ساخت آوایی گونهٔ رسمی و محاوره‌ای زبان فارسی*

وحید مواجی[1] و محرم اسلامی[2]

[1]دانشگاه صنعتی شریف
[2]دانشگاه زنجان

۱۵ آذر ۱۳۹۱

چکیده

گونهٔ رسمی و گونهٔ محاوره‌ای زبان‌ها غالباً با هم تفاوت دارند و این تفاوت در همهٔ سطح‌های زبانی دیده می‌شود. میزان تفاوت بین گونهٔ رسمی و گونهٔ محاوره‌ای، که گاهی از آنها با عنوان تفاوت گفتار و نوشتار یاد می‌شود، از زبانی به زبان متفاوت است. زبان فارسی از جملهٔ زبان‌های است که در آن تفاوت گونهٔ رسمی و گونهٔ محاوره‌ای بسیار زیاد است. در این تحقیق تفاوت‌ها آوایی و به عبارتی فرایندهای آوایی را بررسی می‌کنیم که در زبان فارسی در تبدیل گونهٔ رسمی به گونهٔ محاوره‌ای رخ می‌دهد. مبنای پژوهش حاضر دادگان گفتاری فارسدات تلفنی زبان فارسی است که در آن گفتار پیوستهٔ در دو سطح واجی و آوایی در قالب دو زنجیرهٔ مستقل برچسب خورده است. هم‌گذاری این دو رشته از داده‌ها روشن می‌سازد که در مقایسهٔ این دو گونهٔ زبانی کدام فرایندهای آوایی در تبدیل زنجیرهٔ واجی به زنجیرهٔ آوایی دخیل بودند. در انطباق دو رشتهٔ واجی و آوایی از الگوریتم لونشتاین استفاده کرده‌ایم که مناسب و رایج در انطباق تقریبی رشته‌های متفاوت جهت یافتن فاصلهٔ بین آنها است. در نتیجه تفاوت دو رشتهٔ واجی و آوایی به‌صورت آماری به‌دست آمد. از نتایج این پژوهش می‌توان به لحاظ نظری در توصیف‌های زبان‌شناختی از نظام آوایی زبان فارسی، تهیهٔ منابع محاوره‌ای زبانی فارسی و آموزش زبان فارسی به‌خصوص به غیرفارسی‌زبان‌ها سود جست. از سوی دیگر در فناوری‌های گفتار مانند بازشناسی و بازسازی گفتار، استخراج اطلاعات از متن‌های محاوره‌ای، تبدیل متن به زنجیرهٔ واجی گونهٔ محاوره‌ای زبان فارسی و امکان تبدیل آن به گونهٔ رسمی می‌توان از نتایج این تحقیق استفاده کرد.

کلیدواژه‌ها: ساخت آوایی، گونهٔ رسمی، گونهٔ محاوره‌ای، الگوریتم لونشتاین، فارسدات تلفنی فارسی

۱ مقدمه

گونه‌های رسمی و محاوره‌ای زبان فارسی در تمام سطح‌های زبان تفاوت‌های زیادی با هم دارند که این امر در مقایسه با برخی زبان‌ها بسیار چشمگیر است. در همهٔ زبان‌ها بین گونه‌های رسمی و محاوره‌ای تفاوت‌های هست و در زبان‌های که مسبوق به داشتن نظام نوشتاری دیرینه هستند، این تفاوت بیشتر به چشم می‌خورد. بنابراین زبانی که زبان علم و ادب باشد، که طبیعتاً دارای خط نیز هستند، بین گونه‌های رسمی و محاوره‌ای آن فاصله بیشتر خواهد بود. از دلایل این فاصله می‌توان به محافظه‌کارانه عمل کردن خط در انعکاس تغییرات زبان اشاره کرد که نهادهای مانند فرهنگستان زبان عامدانه در قالب یک برنامه‌ریزی زبانی متولی انجام تغییرات در خط می‌شوند.

*فصلنامه پازند، سال ۸، شماره ۳۰، پاییز ۱۳۹۱، صص ۱۱۸-۱۰۷

دلیل دیگر شاید این باشد که در محیط‌های علمی افراد تحصیل‌کرده کمتر حاضر می‌شوند خلاف سنت نوشتاری زبان بنویسند. این موضوع باعث می‌شود به مرور زمان گونه‌های رسمی و محاوره‌ای مثلاً از حیث نظام آوایی فاصله بگیرند وضعیتی که در زبان فارسی شاهد هستیم.

در این پژوهش کوشیده‌ایم تفاوت‌های آوایی گونه‌های رسمی و محاوره‌ای زبان فارسی را به‌صورت آماری با استفاده از پیکرهٔ زبانی فارسدات تلفنی (Bijankhan et al. (2003 استخراج و دسته‌بندی بکنیم. در فارسدات تلفنی گفتار پیوسته در دو سطح واجی و آوایی برچسب خورده‌اند که مقایسهٔ زنجیره نشان می‌دهد که گفتار رسمی و محاوره‌ای زبان چه تفاوت‌های آوایی با هم دارند. هم‌گذاری خوکار زنجیره‌های واجی و آوایی امر دشواری است، چراکه تحولات آوایی در همهٔ جایگاه‌های صدایی کلمه رخ می‌دهد. اینکه کدام واحد صدایی به کدام واحد صدایی تبدیل شده و کدام واحد صدایی حذف یا درج شده در بررسی خودکار دشواری‌های را ایجاد می‌کند. تغییرات آوایی را می‌توان به چند دسته یا عمل تقسیم کرد:

الف. عمل درج: مثلاً /mehrban/ با درج /a/ به /mehraban/ تبدیل می‌شود.

ب. عمل حذف: مثلاً /dast/ با حذف همخوان پایانی به /das/ تبدیل می‌شود.

ج. عمل جانشینی: مثلاً /?ejtemâ/ با جانشینی /š/ به جای /j/ و حذف همزهٔ پایانی به /?eštemâ/ تبدیل می‌شود.

چناچه در ج) می‌بینیم چندین عمل می‌توانند هم‌زمان روی یک رشته اِعمال شوند. مثلاً در مثال ج) عمل حذف، همزهٔ پایانی کلمه را نیز حذف کرده است. عمل‌های ذکرشده منحصر به یک نویسهٔ واحد نمی‌باشند و می‌توانند روی گروهی از نویسه‌ها اِعمال شوند، مثلاً می‌گویند تبدیل می‌شود به می‌گن که در آن گروه رشتهٔ آوایی /-uyand/ کلاً با گروه /-an/ جانشین شده است.

برای این که تغییرات آواجی را بدست آوریم باید بافتِ آوایی را هم در نظر داشته باشیم که در این مقاله، ما از واج قبلی و واج بعدی به عنوان بافت آوایی استفاده کرده‌ایم. پیکرهٔ فارسدات تلفنی حاوی صورت واجی و آوایی کلماتی است که از ضبط صدای افراد پشت تلفن بدست آمده است. اگر صورت‌های واجی و آوایی را به‌صورت مجموعه یا رشته‌هایی از نویسه‌ها در نظر بگیریم، مسأله به تعیین میزان فاصلهٔ دو رشته کاهش می‌یابد. یعنی هر چه فاصلهٔ دو رشته کمتر باشد مشابهت بیشتری به هم دارند. بنابراین تغییر واجی از روی فاصلهٔ کمینهٔ بین دو رشته بدست خواهد آمد.

الگوریتم‌های مختلفی برای سنجش فاصلهٔ دو رشته وجود دارد که ما در این پژوهش از الگوریتم لونشتاین (Levenshtein (1966 استفاده کرده‌ایم. البته این الگوریتم فاصلهٔ کمینهٔ دو رشته را محاسبه می‌کند، ولی مراحل اِعمال عمل‌های مختلف (درج، حذف، جانشینی) را نمی‌دهد. لذا با تغییری که در این الگوریتم داده شد، مراحل اِعمال عمل‌های واجی را هم بدست آوردیم.

شایان ذکر است که گونهٔ گفتاری معیار زبان فارسی از فارسدات تلفنی در این مقاله مدنظر بوده است.

۲ معیارهای مشابهت

برای ارزیابی میزان شباهت یا عدم شباهت (فاصله) دو رشته نسبت به یکدیگر از معیارهای مشابهت رشته‌ای استفاده می‌شود. معیار مشابهت یک عدد اعشاری است که درجه مشابهت دو رشته از نویسه‌ها را مشخص می‌سازد. مثلاً فاصلهٔ بین دو رشتهٔ رایانه و پایانه کمتر از فاصلهٔ بین دو رشتهٔ رایانه و یارانه است.

از معیارهای مشابهت در زمینه‌های مختلفی منجمله تطابق تقریبی رشته‌ها (Navarro (2001، مقایسه (Stoilos et al. (2005 و جستجوی فازی رشته‌ها (Jin and Li (2005 استفاده می‌شود. یکی از پرکاربردترین زمینه‌های آن، خطایاب‌های املایی (Li et al. (2008 است، بدین صورت که متنی به عنوان ورودی به خطایاب داده می‌شود و خطایاب باید کلمات اشتباه را با نزدیک‌ترین کلمات (کلماتی با بیشترین میزان مشابهت و کمترین فاصله) جایگزین سازد. در ادامه پرکاربردترین معیارهای فاصلهٔ دو رشته معرفی شده‌اند.

۱.۲ فاصلهٔ لونشتاین[۱]

فاصلهٔ لونشتاین (Levenshtein (1966)) به صورت حداقل میزان تغییرات مورد نیاز برای رشتهٔ S به رشتهٔ T تعریف می‌شود که عملیات مجاز در آن عبارتند از: درج، حذف یا جانشینی یک نویسهٔ واحد. برای مثال فاصلهٔ لونشتاین دو رشتهٔ apple و orange با یک ماتریس ساده در جدول ۱ محاسبه شده است. فاصلهٔ این دو رشته برابر ۵ است که در قسمت پایین و سمت راست ماتریس نشان داده شده است. این فاصله بدین معناست که رشتهٔ apple را با درج o، جانشینی r به جای a، a به جای p، p به جای g و g به جای l به رشتهٔ orange تبدیل می‌شود. یک عملیات درج و چهار عملیات جانشینی داریم که مجموعاً برابر ۵ می‌شود.

جدول ۱: مثالی از فاصلهٔ لونشتاین

		A	P	P	L	E
	۰	۱	۲	۳	۴	۵
O	۱	۱	۲	۳	۴	۵
R	۲	۲	۲	۳	۴	۵
A	۳	۲	۳	۳	۴	۵
N	۴	۳	۳	۴	۴	۵
G	۵	۴	۴	۴	۵	۵
E	۶	۵	۵	۵	۵	۵

مراحل ساخته شدن ماتریس شکل ۱ به شرح ذیل است:

۱. ماتریس d با M سطر و N ستون ساخته می‌شود.

۲. به سطر اول مقادیر از 0 تا M و به ستون اول مقادیر از 0 تا N اختصاص داده می‌شود.

۳. هر نویسه‌ای از S (i از 1 تا M) و هر نویسه‌ای از T (j از 1 تا N) را بررسی می‌کنیم.

۴. اگر $S[i]$ با $T[j]$ برابر بود، هزینه برابر 0 و در غیر این صورت برابر 1 است.

۵. مقدار عنصر $d[i,j]$ ماتریس برابر کمینه مقادیر زیر است:

(آ) عنصر بالایی بعلاوهٔ یک: $d[i-1,j]+1$

(ب) عنصر سمت چپ بعلاوهٔ یک: $d[i,j-1]+1$

(ج) عنصر بالا و سمت چپ بعلاوهٔ هزینه: $d[i-1,j-1]+cost$

۶. بعد از این که تکرار مراحل ۳، ۴ و ۵ به پایان رسید، مقدار فاصله در عنصر $d[N,M]$ قرار دارد.

شبه‌کد فاصلهٔ لونشتاین در برنامه ۱ آمده است.

برنامه ۱: شبه‌کد فاصلهٔ لونشتاین

```
1        int LevenshteinDistance(char S[1..M], char T[1..N]){
2            declare int d[0..M, 0..N]
3            for i from 0 to M
4                d[i,0] := i //the distance of any first string to an
                     empty second string
5            for j from 0 to N
```

[۱] Levenshtein distance

```
6                   d[0, j] := j //the distance of any second string to
                        an empty first string
7               for j from 1 to N
8               {
9                   for i from 1 to M
10                  {
11                      if S[i] = T[j] then
12                          d[i, j] := d[i-1, j-1] //no operation
                                required
13                      else d[i, j] := minimum(d[i-1, j] + 1, //a
                            deletion
14                                  d[i, j-1] + 1, //an insertion
15                                  d[i-1, j-1] + 1) //a substitution
16                  }
17              }
18              return d[M, N]
19          }
```

۲.۲ فاصلهٔ همینگ[۲]

فاصلهٔ همینگ (Hamming (1950 فقط عملیات جانشینی را برای تبدیل S به T مجاز می‌شمارد. بنابراین طول S و T باید برابر باشد. از این الگوریتم بیشتر برای تشخیص و تصحیح خطا برای دو رشته با طول مساوی استفاده می‌شود. پیاده‌سازی فاصلهٔ همینگ در شبه‌کد برنامه ۲ آمده است. این الگوریتم، دو رشتهٔ S و T را به عنوان ورودی گرفته و فاصلهٔ همینگ آنها را برمی‌گرداند.

برنامه ۲: شبه‌کد فاصلهٔ همینگ

```
1           int HammingDistance(S[1..M], T[1..N])
2           {
3               If M !=N then return -1
4               declare int HammingDistance=0
5               for i from 1 to M{
6                   if S[i] != T[i] then HammingDistance++
7               }
8               return HammingDistance
9           }
```

۳.۲ فاصلهٔ دامرو–لونشتاین[۳]

فاصلهٔ دامرو–لونشتاین (Damerau (1964 مشابه فاصلهٔ لونشتاین است. تنها فرق آن این است که فاصلهٔ دامرو–لونشتاین یک عملیات دیگر را مجاز می‌شمارد: ترانهش[۴] دو نویسهٔ مجاور. شبه‌کد محاسبهٔ فاصلهٔ دامرو–لونشتاین در برنامه ۳ آمده است.

برنامه ۳: شبه‌کد فاصلهٔ دامرو–لونشتاین

---

[۲] Hamming distance
[۳] Damerau–Levenshtein distance
[۴] Transposition

```
        int DamerauLevenshteinDistance(char S[1..M], char T[1..N])
        {
            declare int d[0..M, 0..N]
            declare int i, j, cost
            for i from 0 to M
                d[i,0] := i //the distance of any first string to an
                    empty second string
            for j from 0 to N
                d[0, j] := j //the distance of any second string to
                    an empty first string
            for i from 1 to M
            {
                for j from 1 to N{
                if S[i] = T[j] then cost := 0
                        else cost := 1
                    d[i, j] := minimum(d[i-1, j] + 1, //a deletion
                                d[i, j-1] + 1, //an insertion
                                d[i-1, j-1] + 1) //a substitution
                if(i > 1 and j > 1 and S[i] = T[j-1] and S[i-1] = M[j
                    ]) then
                    d[i, j] := minimum(
                                d[i, j],
                                d[i-2, j-2] + cost) // transposition
                }
            }
            return d[M, N]
        }
```

در این مقاله از روش فاصلهٔ لونشتاین استفاده کرده‌ایم و در آن تغییراتی داده‌ایم که جابجایی یک گروه با یک گروه دیگر را و اینکه چه مجموعه نویسه‌ای به چه مجموعه نویسه‌ای تبدیل شده است را نیز بدست می‌آوریم. در الگوریتم استاندارد لونشتاین، تغییرات یک نویسه درنظرگرفته می‌شود و درنهایت نیز خروجی الگوریتم برابر فقط حداقل مراحل تغییرات است که در خانه پایین و راست ماتریس موردنظر قرار دارد.

۳ روش کار

در پیکرهٔ زبانی این تحقیق (فارس‌دات تلفنی) مجموعه‌ای شامل ۲۸۲۵۵۳ ردیف داده قرار دارد. اطلاعات موجود در هر سطر این پیکره شامل موارد زیر است:

- صورت واجی (Phonemic) مانند hastid
- صورت آوایی (Phonetic) مانند hastin
- صورت نوشتاری فارسی مانند هستید
- زمان شروع و زمان پایان گفتار.

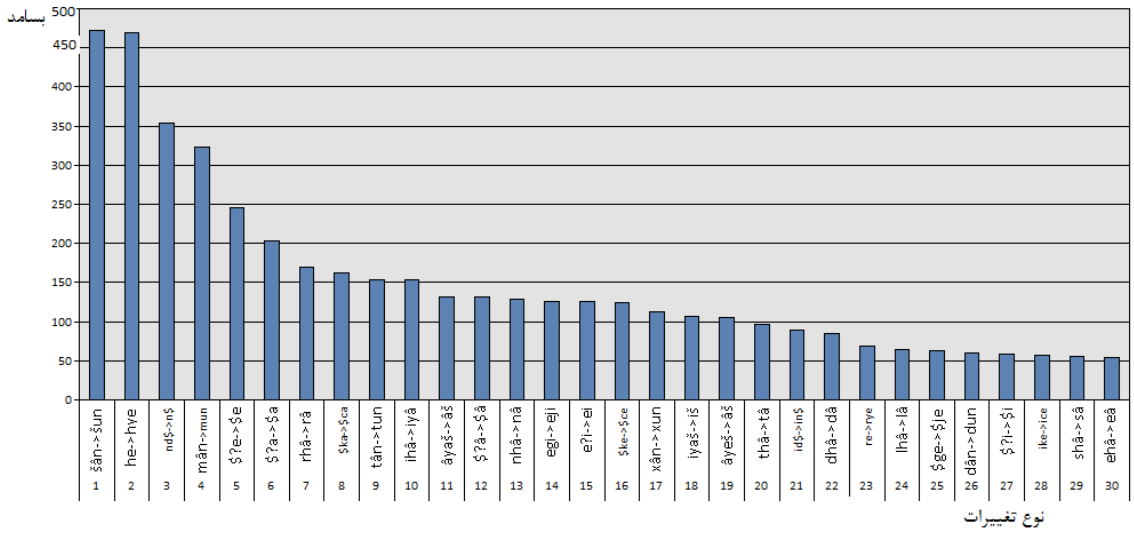
برای کار مورد نظر ما، همهٔ سطرهای این پیکره دارای ارزش اطلاعاتی نبودند، چرا که برخی اطلاعات مانند مکث، تپق، خنده، سرفه و... نیز در این پیکره ذخیره شده است. لذا در مرحله اول کار، پیکره از این گونه اطلاعات پالایش شد و ۲۱۹۵۸۶ سطر از اطلاعات باقی ماند. از این مقدار نیز، مقادیر زیادی دارای صورت واجی و آوایی یکسانی بودند که این گونه اطلاعات نیز پالایش شد و در نتیجه ۱۶۹۵۰ سطر دارای زنجیره‌های واجی و آوایی

متفاوت باقی ماند. سپس الگوریتم تفاوت‌یابی رشته‌های که در مرحله قبل توضیح داده شد روی این تعداد داده اعمال گردید که تعداد ۱۳۵۲۷ تغییر بدست آمد. البته این میزان تغییرات دارای تکرار بود و در بررسی دقیقتر، تعداد تغییرات برابر ۲۷۹۳ نوع بود. ما برای بررسی بسامد تغییرات آوایی، تغییرات با تکرار را در نظر گرفتیم تا به یک تحلیل آماری از میزان تغییرات در آواها هنگام صحبت روزمره برسیم.

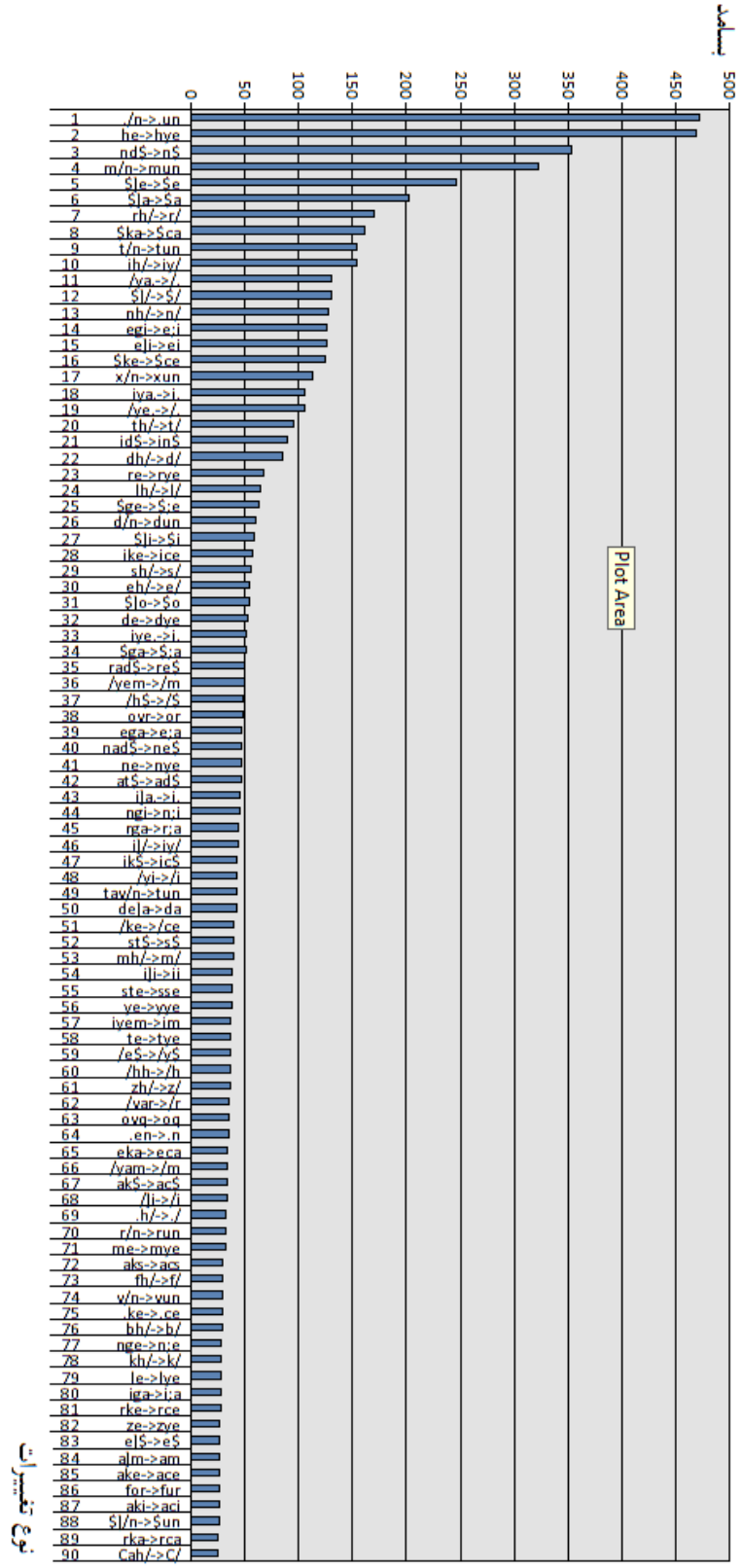
۴ نتایج بدست آمده

تعداد زیادی از این تغییرات دارای بسامد کمی (حتی یک بار) بودند و تعداد کمی از این تغییرات دارای بسامد زیاد. همانطور که در نمودارهای زیر مشاهده می‌کنیم، درصد کمی از تغییرات، بیشترین میزان توزیع بسامدی را اشغال کرده‌اند، یعنی تقریباً ۳۰ تغییر اول بیشترین بسامد را دارند و مابقی تغییرات دارای بسامد و احتمال وقوع کمتری می‌باشند. نمودار توزیع فراوانی ۳۰ تغییر اول در شکل ۵ و نمودار توزیع فراوانی ۹۰ تغییر اول در شکل ۶ آمده است. جدول ۱ نمودار توزیع فراوانی ۳۰ تغییر اول را با جزئیات بیشتری مشخص می‌سازد. مثالی از تغییرات پربسامد به شرح زیر می‌باشد:

- šân → šun
- nd → n
- mân → mun
- ?e → e
- ?a → a
- rh → r
- ka → ca
- tân → tun
- ihâ → iyâ
- âyaš → âš
- ?â → â
- nhâ → nâ



شکل ۱: نمودار توزیع فراوانی ۳۰ تغییر اول



شکل ۲: نمودار توزیع فراوانی ۹۰ تغییر اول

۵ نتیجه‌گیری

در این تحقیق ساخت آوایی گونهٔ رسمی و محاوره‌ای زبان فارسی براساس پیکرهٔ زبانی فارسدات تلفنی مورد بررسی قرار گرفت و در نتیجهٔ آن نوع و بسامد تحولات آوایی در زبان فارسی در چارچوب پیکرهٔ زبانی موردنظر مشخص شد. نوع تحولات آوایی محدود به عمل‌های جانشینی، حذف و درج بود که از مقایسهٔ فاصلهٔ نویسه‌های مربوط به کلمه‌ها در دو سطح واجی و آوایی موجود در فارسدات تلفنی به‌دست می‌آمد. با انجام تغییراتی از الگوریتم لونشتاین برای مقایسهٔ خوکار عمل‌های جانشینی، حذف و درج استفاده کردیم. در ارزیابی نتیجهٔ پژوهش مشخص شد که تعداد زیادی از این تغییرات دارای بسامد کم و تعداد کمی از تغییرات دارای بسامد زیاد بودند. شکل‌های ۵ (نیز در جدول ۱) و ۶ به‌ترتیب توزیع فراوانی ۳۰ و ۹۰ تغییر اول را با جزئیات بیشتری مشخص می‌سازند.

از نتایج این پژوهش می‌توان به لحاظ نظری در توصیف‌های زبان‌شناختی از نظام آوایی زبان فارسی، تهیهٔ منابع محاوره‌ای زبانی فارسی و آموزش زبان فارسی به‌خصوص به غیرفارسی‌زبان‌ها سود جست. از سوی دیگر در فناوری‌های گفتار مانند بازشناسی و بازسازی گفتار، استخراج اطلاعات از متن‌های محاوره‌ای، تبدیل متن به زنجیرهٔ واجی گونهٔ محاوره‌ای زبان فارسی و امکان تبدیل آن به گونهٔ رسمی می‌توان از نتایج این تحقیق استفاده کرد.

مراجع

Hamming , R. W. (1950), “Error detecting and error correcting codes,” The Bell System Technical Journal, 29, 147–160. 4

Levenshtein , V. I. (1966), “Binary Codes Capable of Correcting Deletions, Insertions and Reversals,” Soviet Physics Doklady, 10, 707. 2, 3

Li , C., Lu , J., and Lu , Y. (2008), “Efficient Merging and Filtering Algorithms for Approximate String Searches,” in 2008 IEEE 24th International Conference on Data Engineering, pp. 257–266. 2

Bijankhan, M., Sheykhzadegan, J., Roohani, M. R., Zarrintare, R., Ghasemi, S. Z., and Ghasedi, M. E. (2003), “TFarsDat - The Telephone Farsi Speech Database,” Proc. Eurospeech, Geneva, Switzerland, 1525 – 1528. 2

Damerau, F. J. (1964), “A Technique for Computer Detection and Correction of Spelling Errors,” Commun. ACM, 7, 171–176. 4

Jin, L. and Li, C. (2005), “Selectivity Estimation for Fuzzy String Predicates in Large Data Sets.” vol. 1, pp. 397–408. 2

Navarro, G. (2001), “A Guided Tour to Approximate String Matching,” ACM Comput. Surv., 33, 31–88. 2

Stoilos, G., Stamou, G., and Kollias, S. (2005), “A String Metric for Ontology Alignment,” in The Semantic Web – ISWC 2005, eds. Gil, Y., Motta, E., Benjamins, V. R., and Musen, M. A., Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 624–637. 2