جداسازی و شناسایی حروف تایپی فارسی با استفاده از گشتاورهای مقیاس شده و روش جستجوی ستون به ستون

رضا صفا بخش

استادیار دانشکدهٔ مهندسی کامپیوتر
دانشگاه صنعتی امیرکبیر

وحدت دست پاک

مربی دانشکده مهندسی کامپیوتر
دانشگاه صنعتی امیرکبیر

چکیده:

فرآیند شناسائی متن، از دو بخش اصلی جداسازی کلمات به حروف و شناسائی حروف جدا شده تشکیل می گردد. جداسازی کلمات به حروف مشکل ترین قدم در شناسائی متون فارسی است. زیرا در کلمات فارسی حروف به یکدیگر می چسبند و سیستم شناسائی باید بتواند این حروف را در محل صحیح از یکدیگر جدا کند. در این مقاله روش جدیدی برای شناسائی متن های تایپی یك نوع ماشین تحریر فارسی ارائه شده است، که در آن ابتدا محل تقریبی اتصال حروف با استفاده از نمودار مجموع فواصل نقاط سیاه موجود در هر ستون تا خط مبنای متن تعیین می شود. آنگاه با روش ستون به ستون نقطه دقیق اتصال در همسایگی نقطه تقریبی جستجو می گردد. استفاده از ویژگی گشتاورهای مقیاس شده و دسته بندی آماری بیز *(Bayes)* در روش بالا شناسائی صحیح بیش از ۹۶٪ با سرعت ۳/۷ حرف در ثانیه را بر روی یك سیستم ۸۰۴۸۶ با سرعت ۳۳ مگا هرتز ایجاد نموده است.

Segmentation and Recognition of Typed Farsi Text Using Scaled Moments and Column by Column Search

Reza Safabakhsh

Assist. Prof. of Computer Eng. Dept.
Amirkabir University of Technology

Vahdat Dastpak

Lecturer of Computer Eng. Dept.
Amirkabir University of Technology

Abstract:

Text recognition is generally achieved by segmenting words into characters and recognizing the segmanted characters. Since in Farsi, characters connect to form words, the sementation task is quite difficult and directly influence the performance of the system. In this paper, a new method for recognition of typed Farsi text is presened. The method determines approximate points of connection

for characters based on the sum of distances of the pixels from the baseline. Then, using a column by column search, the exact points of connection are determined. Using scaled moments as features and the Bayes classifier, the method has resulted in a correct recogntion rate of over 96% and an average speed of 3.7 characters per second on a 80486, 33 MHZ machine.

۱ ـ مقدمه

ورودی یک سیستم شناسائی متن، تصویر رقمی شده (Digitized) متن است که از طریق یک دوربین ویدئو و مدارهای واسطهٔ لازم یا دستگاه پویشگر (Scanner) به کامپیوتر وارد می‌شود. چنین سیستمی حروف را بر اساس شکل ظاهری آنها شناسائی می‌کند و متن را در قالب کُدهای از پیش تعریف شده در اختیار قرار می‌دهد.

شناسائی ماشینی متن، کاربردهای زیادی دارد که در همگی آنها توانائی جدید خواندن متون برای کامپیوتر امکانات بسیار قابل توجه و پیشرفته‌ای را فراهم می‌آورد. به عنوان مثال مسأله وارد کردن متن به کامپیوتر جهت ویرایش و چاپ به این طریق به جای وارد کردن آن از طریق صفحه کلید بار بسیار سنگین وارد کردن متون به کامپیوتر به صورت دستی را از سر راه بر می‌دارد و موجب تسریع کار می‌گردد. نمونه دیگری از کاربرد شناسائی متون در اتوماسیون سیستم پُست است که سیستم مکانیزه بتواند با خواندن اطلاعات روی پاکت‌ها نشانی مقصد را استخراج نموده و نامه‌ها را بر حسب آن دسته بندی کند. جدا کردن، شمارش، دسته بندی و خواندن خودکار فرم‌های مختلف مانند برگه‌های رأی، فرم‌های بانکی، فرم‌های ثبت نام و غیره در یک ارگان کاربردهای دیگری از شناسائی ماشینی متن را نشان می‌دهد.

یکی از اولین تحقیقات در زمینهٔ شناسائی متن‌های چاپی فارسی در سال ۱۹۸۱ توسط دکتر پرهامی [۱] انجام شده است. پس از آن در سال ۱۹۸۷ در دانشگاه جیزهٔ مصر، سیستمی برای شناسائی متن‌های چاپی عربی ابداع شده است [۲]. این سیستم، برای شناسائی حروف از روش ضرایب فوریه بهره می‌گیرد و قادر است متن‌های عربی را با دقت نزدیک به ۱۰۰٪ و سرعت تقریبی یک کلمه در ثانیه شناسائی کند. سیستم دیگری که در سال ۱۹۸۸ در مرکز علمی شرکت IBM در کویت ابداع شده است، از ثابت‌های گشتاوری استفاده می‌کند و می‌تواند متن‌های عربی را با دقت نزدیک به ۹۴٪ و سرعت ۱۰/۶ حرف در ثانیه برای یک ماشین IBM/XT شناسائی کند [۳]. سیستم دیگری نیز در سال ۱۹۸۹ به منظور شناسائی متن‌هائی که با نوع خط‌های مختلف تایپ شده‌اند، طراحی شده است [۴]. در این سیستم شناسائی حروف از طریق عبور دادن یک ماسک ۲×۲ بر روی ماتریس شکل حروف و بهره‌گیری از یک درخت تصمیم‌گیری انجام می‌شود. دقت این سیستم، در حدود ۹۰٪ و سرعت آن سه حرف در ثانیه برای یک ماشین IBM/XT گزارش شده است. در سال‌های اخیر چند سیستم شناسائی متن نیز توسط محققین ایرانی ابداع و ارائه شده است. سیستمی که در سال ۱۳۷۲ در دانشگاه تربیت مدرس ارائه گردیده از روش ضرایب فوریه و مکان‌های مشخصه استفاده می‌کند و در شناسائی حروف، دقتی بیش از ۹۹٪ دارد [۵]. سیستم دیگری که در همین سال توسط محققین دانشگاه صنعتی شریف ارائه شده است، بر اساس قواعد مورفولوژیک کار می‌کند و سرعتی در حدود چهار حرف در ثانیه و دقتی بالاتر از ۹۸٪ در شناسائی متن دارد [۱۹]. مؤلفین مقالهٔ حاضر نیز قبلاً دو روش برای شناسائی حروف جداگانهٔ تایپی فارسی ارائه داده‌اند. در روش اول برای شناسائی حروف از گشتاورها و ثابت‌های گشتاوری استفاده می‌شود و طبقه بندی حروف به روش بیز (Bayes) انجام می‌شود [۱۴ و ۱۸]. در روش دوم، شناسائی حروف از طریق موزائیک بندی ماتریس تشکیل دهندهٔ شکل حرف و طبقه بندی به روش «نزدیک ترین میانگین» انجام می‌گیرد [۱۵ و ۱۸]. دقت سیستم اول نزدیک به ۱۰۰٪ و دقت سیستم دوم در بهترین حالت ۹۸/۷۵٪ و سرعت هر دو سیستم نزدیک به ۳۰ حرف در ثانیه برای یک ماشین ۸۰۴۸۶ با سرعت ۳۳ مگاهرتز است.

در عمل قبل از شکستن کلمات، سیستم شناسائی متن بایستی بتواند آنها را در صفحه پیدا کند. برای این کار لازم است ابتدا سطرها تشخیص داده شوند و سپس کلمات تشکیل دهندهٔ هر سطر جدا گردند. بخش۲ این مقاله به بررسی روش جداسازی سطرها و روش جداسازی کلمات می‌پردازد. بخش ۳ روش جداسازی حروف را مورد بررسی قرار می‌دهد و از ترکیب الگوریتم «مجموع فواصل» و الگوریتم «جستجوی ستون به ستون» روش جدیدی را برای

جداسازی حروف تایپی فارسی ارائه می کند. بخش ۴ آزمایش هایی را که برای برآورد کارآئی سیستم انجام شده اند، ارائه می دهد و بالاخره بخش ۵ به بیان خلاصهٔ نتایج به دست آمده و ارائهٔ پیشنهاداتی در جهت بهبود این سیستم می پردازد.

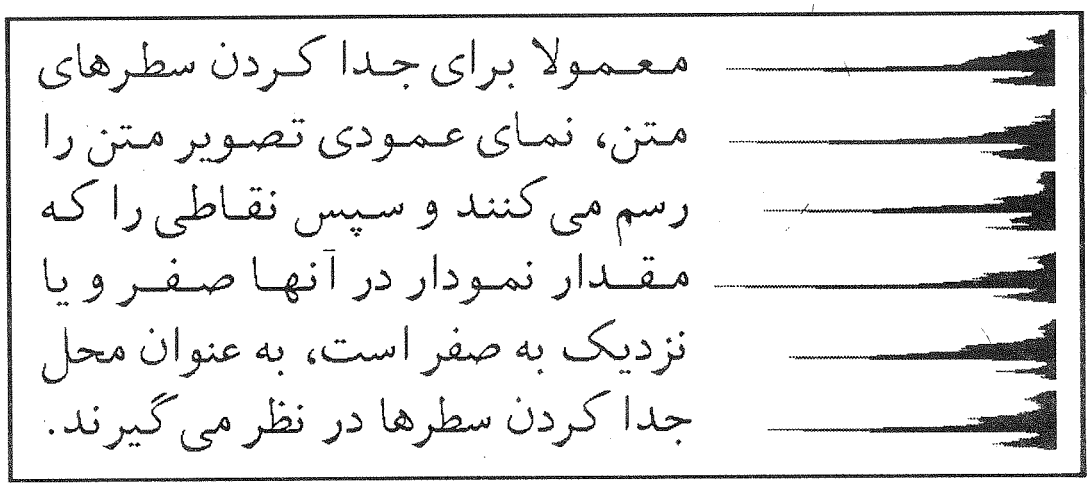
۲ ـ جداسازی سطرها و کلمه ها

متداولترین الگوریتمی که تا کنون برای جداسازی سطرهای متن ارائه شده است از نمای عمودی تصویر متن بهره می گیرد [۲، ۳، ۴، ۵، ۶، ۱۳ و ۱۸]. در این الگوریتم ابتدا نمای عمودی یک صفحه از متن رسم می گردد. نقاط مینیمم این نمودار، مشخص کنندهٔ محل جداسازی سطرها از یکدیگر هستند. در حالت هایی که تصویر متن کاملاً افقی و یا نزدیک به افقی باشد، این الگوریتم به خوبی کار می کند، اما در حالت هایی که تصویر متن چرخیده باشد، این الگوریتم قادر نخواهد بود سطرهای متن را از یکدیگر جدا کند. در این حالت ابتدا باید چرخش متن را تصحیح کرد [۱۲] و سپس به جداسازی سطرها پرداخت. شکل (۱) یک پاراگراف متن را به همراه نمای عمودی آن نشان می دهد. همانطور که در شکل دیده می شود، در محل جدائی هر دو سطر از متن، یک ناحیهٔ خالی در نمای عمودی تصویر متن وجود دارد.

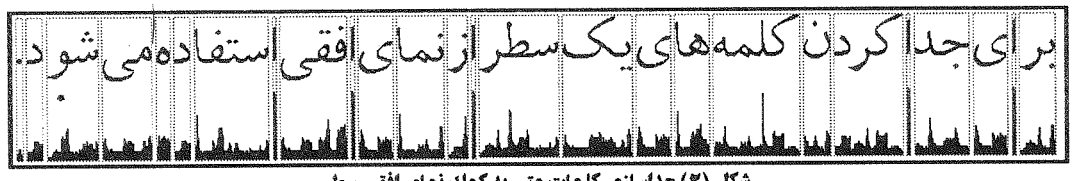
در این تحقیق نیز برای جداسازی سطرهای متن از روش فوق استفاده شده است. سیستم همزمان با دریافت هر ردیف از تصویر متن، تعداد پیکسل های سیاه آن ردیف را می شمارد و با رسیدن به اولین ردیفی که تعداد پیکسل های سیاه آن از حدی کمتر باشد، آن ردیف را به عنوان محل جداسازی سطرها تلقی می کند.

۳ ـ جداسازی کلمات

برای جداسازی کلمات هر سطر، عموماً از نمای افقی سطر استفاده شده است [۲، ۳، ۴، ۱۳ و ۱۸]. در این روش ابتدا نمای افقی یک سطر از متن رسم می گردد. نقاطی که ارتفاع نمای افقی در آنها از یک مقدار آستانه ای کمتر باشد، به عنوان محل جداسازی کلمات تلقی می شوند. شکل (۲) یک سطر از متن را به همراه نمای افقی آن نشان می دهد.



شکل (۱) نحوهٔ جداکردن سطرهای متن به کمک نمای عمودی



شکل (۲) جداسازی کلمات متن به کمک نمای افقی سطر

لازم به ذکر است که این روش الزاماً کلمات را از یکدیگر جدا نمی کند، بلکه اجزائی را جدا می کند که در امتداد عمودی، یکدیگر را نپوشانده باشند. هر یک از این اجزاء ممکن است یک کلمهٔ کامل یا بخشی از یک کلمه باشد. حتی اگر بین هر دو کلمه یک فاصله تایپ نشده باشد، ممکن است دو بخش از دو کلمهٔ متوالی نیز در امتداد عمودی همپوشانی داشته باشند و روش نمای افقی نتواند آنها را از یکدیگر جدا کند. به طور کلی می توان حروف الفبای فارسی را در دو گروه زیر دسته بندی کرد:

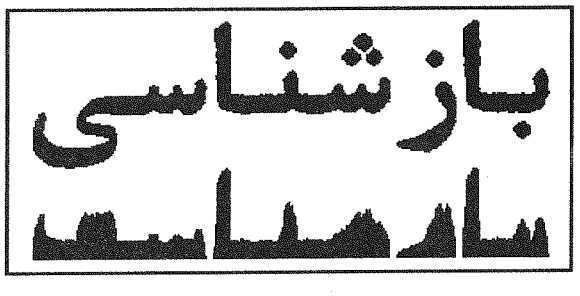
الف) حروفی که در محل خط مبنا (خط کرسی) به حرف مابعد خود می‌چسبند.

ب) حروفی که به حرف مابعد خود نمی‌چسبند.

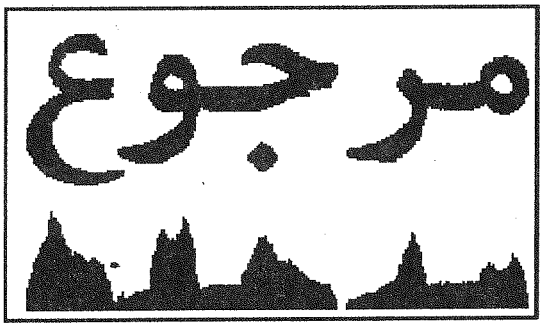
حروف ب پ ت ث ج چ ح خ س ش ص ض ط ظ ع غ ف ق ک گ ل م ن ه ی در محل خط مبنای متن به حرف مابعد خود می‌چسبند اما بقیه حروف یعنی آ ا ب پ ت ث ج چ ح خ د ذ ر ز ژ س ش ص ض ع غ ف ق ک گ ل م ن و ه ی به حرف مابعد خود نمی‌چسبند. مثلاً کلمهٔ «بازشناسی» از چهار بخش «با» ، «ز» ، «شنا» و «سی» تشکیل می گردد (شکل ۳). هر یک از این بخش ها که حاوی یک حرف یا چند حرف به هم چسبیده است، یک قطعهٔ پیوسته را تشکیل می‌دهد. معمولاً می توان با رسم نمای افقی، قطعات پیوستهٔ یک کلمه را از یکدیگر جدا کرد و مسألهٔ شناسائی کلمه را به شناسائی قطعات پیوستهٔ آن کلمه تقلیل داد.

در این روش نقاطی که در آنها مقدار نمای افقی صفر باشد، به عنوان نقاط جدائی دو قطعهٔ پیوسته تلقی می شوند. به عنوان مثال شکل (۳) نمای افقی کلمهٔ «بازشناسی» را نشان می‌دهد. همان طور که در شکل دیده می شود، ارتفاع این نمودار در سه نقطه صفر است و این نقاط، دقیقاً نقاط جدائی قطعات پیوسته هستند. اگر چه این روش در مورد کلمهٔ «بازشناسی» به درستی کار می کند، اما همیشه چنین وضعیتی حاکم نیست. در بسیاری موارد اگر چه دو حرف به یکدیگر نمی‌چسبند اما به خاطر همپوشانی مستطیل محیطی دو حرف، یک مقدار صفر در نمودار نمای افقی ایجاد نمی کنند. شکل (۴) نمای عمودی کلمهٔ «مرجوع» را نشان می‌دهد. در این کلمه سه قطعهٔ پیوسته «مر» ، «جو» و «ع» وجود دارد. اما همپوشانی حروف باعث شده که در نمای افقی، فقط یک نقطهٔ صفر وجود داشته باشد.

بنابراین اگر بخواهیم جداسازی کلمات را از طریق نمای افقی کلمات انجام دهیم، خروجی چنین الگوریتمی الزاماً قطعات پیوسته نیستند، بلکه اجزائی هستند که از یک، دو یا چند قطعهٔ پیوسته تشکیل می شوند. چنین اجزائی را که توسط نمای افقی کلمات از یکدیگر جدا می شوند، اصطلاحاً زیرکلمه می نامیم. به بیان دیگر یک زیرکلمه، بخشی از یک کلمه است که می توان آن را با رسم یک خط راست عمودی از بقیهٔ کلمه جدا کرد به نحوی که این خط راست عمودی، از روی هیچیک از پیکسل های سیاه کلمه عبور نکند.



شکل (۳) قطعات پیوستهٔ کلمهٔ «بازشناسی»



شکل (۴) قطعات پیوستهٔ کلمهٔ «مرجوع»

۴ ـ جداسازی حروف

برای شناسائی حروف تشکیل دهنده زیر کلمات، باید ابتدا آنها را از یکدیگر جدا کنیم. الگوریتم جداسازی حروف الفبای فارسی دو وظیفه بر عهده دارد:

الف) جداسازی حروفی که به حرف مابعد خود نمی‌چسبند اما با آن همپوشانی دارند.

ب) جداسازی حروفی که به حرف مابعد خود می‌چسبند.

در برخی از سیستم‌های شناسائی متن، برای جداسازی حروفی که به حرف مابعد خودنمی‌چسبند اما با آن همپوشانی دارند، از روش برچسب گذاری اجزاء (Component Labeling) استفاده شده است [۵]. در این روش همهٔ پیکسل‌های سیاهی که به یکدیگر پیوسته هستند، با برچسب واحدی مشخص می‌شوند. اشکال مهم این روش آن است که بر اثر اجرای الگوریتم برچسب گذاری اجزاء، نقطه‌های هر حرف نیز از آن جدا می‌شوند و این امر ممکن است چندان مطلوب نباشد.

برای جداسازی حروفی که در محل خط مبنا به حرف مابعد خود می‌چسبند نیز الگوریتم‌هائی ابداع شده است [۲، ۳، ۴]. در روش «ناحیهٔ سکوت»، ابتدا با رسم یک نمای افقی، یک کلمه یا زیرکلمه جدا می‌گردد، سپس یک پنجره فرضی با عرض W از راست به چپ بر روی کلمه عبور داده می‌شود و به ازای هر یک از موقعیت‌های این پنجره، ارتفاع متوسط منحنی پیرامون کلمه محاسبه می‌گردد. در هر موقعیتی که ارتفاع متوسط منحنی مذکور از یک مقدار آستانه‌ای از پیش تعیین شده کوچکتر باشد، آن موقعیت به عنوان محل اتصال دو حرف تلقی می‌گردد. اگرچه این روش، یکی از موفق‌ترین روش‌هائی است که تاکنون برای جداسازی حروف ابداع شده است، اما صحت عملکرد آن مستلزم اطلاع دقیق از ضخامت خط مبنای متن است. زیرا برای تشخیص محل جداسازی حروف، ارتفاع پنجرهٔ فرضی با یک مقدار آستانه‌ای مقایسه می‌شود و مقدار اولیهٔ این مقدار آستانه‌ای کاملاً به ضخامت خط مبنا وابسته است. در این الگوریتم برای تصحیح عرض پنجره و مقدار آستانه‌ای پیش بینی‌هائی شده است، اما اگر ضخامت خط مبنا از ابتدا با چیزی که الگوریتم انتظار دارد متفاوت باشد، این الگوریتم نمی‌تواند کار خود را شروع کند. از طرف دیگر این روش در ضمن جداسازی حروف، موقعیت خط مبنای متن را مطلقاً در نظر نمی‌گیرد، در صورتی که اتصال حروف تنها بر روی خط مبنا پیش می‌آید.

روش دیگری به منظور جداسازی حروف ارائه گردیده [۴] که به طور کامل بر سه قاعدهٔ تجربی استوار است. این قواعد ممکن است در یک نوع خط خاص صدق کنند اما در حالت کلی عمومیت ندارند. به عنوان مثال یکی از این سه قاعده فرض می‌کند که در حروفی که مانند حرف سین بیش از یک دندانه دارند، هیچگاه فاصله بین دو دندانه از ۱/۳ پهنای حرف بیشتر نمی‌شود. اما این فرض همیشه صحت ندارد.

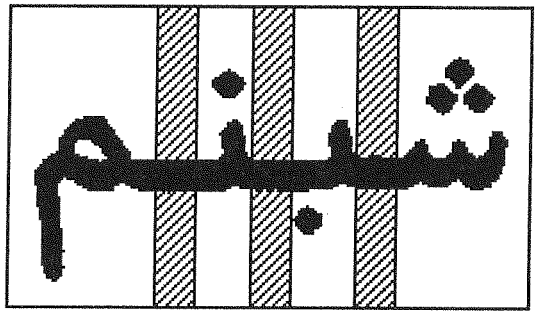
عمومی‌ترین روشی که از شکل ظاهری حروف در یک نوع خط خاص مستقل است، روش جستجوی ستون به ستون است [۲]. در این روش، جداسازی و شناسائی حروف به طور همزمان انجام می‌شود. به این ترتیب که ابتدا یک باریکه به پهنای نازکترین حرف الفبا از کلمه انتخاب و مقدار ویژگی‌ها از آن استخراج می‌گردد. سپس ویژگی‌ها به الگوریتم طبقه بندی حروف فرستاده می‌شوند تا باریکهٔ انتخاب شده شناسائی گردد. الگوریتم طبقه بندی، مقدار خطای طبقه بندی را بازمی‌گرداند و این مقدار، با یک مقدار آستانه‌ای مقایسه می‌گردد. اگر مقدار خطای طبقه بندی از مقدار آستانه‌ای کمتر باشد، حرف پذیرفته و حرف بعدی به همین روش شناسائی می‌گردد. اما اگر مقدار خطای طبقه بندی از مقدار آستانه‌ای کمتر نباشد، یک پیکسل به پهنای باریکه اضافه و همین فرآیند مجدداً تکرار می‌گردد.

ضعف مهم این روش آن است که پهنای اولیهٔ باریکه با نازکترین حرف الفبا مساوی است. در این حالت سرعت شناسائی به شدت کاهش می‌یابد، زیرا برای شناسائی هر حرف باید ده‌ها بار ویژگی‌ها استخراج شوند و الگوریتم طبقه بندی اجرا گردد. از طرف دیگر اگر قسمتی از یک حرف به حرف دیگری شباهت داشته باشد، سیستم به سادگی به اشتباه می‌افتد و یک اشتباه سیستم باعث می‌گردد که در شناسائی باقیماندهٔ حروف کلمه نیز اشتباه پیش بیاید. معمولاً برای اجتناب از این اشتباه، تعداد ویژگی‌ها را زیادتر انتخاب می‌کنند. اما افزایش تعداد ویژگی‌ها باعث کندی الگوریتم طبقه بندی می‌گردد. اگر در این روش مکانیزمی ابداع کنیم که محل تقریبی اتصال حروف را تخمین بزند و سپس به روش شناسائی ستون به ستون محل دقیق اتصال را تعیین کنیم، هر دو ایراد یاد شده برطرف خواهند شد.

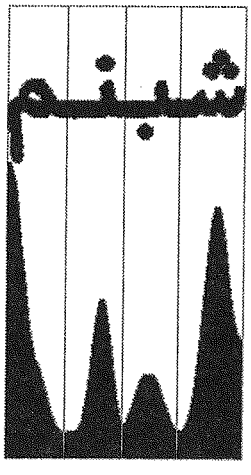
۴-۱ ـ تخمین موقعیت اتصال حروف

در محل اتصال حروف در کلمات فارسی محدوده‌ای وجود دارد که پیکسل‌های سیاه تنها در نزدیکی خط مبنای

متن قرار دارند (ناحیه سکوت). چنین ناحیه ای تقریباً در همهٔ مواردی که دو حرف در محل خط مبنا به یکدیگر می چسبند در شکل حرف ظاهر می گردد و در اکثر موارد کاندید خوبی برای محل جداسازی حروف است. برای مشخص کردن این نواحی اتصال، باید موقعیت خط مبنای متن مشخص گردد. با فرض مشخص بودن موقعیت خط مبنای متن، برای تعیین محل اتصال حروف یک کلمه، در هر یک از ستون های ماتریس تشکیل دهندهٔ یک زیرکلمه، مجموع فواصل کلیهٔ پیکسل های سیاه را تا خط مبنا محاسبه می کنیم. در این حال هر قدر تعداد پیکسل های سیاهی که از خط مبنا دور باشند، بیشتر باشد و هر قدر این پیکسل ها از خط مبنا دورتر باشند، مقدار این مجموع بیشتر و بیشتر خواهد شد [۱۶]، لذا مینیمم های نمودار مذکور نمایانگر محل تقریبی اتصالات می باشند. شکل (۵) محل اتصال حروف و شکل (۶) نمودار مجموع فواصل را برای کلمهٔ «شبنم» نشان می دهد. در نمودار این کلمه سه نقطه مینیمم وجود دارد و این نقاط، محل اتصال حروف کلمه را با اختلاف ناچیزی مشخص می کنند.



شکل (۵)
ناحیه های سکوت در محل اتصال حروف



شکل (۶)
نمودار «مجموع فواصل» کلمهٔ شبنم

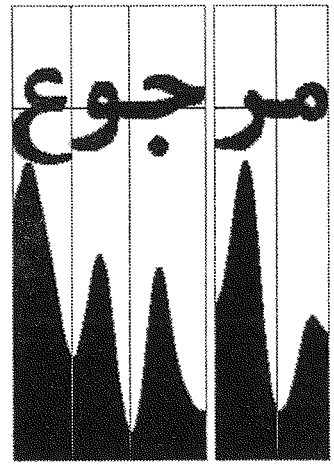
روش فوق برای جداسازی حروفی که در محل خط مبنا به یکدیگر نمی چسبند اما مستطیل محیطی آنها همپوشانی دارد، نیز می تواند به طور مطلوب عمل کند به شرط آنکه تعداد پیکسل هایی که در منطقهٔ همپوشانی وجود دارند بسیار زیاد نباشد. مثلاً در مورد کلمهٔ مرجوع دیدیم که نمای افقی قادر به جداسازی صحیح حروف نیست. شکل (۷) نمودار مجموع فواصل مینیمم های لازم برای جداسازی صحیح کلمه را نشان می دهد. به طور کلی، دو عامل باعث ایجاد یک نقطهٔ مینیمم در نمودار مجموع فواصل می شوند:

الف) در چند ستون متوالی از شکل حرف، پیکسل های سیاه تنها در نزدیکی خط مبنای متن ظاهر شوند.
ب) در نزدیکی خط مبنای متن هیچ پیکسل سیاهی ظاهر نشود اما تعداد کمی پیکسل سیاه، دور از خط مبنای متن قرار بگیرند.

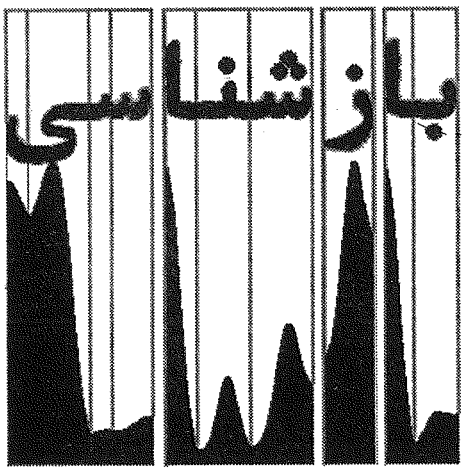
حالت اول برای حروفی پیش می آید که در محل خط مبنای متن به یکدیگر چسبیده باشند و حالت دوم برای حروفی که در محل خط مبنای متن به یکدیگر نچسبیده باشند اما خارج از خط مبنا همپوشانی داشته باشند.

اگر چه روش مجموع فواصل در اکثر موارد تقریب درستی از محل اتصال حروف به دست می دهد، اما در مواردی نیز مرتکب اشتباه می شود. به عنوان مثال در مواردی که بخشی از حروف دارای تعداد کمی پیکسل سیاه در نزدیکی خط مبنا باشد یک مینیمم در نمودار بخش ایجاد می شود که موجب تقسیم حرف در آن محل می گردد. این مسأله در دنباله های انتهائی حروفی مانند ع، ی، س، ش، ص و ض اتفاق می افتد. همچنین موارد دیگر خطا در اثر وجود نواحی سکوت در حروفی که عمدتاً در نزدیکی خط مبنا هستند مانند س، ب، پ، ت، و ث به وقوع می پیوندد. شکل (۸) وجود دو خطای مذکور را در مورد س، و ی در کلمهٔ «بازشناسی» نشان می دهد. این مسأله در بخش بعد مورد بررسی بیشتر قرار می گیرد.

اگرچه نقاط مینیمم نمودار مجموع فواصل به راحتی توسط چشم انسان قابل تشخیص هستند، اما کامپیوتر برای تشخیص آنها باید از الگوریتم های خاصی استفاده کند. در این پروژه الگوریتم ساده ای مورد استفاده قرار گرفته که در آن نمودار مجموع فواصل از راست به چپ مرور می شود و هر بار که جهت تغییر مقدار نمودار، از صعودی به نزولی



شکل (۷) جداسازی حروف کلمهٔ «مرجوع»



شکل (۸) نمودار مجموع فواصل برای کلمهٔ «بازشناسی»

تغییر کند، نقطه مربوطه به عنوان نقطهٔ مینیمم نمودار تلقی می‌شود. اگرچه این الگوریتم برای نمودارهای هموار به خوبی کار می‌کند، اما اگر یک نمودار دارای(Smooth) دندانه های ریز باشد، این الگوریتم محل فرورفتگی هر یک از دندانه ها را به عنوان یک مینیمم در نظر می‌گیرد. برای از بین بردن این اشکال قبل از پیدا کردن نقاط مینیمم یک نمودار مجموع فواصل، باید آن را از یک صافی مناسب عبور دهیم تا کاملاً هموار گردد. به این منظور صافی های میانگین (Mean) و میانه (Median) با همسایگی ۳ تا ۱۵ نقطه آزمایش گردیدند. نتایج نشان داد که دو بار عبور نمودار مجموع فواصل از یک صافی میانگین با همسایگی ۷ نقطه بهترین نتیجه را به دست می‌دهد.

۴-۱-۱ ـ تشخیص موقعیت خط مبنا

روشی که در قسمت قبل برای جداسازی حروف بیان شد، به موقعیت خط مبنای متن وابسته است. متداولترین الگوریتمی که برای تشخیص موقعیت خط مبنای متن وجود دارد، استفاده از نمای عمودی متن است [۳]. در این الگوریتم، نقطهٔ ماکزیمم نمای عمودی هر سطر، به عنوان خط مبنا در نظر گرفته می‌شود. چون در زبان فارسی اتصال حروفی که به یکدیگر می‌چسبند، در محل خط مبنای متن صورت می‌گیرد، معمولاً تعداد پیکسل های سیاهی که بر روی خط مبنا و ردیف های افقی مجاور آن قرار دارند، از تعداد پیکسل های سیاه سایر ردیف های افقی هر سطر از متن بیشتر است. اگرچه این فرض برای تک تک کلمات فارسی صحت ندارد اما اگر یک سطر از متن حاوی چندین کلمه باشد، می‌توان صحت این فرض را پذیرفت.

در این تحقیق، برای تشخیص موقعیت خط مبنا از روش اخیر استفاده شده است. اما معمولاً تصویر متنی که به یک سیستم شناسائی متن وارد می‌شود، کاملاً افقی نیست. کجی متن می‌تواند ناشی از بی دقتی تایپیست و یا اپراتوری باشد که کاغذ را در دستگاه پویشگر قرار می‌دهد. حتی در حالتی که تایپیست و اپراتور با دقت کامل کار خود را انجام دهند، تصویر متن ممکن است به میزان چند درجه از حالت افقی انحراف داشته باشد که با چشم انسان قابل تشخیص نیست. کجی متن باعث می‌شود که ماکزیمم نمای عمودی متن از خط مبنا فاصله پیدا کند. از طرف دیگر در این حالت نمی‌توان یک موقعیت واحد را به عنوان خط مبنای یک سطر از متن در نظر گرفت، زیرا چرخش متن باعث می‌شود که خط مبنای هر کلمه، نسبت به کلمهٔ بعد کمی اختلاف داشته باشد و وقتی موقعیت خط مبنا را در ابتدا و انتهای سطر مقایسه می‌کنیم، این اختلاف چشمگیر خواهد بود. این نکتهٔ مهمی است که در غالب سیستم های شناسائی متن به آن توجه نشده ولی در عمل همهٔ سیستم های شناسائی متن با آن مواجه هستند.

در این تحقیق برای غلبه بر مشکل اخیر، از روش جدیدی استفاده شده است. در مرحلهٔ آموزش سیستم، موقعیت هر حرف نسبت به خط مبنا، به عنوان یکی از اطلاعات آموزشی در اختیار سیستم قرار می‌گیرد. در مرحلهٔ آزمایش سیستم، در ابتدای هر سطر ماکزیمم نمای عمودی سطر، به عنوان خط مبنا در نظر گرفته می‌شود. سپس نمودار «مجموع فواصل» اولین کلمهٔ سطر رسم می‌گردد و سیستم به

کمک نمودار «مجموع فواصل» محل اتصال حروف کلمه را حدس می‌زند و حروف را از یکدیگر جدا و اولین حرف کلمه را شناسائی می‌کند. وقتی اولین حرف شناسائی شد، سیستم با مراجعه به اطلاعاتی که در مرحلهٔ آموزش دریافت کرده و با فرض اینکه حرف اخیر به درستی شناسائی شده است، موقعیت واقعی خط مبنا را با موقعیتی که از روی ماکزیمم نمای عمودی سطر به دست آمده مقایسه می‌کند و در صورتی که این مقادیر با یکدیگر اختلاف داشته باشند، موقعیت خط مبنا را تصحیح می‌نماید و مجدداً نمودار «مجموع فواصل» را براساس موقعیت جدید خط مبنا برای همین کلمه محاسبه می‌کند و به شناسائی حرف بعدی می‌پردازد. به این ترتیب حتی در حالتی که تصویر متن ورودی از حالت افقی انحراف داشته باشد، سیستم به مرور که در هر سطر پیش می‌رود، موقعیت خط مبنا را تصحیح می‌کند.

این الگوریتم همیشه موقعیت خط مبنا را به کمک آخرین حرفی که شناسائی شده است، مشخص می‌کند. اگر سیستم در شناسائی یک حرف مرتکب اشتباه شود، بلافاصله خط مبنا را از موقعیت خود جابجا و مجدداً نمودار «مجموع فواصل» را رسم می‌کند. حال اگر اشتباه ناشی از شناسائی حرف، باعث شود که خط مبنا از موقعیت صحیح خود دور شود، طبیعتاً نمودار مجموع فواصل که کاملاً به موقعیت خط مبنا وابسته است، دستخوش خطا می‌گردد و این خطا باعث اشتباه در شناسائی حرف بعدی می‌شود. در واقع می‌توان الگوریتم اخیر را به یک سیستم کنترل دارای یک حلقهٔ پس‌خور (Feedback) شبیه دانست که به ناگهان به سمت ناپایداری می‌رود.

اگرچه از دید تئوری نکتهٔ اخیر نقطهٔ ضعفی برای الگوریتم به حساب می‌آید، اما اگر الگوریتم شناسائی حروف خوب طراحی شده باشد، تنها ممکن است در شناسائی حروفی که تا حد زیادی به یکدیگر شبیه هستند، مرتکب اشتباه شود. مثلاً حروف «ج» ، «چ»، «ح» و «خ» شباهت زیادی دارند و طبیعتاً هر سیستم شناسائی متن ممکن است هر یک از آنها را به جای دیگری شناسائی کند. این نوع اشتباه، باعث برهم زدن نظم سیستم نخواهد شد. زیرا موقعیت نسبی هر چهار حرف یادشده نسبت به خط مبنا یکسان است.

در حالتی که به صحت عملکرد الگوریتم شناسائی حروف اطمینان نداشته باشیم و این احتمال وجود داشته باشد که دو حرفی که موقعیت نسبی آنها با خط مبنا کاملاً متفاوت باشد به جای یکدیگر شناسائی شوند، می‌توان چارهٔ دیگری اندیشید. یک طبقه بندی کنندهٔ آماری مانند طبقه بندی کنندهٔ بیز یا نزدیکترین میانگین، به ازای هر عمل طبقه بندی، عددی به دست می‌دهد که معرف میزان خطای طبقه بندی (Risk) است. طبیعتاً هر قدر این عدد کوچکتر باشد، احتمال صحت عمل طبقه بندی بیشتر است و هر قدر مقدار آن بزرگتر باشد، احتمال صحت طبقه بندی کمتر است. برای آن که اشتباه در شناسائی یک حرف باعث بروز اختلال در شناسائی حروف بعدی نگردد، می‌توان بعد از شناسائی هر حرف، خطای شناسائی حرف را با یک مقدار آستانه ای مقایسه و تنها در صورتی موقعیت خط مبنا را تصحیح کرد که خطای شناسائی از مقدار آستانه ای انتخاب شده، کمتر باشد. در این حالت نیز هنوز احتمال اشتباه وجود دارد، اما با انتخاب یک مقدار آستانه ای مناسب، می‌توان احتمال بروز خطا را به حداقل رساند.

۴-۲ ـ همپوشانی حروف تایپی در محل اتصال

همانطور که در قسمت ۳ گفته شد، برخی از حروف فارسی در محل خط مبنای متن به حرف مابعد خود می‌چسبند و برخی دیگر به حرف مابعد نمی‌چسبند. معمولاً در ماشین های تحریر، وقتی یکی از حروفی را که به حرف مابعد خود می‌چسبد تایپ می‌کنیم، کاغذ کمی کمتر از پهنای حرف به عقب می‌رود. اگر کاغذ دقیقاً به مقدار پهنای حرف به عقب برود، حرف بعدی کاملاً به این حرف نمی‌چسبد و از زیبائی متن می‌کاهد. اما وقتی کاغذ کمتر از پهنای حرف به عقب برده شود، قسمتی از حرف بعدی، این حرف را در محل خط مبنای متن می‌پوشاند و باعث می‌شود که دو حرف کاملاً به یکدیگر بچسبند. در ماشین های تحریر قدیمی که کاغذ همیشه به یک مقدار حرکت می‌کند، حروفی را که به حرف مابعد خود می‌چسبند، پهن تر از حروف دیگر طراحی می‌کنند.

تأثیر این پدیده در یک سیستم شناسائی متن هنگامی ظاهر می‌گردد که آموزش سیستم با حروف جداگانه انجام شده باشد، اما در مرحلهٔ آزمایش بخواهیم متن های واقعی را شناسائی کنیم. برای روشن تر شدن این پدیده در شکل (۹) کلمهٔ «شبنم» و حروف «شـ» ، «بـ» ، «نـ» و «م» به طور جداگانه نشان داده شده اند. خطوط نقطه چین عمودی، نقاط دقیق اتصال حروف را مشخص می‌کنند. همان طور که دیده

می‌شود، حرف «ش» وقتی به تنهائی تایپ شده باشد، دارای یک دنبالهٔ مثلثی شکل است اما در کلمهٔ «شبنم» این دنبالهٔ مثلثی، توسط حرف بعدی پوشانده شده و دیده نمی‌شود.

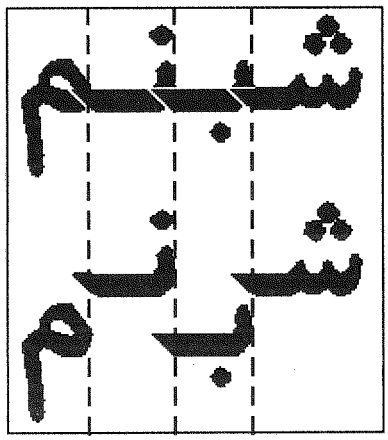
تأثیر این عامل مخصوصاً در مواردی که حساسیت روش شناسائی نسبت به شکل ظاهری حروف زیاد باشد، چشمگیر است و باعث کاهش بیش از انتظار دقت سیستم می‌گردد. برای از بین بردن این اشکال دو راه وجود دارد:

الف) سیستم با نمونه هائی آموزش داده شود که در آنها دنبالهٔ اضافی که توسط حرف بعدی پوشانده می‌شود، وجود نداشته باشد.

ب) این دنبالهٔ اضافی در زمان آموزش و توسط خود سیستم حذف شود.

این دو روش تا حد زیادی به یکدیگر شبیه هستند. در روش اول نمونه هائی که برای آموزش سیستم به کار می‌روند به شیوهٔ خاصی تهیه می‌شوند که در آنها دنبالهٔ اضافی وجود نداشته باشد. به عنوان مثال می‌توان حروف را به طور جداگانه تایپ کرد و بعد از پویش نمونه ها، دنباله های اضافی را از طریق یک برنامهٔ ترسیمی حذف کرد و یا ترتیبی فراهم کرد که سیستم با کلمات کامل آموزش داده شود و در ضمن آموزش، حروف کلمات را با کمک یک اُپراتور از یکدیگر جدا کند. در این حالت وقتی سیستم یک زیرکلمه را جدا می‌کند، محل دقیق اتصال حروف و طبقه ای را که هر حرف در آن قرار می‌گیرد، از اُپراتور می‌پرسد و در خاتمهٔ آموزش، اطلاعات آماری مورد نیاز برای طبقه بندی کننده را ایجاد می‌کند. این روش کاملاً ایده آل است اما هر بار آموزش سیستم ساعت‌ها و یا روزها وقت خواهد گرفت.

در روش دوم، آموزش سیستم با نمونه هائی که به طور جداگانه تایپ شده‌اند، انجام می‌شود. اما اگر نمونه ای که برای آموزش دریافت می‌شود از حروفی باشد که در محل خط مبنا به حرف مابعد خود می‌چسبند، سیستم قبل از محاسبهٔ بردار ویژگی، دنبالهٔ اضافی حرف را حذف می‌کند. آزمایش نشان می‌دهد که وقتی نمونه ها را با دقت ۳۰۰ نقطه در اینچ پویش می‌کنیم، پهنای این دنباله، بین چهار تا شش پیکسل است. به این ترتیب با دریافت هر یک از این نمونه ها، سیستم ابتدا یک ستون به پهنای پنج پیکسل از سمت چپ شکل حرف را حذف و سپس بردار ویژگی را محاسبه می‌کند. شکلی که بدین ترتیب به دست می‌آید، تا حد زیادی به شکل حرفی که از جداسازی حروف یک کلمه حاصل می‌گردد، شباهت دارد. در این حالت در زمان آموزش سیستم باید به ازای هر طبقه، برای سیستم مشخص کرد که حروف این طبقه به حرف مابعد خود می‌چسبند یا خیر. اما این کار نیازی به حضور اُپراتور ندارد و می‌توان برای هر مجموعهٔ آموزشی، یک پروندهٔ متنی ایجاد کرد که حاوی این قلم اطلاعاتی باشد. در این تحقیق، از روش اخیر استفاده شده است.



شکل (۹) حروف تایپی در محل خط مبنای متن یکدیگر را می پوشانند

۴-۳- روش جستجوی ستون به ستون

قبلاً گفتیم که روش «مجموع فواصل» تنها محل تقریبی اتصال حروف را مشخص می‌کند و لی سیستم برای آن که بتواند حروف را به درستی شناسائی کند، به محل دقیق اتصال حروف نیاز دارد. آزمایش نشان می‌دهد که در تفکیک پذیری ۳۰۰ نقطه در اینچ، نقاطی که توسط الگوریتم «مجموع فواصل» برای جداسازی حروف کاندید می‌شوند، بین ۳- تا ۲۴+ پیکسل با محل دقیق اتصال حروف اختلاف دارند. یعنی نقطهٔ واقعی اتصال دو حرف در یک همسایگی ۳ پیکسل از راست تا ۲۴ پیکسل از چپ نقطهٔ کاندید شده قرار دارد. برای آن که دید بهتری از میزان وسعت این همسایگی به دست بیاید، در نظر بگیرید که در تفکیک پذیری ۳۰۰ نقطه، پهنای حرف «الف» تنها چهار پیکسل است. اگرچه وسعت این همسایگی زیاد است، اما آزمایش نشان می‌دهد که مقدار متوسط آن تنها ۵+ پیکسل است. یعنی به طور متوسط، محل صحیح اتصال، به فاصلهٔ

پنج پیکسل در سمت چپ نقطه ای قرار دارد که توسط الگوریتم «مجموع فواصل» پیشنهاد می گردد. به بیان دیگر برای تعیین دقیق محل اتصال دو حرف به طور متوسط باید از نقطه ای که توسط الگوریتم مجموع فواصل پیشنهاد شده است، ۵ پیکسل به سمت چپ حرکت کرد.

در این مقاله، برای تعیین محل دقیق اتصال حروف، از روش «جستجوی ستون به ستون» استفاده شده است. این روش را با مثال شکل (۱۰) توضیح می دهیم. فرض کنید که سیستم حرف «ش» را شناسائی کرده و خط عمودی نقطه چین a نقطهٔ جدائی این حرف را از حرف «ب» مشخص می کند. حال نوبت شناسائی حرف «ب» رسیده است. سیستم ابتدا موقعیتی را که توسط الگوریتم «مجموع فواصل» پیشنهاد شده (خط نقطه چین عمودی c) به عنوان نقطهٔ جدائی دو حرف بعدی می پذیرد و ویژگی ها را استخراج می کند، سپس مقدار ویژگی ها را به الگوریتم طبقه بندی ارسال می کند. الگوریتم طبقه بندی نتیجهٔ طبقه بندی و خطای طبقه بندی را بر می گرداند، سپس خطای طبقه بندی با یک مقدار آستانه ای مقایسه می شود. اگر خطای طبقه بندی از مقدار آستانه ای بزرگتر بود، محل اتصال به میزان یک پیکسل به سمت راست منحرف می گردد و مراحل بالا یعنی استخراج ویژگی ها، طبقه بندی و مقایسهٔ خطای طبقه بندی با مقدار آستانه ای مجدداً تکرار می شود تا جائی که حداقل یکی از سه شرط زیر برقرار شود:

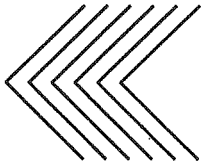
۱) *خطای طبقه بندی از مقدار آستانه ای خطا، کمتر یا با آن مساوی شود.*
۲) *به انتهای کلمه یا زیر کلمه برسیم.*
۳) *به اندازهٔ پهن ترین حرف فارسی (حرف ص) از نقطهٔ انتهائی حرف قبلی (خط نقطه چین عمودی a) دور شده باشیم (خط نقطه چین عمودی d).*

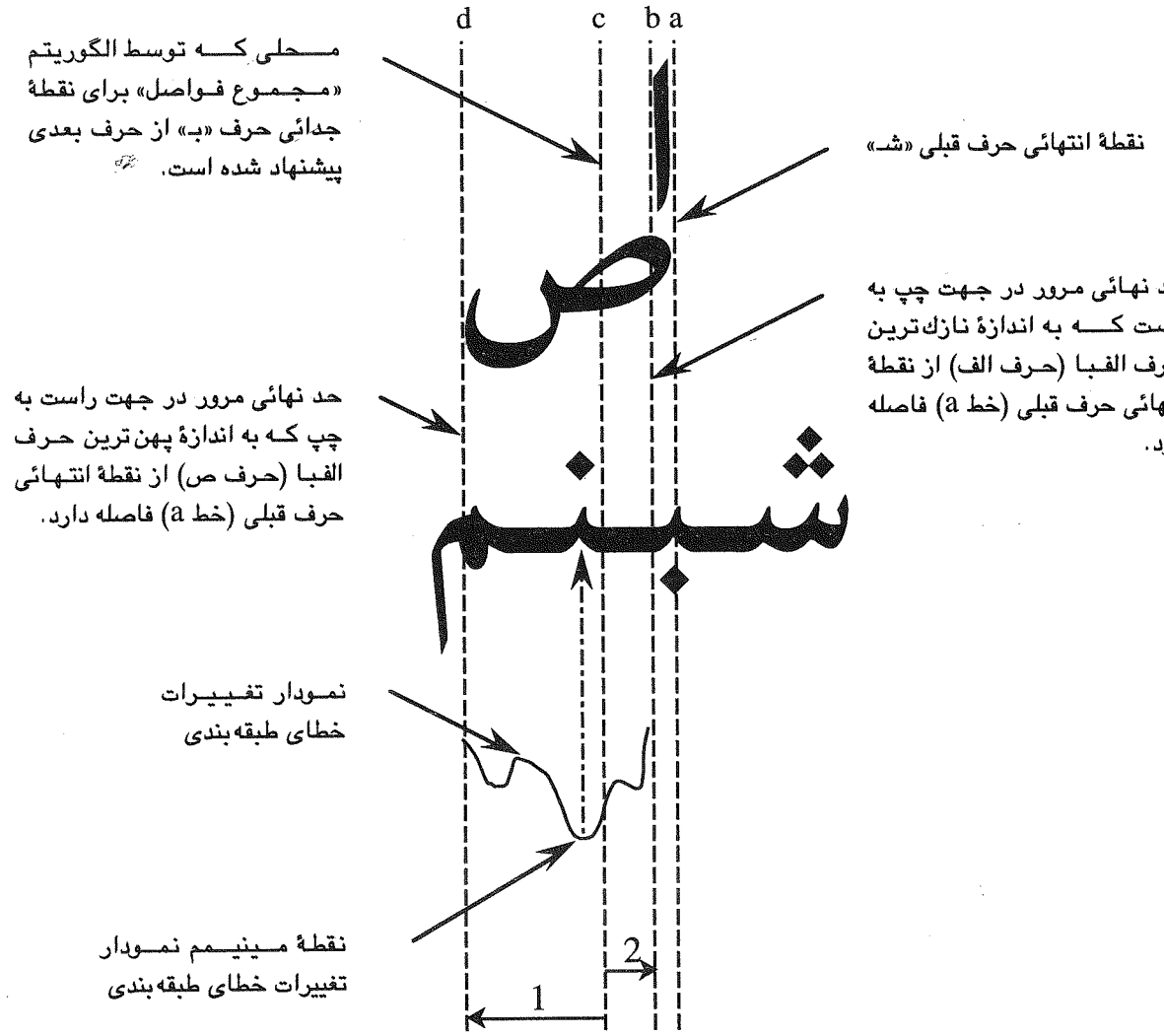
اگر شرط اول برقرار شد، مادامی که این شرط برقرار است، به مرور در جهت راست به چپ ادامه می دهیم تا به حداقل مقدار خطای طبقه بندی برسیم. اما اگر یکی از دو شرط دوم یا سوم برقرار شدند، مجدداً به موقعیتی که توسط الگوریتم «مجموع فواصل» پیشنهاد شده بود (خط نقطه چین عمودی c) باز می گردیم و این بار کلمه یا زیرکلمه را در جهت چپ به راست مرور می کنیم. مرور در این جهت را تاجایی ادامه می دهیم که حداقل یکی از دو شرط زیر برقرار شود:

۱) *خطای طبقه بندی از مقدار آستانه ای خطا، کمتر یا با آن مساوی شود.*
۲) *به اندازهٔ باریک ترین حرف فارسی (حرف الف) به نقطهٔ انتهائی حرف قبلی (خط نقطه چین عمودی a) نزدیک شده باشیم (خط نقطه چین عمودی b).*

در این حالت نیز اگر شرط اول برقرار شد، مادامی که این شرط برقرار است، به مرور در جهت چپ به راست ادامه می دهیم تا به حداقل مقدار خطای طبقه بندی برسیم. اما اگر شرط دوم برقرار شد، یعنی در هیچیک از دو مرور راست به چپ و چپ به راست، خطای طبقه بندی از مقدار آستانه ای کمتر نشد، کمترین خطای طبقه بندی در ضمن دو مرور را انتخاب و موقعیت متناظر با آن را به عنوان محل جدائی دو حرف در نظر می گیریم. در شکل (۱۰) منحنی پایین شکل تغییرات خطای طبقه بندی را در ضمن دو مرور نشان می دهد. همان طور که دیده می شود این نمودار در محل واقعی اتصال دو حرف «ب» و «ن» کمترین مقدار خود را داراست.

انتخاب مقدار آستانه ای خطای طبقه بندی نقش مهمی در دقت و سرعت سیستم دارد. اگر این مقدار آستانه ای خیلی کوچک باشد، هیچگاه خطای طبقه بندی از آن کمتر نمی شود و در نتیجه سیستم برای شناسائی هر حرف، تمام همسایگی راست و چپ را مرور می کند. این وضعیت باعث کندی کار سیستم می گردد. اما اگر این مقدار آستانه ای خیلی بزرگ باشد، مقدار خطای طبقه بندی سریعاً از آن کمتر می شود و در نتیجه دقت سیستم کاهش می یابد. در این پروژه این مقدار آستانه ای در ابتدای کار صفر در نظر گرفته می شود. اما بعد از شناسائی هر حرف، میانگین وزنی مقدار خطای طبقه بندی و مقدار آستانه ای محاسبه می شود و مقداری که به این ترتیب به دست می آید، جایگزین مقدار آستانه ای قبلی می گردد و سیستم با شناسائی هر حرف، این مقدار آستانه ای را تصحیح می کند و خود را با وضعیت موجود وفق می دهد.





شکل (۱۰) شناسائی حرف «ب» در کلمهٔ «شبنم» توسط الگوریتم جستجوی «ستون به ستون»

۴-۳-۱ خاصیت تجمعی گشتاورها

در روش شناسائی ستون به ستون، ابتدا ویژگی‌ها را برای یک باریکه عمودی از شکل حرف محاسبه و این باریکه را شناسائی می‌کنیم. سپس پهنای این باریکه را یک پیکسل افزایش می‌دهیم، مجدداً ویژگی‌ها محاسبه و باریکه را شناسائی می‌کنیم و این کار را تا زمانی که خطای طبقه‌بندی از یک مقدار آستانه‌ای کوچکتر شود، ادامه می‌دهیم. در این روش اگر ویژگی‌های مورد استفاده جمع‌پذیر باشند، وقتی پهنای باریکه را افزایش می‌دهیم، کافی است ویژگی‌ها را تنها برای ستونی که به شکل اضافه می‌شود محاسبه و آنها را با مقدار قبلی ویژگی‌ها جمع کنیم.

تا کنون ویژگی‌های مختلفی برای شناسائی حروف فارسی مورد بررسی قرار گرفته‌اند [۲، ۳، ۴، ۵، ۶، ۱۴، ۱۵]. یکی از موفق‌ترین ویژگی‌ها، «گشتاورهای مقیاس شده» است[۳، ۱۴]. گشتاورهای مقیاس شده از تقسیم گشتاورهای مرکزی μpq به عامل مساحت $(\mu_{oo}^{\frac{p+q}{2}+1})$ به صورت زیر به دست می‌آیند:

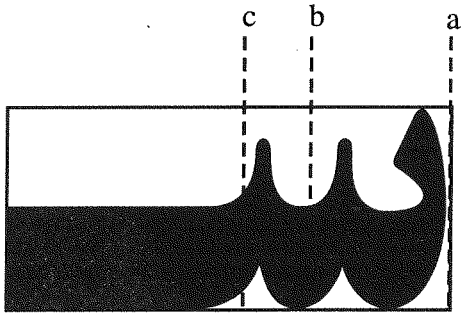
$$\mu'_{pq}=\frac{\mu_{pq}}{\mu_{oo}^{\frac{p+q}{2}+1}} \qquad (۱)$$

$$\mu_{pq}(m,n)=\sum_{x=m}^{n}\sum_{y}(x-\bar{x})^p(y-\bar{y})^q\rho(x,y) \qquad (۲)$$

گشتاورهای مقیاس شده به خودی خود جمع پذیر نیستند. بدین معنی که اگر $c \le b \le a$ باشد:

$$\mu_{pq}(c,a) \neq \mu_{pq}(c,b) + \mu_{pq}(b,a) \qquad (۳)$$

به بیان دیگر وقتی گشتاورهای مقیاس شده را برای یک باریکه از تصویر یک کلمه محاسبه کرده باشیم و یک پیکسل به پهنای باریکه اضافه کنیم، باید مقدار گشتاورها را برای همهٔ شکلی که به این ترتیب به دست می‌آید، مجدداً محاسبه کنیم. این کار باعث کندی عمل محاسبهٔ ویژگی‌ها و در نتیجه کندی کار سیستم می‌گردد. اما می‌توان با استفاده از خاصیت تجمعی (Accumulative) گشتاورها این مشکل را حل کرد.



شکل (۱۱) گشتاورهای مرکزی جمع پذیر نیستند

اما گشتاورهای ساده که از رابطهٔ:

$$m_{pq} = \sum_x \sum_y x^p y^q \rho(x,y) \qquad (۴)$$

محاسبه می‌شوند جمع پذیر هستند [۱]. یعنی:

$$m_{pq}(c,a) = m_{pq}(c,b) + m_{pq}(b,a) \qquad (۵)$$

به این ترتیب اگر بتوانیم گشتاورهای مرکزی را از روی گشتاورهای ساده محاسبه کنیم، می‌توانیم به جای محاسبهٔ مستقیم گشتاورهای مرکزی، با استفاده از خاصیت جمع پذیری، گشتاورهای ساده را محاسبه کنیم و به کمک این روابط، گشتاورهای مرکزی را به دست بیاوریم. چنین روابطی به سادگی بدست آمده‌اند [۱۸].

نکتهٔ مهمی که نباید از نظر دور داشت، این است که مثلاً برای محاسبهٔ گشتاور مرکزی μ_{21} به مقدار پنج گشتاور سادهٔ m_{21}، m_{20}، m_{10}، m_{01} و m_{00} نیاز است و شک نیست که زمان لازم برای محاسبهٔ این پنج گشتاور ساده از زمان لازم برای محاسبهٔ گشتاور مرکزی μ_{21} بیشتر است. در عمل اگر بخواهیم تنها یکی از گشتاورهای مرکزی را محاسبه کنیم، استفاده از خاصیت تجمعی گشتاورها باعث کندی کار می‌گردد و بهتر است مستقیماً از رابطهٔ گشتاور مرکزی استفاده کنیم. اما برای بازشناسی حروف و علائم تایپی فارسی یک گشتاور مرکزی به تنهائی کافی نیست و معمولاً لازم است تعداد بیشتری از گشتاورهای مرکزی را محاسبه کنیم. در این حالت خاصیت تجمعی گشتاورها باعث تسریع کار استخراج ویژگی می‌گردد.

۵ـ آزمایش سیستم

مجموعهٔ الگوریتم‌هائی که در این مقاله برای جداسازی و شناسائی حروف بیان گردید، پیاده سازی شده و میزان دقت آنها ارزیابی گردید. به منظور آزمایش سیستم، به ازاء هر یک از حروف و علائمی که یک ماشین تحریر فارسی تایپ می‌کند، ۱۰۰ نمونه تهیه کردیم. سیستم ویژگی‌ها را از این نمونه‌ها استخراج و اطلاعات مورد نیاز طبقه بندی کنندهٔ بیز را ایجاد کرد، سپس به منظور بررسی صحت عملکرد سیستم، یک صفحه متن ادبی حاوی ۳۲۰ کلمه و ۱۲۴۷ حرف انتخاب گردید (متن شمارهٔ ۱).

در اولین آزمایش از روش استخراج ویژگی «گشتاورهای مقیاس شده» و روش طبقه بندی بیز استفاده شد. در این آزمایش، مقدار آستانه‌ای خطا را بر روی عدد صفر تنظیم نموده و آن را در طول آزمایش ثابت نگه داشتیم. زمان شناسائی متن در این آزمایش ۶ دقیقه بود و سیستم در شناسائی مجموع ۱۲۴۷ حرف در ۴۷ مورد اشتباه کرد. دقت سیستم در این آزمایش ۹۶/۲٪ و سرعت آن ۳/۵ حرف در ثانیه بود.

این آزمایش یکی از نقاط ضعف مهم روش شناسائی ستون به ستون را به وضوح نشان داد. در عنوان متن مورد شناسائی، کلمهٔ «متن» به اشتباه «متوزآ» شناسائی شد. در واقع سیستم دو حرف «م» و «ت» را به طور صحیح شناسائی کرد، اما در شناسائی حرف «ن» مرتکب اشتباه شد. این اشتباه از روش کار الگوریتم جستجوی ستون به ستون ناشی می‌گردد. این الگوریتم، حرف «ن» را به سه بخش تقسیم کرده است (شکل ۱۲). قسمت اول حرف «ن» به «و»

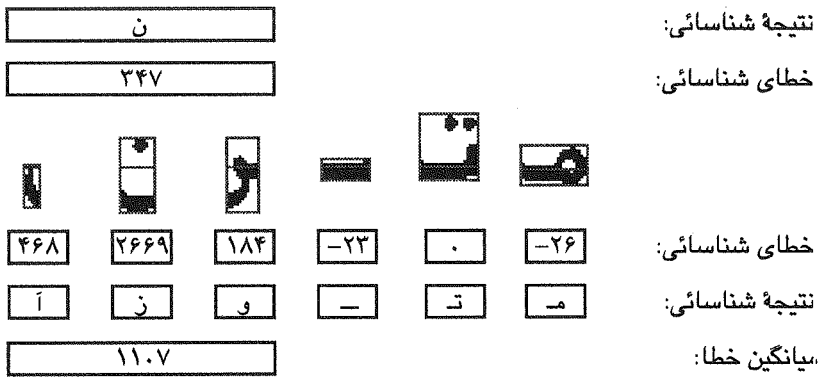
شباهت مختصری دارد. اما شباهت قسمت های بعدی به حروف «ز» و «آ» ناچیز است. اشتباه سیستم در شناسائی حرف «ن» از آنجا ناشی می شود که وقتی الگوریتم جستجوی ستون به ستون، حرف «ن» را در جهت راست به چپ مرور می کند، در منتهی الیه سمت چپ حرف «ن»، مقدار خطای طبقه بندی برابر ۳۴۷ است. چون این عدد از مقدار آستانه ای خطا بیشتر است، الگوریتم جستجوی ستون به ستون جهت پیمایش را عوض می کند و حرف «ن» را در جهت چپ به راست مرور می کند. در ضمن مرور چپ به راست، قسمت ابتدائی حرف «ن» را که به حرف «و» بی شباهت نیست، با خطای ۱۸۴ به عنوان حرف «و» شناسائی می کند. چون عدد ۱۸۴ از عدد ۳۴۷ کوچکتر است، سیستم حرف «و» را به حرف «ن» ترجیح می دهد و این قسمت از حرف «ن» را کنار می گذارد، سپس سعی می کند به همین روش حروف دیگری را جدا کند و چون چیزی که از حرف «ن» باقی مانده به حرف یا علامت دیگری شبیه نیست، بخش دیگری از این حرف را با خطای ۲۶۶۹ به عنوان «ز» و قسمت آخر را با خطای ۴۶۸ به عنوان «آ» جداسازی و شناسائی می کند.

الگوریتم جستجوی ستون به ستون به شکلی که در این پروژه پیاده سازی شده است هدف به حداقل رساندن خطا در شناسائی حرف فعلی را دنبال می کند. اگر این الگوریتم به گونه ای تغییر یابد که میانگین خطای طبقه بندی را در تمام حروف یک زیرکلمه به حداقل برساند، اشکال بالا پیش نخواهد آمد. به عنوان مثال اگر سیستم در مرور راست به چپ حرف «ن» را با خطای ۳۴۷ پذیرفته بود، مقدار متوسط خطای طبقه بندی در شناسائی این قسمت عدد ۳۴۷ بود. اما وقتی سیستم مجموعه حروف «وزآ» را می پذیرد، مقدار متوسط خطای طبقه بندی در شناسائی این سه حرف، میانگین سه عدد ۱۸۴، ۲۶۶۹ و ۴۶۸ یعنی عدد ۱۱۰۷ است. شک نیست که خطای طبقه بندی ۳۴۷ بر ۱۱۰۷ ترجیح دارد. اما وقتی سیستم خطای ۱۸۴ را بر ۳۴۷ ترجیح می دهد هیچ آینده نگری (Lookahead) نسبت به وضعیتی که بعداً پیش خواهد آمد ندارد، و این عامل یعنی فقدان آینده نگری، نقطه ضعف مهم روش جستجوی ستون به ستون است.

یکی از راه هائی که برای برطرف کردن این اشکال به ذهن می رسد، آن است که در الگوریتم جستجوی ستون به ستون، هر زیرکلمه را یک بار در جهت راست به چپ و بار دیگر در جهت چپ به راست پیمایش کنیم و در هر پیمایش، مقدار متوسط خطای طبقه بندی را به دست بیاوریم و پیمایشی را برگزینیم که در آن میانگین خطا کمتر باشد. واضح است که اعمال این روش، سرعت شناسائی متن را تقریباً نصف می کند. اما این هم چارهٔ کار نیست. به عنوان مثال همین کلمهٔ «متن» را در نظر بگیرید. شکل (۱۳) نتیجهٔ پیمایش این کلمه را در جهت چپ به راست نشان می دهد.

در این پیمایش، حرف «م» و «ن» درست شناسائی شده اما سیستم در شناسائی بقیهٔ حروف مرتکب اشتباه شده است و عامل این اشتباه دقیقاً همان فقدان آینده نگری الگوریتم شناسائی ستون به ستون است که در پیمایش راست به چپ باعث اشتباه در شناسائی حرف «ن» شده بود.

روش دیگری که برای برطرف کردن اشکال اخیر به نظر می رسید این بود که پهنای حرف شناسائی شده را نیز به



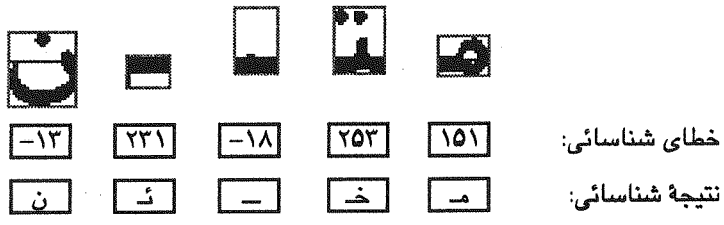
شکل (۱۲) نحوهٔ جداسازی حروف کلمهٔ «متن» توسط الگوریتم جستجوی ستون به ستون

عنوان یک عامل در محاسبهٔ خطای طبقه بندی تأثیر بدهیم. به عنوان مثال در پیمایش راست به چپ پهنای حرف «ن» ۲۸ پیکسل و پهنای بخشی از حرف «ن» که به عنوان «و» شناسائی شده است، ۱۱ پیکسل است. مثلاً اگر خطای طبقه بندی را بر پهنای حرف تقسیم کنیم، مقدار حاصل در دو مورد اخیر به ترتیب ۱۲/۴ و ۱۶/۷ است. یعنی در این حالت سیستم حرف «ن» را به «و» ترجیح می‌دهد و در این مورد خاص، اشکال برطرف می‌شود. اما آزمایش نشان می‌دهد که این روش همیشه درست کار نمی‌کند و تقسیم خطای طبقه بندی بر پهنای حرف، در بسیاری موارد باعث بروز اشتباهات دیگری می‌گردد. شاید به نظر برسد که کم کردن ضریبی از پهنای حرف از مقدار خطای طبقه بندی مناسب‌تر باشد. این روش را نیز با آزمایش بررسی کردیم. اما یافتن ضریبی که همیشه درست کار کند، تقریباً غیر ممکن است.

همان طور که گفته شد، اشتباه در شناسائی حرف «ن» ناشی از فقدان آینده نگری الگوریتم شناسائی ستون به ستون است و این اشکال جز با اعمال یک جستجوی فراگیر (Exhaustive search) برطرف نمی‌شود. در مثال اخیر اگر شرائطی فراهم می‌شد که سیستم بخشی از کلمهٔ «متن» یعنی حرف «مـ» و «تـ» و علامت کشیده را در جهت راست به چپ و حرف «ن» را در جهت چپ به راست شناسائی می‌کرد، این کلمه به درستی شناسائی می‌شد. در واقع تنها راه حل جامعی که وجود دارد، آن است که وقتی قسمتی از یک کلمه به الگوریتم شناسائی ستون به ستون واگذار می‌گردد، این الگوریتم یک بار فرض کند که این قسمت خود یک حرف یا علامت است و خطای طبقه بندی حرف را به دست بیاورد. در مرحلهٔ بعد فرض کند که این قسمت حاوی بیش از یک حرف است. در این حالت برای جدا کردن حرف اول این قسمت، نقطه ای را که در آن خطای طبقه بندی مینیمم باشد، به عنوان محل جداسازی حرف اول در نظر بگیرد و آن را جدا کند، سپس قسمتی را که باقی می‌ماند به همین روش یعنی با فراخوانی بازگشتی (Recursive) همین الگوریتم شناسائی کند و در پایان نقاطی را برای جداسازی حروف بپذیرد که میانگین مقدار خطا را در تمام زیر کلمه به حداقل برسانند. اجرای این الگوریتم به مراتب کندتر از الگوریتمی است که در این گزارش پیاده سازی شده است اما دقت آن بیشتر خواهد بود. پیاده سازی این الگوریتم می تواند به عنوان یکی از جنبه های مهم سیستم حاضر مد نظر قرار بگیرد.

به منظور بررسی میزان کارآیی روش گشتاورهای مقیاس شده، آزمایش دیگری انجام گرفت. در ابتدای این آزمایش مقدار آستانه ای خطای طبقه بندی روی عدد صفر تنظیم شد و از سیستم خواسته شد که پس از شناسائی هر حرف، مقدار خطای طبقه بندی را تصحیح کند. مقدار آستانه ای خطای طبقه بندی در طول این آزمایش وضعیت نسبتاً پایداری داشت و در پایان آزمایش به عدد ۱۷ رسید. در این آزمایش سیستم از مجموع ۱۲۴۷ حرف متن آزمایشی، در ۵۰ مورد اشتباه کرد و شناسائی این متن ۵:۳۹ دقیقه طول کشید. صحت عملکرد سیستم در این آزمایش ۹۶٪ و سرعت آن ۳/۷ حرف در ثانیه بود.

در سومین آزمایش، مقدار آستانه ای خطای طبقه بندی روی عدد ۲۰- تنظیم شد و این مقدار در طول آزمایش ثابت نگه داشته شد. خروجی این آزمایش تنها در سه مورد با حالتی که مقدار آستانه ای خطا صفر بود اختلاف داشت. سیستم در این آزمایش ۴۶ اشتباه داشت و انجام آزمایش ۱۰:۱۳ دقیقه طول کشید. کاهش مقدار آستانه ای خطا همیشه باعث کندی کار سیستم می‌گردد زیرا در این حالت



شکل (۱۳) نتیجهٔ پیمایش کلمهٔ «متن» در جهت چپ به راست

الگوریتم جستجوی ستون به ستون باید برای پیداکردن نقطهٔ جدائی حروف، مسیر طولانی تری را بپیماید.

به منظور بررسی دقیق تر تأثیر پارامتر مقدار آستانه ای خطای طبقه بندی، متن طولانی تری با روش استخراج ویژگی گشتاورهای مقیاس شده شناسائی شد. در این آزمایش مقدار آستانه ای خطا را برابر صفر قرار دادیم و از سیستم خواستیم که پس از شناسائی هر حرف یا علامت، این مقدار را تصحیح کند. در ضمن شناسائی چهار پاراگراف ابتدای متن مقدار آستانه ای خطا وضعیت نسبتاً پایداری داشت و در محدودهٔ صفر تا صد نوسان می کرد. اما در پاراگراف پنجم متن، شناسائی اشتباه چند حرف باعث گردید که مقدار آستانه ای خطا ناگهان از حالت پایداری خارج شود و یک روند افزایشی را طی نماید. معمولاً اشتباهات متوالی در شناسائی حروف باعث می گردند که مقدار آستانه ای خطا افزایش یابد. افزایش این مقدار باعث بروز اشتباه در شناسائی حروف بعدی می گردد و اشتباه در شناسائی باعث افزایش بیشتر و بیشتر مقدار آستانه ای خطای طبقه بندی می شود. این وضعیت چندان هم دور از انتظار نیست زیرا تنظیم مقدار آستانه ای خطای طبقه بندی، عاملی است که حلقهٔ پس خور سیستم را می بندد و طبیعتاً می تواند سیستم را به حالت ناپایداری ببرد. در سیستم فعلی مکانیزم بازدارنده ای برای افزایش ناگهانی مقدار آستانه ای خطای طبقه بندی پیش بینی نشده است. اما این آزمایش نشان می دهد که وجود چنین عامل بازدارنده ای در سیستم لازم و ضروری است.

گشتاورهای مقیاس شده که در این سیستم به عنوان ویژگی مورد استفاده قرار گرفته اند، از اندازهٔ نمونه ها مستقل هستند. به بیان دیگر اگر سیستم را با نمونه هایی که در اندازه استاندارد تایپ شده اند آموزش دهیم، می توانیم نمونه هایی را که در اندازه های بزرگتر یا کوچکتر از اندازهٔ استاندارد باشند شناسائی کنیم. در این سیستم، اندازهٔ نمونه های آزمایشی تنها در یک مورد می تواند بر دقت سیستم تأثیر بگذارد. در روش جستجوی ستون به ستون گفتیم که پیمایش در جهت چپ تنها به اندازهٔ پهنای پهن ترین حرف فارسی یعنی حرف «ص» انجام می شود. طبیعتاً اگر نمونه های آزمایشی بزرگتر از نمونه های آموزشی سیستم باشند، پیمایش به چپ نیز باید در محدودهٔ بزرگتری انجام شود. به این جهت در این سیستم همیشه پیمایش در جهت چپ به میزان ضریبی از پهنای حرف «ص» انجام می شود و برای آن که سیستم قادر باشد متن هایی را که در اندازه های مختلف تایپ شده اند شناسایی کند، مقدار این ضریب بعد از شناسایی هر حرف به طور خودکار تصحیح می شود. این ضریب که در واقع معرف نسبت اندازهٔ حروف آزمایشی به اندازهٔ حروف مورد شناسایی است، پارامتر «مقیاس» نامیده شده است.

نکتهٔ مهمی که در تنظیم پارامتر مقیاس وجود دارد آن است که تنظیم این مقدار نیز به نحوی حلقهٔ پس خور سیستم را می بندد و این عامل نیز ممکن است باعث ناپایداری سیستم شود. به عنوان مثال فرض کنید سیستم بخشی از یک حرف را به اشتباه به عنوان حرف دیگری که تغییر اندازه یافته است، شناسائی کند. این اشتباه باعث کاهش پارامتر مقیاس می گردد. تغییر این پارامتر ممکن است باعث اشتباهات دیگر و در نتیجه کاهش بیشتر پارامتر مقیاس بشود و ناگهان سیستم را به سمت ناپایداری سوق دهد. در این سیستم برای تغییر پارامتر مقیاس نیز هیچ عامل بازدارنده ای پیش بینی نشده است. طراحی یک عامل بازدارنده برای پارامتر مقیاس الگوریتم های پیچیده تری را طلب می کند، زیرا سیستم هیچ فرضی در مورد اندازهٔ متنی که قرار است شناسائی شود، ندارد و هیچ بعید نیست که اندازهٔ متن دو سطر متوالی حتی ده برابر با یکدیگر اختلاف داشته باشد. در واقع سیستم از یک طرف نمی تواند از تغییرات پارامتر مقیاس پیش گیری کند زیرا ممکن است تغییرات این پارامتر، واقعاً ناشی از تغییر اندازهٔ متن مورد شناسائی باشد. از طرف دیگر اشتباهاتی که در شناسائی متن پیش می آیند، ممکن است مقدار نامناسبی را در این پارامتر قرار دهند و سیستم باید تنها در مقابل این گونه تغییرات مقاومت نشان دهد.

جدول (۱) خلاصهٔ نتایج آزمایش های انجام شده را ارائه می دهد.

جدول (۱) نتایج شناسائی متن های تایپی فارسی

مقدارآستانه ای خطا	دقت سیستم	سرعت سیستم
۰ ← ۰	٪۹۶/۲	۳/۵ حرف در ثانیه
۰ ← ۱۷	٪۹۶	۳/۷ حرف در ثانیه
-۲۰ ← -۲۰	٪۹۶/۲	۲ حرف در ثانیه

۶ـ جمع بندی:

در این مقاله سیستمی که به منظور شناسائی متن های تایپی فارسی طراحی و پیاده سازی شده است، ارائه گردید. برای یافتن خط مبنا و جداسازی حروف که از بخش های کلیدی یک سیستم شناسائی متن می باشند روش جدیدی ارائه گردید. آزمایش ها نشان می داد که دقت این سیستم برای شناسائی متن نزدیک به ۹۶٪ است و سرعت سیستم در شناسائی برای یک سیستم ۸۰۴۸۶ با سرعت ۳۳ مگاهرتز به ۳/۷ حرف در ثانیه می رسد.

تنها ملاک سیستم حاضر در شناسائی متن، شکل ظاهری حروف و برخی قواعد نگارشی زبان فارسی در مورد شیوهٔ به هم چسبیدن حروف است. این سیستم از قواعد املائی زبان فارسی استفاده نمی کند. یکی از مهمترین عواملی که می تواند در افزایش دقت این سیستم مورد بررسی قرار بگیرد، استفاده از یک بخش تصحیح املائی کلمات است.

عامل مهم دیگری که می تواند در افزایش سرعت سیستم مؤثر واقع گردد، استفاده از یک درخت تصمیم گیری و یا استفاده از یک طبقه بندی کنندهٔ چند سطحی به جای طبقه بندی کنندهٔ سادهٔ فعلی است. در سیستم فعلی هر حرف مورد شناسائی از طریق طبقه بندی کنندهٔ بیز با همهٔ ۹۵ حرفی که یک ماشین تحریر قادر به تایپ آن است، مقایسه می شود. استفاده از یک درخت تصمیم گیری می تواند تعداد این مقایسه ها را تا حد قابل ملاحظه ای کاهش دهد و باعث افزایش سرعت سیستم بشود.

مکانیزم تصحیح مقدار آستانه ای این سیستم نیز به سادگی دچار اشتباه می شود. در این سیستم هیچ مکانیزم بازدارنده ای برای افزایش ناخواستهٔ مقدار آستانه ای خطا وجود ندارد و چند اشتباه متوالی سیستم می تواند مقدار آستانه ای خطا را از حالت پایداری خارج کند و موجب کاهش ناگهانی دقت سیستم گردد.

منابع:

[1] B. Parhami, M. Taraghi, "Automatic Recognition of Printed Farhi Texts", Pattern Recognition, Vol. 14, No. 1, 1981.

[2] Sherif Sami El-Dabi, Refat Ramsis & Aladin Kamel, Arabic Character Recognition System: A Statistical Approach for Recognizing Cursive TypeWritten Text, Pattern Recognition, pp. 485- 495, 1990.

[3] Talaat S. El-Sheikh and Ramez M. Guindi, Computer Recognition of Arabic Cursive Scripts, Pattern Recognition, vol. 21, no. 4, pp. 293-302, 1988.

[4] Adnan Amin and J. F. Mari, Machine Recognition and Correction of Printed Arabic Text, IEEE Transactions on System, Man & Cybernetics, vol 19, no. 5, Sep/Oct 1989.

[5] See Farsi References.

[6] See Farsi References.

[7] F. Alt, Digital Pattern Recognition by Moments, Journal of ACM, Vol 9, pp 240-258, 1962.

[8] Ming-Kuei Hu, Visual Pattern Recognition by Moment Invariants, IRE Transactions on Information Theory, Feb. 1962.

[9] T.H. Reiss, The Revised Fundamental Theorem of Moment Invariants, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, no. 8, Aug. 1991.

[10] Firooz A. Sadjadi and Ernest L. Hall, Numerical Computations of Moment Invariants for Scene Analysis, in proc. IEEE Conf. on Pattern Recognition and Image Processing, Chicago, IL, pp. 181-186, 1978.

[11] A. Blicher, R.M. Minton and R. Glicksman, Pattern Recognition by

[۵] محمدرضا احمدزاده، احسان‌اله کبیر. شکستن کلمات تایپ شدهٔ فارسی به حروف، گزارش اولین کنفرانس بین‌المللی کامپیوتر در علوم، فنون و پزشکی در ایران، ۵ تا ۷ دی‌ماه ۱۳۷۰.

[۶] احسان‌اله کبیر. گزارش سمینار بازشناسی حروف در مرکز پژوهش‌های علمی و صنعتی ایران، ۳۱ تیرماه ۱۳۷۱.

[۱۴] رضا صفابخش، وحدت دست‌پاک. شناسائی مستقل از اندازه و چرخش حروف تایپی فارسی، مجموعه مقالات کامپیوتر کنفرانس مهندسی برق ایران، اردیبهشت ۷۲، ص ۲۶۷ تا ۲۷۶.

[۱۵] رضا صفابخش، وحدت دست‌پاک. شناسائی حروف تایپی فارسی به روش موزائیک بندی، مجموعه مقالات مخابرات کنفرانس مهندسی برق ایران، اردیبهشت ۷۳، ص ۲۶۷ تا ۲۷۶.

[۱۸] وحدت دست‌پاک. شناسائی ماشینی حروف متنوع فارسی، پایان نامه کارشناسی ارشد، دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، ۱۳۷۲.

[۱۹] حمید فهیمی-بیژن تیمساری. بازشناسی حروف در کلمات تایپ شده فارسی با استفاده از روش مورفولوژی، مجموعه مقالات کنفرانس برق ایران، اردیبهشت ۷۲، ص ۲۷۷ تا ۲۸۵.

Moment Invariants, Proceedings of the IRE, Sept. 1961.

[12] H. S. Barid, Tke Skew Angle of Printed Documents, Proc. SPSE 40th Conf. Symp. Hybrid Imaging Systems, Rochester, NY, pp. 21-24, May 1987.

[13] H. S. Baird and K. Thompson, Reading Chess, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 12, no. 6, June 1990.

[14] See Farsi References.

[15] See Farsi References.

[16] M. F. Tolba, E. Shahdad, On the Automatic Reading of Printed Arabic Characters, IEEE, pp 496-498, 1990

[17] R. G. Casey, Moment Normalization of Handprinted Characters, IBM Journal of Research and Development, pp 548-557, Sept 1970.

[18] See Farsi References.

[19] See Farsi References.