IDMC2008
11-12 NOV
Amir Kabir University

تشخیص وزن عروضی اشعار فارسی: کاربرد جدیدی از متن‌کاوی

محمد مهدی مجیری[۱] ; بهروز مینایی بیدگلی[۲]

چکیده

سابقه تاریخی شعر و ادب فارسی و همچنین دست‌یازی‌های بیگانگان به سوابق تاریخی این عرصه, لزوم ورود تحقیقات کامپیوتری و بین رشته‌ای در این زمینه را طلب می‌نماید. یکی از مشکل‌ترین بخش‌های رشته‌ی ادبیات فارسی تشخیص وزن عروضی است، به صورتی که تشخیص وزن عروضی در بعضی اشعار برای اساتید ادبیات هم مشکل‌زا می‌شود. با توجه به این‌که تاکنون هیچ پژوهشی در این زمینه انجام نشده, ورود به این عرصه ضروری به نظر می‌رسد. ما در این مقاله ابتدا روش و شیوه‌های اخذ دانش و ایجاد پایگاه دانش قوانین وزن عروضی شعر فارسی را بیان می‌نماییم. سپس الگوهای ۳۱گانه وزن عروضی با شیوهای نمایش دانش سیستم‌های خبره تبیین می‌گردد. با استفاده از فنون متن‌کاوی الگوریتمی برای کشف اتوماتیک وزن عروضی هر مصراع و بیت ورودی عرضه می‌نماییم. برای سنجش توانایی برنامه, یک مجموعه هزار بیتی از اشعار حافظ و مولوی جمع‌آوری و آماده‌سازی شده است. در این پژوهش, پس از پنج مرحله پردازش ابتدایی مانند حرکت‌گذاری, تبدیل به رشته صامت-مصوت، اعمال اختیارات و ضرورت‌های شاعری و .... بر روی بیت شعر ورودی، بیت ورودی تبدیل به رشته‌ای از «U» و «-» خواهد شد که «U» در این رشته نشان‌دهنده هجای کوتاه و «-» نشان‌دهنده هجای بلند می‌باشند. سپس این رشته ورودی با استفاده از الگوریتم-های یافتن شباهت دو رشته مانند الگوریتم Levenshtein Distance، شعر ورودی به شبیه‌ترین ۳۱ وزن مشهور فارسی نسبت داده می‌شود. نتایج اولیه به دست آمده صحت بیش از ۶۵٪ را نشان می‌دهد. نتایج این تحقیق میان رشته‌ای به صورت یک پایگاه تحت وب برای دانشجویان، استادان و علاقه‌مندان شعر و ادب فارسی به اشتراک گذاشته می‌شود. نتایج این پژوهش به عنوان یکی از کاربردهای جدید متن‌کاوی در عرصه آموزش شعر و ادب فارسی تحول مثبتی را در پی خواهد داشت.

کلمات کلیدی

وزن عروضی اشعار فارسی، متن‌کاوی، سیستم‌های مبتنی بر قانون، Levenshtein Distance

Persian Poem Rhythm Recognition: A New Application of Text Mining

Mohammad Mahdi Mojiry, Behrouz Minaei-Bidgoli

Abstract

The poem rhythm recognition is one of the most difficult parts of Persian literature. Since there is no previous works in this realm, it is an interesting arena for research. In this article, we first discover the methods to find the rules of detecting the rhythm and to build the database based upon them. Then the 31 patterns of rhythms are described through showing methods of expert systems knowledge. Using the text mining methods, we suggest an algorithm to discover poem rhythm automatically. To assay the ability of the program, a dataset of one thousand verses of Hafez and Molavi poems are collected. After five levels of basic processes like insert tokens, changing to consonant-vowel string, use the authority of poet, etc. on an input verse, it is changed to a 'U' and '-' string which 'U' is the sign of short epigram and '-' is the sign of long epigram. Using classification algorithms like Levenshtein Distance, the input verse is assigned to the

[۱] کارشناس نرم‌افزار، گروه مهندسی کامپیوتر، دانشکده مهندسی، دانشگاه کاشان،

[۲] استادیار دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، b_minaei@iust.ac.ir

most similar rhythm of the 31 rhythms. The results show 65% of accuracy in detection. Results of this research to be shared as a website for teachers, and students, and researchers in Persian literature.

Keywords
Levenshtein Distance, Text Mining, Persian Poem, Automatic Rhythm detection

۱- مقدمه

ادبیات فارسی، زمینی بکر است که به جهات مختلف، کسی جز در موارد نادر حاضر به ورود به این عرصه و زدن حرف نو نیست. برخی بر این باورند که آنچه گذشتگان کرده‌اند فصل‌الخطاب کاری است که می‌توان کرد و برخی از آن می‌ترسند که مورد طعن دیگران قرار گیرند. به عنوان مثال قبل از رایج شدن عروض علمی، عروض فنی غریب بود که کمتر کسی از اهل ادب با معیارهای آن آشنایی داشت و لذا آنان که عروض می‌دانستند در هاله‌ای از اسرار به نظر می‌رسیدند [۱].

هر چه قدر حرف نو زدن در ادبیات مشکل است، ورود کامپیوتر به این عرصه از آن مشکل‌تر. با وجود پیشرفت‌های فراوان در تحقیقات کامپیوتری، (به صورتی که محققان در برخی زبان‌ها همچون انگلیسی و اسپانیایی موفق به سرایش شعر به وسیله کامپیوتر شده‌اند [۸]، [۱۰]، [۱۱]و [۱۳]) هنوز اساتید ادبیات و ادیبان ما حاضر به قبول توانایی‌های این ابزار نیستند. و از طرف دیگر اساتید کامپیوتر نه اطلاع چندانی با مشکلات ادبیات دارند و نه حاضرند در این زمین بکر کاری نو بکنند.

یکی از مشکل‌ترین بخش‌های رشته‌ی ادبیات فارسی تشخیص وزن عروضی است، به صورتی که تشخیص وزن عروضی در بعضی اشعار برای اساتید ادبیات هم مشکل‌زا می‌شود. با توجه به این‌که تاکنون هیچ پژوهشی در این زمینه انجام نشده، ورود به این عرصه ضروری به نظر می‌رسد. البته باید توجه کرد که عمر عروض علمی که ما در این پژوهش از آن استفاده کرده‌ایم به ۵۰ سال هم نمی‌رسد.

۱-۱- وزن عروضی

وزن، نظم و تناسب خاصی است در اصوات شعر (=هجاها) این نظم و تناسب اصوات به انحای گوناگون نزد ملل مختلف مبین نوعی آهنگ و موسیقی است. در شعر سنتی هر زبانی، تساوی هجاهای هر مصراع، در وزن دخیل است. علاوه بر این عامل مشترک، وزن شعر هر زبانی مبتنی بر عامل خاصی است، به عنوان مثال وزن اشعار انگلیسی و آلمانی مبتنی است بر تکیه که بر هجاها واقع می‌شود. شعر فارسی به طور کلی مبتنی بر کمیت (کوتاه و بلندی) هجاهاست [۱].

۱-۲- دلایل تحقیق بر روی عروض

عروض برای شناختن اجزاء شعر و تشخیص وزن صحیح از وزن شکسته و ناصحیح است و یادگیری آن هرگز برای به دست آوردن توانایی لازم برای سرودن اشعار نیست [۳]. وجود وزن برای شعر لازم است و آشنایی شاعران و محققان ادبی با علم عروض واجبتر. امروزه عروض جزو یکی از نظام‌های زبان شناسی محسوب می‌شود و در مطالعات مربوط به فنولوژی(واج شناسی) و سبک‌شناسی و نظام‌های دیگر ادبی کاربرد وسیع یافته است [۱]. آشنایی با عروض و به دست آوردن وزن عروضی یکی از روش‌های تحقیق بر روی اشعار شعرا، مقتضیات زمان شاعر، روحیات شاعر و ... می‌باشد.

۱-۳-سیستم طراحی شده چه کاری نمی‌تواند انجام دهد؟

سیستم طراحی شده تنها قادر به تشخیص ۳۱ وزن معروفی است که از [۱] استخراج شده‌اند. البته باید توجه داشت که [۱۲] تعداد اوزان پرکاربرد را ۲۹ عدد دانسته و [۶] آنها را ۳۳ تا دانسته که البته خود اعتراف می‌کند که این اوزان اضافی بسیار کم کاربردتر از دیگران هستند. تحقیقات نشان می‌دهد که ۹۹ درصد اشعار در همین ۲۹ وزن سروده شده‌اند [۴]. و البته به اعتقاد [۱] قریب ۹۰ درصد و بلکه بیشتر اشعار فارسی به ۳۱ وزن اول سروده شده‌اند. با این توضیحات مشخص می‌شود که به دست آوردن همین ۳۱ وزن از مجموع بیش از ۳۰۰ وزن شناخته شده فارسی، کاری شایسته باشد. البته برای ادامه کار می‌توان تمامی این ۳۰۰ وزن را به مجموعه اضافه کرد. اگر وزنی در مجموعه ۳۱تایی نباشد، سیستم سعی می‌کند آن را به یکی از این اوزان نسبت دهد، بنابراین وزن اشتباه تشخیص داده می‌شود.

همچنین این سیستم قادر به تشخیص درستی و یا نادرستی وزن نیست، یعنی اگر شاعری بیتی را با وزنی اشتباه و یا ناهماهنگ سروده باشد، اگر اختلاف جزیی باشد، سیستم آن را تشخیص نخواهد داد. که این مورد هم می‌تواند زمینه تحقیقات بعدی باشد. در میان اشعار فارسی، به دلیل قواعد عروضی خاص رباعی، این سیستم وزن این نوع شعر را به خوبی تشخیص نمی‌دهد. همچنین این مشکل برای وزن شعر نو نیز وجود دارد.

۲- به دست آوردن وزن عروضی یک شعر

قواعد به دست آوردن وزن عروضی یک شعر را می‌توان در چند صفحه خلاصه کرد و یا در یکی دو ساعت به صورت کامل بیان کرد. ولی این بدان معنی نیست که پس از یادگیری قواعد عروض می‌توان به راحتی وزن عروضی یک شعر را به دست آورد. این امر نیاز به تمرین و ممارست فراوان دارد و مانند مسائل ریاضی اشعاری وجود دارد که پیدا کردن وزن عروضی آنها برای دانشجویان بسیار مشکل است.

علاوه بر مسائل فوق، اختلاف نظر هم بین اساتید وجود دارد، که پس از تحقیقات فراوان ما عروض علمی را انتخاب نمودیم که قابلیت الگوریتمیک کردن آن بسیار آسان‌تر از عروض سنتی است. هم‌چنین بین اساتیدی که از این روش استفاده کرده‌اند، ما از روش دکتر سیروس شمیسا استفاده نموده‌ایم. البته در برخی از مراحل مجبور به استفاده از روش‌های دیگر شدیم، ولی اصل روش از کتاب ایشان [۱] استخراج شده است. برای به دست آوردن وزن عروضی به شیوه علمی باید مراحل زیر را به ترتیب دنبال کرد.

۲-۱- درست خواندن

باید توجه داشت برای به دست آوردن وزن عروضی، صورت ملفوظ شعر برای ما مهم است، نه صورت مکتوب. به همین دلیل است که «ز»، «ض»، «ظ»و «ذ» برای ما تفاوتی ندارد. و در مقابل «و» در سه کلمه «سود»، «تو» و «وجد» برای ما متفاوت است، زیرا به سه صورت مختلف خوانده می‌شود. در اولی به صورت یک مصوت بلند، در دومی به صورت یک مصوت کوتاه و یک صامت و در سومی به صورت یک صامت خوانده می‌شود.

۲-۲- تولید رشته صامت-مصوت

در روش شمیسا و کامیار این مرحله وجود ندارد و در دل مرحله قبل نهفته شده. در این بخش رشته صامت و مصوت (رشته CV) به صورت مخففهای V،C تولید می‌شود. به این صورت که به جای صامت از C و به جای مصوت کوتاه یک V و به جای مصوت بلند دو V قرار می‌دهیم. به این صورت در پیدا کردن هجای کوتاه و بلند در مرحله بعد مشکلی پیدا نمی‌شود.

۲-۳- تقطیع

تقطیع در لغت به معنای قطعه قطعه کردن است و در اصطلاح عروض به معنای قطعه قطعه کردن شعر به هجای کوتاه و بلند است. در این سیستم هجای کوتاه را با «U» و هجای بلند را با «-» نشان می‌دهیم. خروجی این مرحله رشته Udash می‌باشد که بدون اعمال اختیارات شاعری، وزن اصلی شعر نیست، بلکه وزن تقطیعی است.

برای به دست آوردن این رشته، باید از ابتدای رشته صامت-مصوت شروع کنیم. در صورتی که به قالبCVV یا CVC رسیدیم به جای آن «-» قرار دهیم، در صورتی که به CV یا C تنها رسیدیم به جای آن «U» قرار دهیم. باید توجه داشت که در فارسی هیچ هجایی با مصوت (V) آغاز نمی‌شود، بنابراین نباید به گونه‌ای این تقسیم بندی را انجام داد که به V برسیم.

۲-۴-رکن‌بندی

همانطور که در صرف عربی، قالب‌هایی برای کلمات اختراع کرده‌اند و مثلا می‌گویند شاعر بر وزن فاعل، در عروض نیز برای مجموعه چند هجای کوتاه و بلند قالب‌هایی وضع کرده‌اند و مثلا می‌گویند «می‌آیم» (---) بر وزن مفعولن. بنای این قالب بر «ف»، «ع» و «ل» است. البته می‌توان به جای آن هر معادل دیگری به کار برد. مانند «ت» و «تن» که در این صورت «مفاعیلن» می‌شود: «ت تن تن تن». در این مرحله باید با توجه به جدول ارکان، به جای هر چند هجا، یک رکن قرار می‌دهیم. قسمتی از جدول ارکان در جدول (۱) آورده شده است. این جدول از [۱] استخراج شده است.

۲-۵- اعمال اختیارات شاعری

کمتر شعر است که پس از تقطیع شعر به هجای کوتاه و بلند و رکن‌بندی، وزن اصلی شعر را نشان دهد، بلکه وزنی به دست می‌آید که به آن وزن تقطیعی گویند. برای تبدیل وزن تقطیعی به وزن اصلی، توجه به دقایقی که به آنها اختیارات شاعری گویند لازم است.

جدول(۱) قسمتی از جدول ارکان

سه هجایی		چهار هجایی		پنج هجایی	
فَعِلُن	UU–	فاعلاتن	–U––	مُستفعلاتُن	––U––
فاعلن	–U–	فاعلاتُ	–U–U	مُتفاعِلُن	U U –U–
فعولن	U––	فعلاتن	UU––		
مفعولن	–––	فعلاتُ	UU–U		

برخی از اختیارات شاعری در زیر آمده است. البته بعضی از مواردی که در زیر می‌آید در کتب عروض تحت عنوان اختیارات شاعری نیامده، بلکه بعضی تحت عنوان «قواعد تقطیع» و برخی دیگر تحت عنوان «ضرورات و استثنا» آورده شده است.

۱- هجای کوتاه در آخر مصراع، بلند محسوب می‌شود.
۲- شاعر مختار است در آخر مصراع یک یا دو حرف صامت، اضافه بر فرمول بیاورد (یا نیاورد).
۳- شاعر مختار است به جای دو هجای کوتاه (UU) یک هجای بلند بیاورد. عکس این مورد صحیح نمی‌باشد . این اختیار که در اصطلاح به آن «تسکین» می‌گویند، جز در آغاز مصراع (مگر به ندرت) در همه جا قابل اعمال است.
۴- هرگاه بعد از نونی که بعد از مصوت بلند قرار گرفته است، سکون یا مکث باشد، از کمیت مصوت بلند کاسته می‌شود.
۵- شاعر مختار است در برخی اوزان به جای «–U»، «U–» بیاورد، و یا بالعکس عمل کند. این عمل را در اصطلاح «قلب» گویند. قلب در تبدیل مفتعلن به مفاعلن و بالعکس دیده می‌شود.
۶- شاعر مختار است به جای فعلاتن در رکن اول هر مصراع، فاعلاتن بیاورد.

۲-۶- اوزان اشعار فارسی

از اجتماع ارکان مختلف، اوزان گوناگونی پدید می‌آید. در هیچ وزنی کمتر از دو و بیشتر از چهار رکن وجود ندارد. به عبارت دیگر کوتاه‌ترین مصراع شعر سنتی فارسی دارای دو رکن و طویل‌ترین آن دارای چهار رکن است. در عروض سنتی برای هر یک از اوزان اسمی نهاده بودند، مثلا اوزان مبتنی بر «فاعلاتن» را «رمل» و اوزان مبتنی بر «مستفعلاتن» را «رجز» می‌گفتند. در عروض جدید نیازی به حفظ اسامی اوزان نیست و در مواقع لزوم می‌توان به کتب عروضی قدیم مراجعه کرد [۱].

۳- پیش‌نیازهای طرح

قبل از شروع کار، لازم بود برخی مشکلات را حل نموده و همچنین کارهایی را به عنوان پیش‌نیاز انجام داد. در ادامه برخی مشکلات و راه‌حل ارائه شده و همچنین برخی پیش‌نیازهای انجام شده شرح داده می‌شود.

۳-۱- استاندارد خط

یکی از مهمترین چالش‌های خط فارسی عدم وجود الگوی استاندارد نوشتاری برای خط فارسی (یا رعایت نکردن قانون‌های آماده شده تا کنون برای آن) است. روند فارسی‌سازی و استاندارد نمودن زبان فارسی برای رایانه فراز و نشیب‌های زیادی داشته است. در اینجا گوشه‌ای از آن آورده می‌شود [۷].

کُد ascii استاندارد ۱۲۷ نویسه دارد و چون این کُد در یک بایت (۸ بیت) گذاشته می‌شود ۱۲۸ حالت دیگر از ۲۵۶ حالت آن باقی می‌ماند. در آغاز هر نویسه‌ی فارسی به یکی از این جاهای باقی مانده نگاشته می‌شد. این روش پیش از آن برای دیگر زبان‌ها در دیگر کشورها به کار گرفته شده بود. در این روش هر برنامه‌نویس به دلخواه این نگاشت را با برنامه‌نویسی فراهم می‌کرد.

سپس شرکت‌های نرم‌افزاری بزرگ دنیا که علاقه‌مند به فروش نرم‌افزارهای خود در کشورهای دیگر بودند به کمک همین روش (۱۲۸ حالت آزاد) کدی را ساختند که همه‌گیر شد. یکی از بزرگترین دردسرهای این کد برای فارسی رعایت نکردن ترتیب چهار نویسه‌ی ویژه‌ی فارسی (پ،چ،ژ،گ) بود. زیرا که این شرکت‌ها پایه‌ی کار خود را بر زبان عربی گذاشته بودند و سپس فارسی را بدان افزوده بودند.

یونی‌کد یک کد جهانی است که چند شرکت بزرگ نرم‌افزاری دنیا به همراهی سازمان جهانی استاندارد (ISOJTC1/SC2) آن را به وجود آورده‌اند. با چشم‌پوشی از فراز و نشیبهای این روش به سادگی می‌توان گفت که در این روش به جای یک بایت ، دو بایت به کار برده می‌شود. این کدگذاری رویه‌ی پایه‌ی چند زبانی نامیده می‌شود. این کد ۶۵۵۳۵ نویسه دارد.

کد ascii سالها استاندارد رایانه‌ها بوده است بسیاری از سخت‌افزارها، شبکه‌ی جهانی و ... بر پایه‌ی آن ساخته شده‌اند. بنابراین عوض کردن این کد (یک بایتی) اگر ناممکن نباشد بسیار پرهزینه و زمان‌گیر خواهد بود. چاره‌ای که اندیشیده شد به کار بردن کدی با تعداد بایت متغیّر بود که با یونی‌کد نگاشت یک به یک داشته باشد و همچنین ۱۲۸ نویسه ascii را به همان شکل یک بایتی باقی گذارد [۷].

در این پژوهش ما از استاندارد UTF-8 استفاده کرده‌ایم، البته باید توجه داشت که نویسه‌های فارسی در این استاندارد و استاندارد یونی‌کد تفاوت چندانی ندارند. فقط کافی است در هنگام ذخیره فایل‌هایی که حاوی متن هستند، آنها را به صورت UTF-8 ذخیره کنید، هنگام ورود اطلاعات از طریق فیلدهای ورودی هم نگرانی وجود ندارد.

۳-۲- حذف نویسه‌های اضافی

یکی از پیش‌پردازش‌هایی که در کار با متون در هر زبانی انجام می‌شود، حذف نویسه‌های اضافی است. این نویسه‌ها معمولا نویسه‌هایی از قبیل «؟ ، - .» هستند که در قواعد نوشتاری باید نوشته شوند ولی برای پردازش متن نیازی به آنها نیست.

در متون فارسی علاوه بر نویسه‌های معمول در زبان‌های دیگر دو نوع نویسه خاص زیاد مورد استفاده قرار می‌گیرند که باید به آنها توجه شود. این دو نویسه، یکی خط تیره (EmDash) است و دیگری فاصله مجازی (ZERO-WIDTH NON-JOINER) است. نویسه خط تیره معمولا برای تراز کردن انتهای خطوط مورد استفاده قرار می‌گیرد. در کلمه «مهم» اگر چندبار از این نویسه خاص استفاده شود به صورت «مـــــهـــــم» درخواهد آمد. در این پژوهش این نویسه به همراه بقیه نویسه‌های اضافی حذف خواهد شد.

نویسه فاصله اضافی مانند نویسه‌های دیگر به راحتی قابل حذف نیست. زیرا اگر آن را به طور کلی حذف کنیم، ممکن است خواندن کلمه کمی فرق کند، اگر به جای آن فاصله قرار دهیم، کلمه تبدیل به دو کلمه مجزا خواهد شد. بنابراین باید بسته به پردازش مورد نظر با این نویسه برخورد کرد.

۳-۳- ساخت مجموعه داده

یکی از مشکلات موجود بر سر تحقیقات فارسی نبود یک مجموعه داده (Data Set) برای آزمایش برنامه‌های تولیدی می‌باشد. در صورتی که انواع مجموعه داده‌ها به زبان انگلیسی بر روی فضای سایبر وجود دارد.

در مورد این پروژه خاص نیز این مشکل وجود داشت. به همین دلیل بر آن شدیم که یک مجموعه داده مناسب برای این پروژه طراحی بنماییم. برای این کار از اشعار دیوان غزلیات خواجه حافظ شیرازی [۲] و دیوان شمس مولانا جلال الدین [۵] استفاده کردیم. البته متن تایپ شده این اشعار بر روی اینترنت موجود بود، که با وجود غلط‌های تایپی و همچنین عدم رعایت تصحیح‌های معتبر مجبور به استفاده از همین نسخ شدیم.

برای یافتن اوزان اشعار در مورد دیوان حافظ، در [۲] وزن هر یک از غزل‌ها نوشته شده بود، که با کدگذاری و تبدیل به فرمت دلخواه از ان استفاده کردیم. و در مورد دیوان شمس از خانم ده‌نمکی یکی از اساتید ادبیات کمک گرفتیم، که جا دارد در این جا از زحمات ایشان قدردانی نماییم. ایشان وزن عروضی بیش از ۴۵۰ غزل از این مجموعه را استخراج نمودند.

با حذف موارد اختلافی و استفاده از یک انتخاب تصادفی بیش از ۱۰۰۰ بیت از اشعار حافظ و مولانا همراه با کد وزن آنها به صورت قالب استاندارد تعریفی درآوردیم. یکی از مواردی که در ساخت این مجموعه داده مورد توجه قرار گرفته است تعداد هر وزن است. تعداد هر وزن بسته به کاربرد هر وزن تعیین شده است، مثلا برای وزن «مفاعلن فعلاتن مفاعلن فعلن» که پرکاربردترین وزن فارسی است بیشترین تعداد بیت در نظر گرفته شده است. برای بالا بردن دقت اوزان انتخابی تمامی ابیات انتخابی دوبار بررسی گردیده است.

۳-۴- انتخاب زبان برنامه نویسی

برای پیاده سازی این پژوهش زبان‌های مختلفی مورد بررسی قرار گرفت، به دلیل آنکه قصد داشتیم این پژوهش را برای علاقه‌مندان در اینترنت به اشتراک بگذاریم، به دنبال برنامه‌ای تحت وب بودیم که به راحتی بتواند پردازش متن را انجام دهد. زبان برنامه نویسی Python و PHP دو گزینه خوب برای این‌کار بودند، ولی با توجه به اینکه زبان PHP توسط اکثر سرورها پشتیبانی شده و همچنین نیاز به هیچ واسط دیگری ندارد، این زبان برای کار انتخاب شد.

۴- الگوریتم پیشنهادی

برای حل این مساله خاص (یافتن وزن عروضی اشعار فارسی)، می‌توان از روش‌های مختلفی استفاده کرد، بر اساس بررسی‌های انجام شده، ما از عروض علمی و همچنین شیوه دکتر شمیسا با تغییراتی استفاده کرده‌ایم. در این روش ابداعی که نخستین الگوریتم پیشنهادی برای این کار است، مساله یافتن وزن به شش مرحله تقسیم می‌شود که جلوتر توضیح خواهیم داد.

۴-۱- تعریف مساله

برای تقریب به ذهن و همچنین خارج کردن مساله از حالت تخصصی ادبیات مساله تشخیص وزن عروضی اشعار فارسی را به صورت زیر تعریف می‌نماییم: « پس از تغییرهایی بر روی یک رشته ورودی و تبدیل آن رشته خروجی، سعی می‌شود رشته خروجی را در یکی از دسته‌های موجود جای داد.»

رشته ورودی در این‌جا، یک بیت شعر فارسی است.

رشته خروجی، رشته‌ای از «U» و «-» که در آن «U» به معنای هجای کوتاه و «-» به معنای هجای بلند است.

تغییرات، به معنای تغییر حروف به رشته صامت مصوت و سپس تبدیل این رشته به «U» و «-» و همچنین اعمال اختیارات و ضرورت‌های شاعری است.

۴-۲- مرحله‌بندی سیستم

سیستم به شش مرحله مختلف به صورت زیر تقسیم می‌شود:

۱- حرکت‌گذاری
۲- تبدیل مصراع حرکت‌گذاری شده به رشته cv
۳- اصلاح رشته cv بر اساس اختیارات شاعری
۴- تبدیل رشته cv اصلاح شده به رشته udash
۵- اصلاح رشته udash بر اساس اختیارات شاعری
۶- انطباق یکی از اوزان معروف با رشته udash

مراحل یک تا پنج همان روند تبدیل رشته ورودی به خروجی می‌باشد، و مرحله ششم تشخیص الگو.

۴-۲-۱- حرکت‌گذاری

توجه شود که کلمات ما تک تک حرکت‌گذاری خواهند شد. بنابراین قادر به تشخیص حرکت آخر کلمه نخواهیم بود و این امر دقت برنامه را کاهش داده است. دو حالت برای حرکت‌گذاری داریم:

۴-۲-۱-۱- با استفاده از مجموعه لغات

در این برنامه ما از یک مجموعه لغات (Lexicon) فارسی استفاده کرده‌ایم. مجموعه استفاده شده از یک برنامه متن‌باز به نام «واژگان زایای زبان فارسی» برداشته شده و پس از اعمال تغییراتی با فرمت جدید مورد استفاده قرار گرفته‌اند. این لغات شامل نوعی تلفظ به صورت فونتیک می‌باشند، که کار ما را در تبدیلات آتی بسیار آسان می‌نماید. . البته در پایگاه داده این برنامه خصوصیات دیگری از کلمات هم نوشته شده، که ما فقط تلفظ آنها را مورد استفاده قرار دادیم. در این پایگاه داده بیش از ۵۰۰۰۰ کلمه فارسی جمع‌آوری شده است. بعضی از این کلمات در اشعار ما

کاربردی نداشته (مانند آث‌میلان) و برخی از کلمات استفاده شده در اشعار در این پایگاه داده وجود ندارد (مانند حالات مختلف افعال). استفاده از این پایگاه داده برای حرکت‌گذاری کلمات نسبت به حالت بعدی، دقت بسیار بالاتری دارد.

۴-۲-۱-۲- بدون استفاده از مجموعه لغات

در صورتی که کلمه‌ای در پایگاه داده بود، تلفظ آن به صورت دقیق به دست می‌آید، در غیر این صورت با استفاده از قوانین حرکت‌گذاری این کار انجام می‌شود. برخی قوانین حرکت‌گذاری اعمال شده به صورت زیر است:

۱- اگر حرف «الف»، «ی» و یا «و» باشد، روی آن هیچ حرکتی قرار نمی‌دهد، مانند «ابر».
۲- اگر حرف الف نباشد ولی حرف بعدی آن الف باشد، باز هم حرکتی روی حرف قرار نمی‌گیرد، مانند حرف «م» در کلمه «مار».
۳- اگر حرف، آخرین حرف کلمه باشد باز هم حرکتی روی آن قرار نمی‌گیرد.
۴- در غیر از سه مورد بالا، حرکت «ـَ» روی حرف قرار داده می‌شود.

توجه شود که نوع حرکت اهمیتی ندارد. زیرا مثلا دو کلمه «حَسَن» و «حِسِن» هر دو از دو هجای کوتاه و بلند تشکیل شده‌اند. جالب است که با این روش (که تخمین بسیار ضعیفی می‌باشد) و بدون استفاده از مجموعه لغات، تا حدود ۳۰درصد، برنامه وزن‌های ورودی را صحیح تشخیص داد.

۴-۲-۲- تبدیل مصراع حرکت‌گذاری شده به رشته صامت‌مصوت

اگر در مرحله قبل کلمه‌ای در مجموعه لغات یافت شد، تبدیل آن به CV کار بسیار راحتی می‌باشد، این کار به وسیله جدول (۲) و چند قانون ساده به صورت تقریبا دقیق انجام خواهد شد. و در صورتی که کلمه ورودی به وسیله قوانین حرکت‌گذاری، حرکت‌گذاری شده، باید در این مرحله طبق قوانین زیر کلمه حرکت‌گذاری شده تبدیل به رشته CV شود.

۱- به جای هر یک از مصوت‌های کوتاه (ـِـُـَ) یک V قرار می‌دهد.
۲- به جای هر یک از مصوت‌های بلند (ا ی و) VV قرار می‌دهد.
۳- به جای صامت ها، C قرار می‌دهد.

جدول (۲) تبدیل تلفظ‌های موجود در مجموعه لغات به CV

تلفظ حرف	حرف	نشانه	نوع	مثال	تلفظ مثال
A	ا	vv	مصوت بلند	ناسزا	nAsezA
I	ی	vv	مصوت بلند	حصین	hasin
U	او	vv	مصوت بلند	سوز	suz
A	ـَ	v	مصوت کوتاه	بهجت	bahjat
E	ـِ	v	مصوت کوتاه	زرنگ	zerang
O	ـُ	v	مصوت کوتاه	مناظره	monAzere
'	ء	c	صامت	آیه	'Aye
Other	بقیه حروف	c	صامت	بهشت	beheSt

۴-۲-۳- اصلاح رشته CV بر اساس اختیارات شاعری

با توجه به برخی اختیارات شاعری تغییراتی روی رشته CV انجام می‌شود. از جمله اختیارات شاعری که در این مرحله اعمال می‌شود می‌توان به مورد زیر اشاره کرد:

شاعر مختار است در آخر مصراع یک یا دو حرف صامت، اضافه بر فرمول بیاورد(یا نیاورد). بنابراین اگر در انتهای رشته چند C تنها وجود داشت، آنها را حذف می‌نماییم.

۴-۲-۴- تبدیل رشته CV اصلاح شده به رشته udash

هر هجا باید با صامت (C) شروع شود. بنابراین، باید از C شروع کرده و جلو برویم تا هجای کوتاه و بلند مشخص شود:

۱- هجای کوتاه: این هجا را با U نشان می‌دهیم، به دو صورت دیده می‌شود: C و یا CV

۲- هجای بلند: این هجا را با – نشان می‌دهیم. به دو صورت دیده می‌شود: CVV ویا CVC

۴-۲-۵- اصلاح رشته udash بر اساس اختیارات شاعری

در این مرحله نیز بر اساس برخی اختیارات شاعری تغییراتی بر روی رشته Udash انجام خواهد شد. از آن جمله می‌توان به دو مورد زیر اشاره کرد:

۱- هجای کوتاه در آخر مصراع بلند حساب می‌شود. بنابراین هجای آخر مصراع تبدیل به هجای بلند می‌شود. به عبارت دیگر، آخرین کاراکتر رشته به «-» تبدیل می‌شود .

۲- شاعر مختار است به جای فعلاتن در رکن اول هر مصراع، فاعلاتن بیاورد. با بررسی‌های صورت گرفته بر وری اوزان موجود (الگوها) هر کجا در ابتدای مصراع فعلاتن (--UU) آمده پس از آن یا دوباره فعلاتن آمده و یا مفاعلن(U-U-) بنابراین با یک تخمین خوب دو تبدیل زیر را در صورتی که ابتدای رشته باشد انجام می‌دهیم:

فاعلاتن فعلاتن <-- فعلاتن فعلاتن

فاعلاتن مفاعلن <-- فعلاتن مفاعلن

۴-۲-۶- انطباق یکی از اوزان معروف با رشته udash

پس از به دست آوردن رشته Udash نوبت به تطبیق آن با یکی از اوزان مشهور می‌رسد. در این مرحله رشته به دست آمده را با تمامی ۳۱ وزن موجود در پایگاه داده مقایسه کرده و نزدیکترین رشته را به عنوان وزن اعلام می‌نماییم. برای این کار می‌توان از الگوریتم‌های معروف انطباق رشته (String Matching) استفاده کرد ولی به دلیل حالت خاص مسئله و همچنین برای اعمال برخی اختیارات شاعری باید از الگوریتم‌های دیگری هم استفاده کرد.

۵- الگوریتم‌های انطباق رشته

الگوریتم‌های انطباق رشته یا جستجوی رشته یکی از مباحث جذاب در رشته کامپیوتر می‌باشد. این الگوریتم‌ها معمولا به این صورت کار می‌کنند که رشته ورودی را با یک مجموعه رشته مقایسه کرده و برازندگی (fitness) رشته ورودی با هر یک از رشته‌های مجموعه را اعلام می‌نمایند. البته این حداقل کاری است که این الگوریتم‌ها انجام می‌دهند، در نهایت بسته به نوع مساله نزدیکی به رشته دیگر اعلام می‌شود.

۵-۱- Levenshtein Distance

یکی از الگوریتم‌های مورد استفاده برای مرحله ششم، الگوریتم Levenshtein می‌باشد. Levenshtein نام الگوریتمی از دسته الگوریتم‌های انطباق رشته می‌باشد. این الگوریتم توسط شخصی به همین نام در سال ۱۹۶۵ مطرح شد، البته این نام پس از مرگ وی بر این الگوریتم گذاشته شد. با آن که این الگوریتم، الگوریتم جدیدی نیست ولی به دلیل کارایی بالای آن هنوز مورد استفاده است. این الگوریتم کمترین تغییری که نیاز است تا یک رشته تبدیل به رشته دیگر شود را بیان می‌کند [۹].

۵-۲- الگوریتم XOR

علاوه بر Levenshtein، دو الگوریتم ابداعی دیگر نیز مورد استفاده قرار گرفته است، البته این ابداع از روی دو اختیار شاعری تسکین و قلب انجام گرفته است. برای اعمال دو اختیار شاعری ابتدا الگوریتمی به نام XOR تعریف می‌نماییم.

برای این منظور XOR کردن دو نویسه را به این صورت تعریف می‌نماییم: «اگر دو نویسه یکسان بودند، یک و در غیر این صورت صفر برگردان» برای آنکه XOR را بر روی دو رشته انجام دهیم، باید XOR تعریف شده در بالا را بر روی تک تک نویسه‌های دو رشته اعمال کنیم، یعنی نویسه iام رشته اول با نویسه iام رشته دوم مقایسه شود. البته این مقایسه باید تا طول کوچکترین رشته انجام شود. در هربار مقایسه

خروجی‌های مقایسه را اختلاف طول جمع می‌نماییم. در آخر مقدار به دست آمده از طول بزرگترین رشته کم کرده و بر طول بزرگترین رشته تقسیم می‌نماییم.

$$\frac{base - distance}{base}$$

در فرمول بالا base برابر طول رشته بلندتر است. distance هم از جمع اختلاف طول با نتیجه XOR هر یک از نویسه‌ها به دست می‌آید.
البته باید توجه داشت که الگوریتم XOR به صورت تنها مورد استفاده قرار نمی‌گیرد ولی از آن برای اعمال اختیارات شاعری استفاده می‌نماییم.

۵-۳- XOR برای اعمال اختیار شاعری قلب

شاعر مختار است در برخی اوزان به جای «-U»، «U-» بیاورد، و یا بالعکس عمل کند. این عمل را در اصطلاح «قلب» گویند [۱]. به دلیل آن‌که به راحتی نمی‌توان مشخص کرد که آیا شاعر از این اختیار استفاده کرده یا اگر استفاده کرده، در کجای شعر از این اختیار استفاده کرده، نمی‌توان این اختیار را مستقیما در برنامه اعمال کرد. بنابراین باید از یک تابع برازندگی با وزنی مخصوص برای این کار استفاده کرد. به این منظور از الگوریتم XOR که در بالا شرح داده شد، با تعییر ذیل استفاده می‌نماییم:

برای هر نویسه اگر پاسخ برابر یک شد، بررسی که که آیا نویسه‌ی بعدی هم (به صورت معکوس) یک می‌شود. یعنی آیا دو نویسه به صورت «-U» و «U-» در دو رشته قرار دارند و یا به طرز دیگری. اگر به این صورت باشد، خروجی برای هر دو نویسه صفر خواهد شد. محاسبه برازندگی هیچ تفاوتی با XOR نمی‌کند.

۵-۴- XOR برای اعمال اختیار شاعری تسکین

شاعر مختار است به جای دو هجای کوتاه (UU) یک هجای بلند بیاورد. عکس این مورد صحیح نمی‌باشد. این اختیار که در اصطلاح به آن «تسکین» می‌گویند، جز در آغاز مصراع (مگر به ندرت) در همه جا قابل اعمال است [۱]. در این اختیار شاعری هم مانند قبل به دلیل نامعلوم بودن وقوع و مکان وقوع، مجبور به استقاده از یک تابع برازندگی هستیم. در اینجا نیز باز از تابع XOR با تغییراتی استفاده می‌نماییم. در صورتی که خروجی برای یک نویسه یک شد و نویسه رشته اول برابر «U» بود، بررسی می‌کنیم که آیا نویسه بعدی در رشته اول هم برابر «U» است. اگر این گونه بود خروجی را برابر با صفر می‌نماییم.

۶- ترکیب **الگوریتم‌های انطباق رشته**

همانطور که در بالا شرح داده شد، الگوریتم‌های توضیح داده شده هر یک برای کاری مناسب هستند، ولی به‌طور قطع نمی‌توان یکی را بر دیگری ترجیح داد. بنابراین باید به گونه‌ای برازندگی‌های تولید شده از هر یک از این الگوریتم‌ها را با یکدیگر ترکیب کرد.

۶-۱- اولویت

برای ترکیب برازندگی‌ها و به دست آوردن یک برازندگی مناسب، داشتن یک مقدار اولویت ضروری به نظر می‌رسد. البته این مساله با توجه به نوع خاص مسئله و وجود اولویت در بین اوزان به راحتی قابل استخراج است. طبق بررسی‌های انجام شده بر روی اوزان اشعار فارسی از ابتدا تا کنون، محققان هشت کد وزن پرکاربرد فارسی را به ترتیب زیر بیان نموده‌اند [۴]. برخی از کد وزن‌ها و نام وزن‌ها را در جدول (۳) می‌توانید ببینید.
1002، 1003، 1004، 1001، 1009، 1005، 1008، 1006

بنابراین ما هم به این اوزان بر همین ترتیب اولویت بالاتری می‌دهیم. به اولی (۱۰۰۲) اولویت ۱۰۰ به بعدی ۹۹ و به همین ترتیب تا ۹۳، به بقیه اوزان موجود هم اولویت ۹۰ می‌دهیم. این اولویت‌ها کمک می‌کند در مواردی که دو برازندگی نزدیک داشتیم، به یک وزن پرکاربرد نزدیک‌تر شویم.

۶-۲- میانگین وزنی

برای ترکیب دو یا چند برازندگی از میانگین وزنی استفاده خواهیم کرد. به این صورت که برای کنترل تاثیر هر یک از توابع برازندگی به هر یک وزنی داده و با زیاد کردن آن، تاثیر آن تابع را بیشتر و با کم کردن وزن آن، تاثیر تابع را کم می‌نماییم. برای این کار از معادله (۱) استفاده می‌نماییم:

$$\text{fitness}_j = \frac{\sum_{i=1}^{n} \text{fitness}_{ij} \times weight_i}{\sum_{i=1}^{n} weight_i} \times \text{priority}_j$$

معادله (۱)

این تابع میزان برازندگی (fitness) رشته Udash به الگوی jام (وزن jام) را محاسبه می‌کند. برای این کار، ابتدا به وسیله هر یک از الگوریتم‌های انطباق (اگر n الگوریتم وجود داشته باشد) برازندگی رشته Udash به الگوی jام محاسبه می‌شود (fitness_{ij}) سپس این مقدار که عددی بین صفر و یک است، در وزنی که به این الگوریتم داده شده ضرب می‌شود. این مقادیر با یکدیگر جمع شده و سپس تقسیم بر وزن‌ها می‌شوند. این اوزان در برنامه‌ای جداگانه به دست می‌آیند. در آخر مقدار محاسبه شده، در اولویت الگو (priority_j) که بر اساس میزان تکرار این الگو در اوزان فارسی مقدار گذاری شده- ضرب می‌شود. این کار برای تمامی الگوها انجام می‌شود و الگویی که بالاترین عدد را کسب کند، به عنوان برنده اعلام می‌شود.

جدول (۳) برخی از اوزان پرکاربرد به همراه کد

کد	رشته Udash	وزن	نام
۱۰۰۱	UU--UU--UU--UU-	فعلاتن فعلاتن فعلاتن فعلن	رمل مثمن مخبون محذوف
۱۰۰۲	U-U-UU--U-U-UU-	مفاعلن فعلاتن مفاعلن فعلن	مجتث مثمن محذوف
۱۰۰۳	--U-U-UU--U-U-	مفعول فاعلات مفاعیل فاعلن	مضارع مثمن اخرب مکفوف محذوف
۱۰۰۴	-U---U---U---U-	فاعلاتن فاعلاتن فاعلاتن فاعلن	رمل مثمن محذوف
۱۰۰۵	U---U---U---U—U	مفاعیلن مفاعیلن مفاعیلن مفاعیل	هزج مثمن سالم
۱۰۰۶	U---U---U--	مفاعیلن مفاعیلن فعولن	هزج مسدس محذوف
۱۰۰۷	--U-U----U-U--	مفعول فاعلاتن مفعول فاعلاتن	مضارع مثمن اخرب
۱۰۰۸	--UU--UU--UU--	مفعول مفاعیل مفاعیل فعولن	هزج مثمن اخرب مکفوف محذوف
۱۰۰۹	UU--U-U-UU-	فعلاتن مفاعلن فعلن	خفیف مخبون محذوف
۱۰۱۰	-U---U---U-	فاعلاتن فاعلاتن فاعلن	رمل مسدس محذوف

۶-۴- به دست آوردن وزن هر الگوریتم

باید توجه داشت که نمی‌توان برازندگی‌ها را با هم جمع نموده و تقسیم بر تعداد بنماییم، زیرا قدرت تفکیک بعضی از این الگوریتم‌ها بیشتر از دیگری است و همچنین بدون وجود هر یک از این الگوریتم‌ها قدرت تشخیص برنامه پایین می‌آید. بنابراین برنامه‌ای نوشته شد که با تغییر وزن هر یک از برازندگی‌ها بهترین جواب را به دست آورد. برای به دست آوردن وزن توابع آموزش (Train)، مجموعه داده (Data Set) را به دو بخش

تقسیم کردیم. ۷۰۰ بیت را برای به دست آوردن وزن و ۳۰۰ بیت دیگر برای آزمون (Test) برنامه به کار بردیم. وزن‌های به دست آمده برای سه الگوریتم انطباق در جدول (۴) آورده شده.

۷- آزمایش‌های تجربی

پس از تکمیل مراحل کار و همچنین تعیین وزن، نوبت به امتحان نهایی برنامه رسید برای این کار ۱۰۰۰ بیتی که برای آموزش و آزمون به کاربرده بودیم و همچنین تعداد دیگری بیت شعر، برای امتحان نهایی آماده کردیم. برای امتحان نهایی برنامه حدود ۱۳۰۰ بیت شعر آماده شد. از ۱۲۹۷ بیت شعر داده شده، وزن ۸۵۱ بیت صحیح تشخیص داده شده، و ۴۴۶ بیت را به غلط تشخیص داده‌ایم. یعنی برنامه ۶۵.۷ درصد توانسته است جواب بدهد.

۷-۱- نتیجه نهایی هر وزن

برای آنکه بتوان بهتر نتایج را تحلیل کرد و همچنین بتوان روشی برای بهبود نتایج پیدا کرد، هر وزن را به طور جداگانه بررسی کرده تا ببینیم در هر وزن چه درصدی توانسته‌ایم نتیجه بگیریم. مثلا در مورد کد ۱۰۰۲، از ۱۲۸ تعداد بیت در مجموعه داده تعداد ۱۱۸ بیت صحیح تشخیص داده شده، و تعداد ۱۰ بیت نیز غلط تشخیص داده شده. بنابراین ۹۲/۱۹ درصد بیت این وزن صحیح تشخیص داده شده و تعداد ۴۸ بیت از مجموع ۱۲۹۷ بیت به غلط کد ۱۰۰۲ تشخیص داده شده‌اند. البته در برخی اوزان هم نتیجه بسیار بدی گرفتیم مثلا در کد وزن ۱۰۳۰ هیچ بیتی درست تشخیص داده نشده و در کد ۱۰۲۵ فقط ۴ درصد جواب گرفته‌ایم. علت این امر جلوتر توضیح داده خواهد شد. درست است که در کل ۶۵/۶۸ درصد جواب گرفته‌ایم ولی به‌طور میانگین تنها ۵۶/۵۸ درصد هر وزن درست تشخیص داده شده، این مورد بهتر می‌تواند اطمینان به ما دهد که اگر وزنی را این سیستم تشخیص داد، چقدر می‌توان به آن اطمینان کرد.

جدول (۴) وزن‌های به دست آمده برای الگوریتم‌های انطباق

وزن به دست آمده	الگوریتم انطباق
۹	Levenshtein
۴	XOR برای اعمال اختیار شاعری قلب
۳	XOR برای اعمال اختیار شاعری تسکین

۷-۲- دلیل نتایج ضعیف

مطلبی که در وهله اول به نظر می‌رسد که علت اصلی این امر باشد، نزدیکی اوزان به یگدیگر است. به عنوان مثال به رشته Udash دو کد وزن ۱۰۰۴ و ۱۰۳۰ توجه نمایید:

-U---U---U---U- 1004
-U---U---U---U-- 1030

همانطور که می‌بینید، این دو وزن فقط در یک «-» در آخر با هم تفاوت دارند، بنابراین طبیعی است که با یکدیگر اشتباه گرفته شوند. و البته به خاطر ساختار برنامه، وزن ۱۰۰۴ که بالاتر از ۱۰۳۰ است به عنوان وزن صحیح تشخیص داده می‌شود. با بررسی‌های انجام شده مشخص شد که ۴۸ بار وزن‌های دیگر به عنوان وزن ۱۰۰۴ تشخیص داده شده‌اند و همچنین از ۱۸ بیتی که وزن آنها ۱۰۳۰ بوده، ۱۱ مورد ۱۰۰۴ تشخیص داده شده‌اند، که این به معنی تایید حدس ما می‌باشد.

۷-۳- اوزان نزدیک به هم

برای آزمون دلایل ضعف برخی نتایج، تمامی رشته‌های Udash ۳۱ وزن مشهور را به الگوریتم‌های انطباق داده و برازندگی هر یک را به دیگران به دست آوردیم. سپس نزدیکترین وزن به هر یک از اوزان را به همراه مقدار برازندگی محاسبه کرده و مشخص شد، ۴ وزن بیش از ۹۰ درصد به ۴ وزن دیگر نزدیک بودند. این ۸ وزن را در جدول (۵) می‌توانید ببینید. پس از آن هر دو وزن شبیه به هم را یک کد دادیم یعنی به جای ۳۱ وزن، ۲۷ وزن قرار دادیم و تمامی مجموعه داده را مورد آزمون قرار دادیم. نتیجه به دست آمده، صحت نزدیک به ۷۰ درصد را نشان داد. البته

باید به این نکته هم توجه کرد که این ۸ وزن فقط اوزان با شباهت بیش از ۹۰ درصد هستند ولی اوزان دیگری تا ۸۰ درصد شباهت هم موجود داشتیم. که شاید بتوان با ترکیب این اوزان نتایج بهتری به دست آورد که در قسمت پیشنهادات به این مورد اشاره خواهد شد.

جدول (۵) اوزان نزدیک به هم

کد وزن ۱	کد وزن ۲	درصد شباهت
۱۰۰۱	۱۰۲۵	۹۳.۷۵
۱۰۰۲	۱۰۱۴	۹۳.۷۵
۱۰۰۴	۱۰۳۰	۹۳.۷۵
۱۰۱۵	۱۰۲۲	۹۰.۹۱
۱۰۲۲	۱۰۲۳	۹۲.۲۸

۸- نتیجه‌گیری

در این مقاله، به تشریح علم عروض و نیاز به برنامه‌ای برای به دست آوردن وزن عروضی پرداختیم. پیش‌نیازهای طرح را بررسی کرده و برای رفع این نیازها روش‌هایی را ارائه دادیم. الگوریتمی برای مسئله پیشنهاد دادیم و مراحل شش‌گانه‌ی آن را به صورت مبسوط شرح دادیم. پس از پایان یافتن مراحل و انجام آنها، در پایان رشته‌ای از هجای کوتاه و بلند به دست آوردیم، پس از آن الگوریتم‌هایی برای انطباق رشته طراحی شده و آزمونی برای سیستم طراحی کرده و نتایج آنها را تحلیل کردیم. این سیستم تحت وب طراحی شده و به زودی برای استفاده اساتید و دانشجویان علاقه‌مند در اینترنت به اشتراک گذاشته می‌شود.

۹- پیشنهادات

همان‌طور که در مقدمه گفته شد، ادبیات زمینه‌ای بکر -و شاید مهجور- می‌باشد و تحقیقات کامپیوتری در این زمینه به ندرت انجام می‌شود. این پژوهش که اولین پژوهش کامپیوتری در زمینه عروض می‌باشد، بسیار ناقص و در ابتدای کار می‌باشد، توسعه آن مستلزم صرف زمان و انرژی بیشتری است و شاید بدون همکاری اساتید ادبیات و کامپیوتر نتواند به راه خود ادامه پیدا کند.

نتایجی که از این پژوهش در این مدت کم به دست آمد، نشان می‌دهد که می‌توان با صرف زمان بیشتری نتایج بسیار بهتری کسب کرد. برای توسعه و بهبود این پژوهش می‌توان کارهایی از قبیل کارهای ذیل انجام داد:

۱-استفاده و آزمایش دیگر توابع شباهت رشته

۲-استفاده از ساختار سلسله مراتبی برای به دست آوردن وزن‌ها

۳-استفاده از ساختاری دیگر مانند شبکه عصبی برای یافتن شباهت

۴-افزودن تمامی ۳۰۰ وزن فارسی

پیشنهاد دوم بر اساس تحلیل نتایج ضعیف به دست آمده است. به این صورت که با ترکیب اوزان نزدیک به هم، سعی شود با استفاده از الگوریتم‌های انطباق رشته وزن بیت در یک گروه قرار گیرد، پس از آن با استفاده از برخی دیگر از الگوریتم‌های انطباق رشته هر بیت را به یکی از اوزان داخل گروه نزدیک بنماییم.

۱۰- منابع

[۱] شمیسا، س، **آشنایی با عروض و قافیه**، ویراست چهارم، نشر میترا، ۱۳۸۳

[۲] خطیب رهبر، خ، **دیوان غزلیات خواجه حافظ شیرازی**، صفی علی شاه، چاپ ششم، ۱۳۶۹

[۳] ماهیار، ع، **عروض فارسی شیوه‌ای نو برای آموزش عروض و قافیه**، نشر قطره، چاپ پنجم ۱۳۷۴

[۴] وحیدیان کامیار، ت، **فنون و صنایع ادبی** (عروض)، نشر ایران، ۱۳۶۵

[۵] شفیعی کدکنی، م، **گزیده غزلیات شمس** (مولانا جلال الدین) چاپ سپهر، چاپ پنجم ۱۳۶۳

[۶] فرزاد، م، **مجموعه شعر فارسی**، مجله خرد و کوشش، شیراز ص۶۵۰، بهمن ۱۳۴۹

[۷] یوسفان، ا، **یک سیستم بازیابی اطلاعات متنی برای زبان فارسی بر پایه نمایه گذاری معانی پنهان**، پایان‌نامه کارشناسی ارشد، دانشگاه شیراز ایران، دانشکده کامپیوتر، شهریور ۱۳۸۲

[8] H. M. Manurung، G. Ritchie، and H. Thompson. *A flexible integrated architec-ture for generating poetic texts*. Informatics Research Report EDI-INF-RR-0016، University of of Edinburgh، 2000.

[9] Gonzalo Navarro. *A guided tour to approximate string matching*. ACM Computing Surveys, 33(1):31–88, 2001.

[10] Pablo Gervas. *An expert system for the composition of formal Spanish poetry*. Journal of Knowledge-Based Systems، 14(3-4):181-188، 2001.

[11] Pablo Gervas. Wasp: *Evaluation of different strategies for the automatic generation of Spanish verse*. In Proceedings of the AISB-00 Symposium on Creative& Cultural Aspects of AI، pages 93-100، 2000

[12] Elwell، S .*The Persian Metters*، Cambridge، 1976

[13] H. M. Manurung، G. Ritchie، and H. Thompson. *Towards a computational model of poetry generation*. In Proc. of the AISB-00 Symposium on Creative and Cultural Aspects of AI، 2001.CAEPIA 2001