل تا حد امکان بیشترین زیان در درخت نادرست باشد. خطاهای بیش‌تری که یک درخت دارد امتیازش را دورتر از امتیاز درخت درست می‌کند.

تجزیه‌های ممکن برای هر ورودی دلخواه نوعاً به صورت نمایی زیاد خواهد بود و به تبع آن محدودیت‌های حاشیه‌ای نمایی در شکل (۳-۲۰) وجود خواهد داشت.

- Single-best MIRA: یک راه‌حل برای جلوگیری از رشد نمایی تعداد درخت‌ها، ساده کردن بهینه‌سازی با تنها یک محدودیت حاشیه‌ای برای درخت با بیش‌ترین امتیاز $s(x_t, y_t)$ است. این مسئلهٔ بهینه‌سازی در شکل (۳-۲۱) نشان داده شده است.

$$\begin{aligned}&\textbf{min } \|w^{(i+1)} - w^{(i)}\| \\ &\textbf{s.t.} \quad s(x_t, y_t) - s(x_t, y') \geq L(y_t, y') \\ &\qquad \forall (x_t, y_t) \in T, y' \in \operatorname{argmax}_{y'}(x_t, y')\end{aligned}$$

شکل (۳-۲۱) مسئلهٔ بهینه‌سازی در الگوریتم Single-best MIRA [۳۰]

- k-best MIRA: از بروزرسانی مشابه شکل (۳-۲۱) اما با k محدودیت برای k درخت با بیش‌ترین امتیاز نیز استفاده شده است که در شکل (۳-۲۲) آمده است. اگر مجموعهٔ درختان

[1] Standard Quadratic Programming

[۲] مقدار زیان یک درخت وابستگی تعداد واژه‌هایی است که دارای والد اشتباه هستند. بنابراین بزرگ‌ترین زیان در درخت وابستگی برابر طول جمله خواهد بود.

وابستگی ممکن برای جملهٔ ورودی x را با $dt(x)$ نشان دهنـد، مجموعـهٔ k درخـت وابسـتگی در $dt(x)$ که بالاترین امتیاز را با بردار وزن w دارند را با $beat_k(x; w)$ نشان می‌دهنـد. در حـالتی که تساوی رخ دهد، مشکل توسط یک عدد ثابت اما اختیاری بر طرف می‌شود. نشان داده شـده است که مقادیر کوچک k برای رسیدن به بهترین صحت در این الگـو کـافی اسـت. بـه عبـارت دیگر با رشد مقدار k کارایی کاهش می‌یابد که دلیل آن تنظـیم بـیش از انـدازه[1] روی مجموعـهٔ آموزشی است. الگوریتم آیزنر را می‌توان تغییر داد به طوری که k تا از بهترین درخت‌هـا را پیـدا کند. این امر منجر به افزایش زمان $O(k \log k)$ به مجموع زمان اجرا مـی‌شـود. امـا اسـتفاده از این الگو در الگوریتم چو-لیو-ادموندز ناکارآمد خواهد بود. به همین دلیـل توصـیه شـده در ایـن موارد از k=1 استفاده شود.

$$\begin{aligned} &\textbf{min} \ \|w^{(i+1)} - w^{(i)}\| \\ &\textbf{s.t.} \quad s(x_t, y_t) - s(x_t, y') \geq L(y_t, y') \\ &\qquad \forall (x_t, y_t) \in T, y' \in best_k(x_t, w^{(i)}) \end{aligned}$$

شکل (۳-۲۲) مسئلهٔ بهینه‌سازی در الگوریتم k-best MIRA [۲۹]

این الگو مرتبط با الگوریتم متوسط گیری پرسپترون در مقالهٔ [۲۲] است. در آن الگـوریتم یـک درخت یا ساختار با بیش‌ترین امتیاز برای بروزرسانی بردار وزن استفاده می‌شود. الگوریتم بـردار w را طوری بروزرسانی می‌کند که حاشیهٔ بین درخت درست و درخت بـا بـیش‌تـرین امتیـاز را بیشینه کند.

- Factored MIRA: ایـن امکـان وجـود دارد کـه از سـاختار فضـای خروجـی اسـتفاده کـرد و محدودیت‌های حاشیه‌ای با تعداد نمایی را به انـدازهٔ چنـد جملـه‌ای از محـدودیت‌هـای محلـی کاهش داد. برای مسئلهٔ درخت پوشای بیشینهٔ جهت‌دار، می‌توان از یال‌هـای خروجـی اسـتفاده کرد و مسئلهٔ بهینه‌سازی را به فرم درآورد.

$$\begin{aligned} &\textbf{min} \ \|w^{(i+1)} - w^{(i)}\| \\ &\textbf{s.t.} \quad s(l, j) - s(k, j) \geq 1 \\ &\qquad \forall (l, j) \in y_t, (k, j) \notin y_t \end{aligned}$$

شکل (۳-۲۳) مسئلهٔ بهینه‌سازی در الگوریتم Factored MIRA [۳۰]

[1] Overfit

این مسئله بیان می‌کند که وزن یال ورودی درست به واژهٔ x_j و وزن تمام یال‌های ورودی دیگر باید توسط حاشیه‌ای برابر با یک، از هم جدا شوند. بدیهی است زمانی که تمام محدودیت‌ها ادا شود درخت پوشای درست و تمام درختان پوشای نادرست با بیش‌ترین تعداد یال‌های ورودی نادرست مجزا شوند. این بدان دلیل است که امتیاز تمام یال‌های درست کنار گذاشته شده و تنها امتیازها برای خطاهای به دلیل اختلاف در امتیاز کلی محاسبه می‌شوند. چون هر خطا منجر به افزایش حداقل یک امتیاز می‌شود اختلاف کلی امتیاز باید حداقل تعداد خطاها را داشته باشد. در نظر بگیرید n رأس‌ها در گراف G_x وجود داشته باشد. در این صورت تعداد محدودیت‌ها $O(n^2)$ می‌شود زیرا برای هر رأس باید n-1 محدودیت نگهداری شود.

محدودیت‌های اعمال شده در حالت کلی از محدودیت‌های اولیه شدید تر هستند که این امر ممکن است منجر به رد کردن جواب بهینهٔ مسئلهٔ اصلی شود.

در مقالهٔ [۲۹] دو الگوی k-best MIRA و Factored MIRA برای تجزیهٔ وابستگی افکنشی در زبان انگلیسی بررسی شدند که نتیجه نشان داد که k-best MIRA بهتر عمل کرده و آموزش آن سریع‌تر است. علاوه بر این روش k-best MIRA نسبت به تابع زیان انعطاف‌پذیرتر است زیرا تابع زیان را به صورت مجموع عبارات هر وابستگی در نظر نمی‌گیرد.

- **الگوریتم درخت پوشای بیشینهٔ مرتبهٔ بالاتر:** در الگوریتم درخت پوشای بیشینهٔ مطرح شده بخش قبل، روابط وابستگی بین دو واژهٔ سَر و وابسته در نظر گرفته می‌شدند که به صورت امتیاز یال جهت‌دار بین دو واژه محاسبه می‌شد. این نوع برخورد با مسئله برای فواصل کوتاه مناسب است اما اطلاعات کافی برای فواصل طولانی (درخت‌های غیرافکنشی) فراهم نمی‌آورد. پیش‌بینی روابط وابستگی طولانی اغلب متأثر از روابط وابستگی با فاصلهٔ کوتاهی است که تا کنون پیش‌بینی شدند. بنابراین مطلوب است که روابط وابستگی با فاصلهٔ کوتاه نیز در امتیاز فواصل بیشتر به حساب آیند.

برای درک تفاوت الگوی مرتبهٔ اول شکل (۳-۲۴) را در نظر بگیرید. در این شکل با در نظر گرفتن روابط بین واژهٔ سَر و وابسته (به عنوان مثال واژهٔ plays به عنوان سَر و Elianti به عنوان وابسته) الگوی مرتبهٔ اول بدست می‌آید. در این الگو امتیاز وابستگی درخت را به صورت مجموع امتیاز تمام یال‌های درخت تعریف می‌کنند.

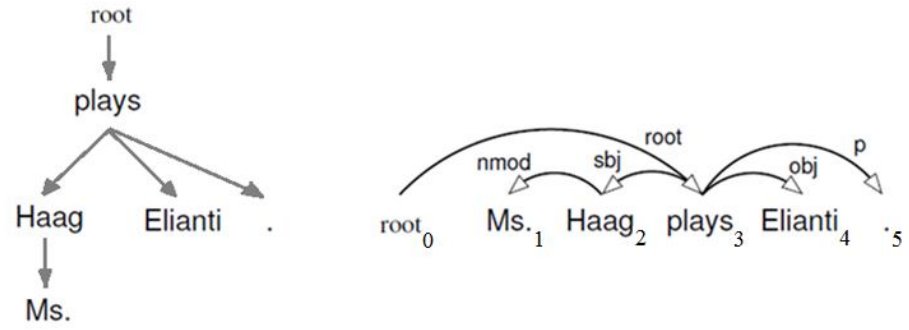
$$s(x,y) = \sum_{(i,j)\in y} s(i,j) = \sum_{(i,j)\in y} w.f(i,j)$$

به عنوان مثال امتیاز درخت وابستگی شکل (۳-۲۴) به صورت زیر محاسبه می‌شود.

$$s(x,y) = s(0,3) + s(3,2) + s(2,1) + s(3,4) + s(3,5)$$

در رابطهٔ فوق $f(i,j)$ یک بازنمایی از خصوصیت دودویی با ابعاد بالا بـرای رابطـهٔ وابسـتگی از x_{i} بـه x_{j} است. به عنوان مثال در درخت وابستگی شکل (۳-۲۴) خصوصیت زیر مقدار ۱ دارد.

$$f(i,j)=\begin{cases}1 & if\ x_i="plays"\ and\ x_j="Elianti"\\ 0 & otherwise\end{cases}$$

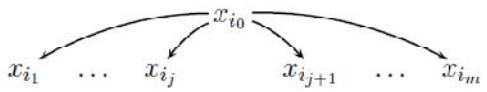


شکل (۳-۲۴) مثال خصوصیات مرتبهٔ بالاتر در درخت وابستگی برچسب‌دار [۳۱]

در مقالهٔ [۱۴] نوعی از بازنمایی خصوصیت‌های مرتبهٔ بالاتر[1] (درخت پوشـای مرتبـهٔ دوم) معرفـی شـده است که در آن تعامل بین سیبلینگ‌ها (واژه‌های دارای یک والد که در درخت سمت چپ شکل (۳-۲۴) مشخص شده است) را در نظر می‌گیرد. به عنـوان مثـال در شـکل (۳-۲۴) تعامـل سـه تـایی «plays»، «Elianti» و «.» یک خصوصیت مرتبهٔ دوم به حساب می‌آید. امتیاز درخـت بـه صـورت مجمـوع امتیـاز یال‌های مجاور در نظر گرفته می‌شود. امتیاز مرتبهٔ دوم درخت وابستگی شـکل (۳-۲۴) بـه صـورت زیـر محاسبه می‌شود.

$$s(i,k,j)=s(0,-,3)+\ s(3,-,2)+\ s(2,-,1)+\ s(3,-,4)+\ s(3,4,5)$$

تابع امتیاز مرتبهٔ دوم معادل امتیاز ایجاد یک جفت یال مجاور از x_{i} به واژهٔ x_{k} و x_{j} است. به عنوان مثال $s(3,4,5)$ امتیاز ایجاد یال از واژهٔ «plays» به «.» و از واژهٔ «plays» بـه «Elianti» اسـت. توابـع امتیـاز نسبت به چپ یا راست والد تعریف می‌شوند و به یال‌ها مجاور در دو سمت مختلف والد اعمال نمی‌شوند. بعنوان مثال $s(3,2,5)$ برای یال‌های مجـاور از «plays» بـه «Haag» و «Elianti» وجـود نـدارد. امتیـاز $s(x_i,-,x_j)$ زمانی که x_j اولین وابستهٔ سمت چپ یا اولین وابستهٔ سمت راست واژهٔ x_i باشد، محاسـبه می‌شود. به صورت رسمی‌تر، واژهٔ x_{i_0} را در نظر بگیرید که فرزندانی بصورت زیر داشته باشد.



[1] Higher-order feature representation

در این صورت امتیاز مرتبهٔ دوم به شکل زیر خواهد بود.

$$\sum_{k=1}^{j-1} s(i_0, i_{k+1}, i_k) + s(i_0, -, i_j) + s(i_0, -, i_{j+1}) + \sum_{k=j+1}^{m-1} s(i_0, i_k, i_{k+1})$$

امتیاز کلی درخت در الگوی مرتبهٔ دوم $s(x,y) = \sum_{(i,k,j)\in y} s(i,k,j)$ خواهد بود که در آن k و j فرزندان مجاور در یک سمت فرزندان درخت y هستند.

در مقالهٔ [۱۳] الگوی تجزیه را با انواع دیگر روابط مرتبهٔ دوم توسعه داده شده است که در آن تعامل بین واژهٔ سَر و وابسته و همچنین فرزندان وابسته را در نظر می‌گیرد. به عنوان مثال در شکل (۳-۲۴) تعامل سه تایی «root»، «plays» و «Haag» یک خصوصیت مرتبهٔ دوم از این نوع به حساب می‌آید.

در مقالهٔ [۳۱] از یک راهکار رده‌بندی ساختار مبتنی بر بخش[1] برای تجزیهٔ وابستگی استفاده شده است. برای جملهٔ داده شدهٔ x در نظر بگیرید Y(x) مجموعهٔ ساختارهای وابستگی پوشای x است که در آن $y \in Y(x)$ به بخش‌های $r \in y$ تجزیه می‌شود. در ساده‌ترین حالت، این بخش‌ها می‌توانند یال‌های وابستگی بین واژه سَر و وابسته باشند که منجر به الگوی «مرتبهٔ اول» یا «مبتنی بر یال» می‌شود. در الگوهای تجزیهٔ مرتبهٔ بالاتر این بخش‌ها می‌توانند شامل تعامل بین بیش از دو واژه باشند. به عنوان مثال برای الگوی مقالهٔ [۱۴] بخش‌ها را تعامل بین سیبلینگ‌ها و برای مقالهٔ [۱۳] بخش‌ها را تعامل بین سیبلینگ‌ها و همچنین تعامل سَر با فرزندان وابسته تعریف کرد. این نوع الگوی مرتبهٔ بالاتر به تجزیه‌کنندهٔ وابستگی اجازه می‌دهد که فرم محدود شده‌ای از حساسیت به بافت[2] را بدست آورد.

◘ تجزیه‌کننده‌های مبتنی بر گراف موجود

- MSTParser: یک تجزیه‌کنندهٔ وابستگی غیرافکنشی که توسط مک‌دونالد[3] در دانشگاه پنسیلوانیای آمریکا توسعه یافته است [۳۲] که به زبان جاوا نوشته شده و به صورت متن باز موجود است[4].
  - برای یادگیری از الگوریتم آن‌لاین با حاشیهٔ بیشینه «MIRA» استفاده می‌کند.
  - به منظور تجزیه امکان انتخاب الگوریتم «آیزنر» [۱۵] و «چو-لیو-ادموندز» [۱۶, ۱۷] را دارد که هر دو الگوریتم نسخه‌های الگوریتم درخت پوشای بیشینه هستند. به صورت پیش‌فرض از الگوریتم چو-لیو-ادموندز استفاده می‌کند زیرا هم روابط بیش‌تری را پوشش می‌دهد و هم

[1] Part-factored
[2] Context-sensitivity
[3] McDonald
[4] http://mstparser.sourceforge.net

صحت بالاتری دارد.

۳-۲-۳- روش‌های ترکیبی

راه‌های قابل تصور زیادی برای ترکیب دو تجزیه کننده وجود دارد. در مقالهٔ [۳۳] الگوهای ترکیبی را به دو دسته تقسیم کرده است:

۱) الگوهایی که تجزیه‌کننده‌های پایه‌ای[1] را در زمان یادگیری ترکیب می‌کنند.

۲) الگوهایی که تجزیه‌کننده‌های پایه‌ای به صورت مستقل آموزش می‌بینند و سپس در زمان تجزیهٔ ترکیب می‌کنند.

ترکیب الگوها در زمان تجزیه، ساده‌تر از ترکیب آن‌ها در زمان یادگیری است اما ترکیب تجزیه‌کننده‌ها در زمان تجزیهٔ راهکارهای جذاب‌تری هستند.

◘ ترکیب در زمان یادگیری

- تجزیه‌کننده‌های پشته‌سازی[2]: ایده این کار اولین بار توسط نیور و مک‌دونالد [۳۴، ۳۵] ارائه شده است. نیور تجزیه‌کنندهٔ مبتنی بر گذار MaltParser و مک‌دونالد تجزیه‌کنندهٔ مبتنی بر گراف MSTParser را پیشنهاد دادند. پیش از این در مقالهٔ [۳۶] ضمن تحلیل خطا تجزیه‌کننده‌های وابستگی پیشرفته مبتنی بر داده در مقیاس بزرگ، نتیجه‌گیری کردند که توزیع خطاهای تجزیه مرتبط با خواص تئوری الگوهای استفاده شده برای «یادگیری» و «استنتاج» است. نهایت نشان دادند که چگونه می‌توان از این نتایج برای بهبود صحت تجزیه با ترکیب این الگوهای بهره برداری کرد. ترکیب این دو سیستم نه تنها صحت کلی را بهبود می‌دهد بلکه مستقیماً از نقاط قوت هر کدام از سیستم‌ها بهره می‌برد. ایده اصلی این روش اجازه دادن به یک الگو برای تولید خصوصیات برای الگوی دیگر است که با این کار صحت از هر دو الگو بهتر خواهد شد.

در «تجزیهٔ مبتنی بر گذار»، الگوی تجزیه بر اساس گذارها از یک حالت به حالت دیگر در یک ماشین حالت انتزاعی تعریف می‌شود. «آموزش» این الگو نوعاً با استفاده از تکنیک‌های رده‌بندی استاندارد انجام می‌شود که در آن‌ها پیش‌بینی یک گذار از میان مجموعهٔ گذارهای ممکن در تاریخچهٔ حالت، آموخته

[1] Base parser
[2] Stacking

می‌شود. «استنتاج» به صورت محلی است زیرا چنین سیستم‌هایی با حالت‌های اولیه محدود شروع بکار کرده و به صورت حریصانه گراف را با انتخاب گذارها با بیش‌ترین امتیاز در هر حالت می‌سازد و تا زمانی که به شرط خاتمه برسد، ادامه می‌دهد.

در «تجزیهٔ مبتنی بر گراف»، الگوی تجزیه بر اساس زیر گراف‌های وابستگی تعریف می‌شود. «آموزش» به صورت سراسری است که الگو به صورت امتیاز عمومی گراف‌های درست، در مقابل نادرست‌ها آموزش می‌بیند. «استنتاج» نیز سراسری است، زیرا تلاش می‌کند گراف با بیش‌ترین امتیاز را از بین مجموعه تمام گراف‌ها بیابد. اکثر راهکارهای این دسته مرتبط با کارهای اولیه آیزنر است.

در هر کدام از این روش‌ها خطاهای متفاوت و مکملی وجود دارد که منجر به دسته‌بندی جداگانه آن‌ها شده است. خطاهای هر سیستم مرتبط با انتظارات مبتنی بر تئوری‌های آن‌هاست. در تحلیل منشأ خطاهای این دو راهکار مقالهٔ [۳۶] به بررسی سه دسته فاکتور اصلی زیر می‌پردازد:

۱) فاکتورهای طول: اکثر الگوهای تجزیه تمایل دارند در جملات با طول[۱] بیش‌تر صحت کمتری داشته باشند. این امر عمدتاً به دلیل افزایش وجود ساخت‌های نحوی پیچیده مانند حروف اضافه، ترکیبات عطفی و غیره است. کارایی این دو الگوی تجزیه قابل تمایز نیست اما MaltParser تمایل دارد روی جملات کوتاه‌تر بهتر عمل کند زیرا الگوریتم استنتاج حریصانه تصمیمات تجزیهٔ کمتری می‌گیرد. در نتیجه احتمال انتشار خطا به صورت چشمگیری هنگام تجزیهٔ این جملات کاهش می‌یابد که به دلیل فضای خصوصیات غنی‌تر نسبت به MSTParser است. نتیجهٔ بررسی‌ها نشان داده است که MSTParser برای یال‌های وابستگی طولانی‌تر به مراتب دقیق‌تر است در حالی که MaltParser در یال‌های وابستگی کوتاه‌تر بهتر است. این رفتار را می‌توان این طور توصیف کرد که یال‌های وابستگی کوتاه‌تر معمولاً در تجزیهٔ حریصانهٔ MaltParser اول تولید می‌شوند و کمتر در معرض انتشار خطا است. در مقابل وابستگی‌های طولانی‌تر نوعاً در گام‌های بعدی الگوریتم تجزیه ساخته می‌شوند و بیش‌تر متأثر از انتشار خطا هستند.

۲) فاکتورهای گراف: ساختار گراف‌های وابستگی پیش‌بینی شده و موجود در پیکرهٔ درختی می‌تواند بین هر الگو متفاوت است. به عنوان مثال محاسبه صحت برای یال‌ها نسبت به فاصلهٔ آن‌ها تا گره ریشهٔ مصنوعی منجر خطاهایی در سطوح مختلف گراف وابستگی می‌شود. برای یک یال داده شده این فاصله را به صورت تعداد یال‌ها در مسیر معکوس تا ریشه محاسبه می‌کنند. برای یال‌های نزدیک به ریشه

[۱] طول وابستگی از w_i به w_j برابر با $|i-j|$ است.

MSTParser خیلی دقیق‌تر است و عکس این مطلب برای یال‌های دورتر از ریشه صادق است. نشان داده شده است که دقت MSTParser با افزایش فاصله تا ریشه کاهش می‌یابد و دقت MaltParser افزایش می‌یابد. اگر نمودار دقت را رسم کنیم، نمودار این دو در جهت خلاف یکدیگر حرکت می‌کند که نقطه تلاقی آن‌ها در میانه است. یال‌های وابستگی دورتر از ریشه معمولاً زودتر در الگوریتم تجزیهٔ MaltParser ساخته می‌شوند.

۳) فاکتورهای زبان‌شناسی: صحت سیستم به مجموعه‌های زبان شناسی مثل برچسب اجزای سخن و انواع وابستگی مرتبط است. نتیجهٔ تحلیل این بوده که صحت MaltParser در مورد «اسامی» و «ضمایر» اندکی بهتر از MSTParser بوده و MSTParser در سایر دسته‌ها مخصوصاً «ترکیبات عطفی» بهتر عمل کرده است.

بر اساس این تحلیل خطای انجام شده الگوی ترکیبی «پشته‌سازی» ارائه شده که ترکیب را در زمان یادگیری انجام می‌دهد و دو سیستم مکمل می‌توانند از یکدیگر بیاموزند. اساس این ترکیب مبتنی بر خصوصیات[1] است. هر دو الگو از یک تابع امتیاز $s: X \rightarrow R$ استفاده می‌کنند با این تفاوت که X در دامنه‌های متفاوتی تعریف می‌شود. برای الگوی مبتنی بر گراف X مجموعه یال‌های وابستگی (i, j, l) است و برای الگوی مبتنی بر گذار X مجموعهٔ جفت حالت-گذار ممکن (c, t) است. در هر دو حالت ورودی با بردار خصوصیت k بعدی $f: X \rightarrow R^k$ بازنمایی می‌شود. در یک تجزیه‌کنندهٔ پشته‌سازی بردار خصوصیات یک الگو که به آن «الگوی پایه‌ای»[2] می‌گویند، توسط تعداد خاصی از خصوصیات تولید شده توسط الگوی دیگر که به نام «الگوی راهنما»[3] شناخته می‌شود، توسعه داده می‌شود. خصوصیات اضافه شده را نیز «خصوصیات راهنما»[4] می‌نامند. نسخه‌ای از الگوی پایه‌ای که توسط بردار خصوصیات توسعه یافته آموزش دیده است را «الگوی هدایت شده»[5] گویند. ایدهٔ اصلی این است که الگوی هدایت شده باید قادر باشد که بیاموزد چه مواقعی باید به «خصوصیات راهنما» اطمینان کرد تا از نقاط قوت «الگوی راهنما» بهره ببرد. در این صورت کارایی تجزیه‌کننده می‌تواند نسبت به الگوی پایه‌ای بهتر شود. با این توصیف دو نوع الگوی هدایت شده زیر را می‌توان تولید کرد:

- الگوی مبتنی بر گراف هدایت شده[6] (MST_{Malt}): در الگوی مبتنی بر گراف MSTParser تابع امتیاز

[1] Feature-based integration
[2] Base model
[3] Guide model
[4] Guide features
[5] Guided model
[6] Guided Graph-based Model

$s(i,j,l) \in \mathrm{R}$ روی وابستگی‌های برچسب‌دار تعریف می‌شود. به صورت دقیق‌تر یال‌های وابستگی (یا جفت یال‌ها) ابتدا توسط بردار خصوصیات با ابعاد بالا $f(i,j,l) \in R^k$ بازنمایی می‌شود که f نوعاً بردار خصوصیت دودویی روی خواص یال است. امتیاز یک یال نیز به صورت رده‌بندی خطی $s(i,j,l) = w.f(i,j,l)$ تعریف می‌شود که بردار وزن w خصوصیتی است که باید آموزش دیده شود. حال در الگوی هدایت شده، این بازنمایی خصوصیات طوری تغییر می‌کند که شامل خصوصیات جدید گراف وابستگی $G_{\mathrm{x}}^{\mathrm{Malt}}$ پیش‌بینی شده توسط MaltParer روی جملهٔ ورودی x باشد. این خصوصیات جدید به صورت خصوصیت با ابعاد بالا $f(i,j,l,G_x^{Malt}) \in R^{k+m}$ بازنمایی می‌شود. این m خصوصیت جدید به عنوان خصوصیت راهنما روی خروجی MaltParser به حساب می‌آورد.

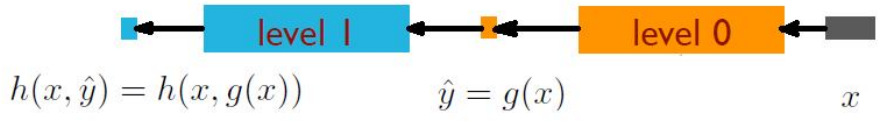
- الگوی مبتنی بر گذار هدایت شده[1] ($Malt_{\mathrm{MST}}$): در الگوی مبتنی بر گذار MaltParser تابع امتیاز $s(c,t) \in R$ روی حالات و گذارها تعریف می‌شود. مجموعهٔ نمونه‌های آموزشی برای این مسئلهٔ یادگیری، مجموعهٔ جفت‌های (c,t) است به صورتی که t گذار درست خروجی در حالت c است توسط رشته‌ای از گذارها گراف وابستگی درست G_{x} برای جملهٔ x تولید می‌شود. هر نمونهٔ آموزشی (c,t) توسط بردار خصوصیت $f(c,t) \in R^k$ بازنمایی می‌شود که این خصوصیات در قالب خواص حالت c شامل حالت پشته σ_c، لیست ورودی β_c و گراف وابستگی نیمه ساخته G_{c} است. حال در الگوی هدایت شده، نمونه‌های آموزشی به سه تایی (c,t,G_x^{MST}) توسعه می‌یابد که G_x^{MST} گراف وابستگی پیش‌بینی شده توسط MSTParser برای جملهٔ x است. در این حالت m خصوصیت راهنما اضافی مبتنی بر G_x^{MST} تعریف می‌شود و بردار خصوصیات به $f(c,t,G_x^{MST}) \in R^{k+m}$ توسعه می‌یابد.

الگوی بدست آمده به این صورت است که در مواقعی که Malt خوب عمل می‌کند، $Malt_{MST}$ اندکی بهتر عمل می‌کند. اما در مواقعی که MST خوب عمل می‌کند، MST_{Malt} اغلب خیلی بهتر است. در حالت کلی صحت MST_{Malt} بهتر از $Malt_{MST}$ است.

در مقالهٔ [۳۷] یک معماری برای تجزیهٔ وابستگی پشته‌سازی پیشنهاد شده است که شامل دو سطح است. شمای کلی این معماری در شکل (۳-۲۵) آمده است. در «سطح ۰» یک تجزیه‌کنندهٔ وابستگی قرار دارد که در زمان اجرا این تجزیه‌کنندهٔ سطح ۰ (g)، رشتهٔ ورودی x را دریافت کرده و در خروجی مجموعه‌ای از یال‌های پیش‌بینی شده را که تخمینی از درخت وابستگی ($\hat{y}_0 = \mathrm{g(x)}$) است را می‌دهد. در «سطح ۱» یک تجزیه‌کنندهٔ وابستگی قرار دارد که از خصوصیات پایه‌ای خود بعلاوهٔ خصوصیات جدید پیش‌بینی یال فراهم شده توسط تجزیه‌کنندهٔ سطح ۰ استفاده می‌کند. تجزیه‌کنندهٔ نهایی درخت تجزیه

[1] Guided Transion-based Model

(h(x,g(x))) را پیش‌بینی می‌کند. بنابراین زمان اجرای کلی مجموع محاسبات h(.) و g(.) خواهد بود.



شکل (۳-۲۵) معماری سیستم تجزیهٔ وابستگی پشته‌سازی

همچنین چارچوبی برای آموزش این سیستم ارائه شده است. فرض کنید D مجموعهٔ نمونه‌های آموزشی $\{\langle x_i, y_i \rangle\}_i$ باشد. آنگاه برای آموزش سیستم مراحل زیر طی می‌شود:

۱) آماده‌سازی دادگان[1] برای آموزش تجزیه‌کنندهٔ سطح ۱

- داده‌های آموزشی D به L قسمت D^1 تا D^L تقسیم می‌شود.
- تعداد L نمونه از تجزیه‌کنندهٔ سطح ۰ را به صورت زیر آموزش داده می‌شود.
  - ❖ نمونهٔ l تجزیه‌کننده (g^l) را روی کل پیکره بدون قسمت l دادگان $(D^{-l} = D \backslash D^l)$ آموزش داده می‌شود.
  - ❖ پس از آموزش g^l از آن برای پیش‌بینی بخش دیده نشدهٔ D^l استفاده می‌شود.
- در پایان دادگان تقویت شده[2] $\tilde{D} = \{\langle x_i, g(x_i), y_i \rangle\}_i$ ساخته شده بطوریکه $\tilde{D} = \mathrm{U}_{l=1}^{L} \tilde{\mathrm{D}}^l$ باشد.

۲) آموزش تجزیه‌کننده‌های هر دو سطح

- تجزیه‌کنندهٔ سطح ۰ (g) را روی دادگان اصلی D آموزش داده می‌شود.
- تجزیه‌کنندهٔ سطح ۱ (h) را روی دادگان تقویت شده $\tilde{D}$ آموزش داده می‌شود.

زمان اجرای این الگوریتم $O(LT_0 + T_1)$ خواهد بود که T_0 و T_1 به ترتیب زمان اجرای آموزش تجزیه‌کنندهٔ سطح ۰ و ۱ هستند.

چارچوب ارائه روی سه نوع الگوی ترکیبی زیر انجام شده است:

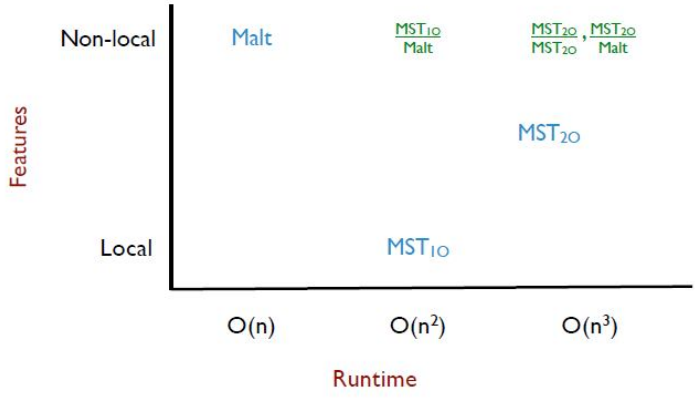
- الگوی پشته‌ای ۱: تجزیه‌کنندهٔ هر دو سطح MSTParser با تقریب مرتبهٔ دوم $(\frac{\mathrm{MST}_{20}}{\mathrm{MST}_{20}})$
- الگوی پشته‌ای ۲: برای تجزیه‌کنندهٔ سطح ۰ از MaltParser و سطح ۱ از MSTParser با

[1] Dataset
[2] Augmented

تقریب مرتبهٔ دو ($\frac{\text{MST}_{20}}{\text{Malt}}$)

- الگوی پشته‌ای ۳: برای تجزیه‌کنندهٔ سطح ۰ از MaltParser و سطح ۱ از MSTParser و مرتبهٔ اول ($\frac{\text{MST}_{10}}{\text{Malt}}$)

نتایج بررسی این سه الگو در شکل (۳-۲۶) آمده است. همان‌طور که از شکل الگوی پشته‌ای ۳ پیچیدگی زمانی کمتری نسبت به دو حالت دیگر دارد. پیشنهاد شده که با انتخاب تجزیه‌کننده سطح ۱ از نوع مبتنی بر یال ساده، پیچیدگی زمانی $O(n^2)$ خواهد شد که در اینصورت صحیح‌تر روش‌های تقریبی خواهد بود. علاوه بر این نشان داده شده است که الگوی پشته‌ای می‌تواند بهتر از تجزیه‌کننده‌هایی که مرتبهٔ دوم را تقریب می‌زنند [۱۴]، عمل کند.



شکل (۳-۲۶) بررسی روش‌های مختلف ساخت الگوی پشته‌ای

◘ ترکیب در زمان تجزیه

- تجزیه‌کننده‌های رای گیری[1]: این تکنیک اولین بار در مقالهٔ [۳۸] عرضه شد و بعدها در مقالهٔ [۳۹] تصحیح و بهبود داده شده است. ایدهٔ اصلی استفاده از مفهوم رای اکثریت[2] است که حداقل نیاز به سه تجزیه‌کننده دارد. اگر دقیقاً سه تجزیه‌کننده موجود باشد بررسی موقعیت‌های زیر مورد نظر است:
  ۱) دو تجزیه‌کننده رأی مخالف دیگری دهند.
  ۲) حالت تساوی که هر تجزیه‌کننده رأی متفاوتی دهند.

مشکل اصلی این است که امکان دارد خروجی منجر به درخت وابستگی معتبر نشود. به عبارت دیگر این الگوی، تضمین می‌کند که مجموعهٔ نهایی وابستگی‌ها بیش‌ترین رأی ممکن را داشته باشند اما تضمین

[1] Voting
[2] Majority vote

نمی‌کند که مجموعهٔ وابستگی‌های رأی آوردهٔ نهایی درخت وابستگی خوش‌ساخت باشد. در حقیقت گراف نهایی ممکن است حتی همبند نبوده و دور در آن وجود داشته باشد. در مقالهٔ [۳۹] نشان داده که این مسئله را می‌توان به حالت خاصی از مسئلهٔ «درخت پوشای بیشینه» که ایدهٔ آن توسط مک‌دونالد [۳۰] برای تجزیهٔ وابستگی مطرح شده، تبدیل کرد. به دلیل استفادهٔ مجدد از یک روش تجزیه، نام راهکار ارائه شده را «تجزیهٔ مجدد»[1] نام گذاشته است. در این راهکار تمام یال‌های وابستگی پیشنهاد شده توسط تجزیه‌کننده‌های مختلف در گراف ذخیره می‌شود و وزن هر یال برابر با تعداد رأی‌های داده شده به آن خواهد بود. نتیجهٔ محاسبه درخت پوشای بیشینه در این گراف، درخت وابستگی بهینه خواهد بود. این الگوی وزن دهی را «الگوی وزن‌دهی پایه‌ای» نام‌گذاری کردند.

در مقالهٔ [۴۰] نشان داده شده است که صحت الگوی رای گیری را می‌توان بازهم بهبود داد. برای این منظور بهینه‌سازی‌هایی در روش وزن‌دهی به رأی‌ها ارائه کرده است. سه خصوصیت که می‌تواند امتیاز را بهبود دهند به صورت زیر معرفی شده است:

۱) رابطهٔ وابستگی واژهٔ وابسته (DEPREL)

۲) برچسب اجزای سخن واژهٔ سَر (H-POS)

۳) رابطهٔ وابستگی واژهٔ سَر (H-DEPREL)

ترکیباتی شامل زیرمجموعهٔ این سه خصوصیت با برچسب اجزای سخن واژهٔ اصلی استفاده شده است.

POS + DEPREL POS + H-POS POS + H-DEPREL

حاصل آزمایشات این است که برچسب اجزای سخن که در الگوی پیش‌فرض استفاده شده است، مهم‌ترین خصوصیت برای وزن‌دهی است. اما سیستم می‌تواند از ترکیب آن با سایر خصوصیات بهره ببرد.

بهینه‌سازی دیگری که انجام شد تلاش برای اعمال «یادگیری کاهش تدریجی»[2] به مسئلهٔ یافتن وزن بهینه است. برای این منظور تابع خطا به صورت زیر تعریف شده است:

$$\varepsilon = \sum_{i} \left(w_i^{ref} - w_i^{hyp} \right)^2$$

که در این رابطه w_i^{ref} وزن مرجع است. اگر یال در تجزیه‌کنندهٔ مرجع باشد ۱ و گرنه ۰ است. همچنین w_i^{hyp} وزن جاری است. نتایج آزمایش بهبود چشمگیر نسبت به الگوی پیش‌فرض نشان نداد که این امر نشان می‌دهد الگوی پیش‌فرض یا بهینه است یا نزدیک به بهینه عمل می‌کند.

[1] Reparsing
[2] Gradient descent learning

تجزیه‌کننده‌های ترکیبی موجود

- MSTParserStacked: یک تجزیه‌کنندهٔ وابستگی ترکیبی است که توسط مارتینز[1] و داس[2] توسعه یافته است [۳۷] که به زبان جاوا نوشته شده و به صورت متن باز موجود است[3]. این تجزیه‌کننده توسعه‌ای بر MSTParser است که الگوی پشته‌سازی را پیاده‌سازی کرده است.

۳-۳- تجزیهٔ وابستگی مبتنی بر دستور

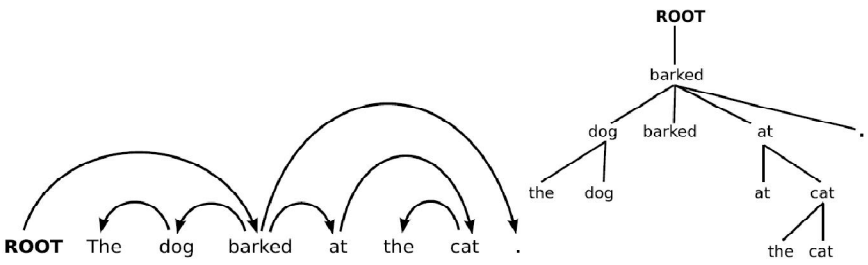
در تجزیهٔ مبتنی بر دستور، در الگوی تجزیهٔ $M = (\mathbb{\Gamma}, \lambda, \mathrm{h})$ برای $\mathbb{\Gamma}$ به جای تعریف مجموعه‌ای از محدودیت‌ها، یک دستور زبان «تعریف‌شده» استفاده می‌شود. در این صورت عمل تجزیه به معنای تحلیل یک جمله با توجه به دستور زبان موجود و شاخص‌های موجود در $h\left(S, \mathbb{\Gamma}, \boldsymbol{\lambda}\right)$ 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 تجزیه‌کننده نتواند خروجی تولید کند، بدین معنا خواهد بود که جملهٔ مورد تجزیه، عضو زبان تعریف شده نیست. رویکرد دیگری که در تجزیهٔ مبتنی بر دستور 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.

۳-۳-۱- تجزیهٔ وابستگی مستقل از متن

این نظریه نخستین بار از سوی گِیف‌من[4] و هِیز[5] 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[6] شامل واژه‌ها می‌شود. در شکل (۳-۲۷) نمونه‌ای از یک درخت وابستگی افکنشی و معادل درخت تجزیهٔ مستقل از متن آن به نمایش

[1] Martins
[2] Das
[3] http://www.ark.cs.cmu.edu/MSTParserStacked
[4] Gaifman
[5] Hays
[6] Non-terminal symbols

گذاشته شده است.



شکل (۳-۲۷) نمونه‌ای از درخت وابستگی افکنشی و معادل مستقل از متن آن [۸]

یکی از حسن‌های استفاده از دستور زبان مستقل از متن، وجود الگوریتم‌های معروف تجزیهٔ سی‌کی‌وای و اِرلی است. روش‌های مستقل از متن به دستهٔ متعارف و دوسویه[1] تقسیم می‌شوند:

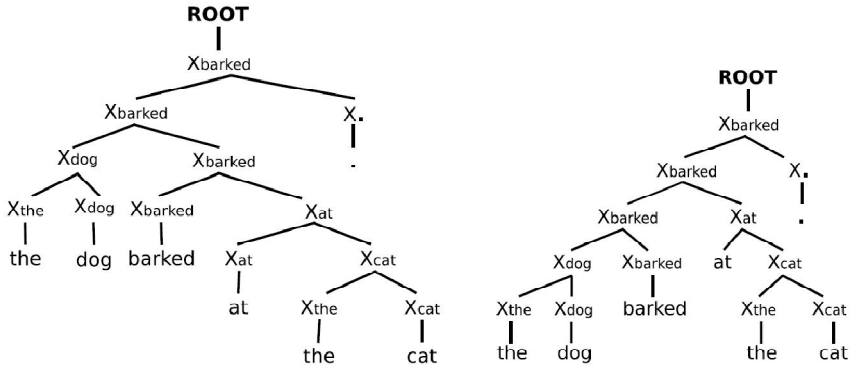
- دستور زبان مستقل از متن متعارف: یک دستور زبان مستقل از متن متعارف $\mathbb{\Gamma}$ به صورت چهارتایی مرتب $(N, \sum, \prod, start)$ تعریف می‌شود، به طوری که:
  - N: مجموعه‌ای محدود از نمادهای غیرپایانی است.
  - $\sum$: مجموعه‌ای محدود از نمادهای پایانی است.
  - $\prod$: مجموعه‌ای قواعد تولید زبان به شکل $X \to \alpha$ است، به طوری که $X \in N$ یک نماد غیرپایانی و α رشته‌ای از نمادهای پایانی و غیرپایانی است.
  - $start \in N$: نماد آغازین است.
- دستور زبان مستقل از متن دوسویه: یک دستور زبان مستقل از متن دوسویه $\mathbb{\Gamma}_B$، دستور مستقل از متن متعارفی 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.

نیور در مقالهٔ [۴۱] ضمن مقایسهٔ این دو روش، بیان می‌کند که 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 .

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 سَر بودن واژه مورد بررسی قرار بگیرد. به همین دلیل پیچیدگی

[1] Bilexical

زمانی به صورت $O(n^3.\min(|\prod|, n^2)$ خواهد بود. از آن‌جایی که $|\prod|$ از n^2 بیش‌تر است، پیچیدگی زمانی به اندازه $O(n^5)$ است. در عین حال در حین تغییر ممکن است چند نوع درخت متفاوت تولید شود؛ به عنوان مثال شکل (۳-۲۸) دو نوع درخت وابستگی دوسویه که برای یک جملهٔ انگلیسی قابل تولید است را نشان می‌دهد.



شکل (۳-۲۸) دو نوع درخت وابستگی دوسویهٔ ممکن برای یک جملهٔ انگلیسی [۸]

برای اجتناب از دو مشکل «پیچیدگی زمانی بالا» و «امکان ساخت چند درخت برای یک جمله»، از نمایش انشعابِ سَر[1] استفاده می‌شود. در این نمایش هر واژهٔ پایانی w_i از درخت وابستگی به صورت دو واژهٔ پایانی w_i^r و w_i^l 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.

$$R_i^r \rightarrow w_i^r, L_i^l \rightarrow w_i^l$$

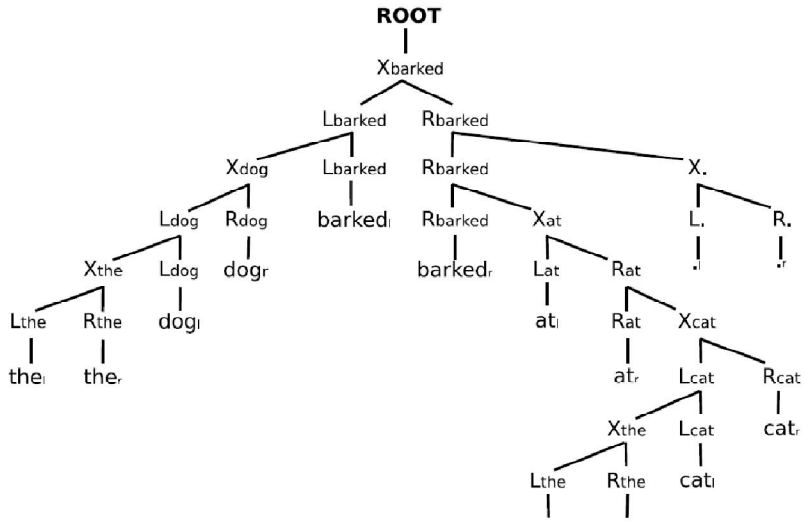
در این صورت برای ساخت یک واژهٔ پایانی، قانون $X_i \rightarrow L_i^l R_i^r$ اضافه خواهد شد. علاوه بر این به دو قانون دیگر برای نشان دادن وابستگی چپ و راست نیاز است:

$$L_i \rightarrow X_j L_i$$
$$R_i \rightarrow R_i X_j$$

در نهایت نیز به قانونی برای ساخت ریشه نیاز است:

بر اساس این قوانین نمایش انشعاب سَر درخت شکل (۳-۲۷) به صورت شکل (۳-۲۹) خواهد شد.

[1] Split-Head



شکل (۳-۲۹) درخت وابستگی با نمایش انشعاب سَر برای جملهٔ شکل (۳-۲۷) [۸]

با استفاده از نمایش انشعاب سَر 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[1] استفاده می‌شود که از برنامه‌ریزی کارکردی[2] اقتباس شده است. در این روش به جای X_i معادل آن یعنی $L_j^l R_j^r$ جایگذاری می‌شود:

$$L_i \rightarrow L_j^l R_j^r L_i$$
$$R_i \rightarrow R_i L_j^l R_j^r$$

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

$$L_i \rightarrow L_j M_{i,j}$$
$$R_i \rightarrow M_{i,j} R_j$$
$$M_{i,j} \rightarrow R_i L_j$$

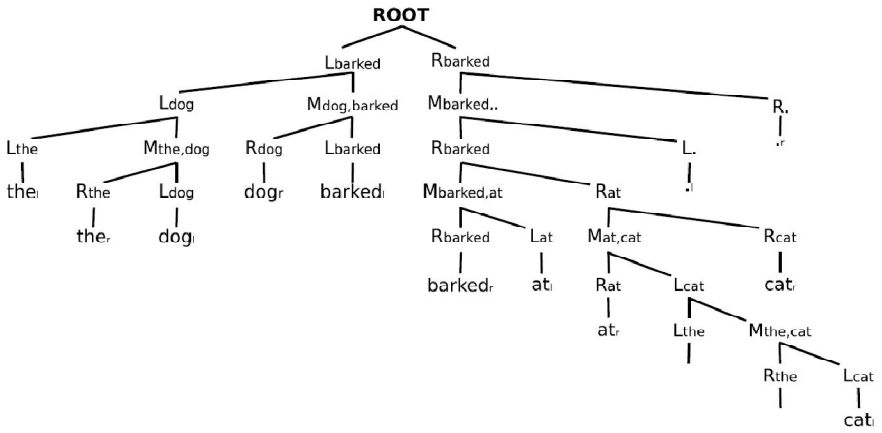
در نهایت قانون ساخت ریشه به صورت زیر خواهد بود:

$$ROOT \rightarrow L_i R_i$$

درخت حاصل از اعمال روش گشودن-تا کردن به صورت شکل (۳-۳۰) در خواهد آمد. با تبدیلات پیشنهاد شده، پیچیدگی زمانی به $O(n^3)$ کاهش می‌یابد. با توجه به این که هر دستور زبان وابستگی افکنشی را به صورت دستور زبان مستقل از متن می‌توان 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[3] استفاده

[1] Unfold-Fold Transformation
[2] Functional Programming
[3] PCFG: Probabilistic Context-Free Grammar

کرد.



شکل (۳-۳۰) درخت وابستگی با روش گشودن-تا کردن [۸]

۳-۳-۲- تجزیهٔ وابستگی مبتنی بر محدودیت

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

۱) مجموعه‌ای از متغیرهای $x = w_0 w_1 \dots w_n$ که نشان‌دهندهٔ مجموعهٔ واژه‌های جمله است.

۲) دامنهٔ متغیرهای w_i که مجموعه $\{w_j | i \neq j \ \& \ 1 \leq j < n\}$ سَرهای ممکن واژه است.

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

از آن جایی که مجموعه مسائل ارضای محدودیت جزو مسائل نمایی غیرقطعی کامل هستند، به استفاده از روش‌های ابتکاری در حل آن‌ها نیاز است.

- 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*[1] یا *فسخ‌شدنی*[2] به مجموعه قواعد اضافه می‌شود. بنابراین برای هر محدودیت c یک وزن λ_c (وزن صفر برای سخت‌ترین محدودیت و وزن یک برای نرم‌ترین محدودیت) 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.

$$weight\ (G) = \prod_{c \in C} \lambda_c$$

- تجزیهٔ وابستگی محدودیت مبتنی بر تبدیل

نتایج حاصل از روش‌های انتشار محدودیت[3] و محدودیت وزن‌دار جالب است ولی مشکلاتی در این روش‌ها وجود دارد. مشکل اصلی این روش‌ها امکان از دست رفتن پاسخ درست به دلیل استفاده از جست و جوهای ابتکاری در فضای حالت محدودیت‌هاست. ساز و کاری برای حل این مشکل در نظر گرفته شده است. در این روش طی هر مرحله از تجزیه یک جمله با استفاده از وزن محدودیت‌ها فرایند تجزیه بهبود می‌یابد. در این روش از کم‌ترین وزن (سخت‌ترین محدودیت) اصلاح آغاز می‌شود. نکتهٔ جالب دربارهٔ این روش *ویژگی هرزمانی*[4] است؛ یعنی در هر زمان دلخواه قابل پایان است.

۳-۴- نتیجه‌گیری

در این فصل مهم‌ترین الگوریتم‌های موجود در تجزیهٔ وابستگی را بررسی کردیم. ابتدا الگوریتم‌های تجزیهٔ مبتنی بر داده، شامل روش‌های مبتنی بر گذار و گراف، و سپس الگوریتم‌های مبتنی بر دستور، شامل روش‌های مستقل از متن و مبتنی بر محدودیت، را مورد بررسی قرار دادیم. در پایان هر بخش پیاده‌سازی‌هایی که به صورت متن باز وجود دارند معرفی شدند.

[1] Soft Constraint
[2] Defeasible
[3] Constraint Propagation
[4] Anytime Property

فصل ۴:

جمع‌بندی و کارهای آینده

۴-۱- جمع‌بندی

تجزیهٔ وابستگی راهی برای تجزیهٔ نحوی زبان طبیعی است که به صورت خودکار به تجزیه و تحلیل ساختار وابستگی جملات پرداخته و برای هر جملهٔ ورودی یک گراف وابستگی ایجاد می‌کند. انواع روش‌های ارائه‌شده را می‌توان به دو دستهٔ کلی تقسیم کرد:

۱) تجزیهٔ وابستگی مبتنی بر داده
   ۱) تجزیهٔ مبتنی بر گذار
   ۲) تجزیهٔ مبتنی بر گراف
۲) تجزیهٔ وابستگی مبتنی بر دستور
   ۱) تجزیهٔ مستقل از متن
   ۲) تجزیهٔ مبتنی بر محدودیت

در سال‌های اخیر اکثر تحقیقات روی روش‌های مبتنی بر داده متمرکز شده است. تجزیه‌کننده‌های مبتنی بر داده از روش‌های یادگیری ماشینی استفاده می‌کنند. بر همین اساس روش‌های این دسته می‌توانند از الگوریتم‌های یادگیری باناظر، بی‌ناظر، نیمه‌ناظر و تقویتی بهره ببرند. از الگوریتم یادگیری تقویتی تنها در چند مقالهٔ خاص استفاده شده است.

یکی از رویکردهای غالب دیگر در این حوزه روش‌های ترکیبی است که تلاش می‌کند روش‌های مبتنی بر داده و گذار را با هم ترکیب کنند که به دو صورت ترکیب در زمان یادگیری و در زمان تجزیه انجام می‌شود.

۴-۲- کارهای آینده

تجزیه‌کننده‌های ارائه شده صحت بالایی ارائه کردند اما اکثر راهکارهای این حوزه تنها روی مجموعهٔ نسبتاً کوچکی از زبان‌ها (عمدتاً انگلیسی) انجام شده است. در این میان تنها کار مشترکی در همایش سالانهٔ یادگیری رایانه‌ای زبان طبیعی، ۱۳ زبان مختلف را مورد ارزیابی قرار دادند. یک مسئلهٔ مهم در این حوزه این است که الگوها و الگوریتم‌ها را طوری توسعه دهیم تا خصوصیات مناسب خاص زبان یا گروهی از زبان‌ها بدست آید. اگرچه تجزیه‌کننده‌های موجود صحت خوبی دارند اما اِعمال آن‌ها به زبان جدید اغلب منجر به کاهش چشمگیر صحت می‌شود [۴۲]. بررسی اثرات عوامل مختلف در طراحی تجزیه‌کننده‌های مبتنی بر داده برای استفاده از آن‌ها در زبان‌های مختلف امری ضروری است.

در میان جنبه‌های مختلف مؤثر در صحت تجزیه‌کننده، نقطهٔ مشترک در این نکته است که خصوصیات خاص زبان می‌تواند نقش کلیدی در بهبود کارایی عمومی تجزیه داشته باشد. در زبان‌های مختلف اطلاعات نحوی مرتبط به روش‌های گوناگونی رمزگذاری می‌شوند و این فرض وجود دارد که ترکیب اطلاعات ساخت‌واژی[1] و نحوی نکتهٔ کلیدی برای رسیدن به صحت بهتر است. همچنین قابل ذکر است که ترکیب این خصوصیات خاص زبان در تجزیه، همواره آسان نیست و بیش‌تر خصوصیات همواره به صورتی که مورد انتظار است، در زبان‌های مختلف کار نمی‌کنند [۴۳].

تلاش‌هایی برای مدیریت جنبه‌هایی از زبان‌های مختلف که قابل پیکره‌بندی نیستند صورت گرفته که نتایج حاصل از این تحقیقات، کمبودهای موجود در روش‌های تجزیهٔ سنتی را نشان داده است [۴۴]. بر همین اساس بررسی بازنمایی‌های ساخت‌واژی و اثرات متقابل آن با الگوریتم‌های تجزیهٔ وابستگی اخیراً آغاز شده است [۴۵]. با وجود اینکه در زبان فارسی هنوز کاری صورت نگرفته است، تحقیقات در زبان‌های ترکی [۴۲]، هندی [۴۳، ۴۶]، آلمانی [۴۷]، کره‌ای [۴۸]، باسک [۴۹، ۵۰]، عربی [۵۱، ۵۲]، و عبری [۵۳، ۵۴] بر روی تجزیه‌کننده‌های مبتنی بر داده انجام شده است.

در این مقالات نشان داده شده است که چطور باید خصوصیات مفید برای تجزیهٔ وابستگی را از زبان مورد نظر استخراج کرد. پس از آن تأثیر برچسب‌های ساخت‌واژی جزئی[2] و برچسب‌های ساخت‌واژی کلی[3] روی تجزیهٔ وابستگی بررسی شده است. نشان داده شده است که داشتن یک برچسب‌های ساخت‌واژی جزئی، مفید است و می‌تواند منجر به بهبود صحت تجزیه شود [۵۵]. اگرچه برچسب‌های ساخت‌واژی بسیار جزئی نیز لزوماً به معنی یک ساخت‌واژهٔ بهتر برای تجزیه نیست، زیرا ممکن است منجر به از دست دادن مفهوم کلی واژه شود. بنابراین بررسی اینکه هر گونه از ویژگی‌های ساخت‌واژی چقدر روی تجزیهٔ وابستگی موثر است، مفید خواهد بود.

در زبان‌های بی‌ترتیب استفاده از تجزیهٔ وابستگی به سایر روش‌ها ترجیح داده می‌شود. از آنجایی که زبان فارسی بی‌ترتیب است[4]، به سراغ تجزیهٔ وابستگی باید رفت. در بررسی‌های انجام شده در زبان‌هایی کاهش صحت در استفاده از تجزیه‌کننده‌های موجود رخ می‌دهد که از نظر ساخت‌واژی غنی هستند که زبان فارسی نیز جزو این دسته از زبان‌هاست. اهمیت دیگر این موضوع اینجاست که پیش از این پیکرهٔ وابستگی برای

[1] Morphological
[2] Fine-grained morphology
[3] Coarse-grained morphology
[4] با توجه به شمس‌فرد(م. شمس‌فرد، "پردازش متون فارسی: دستاوردهای گذشته، چالش‌های پیش رو"، دومین کارگاه پژوهشی زبان فارسی و رایانه، ۱۳۸۴.)، زبان فارسی جزء زبان‌های بی‌ترتیب است.

زبان فارسی وجود نداشت و اخیراً پیکرهٔ درختی زبان فارسی [۴]، ارائه شده است. علاوه بر آن پیکرهٔ درختی دیگری نیز برای زبان فارسی در حال توسعه است [۵۶]. هنوز هیچ تلاشی در این زمینه مخصوص زبان فارسی انجام نشده است. بنابراین ضروری است پیش از آغاز تحقیقات بر روی این پیکره، عوامل موثر در صحت تجزیهٔ زبان فارسی مورد ارزیابی قرار گیرد. بر این اساس پروژه‌ای تحت عنوان «**ارائه ساز و کاری برای کشف تأثیر ویژگی‌های مختلف ساخت‌واژی و صرفی بر روی تجزیهٔ وابستگی زبان فارسی**» پیشنهاد می‌شود.

مراجع

[۱] ا. طبیب زاده، ظرفیت فعل و ساخت‌های بنیادین جمله در فارسی امروز، نشر مرکز، ۲۰۰۷.

[2] J. Nivre, "Dependency grammar and dependency parsing", *MSI report,* vol. 5133, 2005.

[3] J. Nivre, "Inductive dependency parsing", Springer Verlag, 2006.

[4] M .S. Rasooli*, et al.*, "A Syntactic Valency Lexicon for Persian Verbs: The First Steps towards Persian Dependency Treebank", *5th Language & Technology Conference (LTC): Human Language Technologies as a Challenge for Computer Science and Linguistics,* Poznań ,Poland, pp. 227-231, 2011.

[5] M. A. Covington, "A fundamental algorithm for dependency parsing", in *Proceedings of the 39th Annual ACM Southeast Conference*, Athens, Georgia, USA, pp. 95-102, 2001.

[6] J. Nivre and J. Nilsson, "Pseudo-projective dependency parsing", *In Proceedings ACL*, pp. 99-106, 2005.

[7] J. D. Choi, "Dependency Parsing", 2009.

[8] S. Kübler*, et al.*, "Dependency parsing", *Synthesis Lectures on Human Language Technologies,* vol. 1, pp. 1-127, 2009.

[9] J. Nivre, "Bare-Bones Dependency Parsing - A Case for Occam's Razor?" , In *NODALIDA 2011 Conference Proceedings*, 2011.

[10] J. Nilsson*, et al.*, "The CoNLL 2007 shared task on dependency parsing", *In Proceedings of EMNLP-CoNLL 2007*, pp. 915-932, 2007.

[11] J. Hall*, et al.*, "Single malt or blended? A study in multilingual parser optimization", *Trends in Parsing Technology,* pp. 19-33, 2011.

[12] S. Marinov, "Covington variations", In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, Prague, pp. 1144-1148, 2007.

[13] X. Carreras, "Experiments with a higher-order projective dependency parser", In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, Prague, pp. 957–961, 2007.

[14] R. McDonald and F. Pereira, "Online learning of approximate dependency parsing algorithms", *In Proceedings of EACL*, pp. 81–88, 2006.

[15] J. M. Eisner, "Three new probabilistic models for dependency parsing: An exploration", *In the 16th International Conference on Computational Linguistics*, Copenhagen, pp. 340-345, 1996.

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

[17] J. Edmonds, "Optimum branchings", *Journal of Research of the National Bureau of Standards,* 1968.

[18] H. Yamada and Y. Matsumoto, "Statistical dependency analysis with support vector machines", *In Proceedings IWPT*, 2003.

[19] C. Chih-Chung and L. Chih-Jen, "LIBSVM: a library for support vector machines", *Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm,* 2001.

[20] J. Nivre*, et al.,* "Memory-based dependency parsing", *In Proceedings of CoNLL*, pp. 49–56, 2004.

[21] W. Daelemans*, et al.*, "TiMBL: Tilburg Memory-Based Learner, version 5.0, Reference Guide", 2003.

[22] M. Collins, "Discriminative training methods for hidden markov models :Theory and experiments with perceptron algorithms", pp. 1-8, 2002.
[23] G. Attardi, *et al.*, "Multilingual dependency parsing and domain adaptation using DeSR", pp. 1112–1118, 2007.
[24] X. Carreras, *et al.*, "Projective dependency parsing with perceptron", *In Proceedings CoNLL-X*, pp. 181-185, 2006.
[25] J. Nivre, "Incrementality in deterministic dependency parsing", *In Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together (ACL)*, pp. 50-57, 2004.
[26] J. Nivre and M. Scholz, "Deterministic dependency parsing of English text", *In Proceedings COLING*, 2004.
[27] J. Nivre, "Incremental non-projective dependency parsing", In *Proceedings of NAACL HLT 2007*, Rochester, NY, pp. 396-403, 2007.
[28] J. Nivre, *et al.*, "MaltParser: A language-independent system for data-driven dependency parsing", *Natural Language Engineering,* vol. 13, pp. 95-135, 2007.
[29] R. McDonald, *et al.*, "Online large-margin training of dependency parsers", *In Proceedings ACL* ,pp. 91-98, 2005.
[30] R. McDonald, *et al.*, "Non-projective dependency parsing using spanning tree algorithms", *In Proceedings of HLT/EMNLP*, pp. 523-530, 2005.
[31] T. Koo, *et al.*, "Simple semi-supervised dependency parsing", *In Proceedings ACL/HLT* ,pp. 595–603, 2008.
[32] R. McDonald, *et al.*, "Multilingual dependency analysis with a two-stage discriminative parser", pp. 216-220, 2006.
[33] M. Surdeanu and C. D. Manning, "Ensemble models for dependency parsing: cheap and good?", *In NAACL*, pp. 649-652, 2010.
[34] J. Nivre and R. McDonald, "Integrating graph-based and transition-based dependency parsers", *Proceedings of ACL-08: HLT,* pp. 950–958, 2008.
[35] R. McDonald and J. Nivre, "Analyzing and integrating dependency parsers", *Computational Linguistics* ,vol. 37, pp. 197-230, 2011.
[36] 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*, Prague, pp. 122–131, 2007.
[37] A. F. T. Martins, *et al.*, "Stacking dependency parsers", *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 157-166, 2008.
[38] D. Zeman and Z. Žabokrtský, "Improving parsing accuracy by combining diverse dependency parsers", pp. 171-178, 2005.
[39] K. Sagae and A. Lavie, "Parser combination by reparsing", *In Proceedings of NAACL*, pp. 129-132, 2006.
[40] M. Fishel and J. Nivre, "Voting and stacking in data-driven dependency parsing", *NODALIDA, Odense, Denmark,* 2009.
[41] J. Nivre, "Two models of stochastic dependency grammar", 2002.
[42] G. Eryigit, *et al.*, "Dependency parsing of turkish", *Computational Linguistics,* vol. 34, pp. 357-389, 2008.
[43] B. R. Ambati, *et al.*, "Two methods to incorporate local morphosyntactic features in Hindi dependency parsing", *In Proceedings of NAACL-HLT 2010 workshop on*

Statistical Parsing of Morphologically Rich Languages (SPMRL 2010), Los Angeles, CA, pp. 22-30, 2010.

[44] P. Gadde, *et al.*, "Improving data driven dependency parsing using clausal information", *In Proceedings of NAACL-HLT 2010*, Los Angeles, CA, pp. 657-660, 2010.

[45] R. Tsarfaty, *et al.*, "Statistical parsing of morphologically rich languages (SPMRL): what, how and whither", *In Proceedings of NAACL-HLT 2010 workshop on SPMRL*, Los Angeles, CA, pp. 1-12, 2010.

[46] B. R. Ambati, *et al.*, "On the role of morphosyntactic features in Hindi dependency parsing", *In The First Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2010)*, pp. 94-102, 2010.

[47] W. Seeker and J. Kuhn, "On the Role of Explicit Morphological Feature Representation in Syntactic Dependency Parsing for German", In *Proceedings of the 12th International Conference on Parsing Technologies*, Dublin City University, pp. 58-62, 2011.

[48] J. D. Choi and M. Palmer, "Statistical Dependency Parsing in Korean: From Corpus Generation To Automatic Parsing", In *Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011)*, Dublin, Ireland, pp. 1-11, 2011.

[49] K. Bengoetxea and K. Gojenola, "Application of different techniques to dependency parsing of Basque", In *Proceedings of the 1st Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL), NAACL-HLT Workshop*, Los Angeles, USA, pp. 31-39, 2010.

[50] K. Bengoetxea, *et al.*, "Testing the Effect of Morphological Disambiguation in Dependency Parsing of Basque", In *Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011)*, Dublin, Ireland, pp. 28-33, 2011.

[51] J. Dehdari, *et al.*, "Morphological Features for Parsing Morphologically-rich Languages: A Case of Arabic", In *Proceedings of the 2nd Workshop on Statistical Parsing of Morphologically-Rich Languages (SPMRL 2011)*, Dublin, Ireland, pp. 12-21, 2011.

[52] Y. Marton, *et al.*, "Improving Arabic dependency parsing with lexical and inflectional morphological features", *In Proceedings of the 11th Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL) workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL)*, Los Angeles, pp. 13-21, 2010.

[53] Y. Goldberg and M. Elhadad, "Hebrew dependency parsing: Initial results", *In Proceedings of the 11th IWPT09*, Paris, pp. 129-133, 2009.

[54] Y. Goldberg and M. Elhadad, "Easy first dependency parsing of modern Hebrew", In *SPMRL-2010 – a NAACL/HLT workshop on Statistical Parsing of Morphologically Rich Languages*, pp. 103-107, 2010.

[55] G. Zhou, *et al.*, "Improving Dependency Parsing with Fined-Grained Features", presented at the Proceedings of the 5th International Joint Conference on Natural Language Processing, Chiang Mai, Thailand, 2011.

[56] M. Seraji, et al., "Bootstrapping a Persian Dependency Treebank", Linguistic Issues in Language Technology, vol. 7, pp. 1-10, 2012

واژه نامه

بخش الف: واژه نامه فارسی به انگلیسی

ارضای محدودیت ..... Constraint Satisfaction
استنتاج دستور زبان ..... Grammar induction
اشتقاق ..... Derivation
افکنشی ..... Projective
الگوی تجزیه ..... Parsing model
امتیاز ..... Score
امتیاز اتصال ..... Attachment score
انشعاب سَر ..... Split-head
بدون دور ..... Acyclicity
برچسب‌گذاری اجزای سخن ..... POS tagging
برنامه‌ریزی کارکردی ..... Functional programming
بستار متعدی بازتابی ..... Reflexive transitive closure
بیرون انداختن ..... Pop
بیش‌ترین حاشیه ..... Maximum margin
پشته ..... Stack
پیش‌گو ..... Oracle
پیراینده ..... Modifier
پیکرهٔ درختی ..... Treebank
پیکرهٔ نشانه‌گذاری شده ..... Annotated corpus
تابع خصوصیت ..... Feature function
تابع شالوده ..... Kernel function
تبدیل گشودن-تاکردن ..... Unfold-fold transformation
تجزیه ..... Parsing
تجزیهٔ مبتنی بر رده‌بند ..... Classifier-based parsing
تجزیهٔ مبتنی بر نمودار ..... Chart-parsing
تجزیهٔ وابستگی ..... Dependency parsing
تجزیه‌کنندهٔ پشته‌ای ..... Stacked parser
تشکیل یال چپ ..... Left-arc

تشکیل یال راست ..... Right-arc
تغییرات پله‌ای ..... Incrementality
تک‌سَری ..... Single-headed
جابجایی-کاهش ..... Shift-reduce
جزئی ..... Partial
جستجوی فراگیر ..... Exhaustive search
جستجوی ناشیانه ..... Brute-force search
خوش‌ساخت ..... Well-formed
حالت ..... State
درخت پوشای بیشینه ..... Maximum Spanning Tree (MST)
دستگاه بردار پشتیبان (اس.وی.ام.) ..... Support Vector Machine (SVM)
دستور ..... Grammar
دستور زایشی ..... Generative grammar
دستور وابستگی ..... Dependency grammar
دوسویه ..... Bilexical
رده‌بند ..... Classifier
رده‌بندهای آموزش دیده ..... Trained classifier
زایشی ..... Generative
زبان‌شناسی ..... Linguistic
زبان‌های بی‌ترتیب ..... Free-order languages
ساخت‌گرا ..... Structuralist
ساخت‌واژه ..... Morphological
ساختار وابستگی ..... Phrase structure
سازه ..... Constituent
سامانهٔ گذار ..... Transition system
سنجهٔ اف ..... F-measure
شاخص ..... Parameter
شبه‌افکنشی ..... Pseudo-projective
شبه‌پله‌ای ..... Pseudo-incremental
صورت‌گرا ..... Formalist
غیرافکنشی ..... Non-projective

غیر پایانی ..... Non-terminal
غیرقطعی کامل ..... NP-complete
فراخوانی ..... Recall
قطعی ..... Deterministic
کاهش ..... Reduce
لیست پیوندی ..... Linked list
مبتنی بر داده ..... Data-driven
مبتنی بر دستور ..... Grammar-based
مبتنی بر سازه ..... Constituency-based
مبتنی بر فهرست ..... List-based
مبتنی بر گذار ..... Transition-based
مبتنی بر محدودیت ..... Constraint-based
مبتنی بر یال ..... Arc-factored
محدودیت‌های فسخ‌شدنی ..... Defeasible constraints
محدودیت‌های نرم ..... Soft constraints
مستقل از متن ..... Context-free
مشتاق به یال ..... Arc-eager
مقوله‌ای ..... Categorical
نحو ..... Syntax
نزدیک‌ترین همسایه‌ها ..... Nearest neighbors
نقش مفهومی ..... Semantic role
نمادهای غیرپایانی ..... Non-terminal symbols
نمودار ..... Chart
نمونهٔ آموزشی ..... Training instance
وابستگی ..... Dependency
واژه ..... Word
واژهٔ سَر ..... Head
واژهٔ وابسته ..... Dependent
ویژگی هرزمانی ..... Anytime property
همبندی ..... Connectedness
یادگیری ..... Learning

یادگیری باناظر ..... Supervised learning

یادگیری تنبل ..... Lazy learning

یادگیری ماشینی ..... Machine learning

یادگیری مبتنی بر حافظه ..... Memory-based learning

یال ..... Arc

یال معیار ..... Arc-standard

بخش ب: واژه نامه انگلیسی به فارسی

بدون دور ..... Acyclicity
پیکرهٔ نشانه‌گذاری شده ..... Annotated corpus
ویژگی هرزمانی ..... Anytime property
یال ..... Arc
مشتاق به یال ..... Arc-eager
مبتنی بر یال ..... Arc-factored
یال معیار ..... Arc-standard
امتیاز اتصال ..... Attachment score
دوسویه ..... Bilexical
جستجوی ناشیانه ..... Brute-force search
مقوله‌ای ..... Categorical
نمودار ..... Chart
تجزیهٔ مبتنی بر نمودار ..... Chart-parsing
رده‌بند ..... Classifier
تجزیهٔ مبتنی بر رده‌بند ..... Classifier-based parsing
همبندی ..... Connectedness
مبتنی بر سازه ..... Constituency-based
سازه ..... Constituent
ارضای محدودیت ..... Constraint Satisfaction
مبتنی بر محدودیت ..... Constraint-based
مستقل از متن ..... Context-free
مبتنی بر داده ..... Data-driven
محدودیت‌های فسخ‌شدنی ..... Defeasible constraints
وابستگی ..... Dependency
دستور وابستگی ..... Dependency grammar
تجزیهٔ وابستگی ..... Dependency parsing
واژهٔ وابسته ..... Dependent
اشتقاق ..... Derivation

Deterministic ..... قطعی
Exhaustive search ..... جستجوی فراگیر
Feature function ..... تابع خصوصیت
Formalist ..... صورت‌گرا
Free-order languages ..... زبان‌های بی‌ترتیب
Functional programming ..... برنامه‌ریزی کارکردی
F-measure ..... سنجهٔ اف
Generative ..... زایشی
Generative grammar ..... دستور زایشی
Grammar ..... دستور
Grammar induction ..... استنتاج دستور زبان
Grammar-based ..... مبتنی بر دستور
Head ..... واژهٔ سَر
Incrementality ..... تغییرات پله‌ای
Kernel function ..... تابع شالوده
Lazy learning ..... یادگیری تنبل
Learning ..... یادگیری
Left-arc ..... تشکیل یال چپ
List-based ..... مبتنی بر فهرست
Linguistic ..... زبان‌شناسی
Linked list ..... لیست پیوندی
Machine learning ..... یادگیری ماشینی
Maximum margin ..... بیش‌ترین حاشیه
Maximum Spanning Tree (MST) ..... درخت پوشای بیشینه
Memory-based learning ..... یادگیری مبتنی بر حافظه
Modifier ..... پیرایند‌ه
Morphological ..... ساخت‌واژه
Nearest neighbors ..... نزدیک‌ترین همسایه‌ها
Non-projective ..... غیرافکنشی
Non-terminal ..... غیر پایانی
Non-terminal symbols ..... نمادهای غیرپایانی

غیرقطعی کامل ..... NP-complete
پیش‌گو ..... Oracle
شاخص ..... Parameter
تجزیه ..... Parsing
الگوی تجزیه ..... Parsing model
جزئی ..... Partial
ساختار وابستگی ..... Phrase structure
بیرون انداختن ..... Pop
برچسب‌گذاری اجزای سخن ..... POS tagging
شبه‌پله‌ای ..... Pseudo-incremental
شبه‌افکنشی ..... Pseudo-projective
افکنشی ..... Projective
فراخوانی ..... Recall
کاهش ..... Reduce
بستار متعدی بازتابی ..... Reflexive transitive closure
تشکیل یال راست ..... Right-arc
امتیاز ..... Score
نقش مفهومی ..... Semantic role
جابجایی-کاهش ..... Shift-reduce
تک‌سَری ..... Single-headed
محدودیت‌های نرم ..... Soft constraints
انشعاب سَر ..... Split-head
پشته ..... Stack
تجزیه‌کنندهٔ پشته‌ای ..... Stacked parser
حالت ..... State
ساخت‌گرا ..... Structuralist
یادگیری باناظر ..... Supervised learning
دستگاه بردار پشتیبان (اس.وی.ام.) ..... Support Vector Machine (SVM)
نحو ..... Syntax
رده‌بندهای آموزش دیده ..... Trained classifier
نمونهٔ آموزشی ..... Training instance

سامانهٔ گذار ..... Transition system

مبتنی بر گذار ..... Transition-based

پیکرهٔ درختی ..... Treebank

تبدیل گشودن-تاکردن ..... Unfold-fold transformation

خوش‌ساخت ..... Well-formed

واژه ..... Word

Abstract

Dependency-based methods for syntactic parsing have become increasingly popular in natural language processing in recent years. This seminar gives a thorough introduction to the methods that are most widely used today. After an introduction to dependency grammar and dependency parsing, followed by a formal characterization of the dependency parsing problem, surveys the three major classes of parsing models that are in current use: transition-based, graph-based, and grammar-based models. It continues on the comparison of different methods of dependency parsing.

Keywords: Dependency Parsing, Dependency Tree, Transition-based Parsing, Graph-based Parsing

Iran University of Science and Technology
School of Computer Engineering

A Survey on Dependency Parsing

A Seminar Submitted in Partial Fulfillment of the Requirement for the Degree of Master of Science in Artificial Intelligence and Roboic

By:
Mojtaba Khallash

Supervisor:
Dr. Behroz Minaei-Bidgoli

November 2011