تبدیل خودکار درخت‌بانک وابستگی فارسی به درخت‌بانک سازه‌ای

احمد پورامینی[1]، مسعود قیومی[2] و امینه ناصری[3]
[1 و 3] دانشکده برق و کامپیوتر، دانشگاه صنعتی سیرجان، سیرجان، ایران
[2] پژوهشگاه علوم انسانی و مطالعات فرهنگی، تهران، ایران

چکیده

درخت‌بانک‌ها به‌طورمعمول به دو شکل مبتنی بر ساختار وابستگی و مبتنی بر ساختار سازه‌ای ایجاد می‌شوند. هر دوی این ساختارها در حوزهٔ زبان‌شناسی و پردازش زبان طبیعی کاربرد دارند. هم‌اکنون چندین درخت‌بانک وابستگی برای زبان فارسی وجود دارد، اما درخت‌بانک سازه‌ای با حجم بزرگ برای این زبان وجود ندارد. در این مقاله قصد داریم روشی را برای تبدیل یک درخت‌بانک وابستگی به معادل سازه‌ای آن، بر اساس یک الگوریتم موجود ارائه دهیم. الگوریتم مبنا با استفاده از مجموعه‌ای از قواعد تبدیل، زیردرخت‌های سازه‌ای متناظر با یال‌های وابستگی را یافته و با ترکیب آنها ساختار سازه‌ای نهایی را تولید می‌کند. ما این الگوریتم را بر روی ساختارهای وابستگی زبان فارسی اعمال و ضمن ارائه نتایج، اصلاحاتی را در جهت بهبود کارایی آن ارائه می‌کنیم. نشان داده می‌شود که پیمایش یال‌های وابستگی در یک جهت خاص بر روی کیفیت الگوریتم تأثیرگذار است. همین‌طور ما اصلاحاتی را در الگورتیم مربوط به تطبیق قواعد و الگوریتم اتصال زیردرخت‌ها ارائه می‌کنیم. این اصلاحات کارایی الگوریتم را به شکل قابل ملاحظه‌ای افزایش می‌دهند. نتایج عملی بهبودی را به اندازه ۱۶/۴۸٪ نسبت به الگوریتم مبنا نشان می‌دهد.

واژگان کلیدی: پردازش زبان طبیعی، پیکره زبانی، درخت‌بانک وابستگی، درخت‌بانک سازه‌ای.

Converting Dependency Treebank to Constituency Treebank for Persian

Ahmad Pouramini[1*], Masoud Ghayoomi[2] & Amineh Naseri[3]
[1,3]Faculty of Computer Engineering, Sirjan University of Technology, Sirjan, Iran
[2]Faculty of Institute for Humanities and Cultural Studies, Tehran, Iran

Abstract

There are two major types of treebanks: dependency-based and constituency-based. Both of them have applications in natural language processing and computational linguistics. Several dependency treebanks have been developed for Persian. However, there is no available big size constituency treebank for this language. In this paper, we aim to propose an algorithm for automatic conversion of a dependency treebank to a constituency treebank for Persian. Our method is based on an existing method. However, we make modification to enhance its accuracy. The base algorithm constructs a constituency structure according to a set of conversion rules. Each rule maps a dependency relation to a constituency subtree. The constituency structure is built by combining these subtrees. We investigate the effects of the order in which dependency relations are processed on the output constituency structure. We show that the best order depends on the charactersitics of the target language. We also make modification in the algorithm for matching the conversion rules. To match a dependency relation to a conversion rule, we start with detailed infromation and if no match was found, we decrease the details and also change the method for matching. We also make modification in the algorithm used for combining the constituency subtrees. We use statistical data derived from a treebank to find a proper position for attaching a constituency subtree to the projection chain of the

* نویسندهٔ عهده‌دار مکاتبات
* Corresponding author

[DOI: 10.29252/jsdp.14.4.79]

head. The expremental results show that these modifications provide an improvement of 16.48% in the accuracy of the conversion algorithm.

Keywords: Natural language processing, Treebanks, Dependency structure, Phrase structure.

۱- مقدمه

امروزه درخت‌بانک‌ها[1] (پیکره‌های زبانی که از لحاظ نحوی نشانه‌گذاری شده‌اند)، از منابع اساسی پردازش زبان طبیعی محسوب می‌شوند و تأثیر زیادی در بهبود تجزیه‌گرهای[2] آماری دارند. این پیکره‌ها به‌طورمعمول به دو شکل درخت بانک وابستگی[3] (ساختار دستوری[4]) و درخت‌بانک سازه‌ای[5] (جاشناختی[6]) ایجاد می‌شوند. این نام‌گذاری برگرفته از نحوهٔ نمایش ساختار نحوی جملات در این نوع پیکره‌هاست که به‌طورمعمول به دو صورت ساختار وابستگی[7] و یا ساختار سازه‌ای[8] (مبتنی بر دستور زایشی[9]) تحقق می‌یابد.

با توجه به پیشرفت‌هایی که در تجزیه‌گرهای وابستگی ایجاد شده است، شاهد ظهور درخت‌بانک‌های وابستگی در زبان‌های مختلف از جمله زبان فارسی هستیم؛ مانند درخت‌بانک وابستگی UPDT[10] که شامل حدود شش هزار جمله است [16]؛ درخت‌بانک وابستگی PerDT که شامل سی هزار جمله است [13] و درخت‌بانک وابستگی DePerTreeBank [8].

از سوی دیگر، دستور زبان زایشی تحت تأثیر نظریات چامسکی مطرح بوده است و صوری‌سازی‌های محاسباتی ویژه‌ای مانند دستور ساخت سازه‌ای هسته‌بنیان[11] [12]، دستور درخت الحاقی واژگانی‌شده[12] [14]، دستور مقوله‌ای ترکیبی[13] [17] و دستور واژی-نقشی[14] [9] با این رویکرد معرفی شده‌اند و هرکدام دارای کاربردهای خاص خود در پردازش زبان طبیعی هستند.

PerTreebank اولین درخت‌بانک تهیه‌شده برای زبان فارسی است و حاوی به‌طورتقریبی هزار جمله است که براساس دستور ساخت سازه‌ای هسته‌بنیان تهیه شده است [7]. همان‌طور که مشاهده می‌شود، این درخت‌بانک از نظر حجم کوچک است و به‌جز این درخت‌بانک، درخت‌بانک سازه ای عمده‌ای برای زبان فارسی وجود ندارد.

در این مقاله قصد داریم الگوریتم انعطاف‌پذیری را برای تبدیل خودکار درخت‌بانک وابستگی به معادل سازه‌ای آن معرفی کنیم و نتایج اعمال آن را بر روی درخت‌بانک وابستگی فارسی مورد بحث قرار دهیم. این الگوریتم می‌تواند با دقت بالایی یک درخت‌بانک وابستگی موجود را به معادل سازه‌ای آن تبدیل کند. از طرفی، از آنجا که به‌طورمعمول ایجاد ساختار وابستگی یک جمله ساده‌تر است، ابتدا ساختارهای وابستگی جملات را به شکل دستی می‌توان ایجاد کرد و الگوریتم به شکل خودکار آنها را به معادل سازه ای تبدیل می‌کند.

قبل‌از ارائه این الگوریتم، به ساختار وابستگی و سازه‌ای و تفاوت آنها می‌پردازیم.

۱-۱- ارتباط میان ساختار وابستگی و ساختار سازه‌ای

شکل (۱) یک ساختار وابستگی نمونه و ساختار سازه‌ای معادل آن را برای یک جمله نشان می‌دهد. البته در اینجا باید خاطر نشان کرد که این ساختارها قالب استانداری ندارند و ممکن است یک جمله واحد با توجه به نظریه‌های نحوی متفاوت و یا یک کاربرد خاص با ساختارهای متفاوت سازه‌ای یا وابستگی نمایش داده شود.

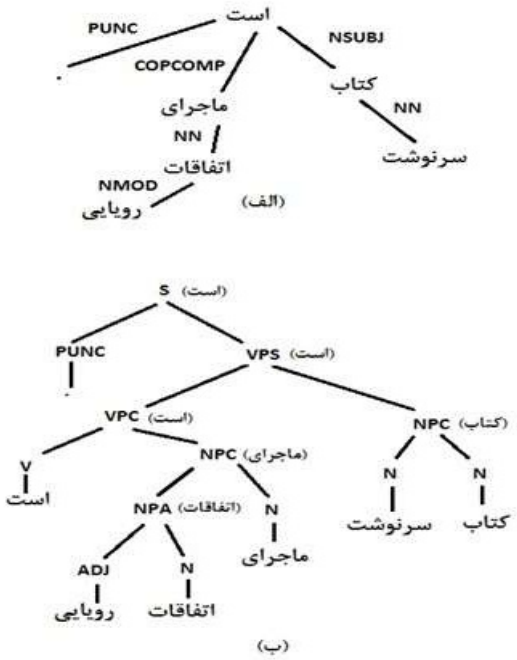
به‌طورکلی هر دو ساختار نوعی درخت هستند. در درخت سازه‌ای، کلمات جمله در برگ‌ها قرار دارند و در قالب واحدهای نحوی (برای مثال گروه اسمی NP) به شکل سلسله‌مراتبی گروه‌بندی می‌شوند. در این ساختار روابط میان کلمات به‌صراحت نمایش داده نمی‌شوند.

در درخت وابستگی هر گره درخت نماینده یک کلمه از جمله است و با توجه به ارتباطی که آن کلمه با سایر کلمات دارد، توسط یال‌هایی به گره‌های متناظر متصل

[1] Treebank
[2] Parser
[3] Dependency Treebank
[4] Tecto-grammatical
[5] Constituency-based Treebank
[6] Typological
[7] Dependency Structure
[8] Phrase Structure
[9] Generative Grammar
[10] Upsala Persian Dependency Treebank
[11] Head-driven Phrase Structure Grammar
[12] Lexicalized Tree Adjoining Grammar
[13] Combinatory Categorical Grammar
[14] Lexical Functional Grammar

[DOI: 10.29252/jsdp.14.4.79]

می‌شود (رابطه میان وابسته و هسته). برخلاف ساختار سازه ای، واحدهای نحوی در این ساختار به‌صراحت نمایش داده نمی‌شوند؛ اما به شکل ضمنی قابل استخراج هستند. برخی دیگر از تفاوت‌های این دو ساختار انتخابی است؛ به‌عنوان‌مثال هر دوی این نمایش‌ها ممکن است از مقوله‌های خالی[1] استفاده کنند؛ اجازه وجود یال‌های متقاطع را بدهند و یا مرتب (براساس ترتیب کلمات در جمله) یا نامرتب باشند.



(شکل-۱): ساختار وابستگی و ساختار سازه‌ای جمله «کتاب سرنوشت، ماجرای اتفاقات رویایی است»

(Figure-1): Dependency structure and its corresponding constituency structure for a sample sentence.

این مقاله در چهار بخش تدوین شده است. در بخش ۲ کارهای مرتبط و الگوریتم‌های موجود برای تبدیل ساختار وابستگی به سازه‌ای مرور می‌شود. در بخش ۳ نتایج الگوریتم انتخاب‌شده بر روی درخت‌بانک فارسی گزارش شده و اصلاحات موردنظر پیشنهاد می‌شود. در بخش ۴، کیفیت الگوریتم با توجه به اصلاحات پیشنهادشده محاسبه شده و نتایج ارائه می‌شود.

۲- کارهای مرتبط

روش‌های متعددی برای تبدیل ساختار وابستگی به ساختار سازه‌ای ارائه شده است. به‌طورکلی این روش‌ها را به دو دسته روش‌های آماری و روش‌های قاعده‌مند می‌توان تقسیم

[1] Empty Categories

کرد. در روش‌های آماری، با استفاده از داده‌های آموزشی زیاد در قالب زوج‌های (DS_i, PS_i) مدلی احتمالاتی تولید می‌شود؛ سپس این مدل با دریافت یک ساختار وابستگی، ساختار سازه‌ای مربوط را تولید می‌کند [5, 18]. مشکل این روش این است که اگر در مرحلهٔ آموزش داده‌های کافی وجود نداشته باشد، الگوریتم ممکن است کارایی لازم را نداشته باشد [4].

در روش‌های قاعده‌مند، الگوریتمِ تبدیل با استفاده از مجموعه‌ای از قواعد، عملیات تبدیل را انجام می‌دهد. به‌طوراصولی تفاوت اصلی میان ساختار وابستگی و ساختار سازه‌ای وجود واحدهای نحوی (مانند گروه فعلی، گروه اسمی و غیره) در ساختار سازه‌ای است که در ساختار وابستگی وجود ندارند و باید به شکلی بازیابی شوند. الگوریتم‌های مختلف ممکن است، فرضیات متفاوتی در مورد ساختار سازه‌ای هدف داشته باشند؛ از آن جمله به تعداد افکنش‌های[2] هر مقوله دستوری[3]، و مکان اتصال زیر درخت مربوط به یک وابسته به زنجیره افکنش هسته[4] می‌توان اشاره کرد. برای مثال [6] روشی را برطبق نظریه ایکس تیره[5] ارائه داد که در آن هر مقوله دستوری مانند X تنها به دو واحد نحوی X' و XP قابل افکنش است؛ همچنین در این روش، وابسته‌های یک هسته به دسته‌های مشخصی تقسیم و هر کدام در مکان مشخصی به زنجیره افکنش هسته متصل می شوند. در روشی دیگر ارائه‌شده در [5]، هر مقوله دستوری مانند X تنها یک افکنش XP دارد، و زیردرخت مربوط به هر وابسته به‌عنوان خواهر گره هسته در درخت سازه‌ای نمایش داده می‌شود.

در مقابل این روش‌ها که فرضیات ثابت و انعطاف‌ناپذیری داشتند، ژای و پالمر در [19] الگوریتمی ارائه کردند که انعطاف بیشتری داشت. هدف اصلی آنها ایجاد ساختارهایی بود که تا حد امکان به ساختارهای درخت‌بانک Penn [11] که به شکل دستی ایجاد شده بودند، شبیه باشند. در این روش، زنجیره افکنش هر مقوله دستوری در قالب یک جدول به الگوریتم داده می‌شود. به‌عنوان مثال طبق این جدول، V به VP و سپس به S افکنده می‌شود. همین‌طور، زیر درخت هر وابسته در پایین‌ترین سطح ممکن به زنجیره افکنش هسته متصل می‌شود. محل این اتصال از

[2] Projection
[3] Part of Speech
[4] head projection chain
[5] X-bar

[DOI: 10.29252/jsdp.14.4.79]

روی دو جدول ورودی دیگر که موضوعات[1] و توصیف‌کننده‌های[2] هر سازه را مشخص می‌کنند، به‌دست می‌آید.

با این حال، همه این روش‌ها شامل برخی محدودیت‌ها و فرضیات ثابت هستند. برای مثال، در همه آنها برای هر مقوله دستوری تنها یک زنجیره افکنش فرض می‌شود؛ درصورتی‌که درعمل ممکن است، یک مقوله دستوری در جملات مختلف دارای افکنش‌های متفاوتی باشد؛ اما از همه مهمتر این است که این روش‌ها از اطلاعات متنوعی که در ساختار وابستگی وجود دارد، مانند نوع وابستگی ((subj, obj, …))، سود نمی‌برند.

ژای و همکارانش در [20] باتوجه به این نواقص، الگوریتم جدیدی ارائه کردند که این محدودیت‌ها را پوشش می‌دهد. فرض اساسی این الگوریتم این است که درخت وابستگی و درخت سازه‌ای مورد نظر، سازگار[3] هستند. شرط سازگاری به این معنی است که درخت وابستگی باید نتیجه الگوریتم مسطح‌سازی[4] درخت سازه‌ای باشد. چنانچه در درخت سازه‌ای، گرهٔ هسته هر زیر درخت مشخص باشد، با الگوریتم مسطح‌سازی درخت وابستگی معادل را می‌توان تولید کرد. برای این منظور، الگوریتم از برگ‌های حاوی کلمات هسته شروع کرده و این کلمات را به گره‌های میانی والد آنها منتقل می‌کند؛ سپس این عمل برای گره‌های میانی نیز تکرار می‌شود و درنهایت کلمه هسته جمله در ریشه درخت و کلمات وابسته آن در زیر درخت‌های متصل به ریشه قرار می‌گیرند. در شکل (۱(ب)) در هر گره میانی، کلمه‌ای که به آن منتقل می‌شود در داخل پرانتز نمایش داده شده است. با پیمایش این درخت از بالا به پایین و اتصال کلمه هسته هر زیر درخت به وابسته‌هایش، درخت وابستگی حاصل می‌شود.

با فرض سازگاری میان ساختار وابستگی و ساختار سازه‌ای، الگوریتم سعی می‌کند تناظری میان روابط (یال‌ها) موجود در درخت وابستگی و زیر درخت‌های موجود در درخت سازه‌ای برقرار کند. از این تناظرها با عنوان «قواعد تبدیل[5]» یاد می‌شود. الگوریتم با استفاده از این قواعد، برای هر یک از یال‌های ساختار وابستگی، زیردرختان متناظر را یافته و درخت سازه‌ای نهایی را با اتصال این زیردرخت‌ها به یکدیگر ایجاد می‌کند. کارایی این الگوریتم برای زبان انگلیسی و با استفاده از درخت‌بانک پن ۸۹/۴٪ گزارش شده است.

البته شرط سازگاری همیشه برقرار نیست. به‌عنوان مثال، چنانچه در درخت سازه‌ای از مقوله خالی استفاده شود، ولی در درخت وابستگی چنین سازوکاری به‌کار گرفته نشود، نتیجه الگوریتم مسطح‌سازی با درخت وابستگی یکسان نخواهد بود [20]. برای برقراری این شرط در درخت وابستگی نیز از مقوله خالی می‌توان استفاده کرد. ژای و همکارانش معتقدند که نسل بعدی درخت‌بانک‌ها باید چندلایه[6] باشد و نحوه نمایش اطلاعات در هر دو ساختار وابستگی و ساختار سازه‌ای به‌صراحت مشخص شده باشد [20]. در این صورت امکان تبدیل خودکار هر ساختار به ساختار دیگر میسر است. این رهنمون در ایجاد درخت‌بانک هندی که دارای سه لایه نشانه‌گذاری وابستگی، معنایی و سازه‌ای است به‌کار گرفته شده است [4].

برای زبان فارسی، دو کار عمده در این زمینه صورت گرفته است که هر دو بر پایه الگوریتم ارائه‌شده توسط ژای و همکاران (۲۰۰۸) قرار دارند. در [1]، سلطان‌زاده و دیگران سعی کرده‌اند، قواعد تبدیل مورد نیاز را به شکل دستی برای زبان فارسی توسعه دهند؛ سپس با استفاده از این قواعد، درخت‌بانک وابستگی PerDT شامل سی‌هزار جمله را به معادل سازه‌ای آن تبدیل کردند [13]. البته الگوریتم در تبدیل حدود دوهزار جمله با محدودیت‌هایی روبه‌رو بوده و ناموفق عمل کرده است. به‌علت عدم وجود یک معیار استاندارد برای ارزیابی الگوریتم، صد جمله به شکل تصادفی انتخاب و به شکل دستی به ساختار سازه‌ای تبدیل شده‌اند. با مقایسه این ساختارها با خروجی الگوریتم، کارایی الگوریتم ۹۶٪/۰۵ گزارش شده است.

در [2]، دهقان و فیلی سعی کردند الگوریتم ارائه‌شده توسط ژای و همکارانش در [20] را در زبان فارسی به‌کار گیرند و اصلاحاتی روی آن اعمال کنند تا کارایی آن در هر دو زبان انگلیسی و فارسی افزایش یابد. این اصلاحات شامل استفاده از یک طبقه‌بند برای تعیین مکان اتصال زیردرختان، و افزودن یک مرحله پساپردازش برای اصلاح ساختارهای نهایی است. در راستای این هدف، آنها قواعد تبدیل مورد نیاز به شکل آماری را از دو درخت‌بانک سازه‌ای PerTreebank [7] و درخت‌بانک وابستگی DepPerTreebank [8] استخراج کردند. DepPerTreebank به شکل خودکار از روی

[1] Argument
[2] Modifier
[3] Consistent
[4] Flattening
[5] Conversion rules
[6] Multi-representational

[DOI: 10.29252/jsdp.14.4.79]

PerTreebank حاصل شده است و هر دو درخت‌بانک حاوی ۱۰۲۸ جمله یکسان هستند. کارایی این الگوریتم در زبان فارسی بدون استفاده از اصلاحات پیشنهادشده ۶۶/۲۱٪ است که و با استفاده از اصلاحات به ۹۲/۰۶٪ افزایش می‌یابد که بهبودی ۲۵/۵۸٪ را نشان می‌دهد. این کار از جنبه‌هایی شبیه به کار انجام‌شده در این مقاله است؛ زیرا هر دو روش از یک الگوریتم مبنا استفاده کرده و از درخت‌بانک‌های یکسانی برای استخراج قواعد و ارزیابی روش استفاده می‌کنند. هرچند، اصلاحات پیشنهادشده در روش پیشنهادی ما متفاوت است که این تفاوت‌ها در هنگام توضیح الگوریتم، بحث خواهند شد.

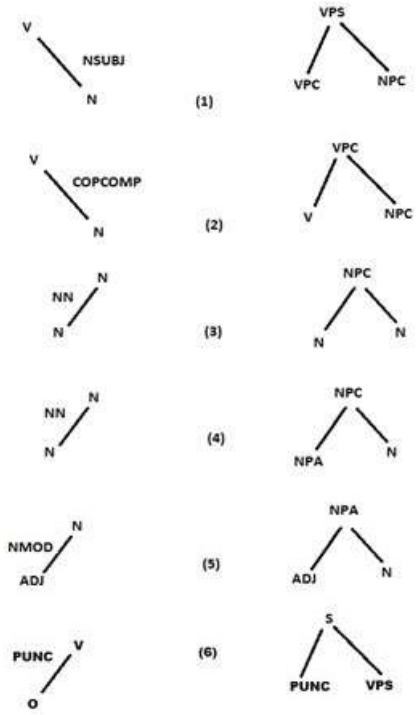
در بخش بعدی، ما الگوریتم ارائه‌شده توسط ژای و همکاران را در [20] مرور و در این مقاله از آن با عنوان «الگوریتم مبنا» یاد می‌کنیم؛ سپس ضمن به‌کارگیری آن در زبان فارسی و بیان نتایج حاصل‌شده، ملاحظات و اصلاحاتی را برای بهبود کارایی این الگوریتم، به‌ویژه برای زبان فارسی ارائه می‌کنیم.

۳- تشریح الگوریتم

۳-۱-قواعد تبدیل

همان‌طورکه گفته شد، الگوریتم مبنا نیاز به تناظرهایی میان یال‌های درخت وابستگی و زیردرخت‌هایی از درخت سازه‌ای داردکه به آنها «قواعد تبدیل» گفته می‌شود.

به‌طورکلی، یک قاعده تبدیل تناظری میان یک الگوی وابستگی و یک الگوی سازه‌ای است. در اینجا، منظور از یک الگوی وابستگی، یک رابطه وابستگی (یال وابستگی) به‌همراه دیگر اطلاعات مربوط به هسته و وابسته، مانند نوع وابستگی و مقوله دستوری هسته و وابسته، است. در سمت دیگر، یک الگوی سازه‌ای وجود دارد که یک درخت تک‌سطحی دوشاخه‌ای است. در شکل (۲)، قواعد تبدیل مربوط به درختان وابستگی و سازه‌ای شکل (۱) نمایش داده شده‌اند. برای مثال، طبق قاعده شمارهٔ ۱، زمانی که در درخت وابستگی گره‌ای با برچسب N به گره V ازسمت راست وابسته و نوع وابستگی SUBJ باشد، الگوی سازه‌ای متناظر آن شامل گره S در ریشه و دوشاخه NPC و VPC است. در اینجا، NPC افکنش وابسته N، و VPC افکنش هسته V محسوب می‌شود. در این مقاله، شاخه مرتبط به افکنش هسته را «شاخه هسته»، و شاخه دیگر را «شاخه وابسته» می‌نامیم.



(شکل-۲): قواعد تبدیل استخراج شده از ساختارهای شکل (۱)
(Figure-2): The conversion rules extracted from the structures in Figure 1.

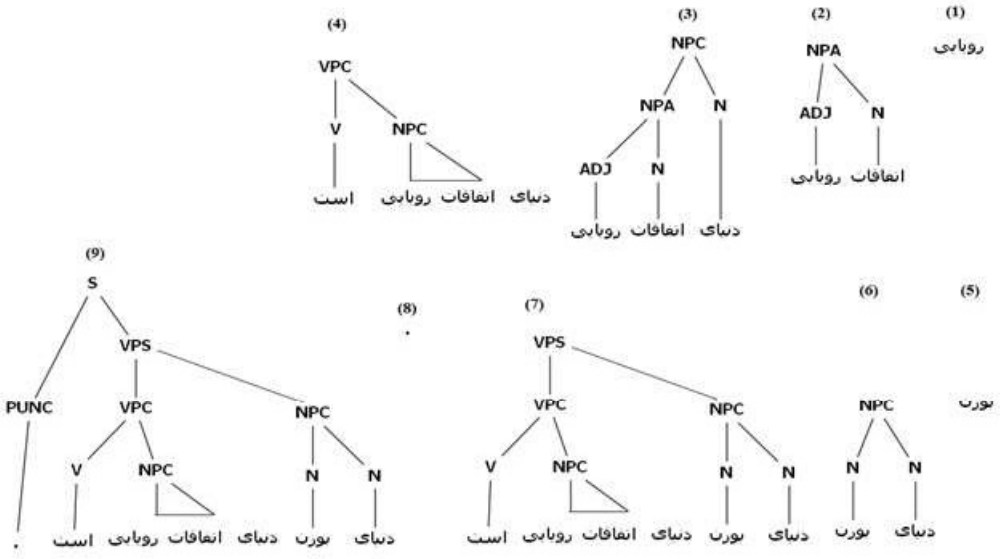
۳-۲- الگوریتم تبدیل

شبه‌کد الگوریتم تبدیل ژای و همکاران در [20] در الگوریتم (۱) نمایش داده شده است. همچنین، مراحل اجرای این الگوریتم بر روی درخت وابستگی شکل (۱) در شکل (۳) نمایش داده شده‌اند.

(الگوریتم -۱) الگوریتم مبنا برای تبدیل ساختار وابستگی به سازه‌ای
(Algorithm 1) The base algorithm for DS to PS conversion.

```
procedure CONVERT(DSNode)
    persistent: Rules, a set of conversion rules
    PSTree ← newPSNode(DSNode)
    if DSNode has no child then
        return PSTree
    else
        for all left Child of DSNode from right to left do
            childTree ← CONVERT(Child)
            rule ← RULES.FIND(DSNode, Child)
            Tree ← INSERT(childTree, rule.PSPattern)
            PSTree ← ATTACH(Tree, PSTree, "Left")
        end for
        Repeat the loop for each right Child of DSNode from
left to right
        return PSTree
    end if
end procedure
```


[DOI: 10.29252/jsdp.14.4.79]



(شکل۳) مراحل تبدیل ساختار وابستگی شکل ۱ به ساختار سازه‌ای

(Figure 3) The steps for converting a sample dependency structure shown in Figure 1 to its constituency equivalent.

الگوریتم تبدیل، یک الگوریتمی بازگشتی است که بر روی ریشه درخت وابستگی فراخوانی می‌شود. چنانچه گره ورودی یک برگ باشد، درخت سازه‌ای شامل یک گره حاوی کلمه متناظر است؛ اما درصورتی‌که گره ورودی دارای فرزندانی (گره‌های وابسته‌ای) باشد، ابتدا فرزندان سمت چپ این گره (بر حسب ترتیب کلمات در نوشتار انگلیسی که معادل فرزندان سمت راست در نوشتار فارسی است)، از سمت راست تا انتها پیمایش می‌شوند. در این پیمایش، در تابعی به‌نام تابع childTree، برای هر فرزند درخت سازه‌ای متناظر آن، به شکل بازگشتی ساخته می‌شود؛ سپس با توجه به یال وابستگی میان این فرزند (وابسته) و والدش (هسته)، قاعده مربوط یافت می‌شود و الگوی سازه‌ای متناظر با آن بازیابی و سپس، زیردرخت فرزند به شاخه سمت چپ این درخت متصل می‌شود. اکنون، در تابعی، به‌نام تابع Attach، درخت حاصل باید به مکان مناسبی در زنجیره افکنش هسته الحاق شود . نحوه این اتصال در قسمت بعدی توضیح داده شده است.

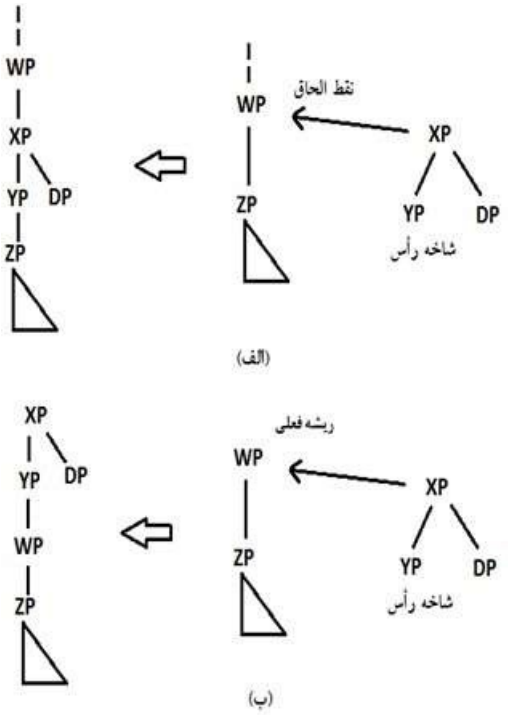
در ادامه این روند، عملیات بالا برای فرزندان سمت راست گره هسته از سمت چپ به ابتدا تکرار و با اتصال کلیه زیردرختان وابسته، درخت سازه‌ای نهایی ایجاد می‌شود. با آزمایش‌های انجام‌شده درعمل دریافتیم که اولویت‌دهی به پیمایش فرزندان سمت چپ نسبت‌به فرزندان سمت راست و یا بر عکس بر دقت الگوریتم تأثیرگذار است که این مطلب در قسمت نتایج بحث خواهد شد.

۳-۳- اتصال زیر درختان به زنجیره افکنش

پس از ایجاد زیردرخت مربوط به وابسته، این زیردرخت باید به مکان مناسبی در زنجیره افکنش هسته الحاق شود. شکل (۴) نحوه این عمل را نشان می‌دهد. حالت (الف) زمانی است که نقطه الحاق مشخص باشد. در این حالت چنانچه نقطه الحاق WP با ریشه زیردرخت XP هم‌نام باشند، این دو گره با یکدیگر ادغام و اگر YP با ZP هم‌نام باشند، این دو گره نیز ادغام می‌شوند. حالت (ب) مربوط به زمانی است که نقطه الحاق مناسبی یافت نشود و در آن صورت کل زنجیره افکنش هسته در زیرگره شاخه هسته YP متصل می‌شود. در این حالت چنانچه WP با YP یکسان باشند، با یکدیگر ادغام می‌شوند. حالت (الف) را «الحاق»، و حالت (ب) را «افزایش» می‌نامیم.

زمانی که وابسته‌های هسته در یک سمت (راست یا چپ) از نزدیکترین وابسته به سمت دورترین وابسته (طبق ترتیب کلمات در جمله) پیمایش می‌شوند، همواره زیردرخت وابسته به زنجیره افکنش هسته افزوده می‌شود (شکل ۴ (ب)). توجه کنید درغیراین‌صورت، ترتیب کلمات در درخت سازه‌ای حاصل رعایت نخواهد شد. این مطلب را می‌توان در مراحل یک تا هفت در شکل (۳) مشاهده کرد.

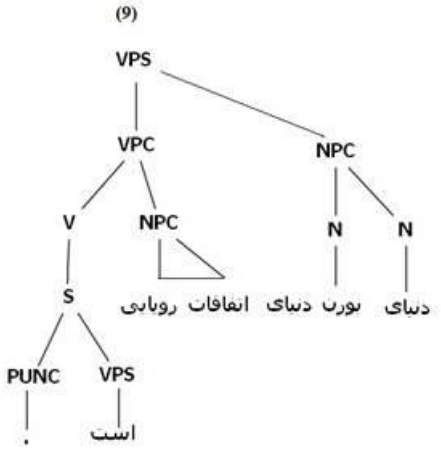
[DOI: 10.29252/jsdp.14.4.79]



(شکل-۴): الف) عمل الحاق زیردرخت وابسته. ب) عمل افزایش زیردرخت وابسته

(Figure-4): Adjoining (top) versus insertion (bottom) to attach a dependent's subtree to the projection chain.

انتخاب نقطه مناسب الحاق زمانی اهمیت پیدا می‌کند که گره هسته دارای وابستگانی در دو طرف خود باشد. در این صورت، با پیمایش وابسته‌ها در یک سمت، زنجیره افکنش هسته گسترش می‌یابد و به هنگام پیمایش وابسته‌ها در سمت دیگر باید نقطه مناسبی در زنجیره افکنش برای الحاق زیردرخت وابسته یافت شود. در الگوریتم مبنا، این نقطه برابر با پایین‌ترین گره در زنجیره افکنش هسته است که اجازه الحاق یک زیر درخت جدید را می‌دهد. برای مثال در شکل (۳) در مرحله ۷، زنجیره افکنش هسته، شامل V، VPC وVPS است که هر کدام نقطه الحاق زیر درخت مربوط به رابطه وابستگی PUNC می‌توانند باشند (قاعده ۶ در شکل ۲). الگوریتم مبنا، گره V را به‌عنوان پایین‌ترین نقطه که شاخه دیگری نیز در سمت راست خود ندارد پیشنهاد می‌کند؛ اما اگر این نقطه انتخاب شود، درخت حاصل مطابق با شکل (۵) خواهد بود که مسلماً دارای خطا است. در این حالت، باید زیردرخت به زنجیره افکنش افزوده شود تا حالت ۹ در شکل (۳) به‌دست بیاید. البته این خطا به نحوه پیمایش گره‌های وابسته نیز مرتبط است، و چنانچه ابتدا وابسته‌های سمت چپ (طبق نوشتار فارسی) و سپس وابسته‌های سمت راست پیمایش شوند، در این مثال خاص خطایی رخ نمی‌دهد؛ اما مثال دیگری می‌توان یافت که این نحوه پیمایش و فرض ثابت‌بودن مکان الحاق باعث خطا خواهد شد.



(شکل-۵): الحاق زیردرخت وابسته با ریشه S به زنجیره افکنش در گره V

(Figure-5): Adjoining the dependent's subtree with the root S to the projection chain at the node V.

برای حل این مشکل و یافتن نقطه اتصال مناسب، دهقان و فیلی (۱۳۹۵) از یک طبقه‌بند[1] استفاده کردند. برای این طبقه‌بند دو طبقه تعریف کردند که عبارت‌اند از پایین‌ترین نقطه (V در مثال قبل) و بالاترین نقطه (VPS در زنجیره افکنش هسته)؛ سپس با تعریف برخی ویژگی‌ها، این طبقه‌بند را توسط ساختارهای سازه‌ای PerTreebank آموزش داده و در الگوریتم خود جهت تعیین نقطه اتصال استفاده کردند. با استفاده از این روش، آنها توانستند به کارایی ۷۶/۳۲٪ برسند که درحدود برابر با ۱۰٪ بهبود نسبت به الگوریتم مبنا است.

ما الگوریتم ویژه‌ای را برای یافتن نقطه مناسب اتصال زیردرخت وابسته طراحی کردیم که راه‌حل‌های متفاوتی را پیشنهاد می‌کند. این الگوریتم در الگوریتم (۲) نشان داده شده است.

طبق این الگوریتم، زنجیره افکنش فعلی هسته را با هدفِ یافتنِ نقطه مناسبِ اتصال، از پایین به بالا پیمایش و برای هر گره در این پیمایش دو شرط را بررسی می‌کنیم. نخستین شرط این است که نباید در گره‌های بالای محل اتصال، در سمتی که قرار است زیردرخت وابسته اضافه شود، زیر درخت دیگری متصل شده باشد. درغیراین‌صورت، ترتیب کلمات در درخت سازه‌ای حاصل رعایت نخواهد شد.

[1] classifier

[DOI: 10.29252/jsdp.14.4.79]

(الگوریتم-۲): الگوریتم اتصال زیردرخت وابسته
(Algorithm-2): Attachment of a dependent's subtree.

```
procedure ATTACH(Tree, PSTree, Side)
    Chain ← PSTree.ProjectionChain
    attached ← false
    for all Node in Chain from bottom to up do
        if There is no subtree attached above the Node on this
Side then
            if CANBEUNDER(Node, Tree.Root) then
                PSTree ← ADJOIN(Tree, Node)
                attached ← true
            end if
        end if
    end for
    if Not attached then
        Attach PSTree on Side of Tree
        PSTree ← Tree
    end if
end procedure

procedure CANBEUNDER(Phrase1, Phrase2)
    if Phrase1 is "S" OR Phrase1 equals Phrase2 then
        return true
    end if
    if NOT UseChainTable then
        return false
    end if
    persistent: ChainTable a collection of extracted projection
chains
    list1 ← All chains in ChainTable where Phrase1 is
above Phrase2
    list2 ← All chains in ChainTable where Phrase1 is
below Phrase2
    if list1.Count >list2.Count then
        return true
    else
        return false
    end if
end procedure
```

شرط دوم الگوریتم در تابع CanBeUnder بررسی می‌کند که آیا ریشه زیردرخت وابسته پایین‌تر از این گره در زنجیره افکنش می‌تواند قرار گیرد یا خیر. در صورت برقراری این شرط زیردرخت وابسته را در این گره الحاق می‌کنیم. درغیراین‌صورت به سراغ گره بالاتر در زنجیره افکنش می‌رویم. درصورتی‌که هیچ نقطه مناسبی یافت نشود، زیردرخت را به زنجیره افکنش هسته می‌افزاییم (عمل افزایش).

در تابع CanBeUnder برای بررسی امکان الحاق زیردرخت وابسته به گره‌ای از زنجیره افکنش چندین شرط بررسی می‌شود. در ساده‌ترین حالت، اگر آن گره همنام ریشه زیردرخت و یا برابر با S به معنی کل جمله باشد، آن گره به عنوان نقطه الحاق انتخاب می‌شود. توجه کنید که نباید چیزی به بالای گره S افزوده شود و تنها عمل الحاق قابل انجام است. طبق آزمایش‌های صورت‌گرفته، این تغییر کوچک اگر با پیمایش خاصی از وابسته‌ها همراه باشد، می‌تواند بهبود قابل ملاحظه‌ای را در دقت الگوریتم به همراه داشته باشد. این روش در قسمت نتایج با عنوان روش «بدون جدول» بحث خواهد شد. برای مثال طبق این روش، در مثال نشان داده‌شده در شکل (۳) زیردرخت وابسته PUNC به زنجیره افکنش در مرحله ۷ افزوده شده و درخت شماره ۹ به درستی ایجاد می‌شود. هرچند این پیشنهاد ممکن است برای زبان فارسی نتایج بهتری را حاصل کند، اما همچنان فرضیات ثابتی دارد و باید روش انعطاف‌پذیرتر و دقیق‌تری را ارائه کرد. به این منظور ما راهکار دیگری را پیشنهاد کرده‌ایم.

برای پاسخ به سؤال که آیا ریشه زیردرخت وابسته در پایین یک گره در زنجیره افکنش می‌تواند قرار گیرد یا خیر، نیاز به جدولی داریم که ترتیب معمول واحدهای نحوی در زنجیره افکنش هسته‌ها را مشخص کند. البته این ترتیب همیشه یکسان نیست و برای مثال واحد نحوی VPA[1] ممکن است در برخی موارد پایین واحد نحوی VPC[2] و در برخی موارد بالای آن قرار گیرد.

برای ایجاد جدول مذکور، کلیه زنجیره‌های افکنش هسته‌ها در درخت‌بانک سازه‌ای PerTreebank را استخراج کردیم. به این ترتیب برای هر کلمه یک زنجیره حاصل می‌شود (۲۶۳۵۰ زنجیره). پس از حذف زنجیره‌های تکی (حاوی یک عنصر)، ۱۳۵۳۱ زنجیره با دو یا بیش از یک عنصر حاصل می‌شود که از این تعداد، ۱۲۵۷ زنجیره متفاوت هستند. نمونه‌ای از زنجیره‌ها در جدول ۱ نمایش داده شده است. برای مثال زنجیره‌های S/VPS/VPC/V، NPC/N، NPA/N و PUNC از ساختار سازه‌ای شکل (۱) قابل استخراج هستند. در این میان، زنجیره NPC/N دو بار تکرار شده و زنجیرهPUNC حاوی یک افکنش است که در جدول استفاده نمی‌شود؛ سپس تعداد مواردی که یک واحد نحوی در زیر واحد نحوی دیگر قرار گرفته و تعداد مواردی که در بالای آن قرار گرفته است مقایسه می‌شوند و درصورتی‌که تعداد مواردی که در آنها ریشه زیردرخت وابسته پایین‌تر از گره موردنظر است بیشتر باشد، این گره به‌عنوان نقطه الحاق انتخاب می‌شود (به تابع CanBeUnder رجوع کنید). این عمل تأثیر قابل ملاحظه‌ای در کارایی الگوریتم دارد که در قسمت بعدی با عنوان روش «با جدول» در مورد آن بحث خواهد شد.

[1] Verb Phrase Adjunct
[2] Verb Phrase Complement

[DOI: 10.29252/jsdp.14.4.79]

(جدول-۱): چند نمونه از زنجیره‌های افکنش در جدول افکنش هسته‌ها.

(Table-1): Samples fo projection chains in the projection chaing table.

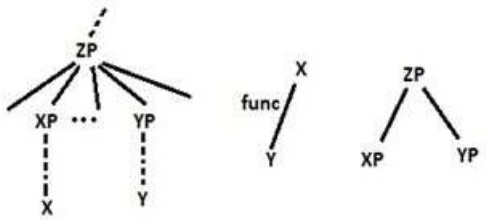
S/VPS/VPC/V
S/VPA/VPSD/VPC/MV/V
VPS/VPA/VPC/V
VPC/MV/V
NPC/N
NPC/PRON
ADJPC/ADJ

۴- نتایج

برای بررسی کارایی الگوریتم، نیاز به یک درخت‌بانک وابستگی و معادل سازه‌ای آن داریم تا خروجی الگوریتم را با درخت‌بانک سازه‌ای مقایسه کنیم. برای این کار، از درخت‌بانک وابستگی DepPerTreebank [8] و معادل سازه‌ای آن PerTreebank [7] استفاده کردیم. هر چند تعداد جملات متناظر در این دو درخت‌بانک تنها حدود ۱۰۲۸ جمله است، اما میانگین طول جمله‌ها در حدود ۲۶ کلمه است که نشان‌دهنده وجود جملات مرکب و پیچیده در این حجم داده است. این تعداد جمله برای تخمین کارایی الگوریتم مناسب است.

برای فراهم‌کردن قواعد تبدیل، این قواعد را به شکل خودکار از درخت‌بانک سازه‌ای استخراج کردیم. روش این کار همانند الگوریتم تبدیل ساختار سازه‌ای به ساختار وابستگی (الگوریتم مسطح‌سازی) است. در این روش، پس‌از این که فرزند هسته هر گره میانی در درخت سازه‌ای مشخص شد، گره‌های درخت را با هدف استخراج قواعد پیمایش می‌کنیم.

الگوی استخراج قواعد در شکل (۶) نمایش داده شده است [20]. مطابق شکل، برای یک گره میانی مانند ZP با بیش‌از یک فرزند، چنانچه XP فرزند مرتبط با هسته آن باشد، برای هر فرزند غیر هسته YP، قاعده نمایش داده‌شده در شکل استخراج می‌شود. در یک سمت این قاعده رابطه وابستگی میان X به‌عنوان جزء کلام کلمه هسته XP و Y به‌عنوان جزء کلام کلمه هسته YP قرار دارد و در سمت دیگر یک درخت دودویی تک سطحی متشکل از ZP به‌عنوان ریشه و XP و YP به عنوان گره قرار دارد. البته اطلاعاتی مانند نقش دستوری کلمه وابسته (func) از ساختار متناظر در درخت‌بانک وابستگی گرفته شده است. این الگوریتم مستقل از زبان و درواقع روشی برای تجزیه یک درخت سازه‌ای براساس ساختار وابستگی جمله است. شرط این تجزیه، وجود یک هسته در هر زیردرخت از درخت سازه‌ای است که این وضعیت در درخت‌بانک سازه‌ای PerTreebank وجود دارد. برای مثال هسته واحد نحوی S به‌طورمعمول نخستین گره VP از سمت راست است.



(شکل-۶): الگوی استخراج قواعد از درخت‌بانک سازه‌ای

(Figure-6): The pattern for extracting conversion rules from a constituency treebank.

۴-۱-کارایی الگوریتم در ایجاد درخت سازه‌ای

برای اندازه‌گیری کارایی الگوریتم، ما آن را بر روی درختان درخت‌بانک وابستگی آزمایش کردیم. در ابتدا سعی کردیم دقت الگوریتم در تولید درخت سازه‌ای را ارزیابی کنیم. برای این کار، برای هر یال در ساختار وابستگی به‌طوردقیق از قاعده استخراج‌شده برای آن یال استفاده کردیم. در این حالت، کارایی الگوریتم تنها وابسته به نحوه اتصال الگوهای سازه‌ای برای ایجاد درخت سازه‌ای است.

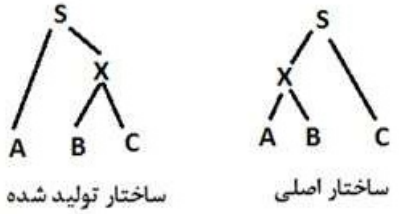
برای اندازه‌گیری کارایی از معیاری به نام PARSEVAL استفاده کرده‌ایم که روشی استاندارد برای اندازه‌گیری کارایی تجزیه‌گرهای نحوی است [3]. ما از نسخه پیاده‌سازی این الگوریتم با نام evalb استفاده نمودیم [15]. در این معیار، سه مقیاس برای مقایسه ساختارهای تولیدشده با ساختارهای اصلی محاسبه می‌شود: ۱- دقت[1]، ۲- پوشش[2] و ۳- تعداد براکت‌های متقاطع[3].

دقت، بیان‌گر نسبت واحدهای درست در ساختار تولیدشده به کل واحدهای نحوی در این ساختار است. پوشش، بیان‌گر نسبت این واحدها به کل واحدهای ساختار اصلی است. منظور از واحد درست، واحدی است که برچسب و کلمات دربرگیرندهٔ آن (شروع و خاتمه آن) با واحدی در ساختاراصلی منطبق باشد. اگر برچسب واحد نحوی برای ما مهم نباشد دقت و پوشش بی‌برچسب[4] و درغیراین‌صورت

[1] Precision
[2] Recall
[3] Crossing Brackets
[4] Unlabeled

[DOI: 10.29252/jsdp.14.4.79]

بابرچسب[1] خوانده می‌شوند. مقیاس سوم مربوط به ساختار درخت تولید شده است. منظور از براکت همان واحد نحوی است (ابتدا و انتهای هر واحد نحوی در جمله با یک پرانتز یا براکت مشخص می‌شود). زمانی دو براکت یا واحد نحوی متقاطع خوانده می‌شوند که محدوده هر کدام (ابتدا و انتهای آنها) به شکل جزئی با هم هم‌پوشانی داشته باشد. برای مثال، دو ساختار ((A (B C)) و ((A B) C)) هر کدام دارای یک براکت متقاطع هستند، زیرا B در دو براکت (B C) و (A B) مشترک است. این ساختارها در شکل (۷) نمایش داده شده‌اند.



(شکل-۷): یک خطای براکت متقاطع میان ساختار اصلی و ساختار تولیدشده

(Figure-7): A crossing branch between the original structure and the output structure.

در این مثال، با فرض بر این که ساختار نخست ساختار اصلی باشد، ساختار تولیدشده دارای دو براکت یا واحد نحوی است که یکی از آنها درست (واحد نحوی S) و در نتیجه دقت آن برابر با ۵۰٪ است. از آنجا که درخت اصلی نیز حاوی دو واحد نحوی است، مقیاس پوشش نیز ۵۰٪ و متوسط براکت متقاطع نیز یک است. (با توجه به اینکه تنها یک ساختار مقایسه شده و داری یک براکت متقاطع است).

(جدول-۲): کیفیت الگوریتم با استفاده از جدول افکنش رئوس

(Table-2): The results of using "The projection chain table"

	نخست سمت راست		نخست سمت چپ	
	متوسط براکت‌های متقاطع	دقت پوشش f-score	متوسط براکت‌های متقاطع	دقت پوشش f-score
الگوریتم مبنا	4.12	78.8 82.95 80.8	4.47	81.5 88.1 84.65
بدون جدول	0.34	98.0 96.9 97.44	1.5	91.0 93.1 92.03
با جدول	0.18	98.6 97.4 97.99	0.19	97.8 97.5 97.64

طبق این تعاریف، جدول (۲) کارایی الگوریتم را برحسب دقت و پوشش بابرچسب و متوسط براکت‌های متقاطع در حالات مختلف را نشان می‌دهد. f-score نشان‌دهنده میزان تأثیر دقت و پوشش برهم است. برای محاسبه کیفیت، الگوریتم بر روی کلیه جملات درخت‌بانک DepPerTreebank اجرا شده و خروجی تبدیل هر جمله با ساختار اصلی در PerTreebank مقایسه شده است. همان‌طور که گفته شد، ازآنجاکه در مرحله استخراج قواعد، شناسه قواعد متناظر با هر یال وابستگی ذخیره شده است، همان قاعده استخراج‌شده برای هر یال وابستگی به کارگرفته شده است. به معنی دیگر خطایی در انتخاب قواعد وجود ندارد.

منظور از «نخست سمت چپ» این است که در الگوریتم ابتدا فرزندان سمت چپ گره هسته و سپس فرزندان سمت راست آن پیمایش می‌شوند. در «نخست سمت راست» این ترتیب برعکس است. کلمات طبق نوشتار فارسی از راست به چپ مرتب می‌شوند.

منظور از «با جدول» و «بی جدول» این است که در الگوریتم اتصال (الگوریتم ۲) از جدول افکنش هسته‌ها استفاده شود یا خیر (شرط UseChainTable).

همان‌طورکه مشاهده می‌شود، اصلاحات انجام‌شده در هر دو حالت «بدون جدول» و «با جدول» بهبودی بیش از ۱۲٪ را نسبت به دقت الگوریتم مبنا ۸۴/۶۵٪ فراهم می‌آورند. همین‌طور، این اصلاحات تأثیر زیادی بر متوسط براکت‌های متقاطع دارند. استفاده از جدول افکنش هسته‌ها در هر دو روش پیماش تأثیر مثبتی بر دقت الگوریتم دارد؛ هرچند این تأثیر در پیمایش «نخست سمت چپ» به اندازه ۶/۸٪ و در پیمایش دیگر به اندازه ۰/۶٪ است. همچنین، استفاده از جدول، باعث بهبود قابل توجه مقیاس متوسط براکت‌های متقاطع در حالت «نخست سمت چپ» شده است. بنابراین درصورت عدم استفاده از جدول افکنش هسته‌ها بهتر است از پیمایش «نخست سمت راست» استفاده شود که در هر دو حالت دقت بالایی دارد.

ما سعی کردیم دلیل تأثیر نحوه پیمایش وابسته‌ها بر کارایی الگوریتم و به‌ویژه تأثیر استفاده از جدول در پیمایش «نخست سمت چپ» را بررسی کنیم. این تأثیر مربوط به نحوه ایجاد درخت سازه‌ای و ماهیت ساختارهای سازه‌ای جملات فارسی است که در ادامه توضیح داده می‌شود.

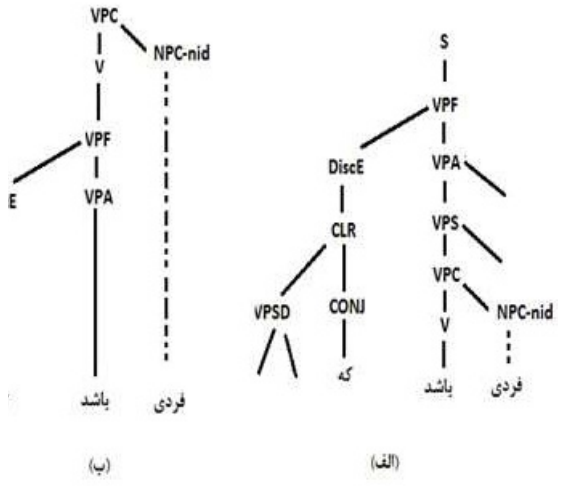
[1] Labeled

[DOI: 10.29252/jsdp.14.4.79]

۱-۱-۴- تأثیر نحوه پیمایش بر کارایی الگوریتم

طبق الگوریتم اتصال (الگوریتم ۲)، چنانچه نقطه مناسبی در زنجیره افکنش هسته پیدا نشود، زیردرخت به زنجیرهٔ افکنش افزوده می‌شود. این اتفاق به‌طورمعمول زمانی می‌افتد که فرزندان یک سمت به‌ترتیب پیمایش می‌شوند. در این حالت، زیردرختان به‌ترتیب بر روی یکدیگر سوار شده و زنجیره افکنش را کامل می‌کنند. برای مثال می‌توان این مطلب را در مراحل یک تا هفت ایجاد درخت سازه‌ای در شکل(۳) مشاهده کرد. اکنون چنانچه گره هسته دارای فرزندانی در سمت دیگر خود باشد، یافتن محل مناسب اتصال این فرزندان در زنجیره افکنش ایجادشده اهمیت پیدا می‌کند.

در شکل (۸ (الف)) قسمتی از ساختار سازه‌ای جمله «شاید او فردی باشد که ما می‌خواهیم» نمایش داده شده است. کلمه هسته این جمله کلمه «باشد» است که دارای کلمات وابسته‌ای در دو سمت خود است. با توجه به این مثال، قصد داریم دلیل دقت پایین الگوریتم در حالت «نخست از چپ» و «بدون جدول» را توضیح دهیم.



(شکل-۸): ترتیب اتصال زیر درختان برای جمله «شاید او همان فردی باشد که ما می‌خواهیم» با پیمایش «اول سمت چپ»

(Figure-8): The order of the attachment of the subtrees for a sample sentence according to "Frist Left Side" navigation of dependents.

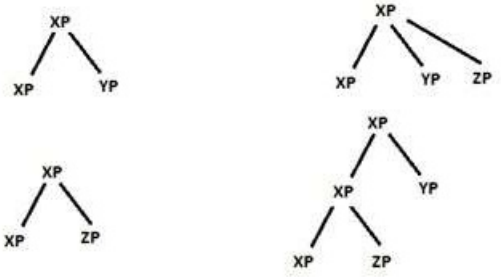
زمانی که ابتدا کلمات سمت چپ پیمایش می‌شوند، با توجه به نقطه اتصال زیردرخت مربوط که در بالای زنجیره است، زنجیره افکنش شامل VPF و VPA خواهد بود که VPA به‌طورمستقیم به هسته متصل است (شکل ۸ (ب)). در این حالت زمانی که کلمه «فردی» که در سمت راست است، پیمایش می‌شود، ازآنجاکه ریشه الگوی سازه‌ای متناظر آن (یعنی VPC) در زنجیره افکنش وجود ندارد، و همین‌طور هیچ نوع اطلاعاتی برای تشخیص اینکه آیا VPC یا V از VPA پایین‌تر هستند یا خیر نیز وجود ندارد، الگوریتم نقطه اتصال را نیافته و کل زنجیره افکنش به شاخه V متصل می‌شود که مسلماً خطا است؛ اما چنانچه از جدول افکنش هسته‌ها استفاده شود، چون VPC به‌طورمعمول پایین‌تر از VPA است، زیردرخت وابسته، پایین‌تر از VPC الحاق می شود. در این مثال، اگر ابتدا فرزندان سمت راست پیمایش شوند، حتی در نبود جدول افکنش هسته‌ها، وابسته‌ها به‌ترتیب به زنجیره افکنش هسته اضافه شده و درخت سازه ای صحیح حاصل می‌شود. البته در این روش پیمایش نیز مواردی وجود دارد که نقطه اتصال نخستین وابسته سمت راست در بالای زنجیره افکنش، و با فاصله چندین افکنش از کلمه هسته قرار دارد. در این حالت، نیز عدم استفاده از جدول، مشکلی مشابه مثال بالا را ایجاد می‌کند؛ اما چنین مواردی نسبت به موارد مشابه پیمایش «نخست از چپ» در درخت‌بانک وابستگی کمتر هستند. با بررسی درخت‌بانک وابستگی مشخص شد، حدود ده درصد از هسته‌ها دارای وابستگانی در دو سمت خود هستند که این امر بیشتر مربوط به جملات و کلمات همپایگی و جملات مرکب و یا خروج بند موصولی از مکان اصلی خود (مانند مثال بالا) بود. در این میان، خروج بند موصولی و جملات مرکب به‌علت این که زیردرخت سمت چپ در افکنش‌های سطح بالاتری به این زنجیره متصل می‌شوند، بیشترین خطا را ایجاد می‌کنند. این خطا تأثیر خود را به‌ویژه در افزایش براکت‌های متقاطع نشان می‌دهد.

در الگوریتم مبنا، که کارایی آن در جدول (۲) نمایش داده شده است، از آنجا که پایین‌ترین گره در زنجیره افکنش انتخاب می‌شود، پیماش «نخست سمت چپ» دارای دقت بالاتری است. این امر با توجه به توضیحات بالا قابل انتظار است. برای مثال، اگر مراحل نشان داده‌شده در شکل (۳) با این روش پیمایش و با استفاده از الگوریتم مبنا طی شود، خطای نشان‌داده‌شده در شکل (۸) رخ نخواهد داد.

هر دو روش پیمایش، درصورت استفاده از جدول، دقت بالایی دارند؛ اما همچنان برخی دلایل مانع رسیدن به دقت و پوشش صددرصد می‌شوند. در برخی موارد ممکن است، حالت‌های مختلفی برای ترکیب زیردرختان یکسان وجود داشته باشد، که الگوریتم با انتخاب یک حالت، حالت دیگر را تحت پوشش خود قرار نخواهد داد. برای مثال، در

[DOI: 10.29252/jsdp.14.4.79]

شکل (۹) دو روش برای ترکیب زیردرختان سمت چپ وجود دارد. البته ممکن است، این ابهام با درنظرگرفتن اطلاعات بیشتر رفع شود؛ اما چنانچه در درخت‌بانک سازه‌ای برای یک مفهوم از چندین روش ناسازگار استفاده شود، رفع این خطا مشکل‌تر خواهد شد.



(شکل-۹): دو حالت ممکن برای ترکیب زیردرختان سمت چپ.
(Figure-9): Two possible ways to combine the subtrees on the left side.

منشاء دیگر خطا، ناسازگاری در ترتیب واحدهای نحوی در زنجیره افکنش است. هر چند ما همیشه از ترتیب معمول (با تکرار بالا) استفاده می‌کنیم، اما مواردی نیز وجود دارد که در آنها این ترتیب برعکس است.

همین‌طور منشاء عمده دیگر خطا، انتخاب پایین‌ترین نقطه ممکن برای اتصال زیردرخت وابسته به زنجیره افکنش هسته در الگوریتم اتصال است (پیمایش زنجیره افکنش از پایین به بالا در الگوریتم (۲)) که ممکن است در ساختار اصلی همیشه به این شکل نباشد و زیردرخت وابسته در نقطه بالاتری متصل شده باشد. علاوه‌بر اینها وجود افکنش‌های تک‌انشعابی (تک‌فرزندی) در ساختارهای سازه‌ای اصلی عامل دیگر عدم انطباق خروجی الگوریتم با این ساختارها می‌تواند باشد. ازآنجاکه الگوی سازه‌ای در قواعد تبدیل به شکل یک درخت دوانشعابی تک‌سطحی است، افکنش‌های تک‌انشعابی را پوشش نمی‌دهد. در جدول (۳)، درصد فراوانی انواع انشعاب برای گره‌های داخلی ساختارهای سازه‌ای درخت‌بانک PerTreebank آورده شده است. همان‌طورکه ملاحظه می‌شود، حدود نه درصد موارد مربوط به افکنش‌های تک‌انشعابی است.

(جدول-۳): درصد فراوانی انواع انشعاب در ساختارهای سازه‌ای PerTreebank

(Table-3): The frequency of each type of branches (one, two and more)

تک انشعابی	دوتایی	سه‌تایی یا بیشتر
۹/۵	۸۹/۵	۱

مجموعه این موارد باعث شدند که حتی درصورت استفاده از قواعد صحیح، همچنان به دقت صد درصد نرسیم که دو درصد خطا ناشی از موارد مذکور است؛ اما درعمل همواره قواعد صحیح انتخاب نمی‌شوند. در قسمت بعد به این مطلب می‌پردازیم.

۲-۴-کارایی الگوریتم در انتخاب درست قواعد

در قسمت قبل برای اندازه‌گیری کارایی الگوریتم، در ترکیب الگوها و بازسازی درخت سازه‌ای، به‌عمد از الگوی متناظر هر یال وابستگی استفاده کردیم. اما درعمل، الگوریتم باید قادر باشد تا با توجه به اطلاعات مربوط به یال وابستگی، قاعده درست را انتخاب کند؛ اما یک تناظر یک‌به‌یک میان الگوهای وابستگی و الگوهای سازه‌ای وجود ندارد. برای مثال در قواعد شکل ۲ در سمت چپ قواعد ۳ و ۴ الگوی وابستگی (یال وابستگی) یکسانی وجود دارد؛ و یا به عبارت دیگر، یک الگوی وابستگی با دو الگوی سازه‌ای متناظر شده است. به چنین الگویی «الگوی مـبهم[1]» می‌گوییم. البته درصورت استفاده از اطلاعات بیشتر در مورد الگوی وابستگی این ابهام را تا حدی می‌توان برطرف کرد. برای مثال در همان شکل، چنانچه برچسب یال وابستگی (نقش کلمه وابسته) را در نظر نگیریم، الگوهای وابستگی قواعد ۱ و ۲ نیز یکسان هستند؛ اما با لحاظ این برچسب، ابهام برطرف می‌شود.

با استفاده از ساختارهای درخت‌بانک سازه‌ای PerTreebank و درخت‌بانک وابستگی DepPerTreebank، ۲۶۸۰۲ قاعده استخراج کردیم که این قواعد شامل ۶۱۹۷ الگوی وابستگی غیرتکراری (برحسب مقوله‌های دستوری هسته و وابسته و جهت وابستگی) و ۲۹۱۷ الگوی متفاوت سازه‌ای بود. مقیاس «درجه ابهام» را به‌عنوان حاصل تقسیم تعداد کل قواعد (الگوهای وابستگی) بر تعداد الگوهای سازه‌ای متفاوت تعریف می‌کنیم؛ برای مثال در حالت مذکور، درجه ابهام برابر با ۲/۱۲ = ۲۹۱۷ / ۶۱۹۷ به‌دست می‌آید. برای کاهش ابهام الگوها، الگوریتم مبنا [20] اطلاعات بیشتری را برای ایجاد الگوهای وابستگی لحاظ کردند. این اطلاعات عبارت بودند از:

- الف) مقوله دستوری هسته و وابسته و موقعیت مکانی (راست یا چپ) وابسته نسبت به هسته
- ب) اطلاعات بالا به‌علاوهٔ برچسب یال وابستگی

[1] ambiguous pattern

— پ) تمامی اطلاعات بالا به‌علاوهٔ برچسبی که نشان می‌دهد گره وابسته برگ است یا خیر.
— ت) تمامی اطلاعات بالا به‌علاوهٔ برچسبی که نشان می‌دهد گره هسته وابسته‌های دیگری نیز دارد یا خیر.

همین‌طور طبق الگوریتم مبنا، زمانی که همچنان چندین قاعده با الگوهای بالا منطبق می‌شود از دو راهبرد زیر برای انتخاب قاعده استفاده می‌شود:

— ث) از میان قواعد یافت‌شده، قاعده‌ای که بیشترین کاربرد (بیشترین بسامد در قواعد استخراج شده) را دارد، انتخاب شود.
— ج) از میان قواعد یافت‌شده، قاعده‌ای که بیشترین تطابق با زنجیره افکنش فعلی را دارد، انتخاب شود (برچسب شاخه هسته آن در زنجیره وجود دارد).

ما کارایی الگوریتم در زبان فارسی را با استفاده از هر الگو اندازه‌گیری کردیم که این نتایج در جدول (۴) آمده است. برای این کار، کلیه قواعد استخراج‌شده از دو درخت‌بانک سازه‌ای و وابستگی استفاده شده‌اند. با استفاده از این قواعد، الگوریتم را بر روی کلیه جملات درخت‌بانک وابستگی اجرا کردیم و خروجی را با ساختارهای سازه‌ای مقایسه کردیم. در این حالت، دقت الگوریتم وابسته به انتخاب صحیح قواعد و ایجاد درست ساختار سازه‌ای است. همانطور که مشاهده می‌شود، با بکارگیری اطلاعات بیشتر درسمت الگوی وابستگی، تعداد الگوهای وابستگی قابل تفکیک بیشتر می‌شوند و به عبارت دیگر«درجه ابهام» قواعد کمتر می‌شود. در ستون "قواعد صحیح"، درصد قواعدی که در هر حالت به شکل صحیح توسط الگوریتم انتخاب شده‌اند، نشان داده شده است. همان‌طورکه مشاهده می‌شود با کاهش درجه ابهام، انتخاب قواعد صحیح‌تر صورت می‌گیرد و دقت الگوریتم افزایش می‌یابد.

همچنین، برای کلیه الگوها به جز الگوی (الف)، راهبرد (و) تأثیر بیشتری بر دقت الگوریتم داشت و بیشترین دقت از ترکیب (د) و (و) برابر با ۸۱/۳٪ به‌دست آمد. نکته جالب توجه این است که زمانی که درجه ابهام قواعد بالا است (الگوی الف)، راهبرد (ه) مبنی بر انتخاب قاعده‌ای با بیشترین بسامد (احتمال استفاده) باعث ایجاد دقت بالاتری می‌شود. همین‌طور برای کلیه الگوها راهبرد (ه) دارای درصد قواعد صحیح بالاتری است و میزان پوشش بالاتری را نیز دارد و یا به‌عبارتی واحدهای نحوی ساختار اصلی را بهتر تحت پوشش خود قرار می‌دهد. با این حال، به‌جز حالت (الف)، راهبرد (و) در بقیه حالت‌ها به شکل جزئی دارای دقت بالاتری است. دلیل این تفاوت این است که راهبرد (و) قاعده‌ای را انتخاب می‌کند که بیشترین تطابق با زنجیره افکنش هسته را دارد؛ درنتیجه گره‌های اضافی کمتری تولید شده و دقت بالاتری دارد؛ اما در حالت (الف)، از آنجا که قواعد بیشتری با الگوی وابستگی منطبق می‌شوند، احتمال خطا با استفاده از راهبرد (و) افزایش پیدا می‌یابد و استفاده از بسامد قواعد راهبرد مطمئن‌تری است. یکی از نتایجی که از این جدول حاصل می‌شود این است که در صورت استفاده از الگوهای کلی‌تر حالت (الف) راهبرد (ه) به معنای استفاده از قاعده‌ای با بیشترین بسامد بهتر عمل می‌کند؛ و برای الگوهای جزئی‌تر حالت (د)، راهبرد (و) به معنی تطبیق قاعده با زنجیره افکنش، گزینه بهتری است.

(جدول-۴): کیفیت الگوریتم در انتخاب قواعد صحیح برای الگوهای مختلف وابستگی

(Table-4): The results of conversion algorithm based on different rule matching methods.

	درجه ابهام قواعد	کارایی (ه)		کارایی (و)	
		قواعد صحیح	دقت پوشش	قواعد صحیح	دقت پوشش
			f-score		f-score
الف	2.17	65.4	70.29 84.11	59.4	67.6 80.18
			76.58		73.35
ب	1.68	72.3	78.67 93.94	69.9	78.9 93.16
			85.62		85.44
ج	1.57	75.6	80.22 94.04	74.3	80.55 93.6
			86.58		86.56
د	1.51	76.8	81.1 94.24	76.1	81.3 93.5
			87.17		86.97

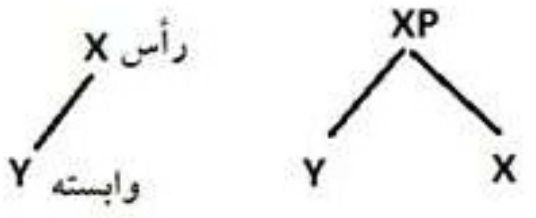
منشاء اصلی کاهش کارایی در این قسمت، وجود ابهام در قواعد و انتخاب نادرست قواعد است. با استفاده از اطلاعات بیشتر این ابهام را تا حدی می‌توان کاهش داد. چنانچه ساختار وابستگی تمایزی میان دو مفهوم قائل نشود، اما چنین تمایزی در ساختار سازه‌ای وجود داشته باشد، این ابهام اجتناب ناپذیر است؛ زیرا اطلاعات لازم برای رفع ابهام در اختیار نخواهد بود. همچنین، وجود ناسازگاری در نمایش ساختارهای مربوط به یک مفهوم در درخت‌بانک سازه‌ای

[DOI: 10.29252/jsdp.14.4.79]

منشاء این خطا می‌تواند باشد. از طرفی استفاده از الگوهای پیچیده‌تر لزوماً به کارایی بالاتر در پوشش ساختارهای دیده‌نشده منتهی نمی‌شود. در این حالت، الگوها بسیار جزئی شده و ممکن است در بسیاری موارد با روابط وابستگی در مجموعه‌ای جدا از مجموعه آموزشی منطبق نشوند و درعمل بهبود چندانی را حاصل نکنند. به‌عنوان کارهای آینده، بر روی ایجاد الگوهای پیچیده‌تر و در عین حال عمومی می‌توان پژوهش کرد.

۳-۴- کارایی الگوریتم برای پوشش روابط وابستگی دیده نشده

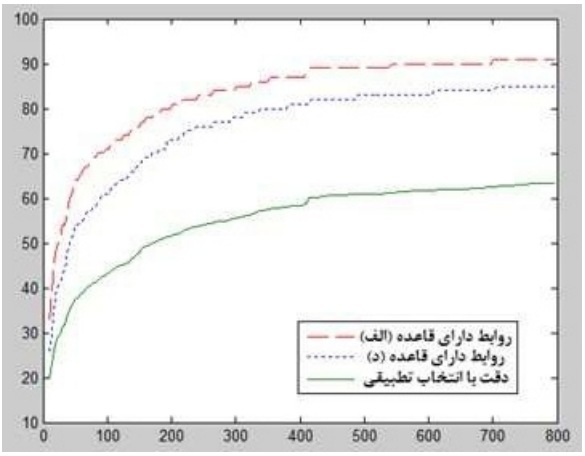
در قسمت قبل، ما مطمئن بودیم که برای هر رابطه وابستگی قاعده‌ای وجود دارد و خطاهای الگوریتم ناشی از انتخاب نادرست قواعد بود؛ اما چنانچه بخواهیم با همین مجموعه قواعد، الگوریتم را به ساختارهای وابستگی جدیدی اعمال کنیم، ممکن است برای برخی از الگوهای وابستگی قاعده‌ای یافت نشود. با افزایش داده آموزش (داده استخراج قواعد) این مشکل می‌تواند کاهش یابد.



(شکل-۱۰): الگوی سازه‌ای پیش‌فرض برای یک الگوی وابستگی.
(Figure-10): The default constituency pattern for a dependency pattern.

برای بررسی تأثیر اندازه داده آموزش بر کارایی الگوریتم، داده‌های موجود (۱۰۲۸ جمله) را به دو دسته تقسیم کردیم. ۲۲۸ جمله را به‌طور ثابت به‌عنوان داده‌های آزمایشی کنار گذاشتیم و به‌ترتیب از ده تا هشتصد جمله را برای استخراج قواعد استفاده و در هر مورد با استفاده از قواعد استخراج‌شده الگوریتم را روی داده‌های آزمایشی اعمال کردیم. در مواردی که برای یک الگوی وابستگی قاعده‌ای وجود نداشت، از یک الگوی سازه‌ای پیش‌فرض مطابق شکل (۱۰) استفاده نمودیم.

نمودار نشان‌داده‌شده در شکل (۱۱) دقت بابرچسب الگوریتم را بر حسب اندازه داده آموزش نشان می‌دهد. ما دقت الگوریتم را مشخصاً با استفاده از دو الگوی وابستگی (الف) و (د) که در قسمت قبل معرفی شدند، اندازه‌گیری کردیم. مطابق شکل، زمانی که اندازه داده آموزش کم است، هرچه الگوی وابستگی جزئی‌تر باشد (حالت (د))، روابط وابستگی کمتری با آن منطبق می‌شوند و دقت آن نیز کمتر است؛ درمقابل، دقت الگوی (الف) که کلی‌تری است، بالاتر است. با افزایش داده‌های آموزش، قواعد و الگوهای سازه‌ای بیشتری استخراج می‌شوند و همزمان درجه ابهام قواعد نیز بالاتر می‌رود. در این حالت، الگوهای کلی‌تر (حالت (الف)) ابهام بالاتری و الگوهای جزئی‌تر ابهام کمتری خواهند داشت. درنتیجه فاصله میان دقت الگوی (د) و الگوی (الف) کاهش می‌یابد و درنهایت الگوی (د) به دقت بالاتری دست می‌یابد. در هر دوی این حالت‌ها برای انتخاب یکی از قواعد در الگوهای مبهم از راهبرد (و) استفاده شده‌است.

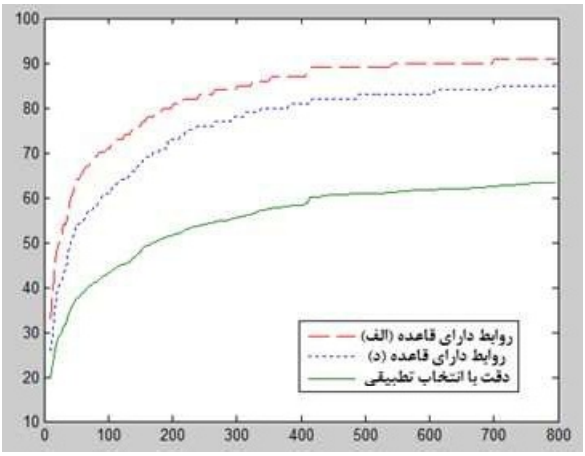


(شکل-۱۱): دقت الگوریتم بر روی داده‌های آزمایشی در حالات مختلف برحسب اندازه مجموعه آموزشی
(Figure-11): The results of the accuracy of the algorithm on test data versus the size of training data.

در اینجا روش دیگری را برای انطباق الگوهای وابستگی و انتخاب قواعد به‌کار بردیم که دقت آن فارغ از اندازه داده آموزش از دقت کلیه الگوها بالاتر است. این روش را «انتخاب تطبیقی» نامیدیم. در این روش ابتدا سعی می‌کنیم با استفاده از الگوی (د) و راهبرد (و) قاعده‌ای را برای یک رابطه وابستگی بیابیم. چنانچه قاعده‌ای یافت نشود از اطلاعات کمتری در سمت الگوی وابستگی استفاده و به جای الگوی (د) از الگوی (ج) استفاده می‌کنیم. به همین ترتیب اگر باز قاعده‌ای یافت نشد، به سمت استفاده از الگوهای کلی‌تر (ب) و (الف) می‌رویم. همین‌طور برای الگوهای کلی‌تر با درجه ابهام بالا از راهبرد (ه) به جای (و) استفاده می‌کنیم. همان‌طورکه درقبل گفته شد، زمانی که

[DOI: 10.29252/jsdp.14.4.79]

تعداد قواعد منطبق با یک الگو زیاد هستند، راهبرد (ه) مبنی بر انتخاب قاعده با بیشتری بسامد بهتر عمل می‌کند. دقت الگوریتم با استفاده از این روش در شکل (۱۲) با نام «انتخاب تطبیقی» نمایش داده شده است. دلیل این نام‌گذاری انعطاف و تطبیق الگوریتم در استفاده از اطلاعات رابطه وابستگی در یافتن و انتخاب قاعده مناسب است. همان‌طور که مشاهده می‌شود، این حالت همواره از دو حالت قبل کارایی بالاتری دارد. کارایی آن از تطبیق الگوی (د) بیشتر است؛ زیرا هر رابطه‌ای را که الگوی (د) پوشش دهد، این روش نیز پوشش می‌دهد. همین‌طور کارایی آن از تطبیق الگوی (الف) بیشتر است؛ زیرا ابتدا از الگوهای جزئی‌تر با درجه ابهام کمتر شروع می‌کند و درصورت شکست به سمت استفاده از الگوی کلی‌تر (الف) می‌رود. در نتیجه در انتخاب قواعد از بیشینه اطلاعات موجود بهره می‌برد و قواعد دقیق‌تری انتخاب می‌کند.



(شکل-۱۲): نمودار دقت الگوریتم و درصد روابط دارای قاعده برحسب اندازه مجموعه آموزشی

(Figure-12): The relation between the accuracy of algorithm (green line) and the present rules (red, blue) versus the size of training data.

شکل (۱۲) درصد روابط وابستگی را در داده آموزش که با الگویی در قواعد استخراج‌شده منطبق می‌شوند، برای دوالگوی (الف) و (د) نشان می‌دهد. همان‌طور که مشاهده می‌شود، با افزایش داده‌های آموزش روابط وابستگی بیشتری تحت پوشش قرار می‌گیرند؛ اما این افزایش خطی نیست. طبق این نمودار با یک داده آموزش به اندازه هشتصد جمله ۸۵٪ روابط در داده آموزش طبق الگوی (د) تحت پوشش قرار می‌گیرد. ازآنجاکه نمودار لگاریتمی است، برای رسیدن به درصدهای بالاتر از نود درصد به داده آموزش بسیار بزرگتری احتیاج داریم. افزایش داده آموزش باعث افزایش دقت الگوریتم و از طرفی با افزایش قواعد، ابهام الگوها نیز بیشتر می‌شود که تأثیر معکوسی بر دقت الگوریتم دارد.

جدول (۵) کارایی الگوریتم بر روی ساختارهای دیده‌نشده را در مقایسه با الگوریتم مبنا و الگوریتم ارائه‌شده توسط دهقان و فیلی [2] نشان می‌دهد. برای ایجاد شرایط یکسان، از نود درصد PerTreebank برای داده آموزش و از ده درصد باقیمانده به عنوان داده آزمون استفاده شد و کارایی بدون برچسب براساس میزان تأثیر دقت و پوشش برهم (f-score) به‌عنوان معیار ارزیابی محاسبه و رقم ۸۲/۶۲٪ حاصل شد.

(جدول-۵): کارایی الگوریتم تبدیل پیشنهادی در مقایسه با سایر الگوریتم‌های مرتبط

(Table-5): The results of our proposed alogorithm compared with other existing methods.

		دقت	پوشش	f-score
الگوریتم مبنا (بی‌برچسب)		55.29	82.34	66.15
الگوریتم دهقان و فیلی(طبقه بند) (بی‌برچسب)		64.88	92.65	76.32
الگوریتم دهقان و فیلی(طبقه بند + مکاشفه‌ای) (بی‌برچسب)		95.39	88.95	92.06
الگوریتم پیشنهادی (باجدول + انتخاب تطبیقی)	با برچسب	69.08	85.64	76.47
	بی برچسب	74.63	92.53	82.62

تنظیمات الگوریتم پیشنهادی شامل استفاده از جدول افکنش هسته‌ها در اتصال زیردرختان و راهبرد انتخاب تطبیقی در انتخاب قواعد است. این نتایج ۱۶/۴۷٪ بهبود را نسبت‌به الگوریتم مبنا و ۶/۳٪ بهبود نسبت به الگوریتم دهقان و فیلی (۱۳۹۵) در حالت استفاده از طبقه‌بند نشان می‌دهد. دهقان و فیلی یک مرحله پس‌پردازش مکاشفه‌ای را نیز به الگوریتم خود اضافه کردند که هدف آن اصلاح ساختارهای تولید شده است. این مرحله شامل اصلاح برخی انشعاب‌های تکی در ساختارهای خروجی است. این انشعاب‌ها ممکن است، به‌علت خطا در انتخاب قواعد و یا خطا در اتصال زیردرختان حاصل شوند. به این منظور، تمام ساختارهای موجود در داده آموزش پردازش شده و فهرستی از انشعاب‌های تکی ذخیره می‌شود. در مرحله پس‌پردازش، چنانچه یک انشعاب تکی در این فهرست نبود، با جایگزین‌کردن گره والد با تک‌فرزند خود اصلاح می‌شود. با انجام این اصلاحات، کارایی بی‌برچسب الگوریتم نهایی ۹۲/۰۶٪ گزارش شده است. البته در این حالت، میزان

[DOI: 10.29252/jsdp.14.4.79]

پوشش الگوریتم کاهش یافته است که به‌علت حذف برخی برچسب‌ها است. این مرحله پس‌پردازش در الگوریتم فعلی ما وجود ندارد و می‌تواند به‌عنوان کارهای آینده مورد ملاحظه قرار گیرد. با این حال، مورد مناسب برای مقایسه الگوریتم پیشنهادی در این مقاله و الگوریتم دهقان و فیلی [2]، استفاده از طبقه‌بند برای اتصال زیردرخت‌ها است. در این حالت استفاده از جدول افکنش هسته‌ها برای این عمل در کنار روش انتخاب تطبیقی برای انتخاب قواعد ۶/۳٪ بهبود را نشان می‌دهند. همین‌طور معیار پوشش الگوریتم ارائه‌شده از الگوریتم نهایی دهقان و فیلی (طبقه‌بند و مکاشفه‌ای) به اندازه ۳/۵٪ بیشتر است.

۵- نتیجه‌گیری

در این مقاله، با بررسی کارهای مرتبط، یکی از بهترین الگوریتم‌های موجود برای تبدیل یک درخت‌بانک وابستگی به معادل سازه‌ای را انتخاب کردیم و ضمن به‌کارگیری آن در زبان فارسی، اصلاحاتی را برای بهبود کارایی آن پیشنهاد کردیم. در این روش با استفاده از مجموعه‌ای از قواعد تبدیل ساختارهای سازه‌ای مورد نظر را می‌توان ایجاد کرد. یک قاعده تبدیل تناظری میان یک الگوی وابستگی و یک الگوی سازه‌ای است. این قواعد می‌توانند دست‌ساز بوده و یا از یک درخت‌بانک سازه‌ای موجود استخراج شوند.

ما الگوریتم تبدیل را بر روی یک درخت‌بانک وابستگی فارسی اعمال و قواعد را به طور خودکار از درخت بانک سازه‌ای متناظر آن استخراج کردیم. با فراهم‌بودن این قواعد، الگوریتم با دقت ۹۸٪ درخت‌بانک وابستگی را به معادل سازه‌ای آن تبدیل می‌کند. در این زمینه ما عوامل مؤثر بر دقت الگوریتم را مورد بررسی قرار دادیم. مشخص شد نحوه پیمایش گره‌های وابسته یک هسته («نخست از راست» یا «نخست از چپ») بر دقت الگوریتم تأثیرگذار است. همین‌طور استفاده از جدول افکنش هسته‌ها برای اتصال زیردرختان و تولید ساختار نهایی، تأثیر قابل ملاحظه‌ای بر دقت الگوریتم دارد و تأثیر نحوه پیمایش را خنثی می‌کند؛ سپس دقت الگوریتم را در انتخاب قاعده صحیح برای یک الگوی وابستگی بررسی کردیم. خاطر نشان کردیم که این الگوها دارای ابهام هستند و با افزایش اطلاعات در سمت الگوی وابستگی انتخاب دقیق‌تری داشت و به دقت بالاتری می‌توان دست یافت. با این حال، درصورت عدم تمایز میان دو مفهوم در درخت‌بانک وابستگی (عدم وجود اطلاعات برای رفع ابهام) و یا ناسازگاری در نمایش یک رابطه وابستگی در درخت بانک سازه‌ای (تناقض در پوشش یک مفهوم) چنین ابهامی اجتناب ناپذیر است.

در انتها، ما دقت الگوریتم را برای اعمال بر روی ساختارهای خارج از داده‌های آموزشی مورد بررسی قرار دادیم. در این حالت، اصلی‌ترین عامل خطا، عدم تحت پوشش قراردادن برخی از روابط وابستگی توسط قواعد موجود است. در این وضعیت، هرچه الگوهای وابستگی کلی‌تر باشند، روابط وابستگی بیشتری را تحت پوشش قرار می‌دهند؛ اما همزمان الگوهای سازه‌ای قابل تطبیق با یک رابطه وابستگی افزایش می‌یابند و انتخاب الگوی سازه‌ای صحیح سخت‌تر شود. برای رفع این مشکل یک روش ویژه با نام «انتخاب تطبیقی» را ارائه کردیم. در این روش، ابتدا سعی می‌کنیم با استفاده از الگوهای جزئی‌تر، قاعده‌ای را بیابیم و درصورت عدم موفقیت، اطلاعات الگوی وابستگی را کاهش می‌دهیم تا درنهایت قاعده‌ای برای رابطه وابستگی یافت شود.

با ارزیابی الگوریتم ارائه‌شده به کارایی بدون برچسب براساس میزان تأثیر دقت و پوشش بر هم (f-score) به رقم ۸۲/۶۲٪ دست پیدا کردیم. این نتایج ۱۶/۴۷٪ بهبود را نسبت به الگوریتم مبنا و ۶/۳٪ بهبود را نسبت به الگوریتم دهقان و فیلی [2] در حالت استفاده از طبقه‌بند نشان می‌دهند.

۶- مراجع 6-References

[۱] سلطان‌زاده ف.، بحرانی م. و اسلامی م. "ارائه راهکاری قاعده‌مند جهت تبدیل خودکار درخت تجزیه نحوی وابستگی به درخت تجزیهٔ ساخت‌سازه‌ای برای زبان فارسی"پردازش علائم و داده‌ها .جلد ۱۲، شماره ۴، صفحه ۱۱۵-۹۵، ۱۳۹۴.

[1] Soltanzadeh F, Bahrani M, Eslami M. "A Rule-Based Approach in Converting a Dependency Parse Tree into Phrase Structure Parse Tree for Persian", JSDP. vol. 12 (4), pp. 95-115, 2016.

[۲] دهقان، م.،فیلی، ه. "تولید درختبانک سازه‌ای زبان فارسی به روش تبدیل خودکار". پردازش علائم و داده‌ها. جلد ۲۸، شماره ۲، صفحه ۱۲۱-۱۳۷، ۱۳۹۵.

[2] Dehghan M H, Faili H. "Generating the Persian Constituency Treebank in an Automatic Converting Method", JSDP, vol. 13 (2), pp.121-137, 2016.

[DOI: 10.29252/jsdp.14.4.79]

[3] Black, Ezra, et al. "A procedure for quantitatively comparing the syntactic coverage of English grammars." Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991.

[4] Bhatt, Rajesh, and Fei Xia. "Challenges in converting between treebanks: a case study from the hutb." META-RESEARCH Workshop on Advanced Treebanking. 2012.

[5] Collins, Michael, et al. "A statistical parser for Czech." Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics. Association for Computational Linguistics, 1999.

[6] Covington, Michael A. "An empirically motivated reinterpretation of Dependency Grammar." arXiv preprint cmp-lg/9404004, 1994.

[7] Ghayoomi, Masood. "Bootstrapping the Development of an HPSG-based Treebank for Persian." Linguistic Issues in Language Techno-logy vol. 7, no. 1, pp. 1-13. 2012.

[8] Ghayoomi, M., & Kuhn, J. “Converting an HPSG-based Treebank into its Parallel Dependency-based Treebank”. In LREC, pp. 802-809, 2014.

[9] Kaplan, Ronald M. "The formal architecture of lexical-functional grammar." Formal issues in lexical-functional grammar, vol. 47, pp. 7-27, 1995.

[10] Klein, A., “From dependency to constituency: Automatic generation of Penn Treebank trees from LFG f-structures”, M.S. Thesis , University of Heidelberg, Germany, 2009.

[11] Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. “Building a large annotated corpus of English: The Penn Treebank”, Computational linguistics, vol. 19, no. 2, pp 313-330, 1993.

[12] Pollard, Carl, and Ivan A. Sag. Head-driven phrase structure grammar. University of Chicago Press, 1994.

[13] Rasooli, M. S., Kouhestani, M., & Moloodi, A. “Development of a Persian syntactic dependency treebank”. In Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 306-314, 2013.

[14] Schabes, Y., Abeille, A., & Joshi, A. K. “Parsing strategies with 'lexicalized' grammars: application to tree adjoining grammars”, In Proceedings of the 12th conference on Computational linguistics, Association for Computational Linguistics, vol. 2, pp. 578-583, 1988.

[15] SekineS. & Collins.M. J, The evalb software, 1997. Available: http://cs.nyu.edu/cs/projects/proteus/evalb. [Accessed: 01- Oct- 2017].

[16] Seraji, M., Megyesi, B., & Nivre, J. “Bootstrapping a Persian dependency treebank”. Linguistic Issues in Language Technology, vol. 7, no. 18, pp 1-10, 2012.

[17] Steedman, M. The syntactic process, vol. 24. Cambridge: MIT press, 2000.

[18] Wang, Z., & Zong, C. “Phrase structure parsing with dependency structure”, In Proceedings of the 23rd International Conference on Computational Linguistics: Posters, Association for Computational Linguistics, pp. 1292-1300, August. 2010.

[19] Xia, F., & Palmer, M. “Converting dependency structures to phrase structures”, In Proceedings of the first international conference on Human language technology research, Association for Computational Linguistics, pp. 1-5, March. 2001.

[20] Xia, F., Rambow, O., Bhatt, R., Palmer, M., & Misra Sharma, D. “Towards a multi-representational treebank”, LOT Occasional Series, vol. 12., pp. 159-170, 2008.

احمد پورامینی فارغ‌التحصیل مهندسی نرم‌افزار از دانشگاه صنعتی شریف است و مدرک کارشناسی ارشد خود را در رشته هوش مصنوعی از دانشگاه علم و صنعت اخذ کردند. ایشان از سال ۱۳۹۰ به‌عنوان عضو هیئت علمی دانشگاه صنعتی سیرجان مشغول به تدریس و پژوهش هستند. زمینه پژوهشی مورد علاقه ایشان پردازش زبان طبیعی، یادگیری ماشین و داده‌کاوی است.

نشانه رایانامه ایشان عبارت است از:

pouramini@gmail.com

مسعود قیومی عضو هیأت علمی پژوهشگاه علوم انسانی و مطالعات فرهنگی است. وی فارغ‌التحصیل مقطع دکترای رایانه با گرایش زبان‌شناسی رایانشی در سال ۲۰۱۴ از دانشگاه آزاد برلین آلمان است. ایشان در سال ۲۰۰۹ از دانشگاه سارلند آلمان و در سال ۲۰۰۸ از دانشگاه نانسی۲ فرانسه موفق به اخذ مدرک کارشناسی ارشد در رشته زبان‌شناسی رایانشی شد. همچنین وی در سال ۱۳۸۳ دوره کارشناسی ارشد خود را در رشته زبان‌شناسی همگانی در دانشگاه آزاد واحد تهران مرکز به پایان رساند. ایشان در سال ۱۳۸۰ در رشته مترجمی

[DOI: 10.29252/jsdp.14.4.79]

زبـان انگلیسـی از دانشـگاه آزاد اسـلامی قـم فارغ‌التحصـیل شد. زمینه‌های تخصصی مـورد علاقـهٔ ایشـان پـردازش زبـان طبیعی، مدل‌سازی زبانی، نحو و معناشناسی واژگانی است.
نشانی رایانامه ایشان عبارت است از:
M.Ghayoomi@ihcs.ac.ir

امینه ناصری مدرک کارشناسی خود را در رشته مهندسی کامپیوتر از دانشگاه شیراز در سال ۱۳۸۵ و مدرک کارشناسی ارشد خود را در رشته هوش مصنوعی از دانشگاه شاهرود اخذ کردند و هم‌اینک به‌عنوان عضو هیئت علمی دانشگاه صنعتی سیرجان مشغول فعالیت هستند. زمینه‌های پژوهشی مورد علاقه ایشان پردازش تصویر و یادگیری ماشین است.
نشانی رایانامه ایشان عبارت است از:
naseri.amine@yahoo.com

[DOI: 10.29252/jsdp.14.4.79]