
دانشکده زبان‌ها و زبان‌شناسی

**پروژه درس**

# آشنایی با زبان‌شناسی رایانشی

یادگیری بی‌سرپرست مورفولوژی زبان فارسی

با استفاده از Morfessor

استاد: دکتر محمد بحرانی

وحید مواجی
89702855

تابستان 1390

**این پروژه طبق موافقت و هماهنگی بعمل آمده با اساتید محترم به طور مشترک برای این درس و درس روش‌های آماری در پردازش زبان طبیعی (دکتر وزیرنژاد) انجام شده است.*

## 1. مقدمه

طبق تئوری‌های صرف زبانشناختی، تکواژها کوچکترین واحدهای حامل معنی زبان هستند. هر شکل کلمه را می‌توان به صورت ترکیبی از تکواژها نشان داد. مثلاً در کلمات انگلیسی زیر:

arrange+ment+s, foot+print, mathematic+ian+’s, un+fail+ing+ly

تحلیل خودکار صرفی برای بسیاری از کاربردهای زبان طبیعی که با واژگان بزرگی سروکار دارد مفید فایده است؛ مثلا در پردازش گفتار و ترجمه ماشینی. بسیاری از برنامه‌های موجود، از کلمات به عنوان واحدهای پایه واژگان استفاده می‌کنند. با این حال برای زبان‌های بسیار تصریفی مثل فنلاندی، ترکی و استونیایی، این امر غیرممکن است چرا که تعداد اشکال کلمه ممکن بسیار زیاد است. همین امر (به میزان کمتر) برای زبان‌های ترکیبی مثل آلمانی، سوئدی و یونانی صادق است. تحلیلگرهای صرفی وجود دارد که توسط متخصصین زبان‌های بخصوص ساخته شده است. ولی دانش متخصص و هزینه نیروی کار او زیاد است. علاوه بر این باید برای هر زبان به طور جداگانه تحلیلگر ساخته شود و این تحلیلگرها باید مرتباً و به طور پیوسته به روز شوند تا با تغییرات زبانی همخوانی داشته باشد (مثلاً بعلت ایجاد کلمات جدید و تصریفات آنها).

در کنار سیستم‌هایی که ذکر شد، الگوریتم‌هایی وجود دارد که به طور بی‌سرپرست کار می‌کنند و تقطیع‌های تکواژی کلمات را در یک پیکره که نشانه‌گذاری نشده است پیدا می‌کنند. Morfessor یک مدل کلی برای یافتن تصریف بی‌سرپرست و ساده کلمات از روی یک متن خام می‌باشد. Morfessor طوری طراحی شده است که برای زبانهایی عمل می‌کند که صرف تسلسلی[1] دارند و تعداد تکواژها به کلمات زیاد تغییر می‌کند و از پیش معلوم نیست. این امر Morfessor را از مدل‌های مشابه مثل Linguistica (Goldsmith) که فرض می‌کنند کلمه از یک پایه و احتمالاً تعدادی پسوند و پیشوند تشکیل می‌شود، متمایز می‌سازد.

## 2. MDL[2] چیست؟

از آنجا که روش Morfessor (و کلاً روش‌های یادگیری بی‌سرپرست) در پایه و اساس خود از MDL استفاده می‌کنند، شرح کوتاهی بر آن لازم به نظر می‌رسد. هر مجموعه‌ای از داده‌ها را می‌توان به طورت رشته‌ای از نشانه‌ها از یک الفبای محدود نمایش داد. ایده اصلی پشت MDL این است که هر قاعده‌مندی درون هر مجموعه‌ای از دادگان را می‌توان برای فشرده‌سازی آن دادگان بکار برد یعنی توصیف دادگان با نشانه‌های کمتر. برای انتخاب فرضیه‌ای که بیشترین قاعده‌مندی درون دادگان را بیابد، دانشمندان به دنبال این هستند که کدام فرضیه به بیشترین فشردگی داده منتج می‌شود. برای نیل بدین منظور از یک کد برای فشرده‌سازی دادگان استفاده می‌شود که غالباً یک زبان برنامه نویسی است. یک برنامه به آن زبان نوشته می‌شود که دادگان را تولید کند (یعنی این برنامه نشانگر آن دادگان است)؛ طول کوتاه‌ترین برنامه‌ای که دادگان را تولید می‌کند، پیچیدگی کولموگوروف[3] آن دادگان نام دارد. ایده اصلی تئوری کلی ری سلیمانف[4] در استدلال استقرایی همین می‌باشد.

[1] Concatenative
[2] Minimum Description Length
[3] Kolmogorov
[4] Ray Solomonoff

با این حال این نظریه ریاضی راهی عملی برای انجام استنتاج ارائه نمی‌دهد. دلایل اصلی این امر بدین شرح است:

- پیچیدگی کولموگوروف غیرقابل محاسبه است؛ هیچ الگوریتمی وجود ندارد که ورودی آن دنباله دلخواهی از دادگان بوده و خروجی آن کوتاهترین برنامه‌ای که آن دادگان را تولید کند.
- پیچیدگی کولموگوروف بستگی به این دارد که از چه زبان برنامه نویسی استفاده شود. این امر یک انتخاب دلخواه است ولی به شدت روی مسأله پیچیدگی تأثیر می‌گذارد.

MDL سعی بر این دارد تا با موارد زیر بر این مسأله فائق آید:

- محدود کردن مجموعه کدهای مجاز به نحوی که یافتن کوتاهترین طول کد دادگان امکانپذیر و قابل محاسبه باشد.
- انتخاب کدی که کارا و قابل قبول باشد و به دادگان وابستگی نداشته باشد. این مسأله کمی غامض است و تحقیقات زیادی هنوز روی آن در دست انجام می‌باشد.

به جای برنامه‌ها، در MDL از فرضیات، مدل‌ها یا کدها صحبت می‌شود. مجموعه کدهای مجاز را آنگاه کلاس مدل[5] می‌نامند. سپس کدی انتخاب می‌شود که مجموع توصیف آن کد و توصیف دادگان کمینه باشد.

### 2.1.مثالی از MDL:

سکه‌ای 1000 بار پرتاب می‌شود و تعداد شیرها و خطهای آن یادداشت می‌شود. دو کلاس مدل را در نظر بگیرید:

اولی کدی است که خروجی 0 برای شیرها یا 1 برای خطها را می‌دهد. این کد نشانگر این فرضیه است که سکه سالم و متقارن است. طول کد برای این کد دقیقاً 1000 بیت است.

دومین کلاس مدل شامل همه کدهایی است که برای سکه‌ای که نقص خاصی دارد کارا می‌باشند؛ نشانگر این فرضیه که سکه سالم نیست. فرض کنید 510 شیر و 490 خط می‌بینیم. آنگاه طول کد بنا بر بهترین کد در این کلاس مدل دوم از 1000 بیت کمتر می‌باشد.

بهمین دلیل یک روش آماری ساده ممکن است مدل دوم را بعنوان توصیف بهتری از دادگان در نظر بگیرد ولی MDL یک کد واحد براساس فرضیات می‌سازد به جای اینکه فقط از بهترین کد استفاده کند. برای انجام این امر ساده‌تر است که از یک کد دوقسمتی استفاده کنیم که در آن عناصر کلاس مدلی که بیشترین بازدهی را دارد مشخص شده باشد. سپس دادگان را با این کد مشخص می‌سازیم. بیتهای زیادی لازم است تا مشخص شود از کدام کد باید استفاده کنیم؛ بنابراین طول کد کلی مبتنی بر کلاس مدل دوم ممکن است از 1000 بیت بیشتر شود.

در مورفولوژی یا صرف هم از MDL استفاده می‌شود بدین طریق که دادگان ما یک پیکره خام بدون برچسب است و کد ما همان مدل مورفولوژی می‌باشد که برای توصیف این دادگان بکار می‌رود. یعنی تا آنجا که ممکن است مدلی باید تولید شود که بیشترین قاعده‌مندی در توصیف ساختار درونی کلمات را داشته باشد (و لذا حجم پیکره را کاهش دهد) و

[5] Model Class

کمترین طول قواعد را داشته باشد. یعنی مجموع این دو پارامتر باید کمینه شوند. برای روشن شدن بیشتر مطلب فرض کنید در یک پیکره 1000 بار کلمه دانشجو با صورتهای تصریفی مختلف خود آمده است: دانشجوهایش، دانشجوهای، دانشجویتان و .... حال اگر از قاعده‌مندی موجود در صرف استفاده کنیم لازم است فقط یک مدخل برای دانشجو داشته باشیم و برای مداخل دیگر فقط وندهایی که به پایه چسبیده‌اند را ذخیره کنیم. از طرف دیگر مدل نباید تعداد وندهای زیادی تولید کند که عکس غرض حاصل شود یعنی حجم پیکره بیشتر از نسخه اصلی آن شود.

## 3. فرمول‌های ریاضی

هدف اصلی، یافتن یک مدل زبانی به روش بی‌سرپرست از یک پیکره متنی خام می‌باشد. مدل زبان (M) از یک واژگان مورف و یک گرامر تشکیل می‌شود. هدف یافتن مدل بهینه زبانی برای تقطیع پیکره می‌باشد، یعنی مجموعه‌ای از مورف‌ها که مختصر باشد و یک نمایش مختصر و کوتاه از پیکره بدست دهد. روش پس‌بیشینه[6] یا MAP پارامترهایی را که باید بیشینه شوند ارزیابی می‌کند:

$$\arg\max_{M} P(M|corpus) = \arg\max_{M} P(corpus|M).P(M) \quad (1)$$

که

$$P(M) = P(lexicon, grammar) \quad (2)$$

همانطور که دیده می‌شود، MAP از دو قسمت تشکیل شده است: احتمال مدل زبانی P(M) و بیشترین احتمال[7] (ML) پیکره به شرط مدل زبانی که بصورت P(corpus|M) نوشته شده است. احتمال مدل زبانی، احتمال مشترک واژگان و گرامر است. این فرمول یک احتمال بیزی است یعنی استفاده از احتمالات برای نشان دادن درجه اعتقاد قبلی به جای شمارش بسامد نسبی وقوع رخداد در آزمایش واقعی. در ادامه، اجزای مدل Morfessor با جزئیات بیشتری و با شرح مفاهیم واژگان، گرامر و پیکره توضیح داده خواهد شد.

## 4. واژگان[8]

واژگان شامل یک مدخل برای هر مورف در پیکره تقطیع شده می‌باشد. منظور از اصطلاح واژگان، محلی است برای ذخیره هر گونه اطلاعاتی که شخصی بخواهد بعنوان مجموعه‌ای از مورف‌ها و روابط بین آنها.

فرض کنید که واژگان از M مورف متمایز تشکیل شده است. احتمال رسیدن به یک مجموعه از M مورف $\mu_1, \ldots, \mu_M$ که واژگان را بسازند بدین ترتیب است:

$$P(lexicon) = M!.P(properties(\mu_1),\ldots, properties(\mu_M)). \quad (3)$$

این فرمول بیانگر این احتمال مشترک است که مجموعه‌ای از مورف‌ها که هر کدام مجموعه خاصی از ویژگی‌ها را دارند ایجاد شوند. عامل M! بدین طریق توجیه می‌شود که M! ترتیب ممکن برای مجموعه‌ای از M عنصر موجود است و واژگان مستقل از ترتیب ایجاد مورف‌ها می‌باشد.

[6] Maximum A Posteriori
[7] Maximum Likelihood
[8] Lexicon

در مدل پایه Morfessor تنها ویژگی‌هایی که برای هر مورف در واژگان ذخیره می‌شود، بسامد مورف در پیکره و رشته حروف تشکیل دهنده آن مورف می‌باشد. نمایش رشته حروف طبیعتاً شامل دانشی از طول آن مورف است یعنی تعداد حروف تشکیل دهنده آن.

فرض می‌شود که بسامد و حروف تشکیل‌دهنده از یکدیگر مستقل هستند، لذا داریم:

$$P(properties(\mu_1),\ldots, properties(\mu_M)) = P(f_{\mu 1}, \ldots, f_{\mu M}).P(s_{\mu 1}, \ldots, s_{\mu M}) \quad (4)$$

که f نشانگر بسامد مورف و s نشانگر رشته مورف می‌باشد.

**5. گرامر**

گرامر شامل اطلاعاتی است که نشان می‌دهد چگونه واحدهای زبان با یکدیگر ترکیب می‌شوند. در نسخه‌های کاملتر Morfessor از یک مورفوتاکتیک[9] ساده (نحو داخل کلمات) استفاده شده است. در مدل پایه، هیچ وابستگی به متنی[10] وجود ندارد و لذا گرامری هم موضوعیت ندارد. احتمال P(lexicon, grammar) در معادله 2 به P(lexicon) کاهش می‌یابد. عدم وجود گرامر بدین معناست که مهم نیست چه مورف‌هایی بعد یا قبل یک مورف می‌آیند یا یک مورف در ابتدا، وسط یا انتهای کلمه قرار دارد.

احتمال یک مورف $\mu_i$ یک تخمین بیشینه است. مساوی بسامد مورف در پیکره، $f_{\mu i}$ تقسیم بر تعداد کل توکن[11]های مورف، N می‌باشد. مقدار N برابر مجموع بسامد هر کدام از M نوع مورف است:

$$P(\mu_i) = f_{\mu i} / N = f_{\mu i} / \sum^{M}_{j=1} f_{\mu j} \quad (5)$$

**6. پیکره**

هرشکل کلمه در پیکره را می‌توان به صورت دنباله‌ای از مورف‌ها که در واژگان موجود می‌باشند نشان داد. معمولاً چندین تقطیع برای هر کلمه وجود دارد. در مدل MAP تقطیعی که احتمال بیشتری دارد انتخاب می‌شود. احتمال پیکره به شرط یک مدل خاص زبانی و تقطیع مورف به شکل زیر است:

$$P(corpus|M) = \Pi^{W}_{j=1}\Pi^{n_j}_{k=1}P(\mu_{jk}) \quad (6)$$

ضرب روی W کلمه درون پیکره انجام می‌شود (token count) که هر کدام به $n_j$ مورف شکسته می‌شوند. k امین مورف در j امین کلمه، $\mu_{jk}$ دارای احتمال $P(\mu_{jk})$ می‌باشد که از روی معادله 5 محاسبه می‌شود.

**7. الگوریتم جستجو**

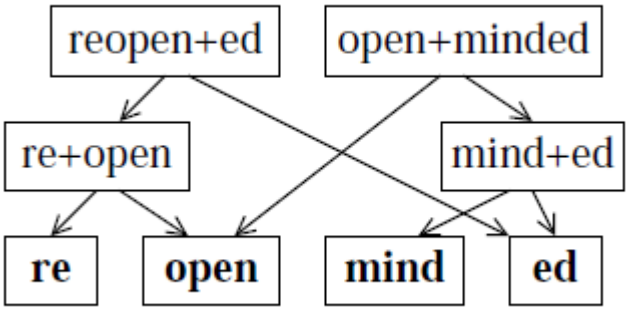
[9] Morphotactics
[10] Context-Sensitivity
[11] Token

برای یافتن واژگان مورف و تقطیع بهینه، از یک الگورتیم جستجوی حریصانه[12] استفاده می‌شود. ابتدا هر کلمه درون پیکره خودش یک مورف است. تقطیع‌های مختلفی امتحان می‌شوند و تقطیعی که بیشترین احتمال را داشته باشد انتخاب می‌شود. این روند با اصلاح تقطیع‌ها پیش می‌رود تا جاییکه هیچ پیشرفت قابل ملاحظه‌ای بدست نیاید.

به جای محاسبه احتمالات، از منفی لگاریتم احتمالات (logprobs) استفاده می‌شود و همه ضرب‌ها با جمع جایگزین می‌شوند. logprob های منفی را می‌توان بعنوان طول کد[13] در چارچوب MDL درنظر گرفت. طول کد مشاهده‌ای مثل x بدین ترتیب با احتمال x مرتبط است:

$$L(x) = -\log P(x)$$

الگوریتم جستجو از یک ساختمان داده استفاده می‌کند به نحوی که هر شکل کلمه متمایز در پیکره درخت تجزیه دودویی خود را دارد. در شکل زیر درخت‌های تجزیه فرضی کلمات انگلیسی reopened و openminded نشان داده شده است. گره‌های برگ این ساختار تجزیه ناپذیرند و نشانگر مورف‌هایی هستند که در واژگان مورف آمده است. برگها تنها گره‌هایی هستند که بر طول کد کلی مدل تأثیرگذارند، چرا که گره‌های سطوح بالاتر فقط در جستجو مورد استفاده قرار می‌گیرند.




به هر گره یک بسامد نسبت داده می‌شود که نمایانگر تعداد دفعات وقوع آن در پیکره می‌باشد. تعداد وقوع یک گره همیشه با حاصلجمع تعداد پدرانش برابر می‌باشد. مثلاً در شکل بالا تعداد مورف open برابر جمع تعداد reopen و openminded است.

در طی فرایند جستجو، اصلاحات در تقطیع جاری مورف براساس عملیات resplitenode انجام می‌شود که شبه-کد آن در الگوریتم 1 آمده است:

**Algorithm 1 resplitnode**(*node*)
**Require:** *node* corresponds to an entire word or a substring of a word
// REMOVE THE CURRENT REPRESENTATION OF THE NODE //
**if** *node* is present in the data structure **then**
**for all** nodes m in subtree rooted at *node* **do**
decrease count(m) by count(*node*)
**if** m is a leaf node, i.e., a morph **then**
decrease L(*corpus* jM) and L($f_{-1}$; : : : ; $f_{M}$) accordingly

[12] Greedy Search
[13] Code Length

```
                    if count(m) = 0 then
                        remove m from the data structure
                        subtract contribution of m from L(s_1; : : : ; s_M) if m is a
                        leaf node
    // FIRST, TRY WITH THE NODE AS A MORPH OF ITS OWN //
    restore node with count(node) into the data structure as a leaf node
    increase L(corpus jM) and L(f_1; : : : ; f_M) accordingly
    add contribution of node to L(s_1; : : : ; s_M)
    bestSolution [L(Mj corpus); node]
    // THEN TRY EVERY SPLIT OF THE NODE INTO TWO SUBSTRINGS //
    subtract contribution of node from L(Mj corpus), but leave node in data
    structure
    store current L(Mj corpus) and data structure
    for all substrings pre and suf such that pre _ suf = node do
        for subnode in [pre; suf] do
            if subnode is present in the data structure then
                for all nodes m in the subtree rooted at subnode do
                    increase count(m) by count(node)
                    increase L(corpus jM) and L(f_1; : : : ; f_M) if m is a
                    leaf node
            else
                add subnode with count(node) into the data structure
                increase L(corpus jM) and L(f_1; : : : ; f_M) accordingly
                add contribution of subnode to L(s_1; : : : ; s_M)
        if L(Mj corpus) < code length stored in bestSolution then
        bestSolution [L(Mj corpus); pre; suf]
        restore stored data structure and L(Mj corpus)
    // SELECT THE BEST SPLIT OR NO SPLIT //
    select the split (or no split) yielding bestSolution
    update the data structure and L(Mj corpus) accordingly
    if a split was selected, such that pre _ suf = node then
        mark node as a parent node of pre and suf
        // PROCEED BY SPLITTING RECURSIVELY //
        resplitnode(pre)
        resplitnode(suf)
```

در جستجو، تمام شکل کلمات متمایز در پیکره به ترتیب تصادفی مرتب می‌شوند و در هر کلمه به نوبه خود به تابع resplitenode داده می‌شود که یک درخت تجزیه دودویی برای آن کلمه ایجاد می‌کند. ابتدا کل کلمه بعنوان یک مورف به واژگان اضافه می‌شود. تجزیه‌ای که کمترین طول کد را دارد انتخاب می‌شود. در صورت تجزیه، تجزیه دو قسمت آن به طور بازگشتی ادامه می‌یابد و وقتیکه هیچ بهره‌ای روی طول کد کلی حاصل نشود متوقف می‌شود. بعد از اینکه همه کلمات یکبار پردازش شدند، بار دیگر به صور تصادفی مرتب می‌شوند و هر کلمه دوباره توسط resplitenode پردازش می‌شود. این فرایند تا جایی ادامه می‌یابد که طول کد کلی مدل و پیکره از مرحله‌ای به مرحله بعد کاهش چشمگیری نداشته باشد.

هر کلمه در هر مرحله یکبار پردازش می‌شود، ولی به خاطر ترتیب تصادفی کلمات، ترتیب پردازش کلمات از هر مرحله به مرحله بعد تفاوت می‌کند. این امکان وجود دارد که از یک روش تعیین‌پذیر[14] استفاده کرد به نحوی که همه کلمات به ترتیب خاصی پردازش شوند ولی روش تصادفی مرجح است، چرا که روش‌های تعیین‌پذیر ممکن است نتایج بایاس شده‌ای داشته باشند.

طبیعت تصادفی الگوریتم متضمن این است که خروجی‌ها وابسته به یک سری اعداد تصادفی باشند که با مولدهای تصادفی تولید شده اند.

**8. نتایج بدست آمده**

با استفاده از برنامه Morfessor که به زبان پرل نوشته شده است، دو پیکره Flexicon و Bijankhan مورد آزمایش قرار گرفتند. برای مقایسه، برنامه Pars Morph که خودمان نوشته‌ایم و مبتنی بر قاعده می‌باشد نیز روی دادگان Flexicon مورد آزمایش قرار گرفت. مدت زمان اجرای برنامه بی‌سرپرست بسیار کمتر و حدود یک دهم برنامه مبتنی بر قاعده بود ولی دقت برنامه مبتنی بر قاعده بسیار بالاتر است. چون به هرحال روش بی‌سرپرست یک روش آماری است و لزوماً همیشه جواب صحیح نمی‌دهد. در بررسی‌هایی که روی خروجی برنامه و گزارش‌های خود تولیدکنندگان Morfessor هم انجام گردید مشخص شد که این امر برای زبانهای دیگر از جمله انگلیسی و فنلاندی هم صادق است و مختص به زبان فارسی نمی‌باشد.

علی ای حال اگر بخواهیم مقایسه‌ای اجمالی انجام دهیم، برنامه مبتنی برقاعده حالتهای مختلف زیادی از قبیل تصریف، اشتقاق و ترکیب را در نظر می‌گیرد و برای هر کلمه ممکن است بیش از یک تجزیه بدست آورد، تجزیه‌های بدست آمده هم تا حد امکان مقوله بندی دستوری دارند ولی سرعت اجرای برنامه پایین‌تر است و الگوریتم فقط خاص زبان فارسی است. در طرف مقابل برنامه Morfessor که به روش بی‌سرپرست عمل می‌کند، دارای سرعت خیلی زیادی است، به پایگاه داده نیازی ندارد و برای همه زبان‌ها عمل می‌کند، ولی خروجی آن دارای مقوله نحوی و دستوری نیست و اشتباهات آن بیشتر از روش مبتنی بر قاعده است.

برای مقایسه، قسمتی از خروجی بدست آمده توسط دو روش روی واژگان Flexicon را در ادامه می‌آوریم. خروجی کامل برنامه‌ها پیوست این مستند می‌باشد. این خروجی‌های مثال برای کلمات زیر بدست آمده است:
آباد، ابد، ابعاد، ابدا، آبادان، ابدأ، آبدان، آبادانی، آباده، آبادگر، آبادگری، آبادی، ابدی، ابعادی، ابدیت، ابدالدهر، ابدالآباد، آبادسازی.

الف. خروجی برنامه Morfessor:

1 آباد
1 ا + بد
1 اب + عاد
1 ا + بد + ا
1 آباد + ان
1 ا + بد + أ
1 اب + دان
1 آباد + ان + ی
1 آباد + ه

[14] Deterministic

1 آباد + گر
1 آباد + گر + ی
1 آباد + ی
1 ا + بد + ی
1 اب + عاد + ی
1 ا + بد + ی + ت
1 ا + بد + ال + ده + ر
1 ا + بد + ال + آباد
1 آباد + سازی

## ب. خروجی برنامه Pars Morph:

```
##################################
:inflection
آباد
:derivation
آب + اد
----------
##################################
:inflection
ابد
----------
##################################
:inflection
ابعاد
:compounding
اب + عاد
----------
:inflection
ابعاد
:compounding
اب + عاد
----------
##################################
:inflection
ابدا
:derivation
ابد + ا
----------
##################################
:inflection
آبادان
:derivation
آب + اد + ان
----------
:inflection
آبادان
:compounding
آباد + ان
----------
##################################
:inflection
ابداً
----------
##################################
:inflection
ابدان
:derivation
اب + دان
----------
:inflection
```

ابد + ان
---
:inflection
ابدان
:compounding
ابد + ان | اب + دان
---
##########
:inflection
آبادانی
:derivation
آب + اد + انی
---
:inflection
آباد + ان + ی
:derivation
آب + اد
---
##########
:inflection
آباده
:derivation
آباد + ه
---
##########
:inflection
آبادگر
:derivation
آب + اد + گر
---
:inflection
آبادگر
:compounding
آباد + گر
---
##########
:inflection
آبادگری
:derivation
آب + اد + گری
---
:inflection
آبادگر + ی
:derivation
آب + اد + گر
---
:inflection
آبادگری
:compounding
آباد + گری
---
##########
:inflection
آبادی
:derivation
آباد + ی
---
:inflection
آباد + ی
:derivation

آب + اد
---
##########
:inflection
ابدی
:derivation
ابد + ی
---
:inflection
ابدی
:compounding
اب + دی
---
##########
:inflection
ابعادی
:derivation
ابعاد + ی
:compounding
اب + عاد
---
:inflection
ابعادی
:compounding
اب + عادی
---
##########
:inflection
ابدیت
:derivation
ابد + یت
---
:inflection
ابد + یت
---
:inflection
ابدیت
:compounding
اب + دیت
---
##########
:inflection
ابدالدهر
:compounding
ابدال + دهر
---
:inflection
ابدالدهر
:compounding
ابدال + دهر
---
##########
:inflection
ابدالآباد
:derivation
ابد + ال + آباد
---
:inflection
ابدالآباد
:compounding

ابدال + آباد
---
##########
:inflection
آبادسازی
:compounding
آباد + سازی
---
:inflection
آبادسازی
:compounding
آباد + سازی
---
##########