

## بازیابی پیشرفته اطلاعات

### فاز اول پروژه

#### اعضای گروه:

حامد علی محمدزاده - ۹۶۱۰۲۰۲۹

حمیدرضا هدایتی - ۹۶۱۰۹۹۳۹

آرمین سعادت بروجنی - ۹۶۱۰۵۸۲۹

## بخش 0. کنسول

ما به طور کلی از کتابخانه `tkinter` پایتون برای رابط کاربری استفاده کردیم (که با `sudo apt-get install python-tk` باید آن را بر روی کامپیوتر خود نصب کنید) و به ازای هر بخش از پروژه، قسمتی در رابط کاربری برای آن بخش مورد نظر اختصاص دادیم، در این اسکرین‌شات شکل کلی رابط کاربری ما را می‌توانید مشاهده کنید:

MIR Project		
Prepare CSV documents		Prepare XML documents
Enter document desc, Hello! etc.	Enter document title, Intro etc.	Add single document
Enter document ID, 32198 etc.		Delete single document
Enter term, Hello etc.		Show posting-list of a term
Enter term, Hello etc.	Enter document ID, 32198 etc.	Show positions of term in document
Enter bigram terms, ba etc.		Show terms fit in this bigram
Without Compression	Save index	Load index
Enter your query, Shakespeare book etc.		Enter your window size, 5 etc.
Correct my query	LNC-LTC search	Proximity search

برای مثال بخش `prepare` کردن اطلاعات در دو دکمه بالا پیاده‌سازی شده، برای تغییر پویای نمایه از `add single document` و `delete single document` و ...

## بخش ۱. پیش پردازش اولیه

ما در xml فرض را بر این گذاشتیم که تعریف یک document یک text و title به ازای هر page در صفحات ویکی‌پدیا است و در csv فرض‌مان این بود که یک title و description در هر سطر تعریف یک document است، و در نهایت هر دو تعریف درون پروژه ما تبدیل به document هایی می‌شود که هر کدام title و description خاص خودشان را دارند. در این پروژه توانایی اضافه کردن csv و xml به هر زبانی دارید و پروژه خود با تشخیص زبان داده import شده آن را پیش‌پردازش می‌کند و نمایه مربوطه را می‌سازد. دو فایل کلی و اسه پیش‌پردازش به هر دو زبان وجود دارد، preprocess\_eng.py و preprocess\_per.py که هر دو در پکیج preprocess قرار گرفته‌اند، و در هر دو تابعی به نام prepare\_text وجود دارد که ورودی آن یک دیتافریم است که فرض شده ستون‌های title و description دارند. برای پیش‌پردازش فارسی از کتابخانه hazm و انگلیسی از کتابخانه nltk استفاده می‌کنیم.

### ۱. جداسازی

از تابع word\_tokenize کتابخانه nltk برای زبان انگلیسی و از تابع word\_tokenize کتابخانه hazm برای زبان فارسی استفاده کردیم که raw را به لیستی از token‌ها تبدیل کنیم.

### ۲. نرمال‌سازی متنی

برای زبان فارسی از تابع normalizer استفاده کردیم و پس از آن جدا سازی را انجام دادیم.

### ۳. حذف علام نگارشی

برای نرمال کردن و حذف token های غیر حرفی از تابع استاندارد پایتون isalpha()) استفاده کردیم و اگر کلمه‌ای isalpha بود آن را نگه داشتیم.

### ۴. بازگرداندن کلمات به ریشه

در زبان فارسی از دو تابع stemmer.stem و lemmatizer.lemmatize استفاده کردیم که به ترتیب stemming و lemmatizing را انجام می‌دهند.

در زبان انگلیسی پس از stem کردن به وسیله snowball.stem اول کلمات را به حالت گذشته آن‌ها برده

word, pos=v lemma.lemmatize و سپس به حالت کنونی می‌آوریم pos=n

### ۵. یافتن و حذف نکات پرتکرار

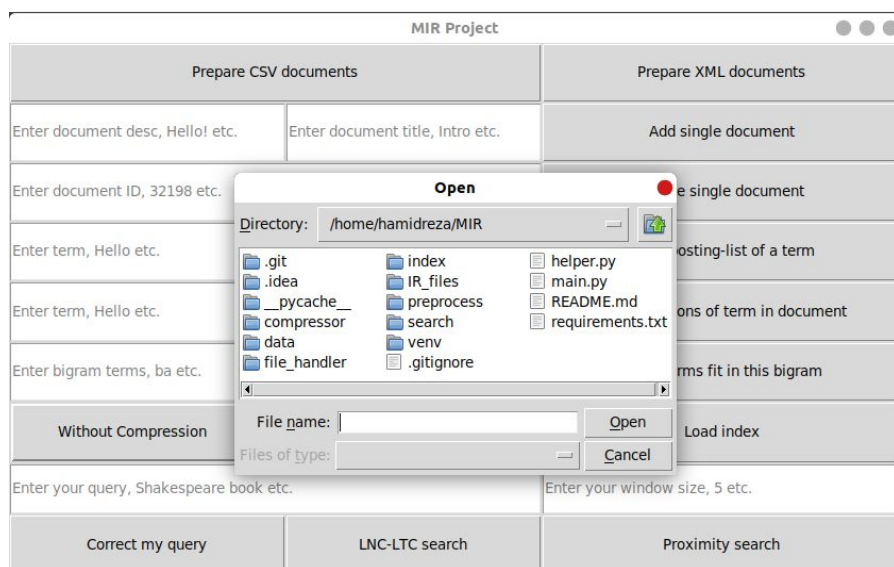
بعد از clean کردن یک raw و جداسازی آن‌ها وقت آن می‌رسد که کلمات پرتکرار را پیدا کرده و آن‌ها را حذف کنیم که در اینجا از تابع پیاده‌سازی شده توسط خودمان به اسم find\_stop\_words استفاده کردیم که کلمات را بر اساس تعداد تکرار آن‌ها در تمامی document ها مرتب می‌کند و به اندازه درصد تعیین شده (stop\_words\_ratio) کلمات پرتکرار را باز می‌گرداند و پس از آن از دیتافریم حذف می‌شود.

## بارمبندی:

### ۱. گرفتن متن از کاربر و نمایش لغات آن بعد از پیش‌پردازش متنی

بعد از انجام مراحل پیش‌پردازش، جدول مربوطه پوشه‌ها را دوباره به کاربر نشان می‌دهیم که مراحل آن به شکل زیر است:

با فشار دکمه prepare csv یا prepare xml صفحه زیر برای انتخاب داده مربوطه نشان داده می‌شود.



و بعد از انتخاب فایل داده، لغات بعد از پیش پردازش نشان داده می‌شود:

MIR Project		Parsed Document	
Prepare CSV documents		Parsed Document	
Enter document desc, Hello! etc.	Index	Title	Description
Enter document ID, 32198 etc.	1	school, kill, creativ	sir, ken, robinson, entertain, profound, r
Enter term, Hello etc.	2	avert, climat, crisi	same, humor, human, exud, inconveni
Enter term, Hello etc.	3	simplic, sell	new, york, time, columnist, david, pogu
Enter bigram terms, ba etc.	4	green, ghetto	emot, charg, activist, majora, carter, de
Without Compression	5	best, stat, ever, see	never, see, data, present, like, drama, u
Enter your query, Shakespeare book etc.	6	whi	toni, robbin, discus, invis, forc, motiv, e
Correct my query	7	let, go, god	when, two, young, mormon, missionar,
	8	behind, design, seattl, librari	architect, joshua, take, audienc, dazzi, c
	9	let, teach, religion, all, religion, school	philosoph, dan, dennett, call, religion, al
	10	life, purpos	pastor, rick, warren, author, life, reflect,
	11	my, wish, call, architectur	accept, ted, prize, cameron, sinclair, de
	12	my, wish, global, day, film	jehan, noujaim, unveil, her, ted, prize, w
	13	my, wish, help, me, stop, pandem	accept, ted, prize, larri, brilliant, smallp
	14	radic, promis, interfac	jeff, han, show, off, cheap, scalabl, com
	15	one, laptop, per, child	nichola, negroport, founder, mit, mediu
	16	magic, violin	violinist, sirena, huang, give, technic, br
	17	improvis, piano, age	pianist, compos, jennif, lin, give, magic,
	18	simpl, design, save, life	fume, indoor, cook, fire, kill, more, than
	19	organ, design, inspir, by, natur	design, ross, lovegrov, expound, philosoc
	20	birth, wikipedia	jimmi, wale, recal, assembl, ragtag, bar
	21	birth, learn, revolut	visionari, richard, baraniuk, explain, visi
	22	nerdcor, comedi	perform, web, toymak, ze, frank, deliv,
	23	meet, founder, blog, revolut	find, mother, blog, revolut, movabl, typ
	24	whi, love, whi, cheat	anthropologist, helen, fisher, take, trick
	25	happi, bodi, soul	eve, ensler, creator, vagina, monologu,
	26	chemic, scum, dream, distant, quasar	legendari, scientist, david, deutsch, put,
	27	whi, univers, seem, so, strang	biologist, richard, dawkin, case, think, ir
	28	freakonom, crack, deal	freakonom, author, steven, levitt, prese
	29	choic, happi, spaghetti, sauc	tip, point, author, malcolm, gladwel, get
	30	surpris, scienc, happi	dan, gilbert, author, stumbl, happi, chall
	31	paradox, choic	psychologist, barri, schwartz, take, aim,
	32	meet, futur, cancer, research	eva, vert, onli, when, give, discus, her, j
	33	roadmap, end, age	cambridg, research, aubrey, de, grey, ar
	34	mobil, phone, fight, poverti	iqbal, quadir, tell, experi, kid, poor, ban
	35	invest, africa, own, solut	jacquelin, novogratz, applaud, heighten
	36	whi, should, invest, free, press	free, press, paper, magazin, radio, tv, bl
	37	rebuild, break, state	ashraf, ghani, passion, power, emphas,
	38	real, futur, space, explor	passion, legendari, spacecraft, design, t
	39	whi, i, ski, north, pole	arctic, explor, ben, saunder, recount, ha
	40	my, wish, three, action, africa	musician, activist, bono, accept, ted, pri
	41	my, wish, manufactur, landscap, green	accept, ted, prize, photograph, edward,
	42	my, wish, three, unusu, medic, invent	accept, ted, prize, inventor, robert, fisch
	43	juri, fool, by, statist	oxford, mathematician, peter, donnelli,

که index، آیدی داکيومنت مورد نظر و title و description اطلاعات تمیز شده مربوط به آن داکيومنت است.

## 2. نمایش متون پرتکرار

همزمان با نشان دادن لغات پرتکرار به همراه تعداد تکرارهای آن‌ها در جدولی جداگانه نشان‌دهنده می‌شود:

The screenshot displays a software interface with several components:

- process\_per.py**: A terminal window showing the execution of a script that processes documents and updates NLTK data.
- MIR Project**: A window with tabs for "Prepare CSV documents" and "Prepare XML documents". It includes input fields for document descriptions, titles, IDs, and terms, along with buttons for adding, deleting, and saving documents.
- Stopwords found (TOP 0.3%)**: A table listing the most frequent words in the documents. The table has columns for the word and its frequency.
- Index**: A window showing a list of documents with their titles and descriptions.
- parsed\_document**: A window showing the parsed content of a document, including a list of words and their frequencies.

به طور مشابه بعد از import کردن داده فارسی:

The screenshot displays a software interface with several components:

- Toplevel**: A window showing the project structure and a list of documents.
- MIR Project**: A window with tabs for "Prepare CSV documents" and "Prepare XML documents". It includes input fields for document descriptions, titles, IDs, and terms, along with buttons for adding, deleting, and saving documents.
- Stopwords found (TOP 0.06%)**: A table listing the most frequent words in the documents. The table has columns for the word and its frequency.
- Index**: A window showing a list of documents with their titles and descriptions.
- parsed\_document**: A window showing the parsed content of a document, including a list of words and their frequencies.

## بخش ۲. نمایه‌سازی

بعد از پیش‌پردازش متن، پوشه‌ها را تک تک به نمایه‌مان اضافه می‌کنیم. به طور کلی **state** کلی برنامه بدین صورت است:

### **:positional**

یک دیکشنری که کلید آن آیدی یک توکن و مقدار آن برابر پوسیتینگ لیست جایگاهی آن توکن است.

### **:bigram**

یک دیکشنری که کلید آن یک دو حرفی است و مقدار آن ترم‌هایی که آن بای‌گرام در آن‌ها مشاهده شده.

### **:all\_tokens**

لیست تمامی توکن‌ها به صورتی که شماره ۱۰ آن برابر با ۱۰مین توکنی‌ست که مشاهده شده.

### **:doc\_is\_available**

لیستی از پوشه‌های به صورتی که خانه شماره ۱۰ آن در صورتی که **true** باشد یعنی داکيومنت شماره ۱۰ حذف نشده است و هنوز وجود دارد.

### **:normalized\_doc**

مربع اندازه بردار هر داکيومنت که در سرچ از آن استفاده می‌شود.

### **:stop\_words**

لیست لغات پرتکرار که در پیش‌پردازش بدست آورده‌ایم.

به جز اطلاعات بالا، داده‌های دیگری نیز که برای افزایش سرعت و راحتی مورد نیاز است وجود دارد که بدین ترتیب‌اند:

### **:Token\_map**

دیکشنری‌ای است که معکوس آرایه قبلی می‌باشد و به این معنی است که با دادن توکن می‌توان به آیدی آن رسید.

از آنجا که **positional** و **bigram** هر دو دیکشنری هستند عملاً از دادساختار **hashmap** برای پیاده‌سازی **index** هایمان استفاده کرده‌ایم، بعد از اضافه شدن هر داکيومنت به نمایه ما پس از تبدیل کردن **token** به **token\_id** برای اضافه کردن یک **position** جدید به نمایه جایگاهی‌مان نخست یک باینری سرچ اجرا کرده و به **document** مربوطه می‌رسیم و پس از آن برای **insert** کردن **position** جدید در لیست مربوطه یک باینری سرچ دیگر را اجرا می‌کنیم و پس از یافتن نقطه مناسب آن **position** را **insert** می‌کنیم، تلاش شده که اضافه کردن یک **document** در بهترین ار در زمانی انجام شود که عملاً پیچیدگی زمانی آن از **order** زیر است:

$$O(\text{number\_of\_tokens\_in\_document} * \log(\text{df}(t)) * \log(\text{cf}(t)))$$

و از آن جا که ما عملاً اضافه کردن یک فایل داده را به اضافه کردن یک مجموعه داده تبدیل می‌کنیم که ما قابلیت چند بار اضافه کردن فایل‌ها و همچنین هر مقدار اضافه کردن تنه‌ای اطلاعات را دارد.

## بارمبندی:

### 1. نمایه‌سازی از روی پوشه‌های در اختیار قرار داده شده

پس از انتخاب فایل برای اضافه کردن به نمایه یا اضافه کردن یک داکيومنت تنها به نمایه، ما استیت برنامه (متغیرهای گفته شده در بالا) را تغییر می‌دهیم، همانطور که گفته شد برای هر دو نمایه **positional** و **bigram** یک باینری سرچ اجرا می‌کنیم که روند اضافه کردن بسیار سریع‌تر شود و پس از هر بار ورودی کاربر نمایه ما آپدیت می‌شود. برای مثال با اضافه کردن یک داکيومنت تنها به نمایه داریم:

hello my dear	letter	Add single document
---------------	--------	---------------------

بعد از فشردن دکمه:

● ● ●

MIR Project		
Prepare CSV documents		Prepare XML documents
hello my dear	letter	Add single document
Enter document ID, 32198 etc.		Delete single document
Enter term, Hello etc.		Show posting-list of a term
Enter term, Hello etc.		Show positions of term in document
Enter bigram terms, ba etc.		Show terms fit in this bigram
Without Compression	Save index	Load index
Enter your query, Shakespeare book etc.		Enter your window size, 5 etc.
Correct my query	LNC-LTC search	Proximity search

**Info**

💡 **Your enter document added with ID 2551**

OK

همچنین برای حذف:

2551	Delete single document
------	------------------------

بعد از فشردن دکمه:

MIR Project

Prepare CSV documents		Prepare XML documents
hello my dear	letter	Add single document
2551		Delete single document
Enter term, Hello etc.		Show posting-list of a term
Enter term, Hello etc.	Enter	Show positions of term in document
Enter bigram terms, ba etc.		Show terms fit in this bigram
Without Compression	Save index	Load index
Enter your query, Shakespeare book etc.		Enter your window size, 5 etc.
Correct my query	LNC-LTC search	Proximity search

**Info**

**Document with ID 2551 removed successfully**

## 2. نمایش posting list کلمه ورودی توسط کاربر

با استفاده از نمایه positional این درخواست را پاسخ می‌دهیم:

ted	Show posting-list of a term
-----	-----------------------------

و بعد از فشردن دکمه:

**Posting list of 'ted'**

Document 11:  
Position 1 of description  
Position 13 of description

Document 12:  
Position 4 of description

Document 13:  
Position 1 of description

Document 37:  
Position 17 of description

Document 40:  
Position 4 of description

Document 41:  
Position 1 of description

Document 42:  
Position 1 of description

Document 64:  
Position 1 of description

Document 65:  
Position 2 of description

Document 66:  
Position 1 of description  
Position 11 of description

Document 83:  
Position 10 of description

Document 95:

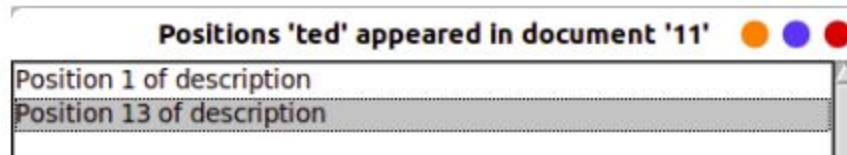


### 3. نمایش جایگاه کلمه وارد شده توسط کاربر در هر سند

به طور مشابه می‌توانیم یک توکن و یک داکيومنت را وارد کنیم و مکان‌های مختلف آن‌ها را مشاهده کنیم:

ted	11	Show positions of term in document
-----	----	------------------------------------

و بعد از فشردن دکمه:

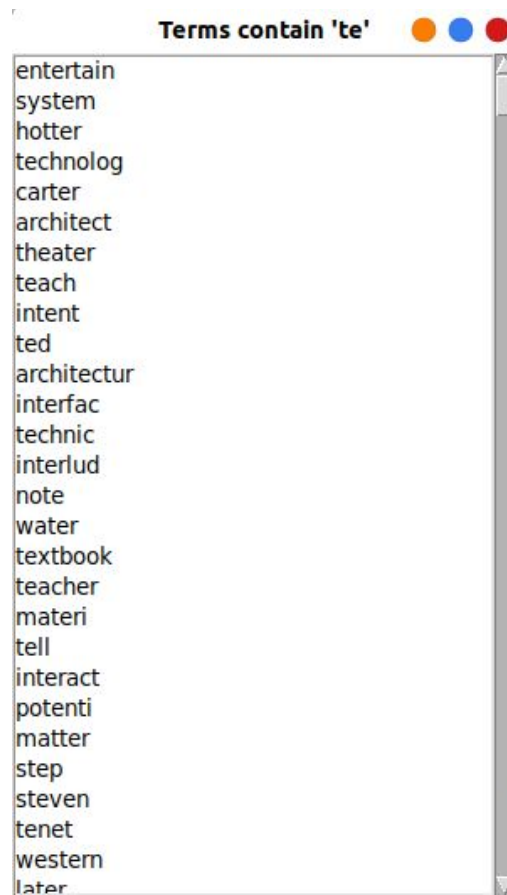


### 4. مشاهده تمام کلماتی که دارای یک bigram خاص درون خود هستند

با استفاده از نمایه bigram که یک دیکشنری بر اساس biwordهاست این درخواست‌ها را پاسخ می‌دهیم: (دقت کنید که برای محاسبه biwordها به شروع و پایان کلمه‌ها \$ اضافه می‌کنیم)

te	Show terms fit in this bigram
----	-------------------------------

و بعد از فشردن دکمه:



### بخش ۳. فشردسازی نمایه‌ها

روش‌های فشردسازی `gamma-code` و `variable-byte` مطابق با اسلایدهای درس پیاده‌سازی شدند. در نهایت ۴ `class` طراحی شد که به شکل زیر است:

```
GammaCodeCompressor
GammaCodeDecompressor
VariableByteCompressor
VariableByteDecompressor
```

که مسئول فشردسازی و استخراج به حالت عادی هستند.

همانطور که در دستور کار پروژه ذکر شده بود، تنها `posting list` ها فشرده شدند و روی دیکشنری فشردسازی صورت نگرفته است.

توابع مربوط به فشردسازی به صورت جنریک پیاده‌سازی شده‌اند به این صورت که هم `positional` و هم `bigram` به هر دو صورت می‌توانند فشرده شوند.

در ضمن در فشردسازی `positional`، هم شناسه داکيومنت‌ها (در واقع اختلافشان) و هم ایندکس تعیین جایگاه (آن هم اختلافشان) فشرده شده است.

علاوه بر آن مطابق اسلایدهای درس، عدد صفر نمایش `gamma-code` ندارد که در این پیاده‌سازی این قضیه نیز مورد بررسی قرار گرفت و عدد صفر نیز ذخیره می‌شود.

همچنین برای اینکه اختلاف اعداد (چه شناسه داکيومنت‌ها چه جایگاه‌ها) ذخیره می‌شود، `posting-list` همواره به صورت `sorted` ذخیره و نگهداری می‌شود تا از این بابت مشکلی نباشد و عملکرد سامانه نیز بهینه باشد.

در ایندکس جایگاهی، `posting list` به صورت زیر است:

```
Posting = [ [doc_number1, [10,20,30]], [doc_number2, [5, 10]], ... ]
```

که نشان می‌دهد هر ترم در چه داکيومنت‌هایی و در چه جایگاهی از آن‌ها آمده‌اند.

برای فشردسازی این نوع `posting`، ابتدا شماره داکيومنت، سپس تعداد دفعات تکرار ترم در آن داکيومنت (که همان طول آرایه جایگاهی است) فشرده می‌شود. پس از آن جایگاه‌ها فشرده می‌شوند. دلیل فشردسازی طول آرایه این است که در زمان `decode` معلوم شود چه زمان به سراغ داکيومنت بعدی باید رفت.

در ایندکس بایگرم، `posting list` به صورت زیر است:

```
Posting = [doc_number1, doc_number2, ...]
```

که صرفاً نشان می‌دهد هر ترم در چه داکيومنت‌هایی ظاهر شده است. برای فشردسازی اختلاف این اعداد که سورت شده هستند ذخیره می‌شود.

برای ذخیره‌سازی اطلاعات از `json` استفاده شد. اطلاعاتی مانند `stop_words` یا `all_token` که هر عدد را به یک `term` مپ کرده است بدون فشردسازی در فایل جدا ذخیره شدند.

برای ذخیره‌سازی `positional` و `bigram` ابتدا مطابق با `compress_type` مشخص شده فشردسازی صورت گرفت و نتیجه در فایل جداگانه برای هر کدام ذخیره شد.

همچنین پس از ذخیره، حجم فایل ذخیره شده محاسبه شده است.

برای خواندن از فایل نیز از `json` استفاده شد. پس از خواندن اطلاعات ذخیره شده، بعضی اطلاعات مانند

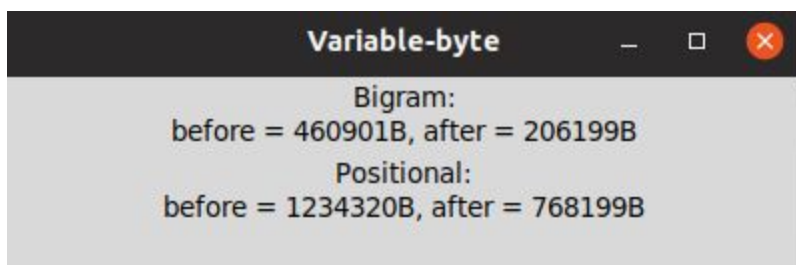
`token_map` تولید می‌شود. `Token_map` یک داده ساختار است که هر `term` را به یک عدد `map` می‌کند و از روی `all_token` که ذخیره شده بود به دست می‌آید. سامانه به هر دوی این داده ساختارها نیاز دارد اما چون از روی هم به دست می‌آیند، تنها یکی از آن‌ها ذخیره می‌شود و دیگری به دست می‌آید تا سربار خواندن و نوشتن از فایل بهینه شود.

در زمان خواندن همچنین `positional` و `bigram` به توابع `decompressor` داده شده و از حالت فشرده خارج می‌شوند. و در نهایت ایندکس عادی به سامانه تحویل داده می‌شود.

## بارمبندی:

### 1. پیاده‌سازی و نمایش میزان حافظه اشغال شده قبل و بعد از اعمال variable-byte

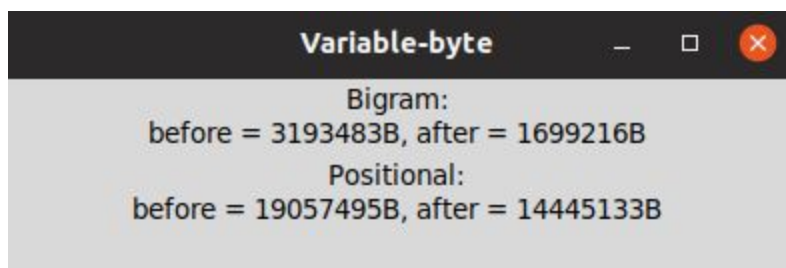
فشرده‌سازی روی داده‌های CSV:



مشاهده می‌شود که bigram از حدود 460KB به حدود 206KB کاهش یافته که بیش از ۵۰ درصد فشرده شده است.

مشاهده می‌شود که positional از حدود 1.2MB به حدود 0.7MB کاهش یافته که حدود ۴۰ درصد فشرده شده است.

فشرده‌سازی روی داده‌های xml:

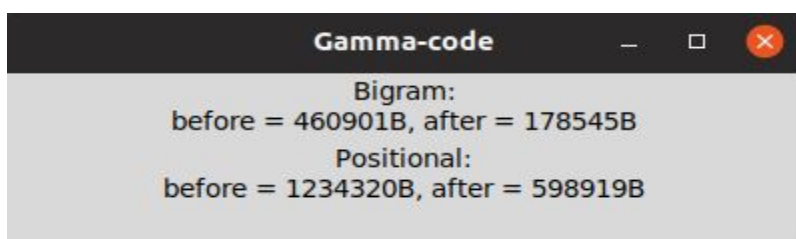


مشاهده می‌شود که bigram از حدود 3.2MB به حدود 1.7MB کاهش یافته که حدود ۵۰ درصد فشرده شده است.

مشاهده می‌شود که positional از حدود 19MB به حدود 14.5MB کاهش یافته که حدود ۲۵ درصد فشرده شده است.

### 2. پیاده‌سازی و نمایش میزان حافظه اشغال شده قبل و بعد از اعمال gamma-code

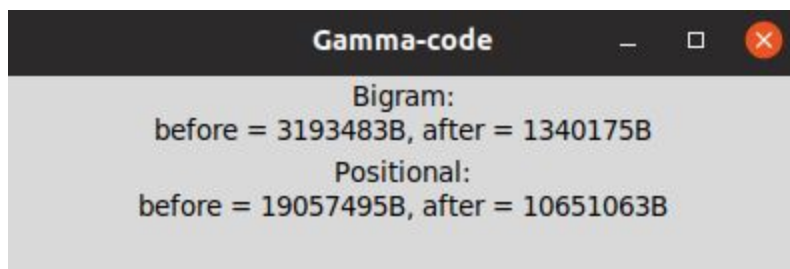
فشرده‌سازی روی داده‌های CSV:



مشاهده می‌شود که bigram از حدود 460KB به حدود 178KB کاهش یافته که بیش از ۶۰ درصد فشرده شده است.

مشاهده می‌شود که positional از حدود 1.2MB به حدود 0.6MB کاهش یافته که حدود ۵۰ درصد فشرده شده است.

فشرده‌سازی روی داده‌های xml:

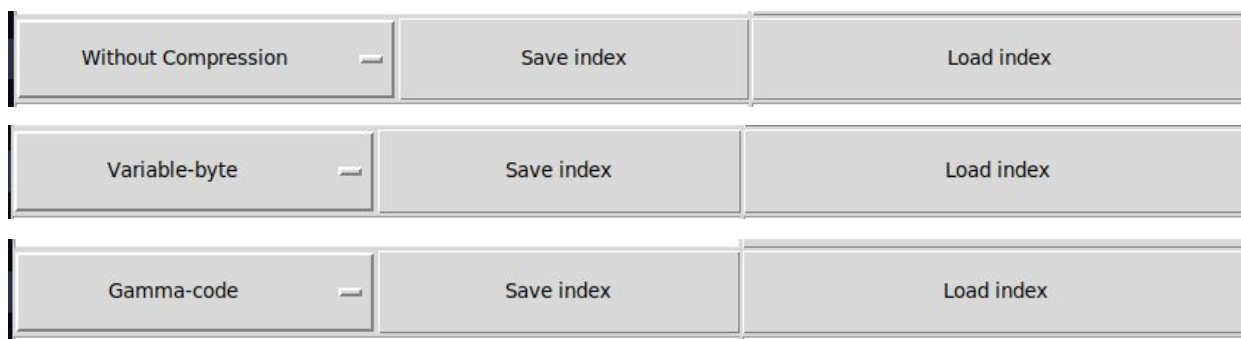


مشاهده می‌شود که bigram از حدود 3.2MB به حدود 1.3MB کاهش یافته که حدود ۶۰ درصد فشرده شده است.

مشاهده می‌شود که positional از حدود 19MB به حدود 10.5MB کاهش یافته که حدود ۴۵ درصد فشرده شده است.

### 3. ذخیره‌سازی نمایه‌ها در فایل و بارگذاری از آن

پس از انتخاب شیوه فشرده‌سازی، برای ذخیره کافی است دکمه **save** و برای بارگذاری دکمه **load** زده شود. پس از **save** حجم فایل فشرده شده و فایل اولیه نمایش داده می‌شود. پس از **load** داده‌ها در سامانه بارگذاری شده و سامانه آماده کار است.



همانطور که مشاهده می‌شود، ۳ حالت برای فشرده‌سازی وجود دارد:

Without Compression  
Variable-byte  
Gamma-code

دقت شود که حالت فشرده‌سازی یک فایل در هنگام **save** و **load** باید یکسان باشد.

## بخش ۴. اصلاح پرسمان

با استفاده از نمایه bigram می‌خواهیم که ۱۰ (top\_jaccard\_items\_candidate) کاندید نخست فاصله jaccard با term وارد شده را پیدا کنیم، برای انجام اینکار، نخست biwordهای term وارد شده برای اصلاح را بررسی می‌کنیم، به ازای هر کدام از آن‌ها کلماتی که آن biword دارند را بررسی می‌کنیم و تعداد biwordهای مشترک آن کلمه را با کلمه وارد شده محاسبه می‌کنیم و در نهایت jaccard distance را بدست می‌آوریم (که برابر با یک منهای اشتراک مجموعه biwordها تقسیم بر اجتماع مجموعه هر دو است). پس از آن‌هایی که کمترین فاصله را دارند مرتب می‌کنیم (تمامی منطق گفته شده در تابع correct در فایل spell\_checker زده شده) و آن‌ها را برای محاسبه edit distance به تابع edit\_distance ورودی می‌دهیم.

برای محاسبه فاصله ویرایش، الگوریتم dynamic programming نوشته شده که بدین ترتیب تعریف شده:

$$dis[i][j] = \max(i,j) \text{ if } i=0 \text{ or } j=0 \text{ else } \min(dis[i-1][j], dis[i][j-1], dis[i-1][j-1] + 1 \text{ if } a[i] == b[j])$$

که به این معنی است که اگر بخواهیم تا پریفیکس i استرینگ اول را به پریفیکس j استرینگ دوم تبدیل کنیم چقدر هزینه خواهد داشت.

## بارمبندی:

### 1. نمایش پرسمان اصلاح شده

بعد وارد کردن کوئری در فیلد خواهیم داشت:

index	
Correct my query	LNC-LTC search

و بعد از زدن دکمه correct my query داریم:

Corrected query

Original query: index
Cleaned query: index
Corrected query: index
Jaccard distance: 0.5
Edit distance: 1

و در خود فیلد هم کلمه وارد شده تغییر می‌کند که برای search کردن کارمان ساده‌تر شود.

### 2. محاسبه فاصله جاکارد دو کلمه

همانطور که در بالا می‌بینیم فاصله جاکارد کلمه تمیز شده و کلمه اصلاح شده نشان داده شده است.

### 3. محاسبه فاصله ویرایش دو کلمه بدون استفاده از کتابخانه‌های آماده

همانطور که در بالا می‌بینیم فاصله ویرایشی کلمه تمیز شده و کلمه اصلاح شده نشان داده شده است.

## بخش ۵. جستجو و بازیابی اسناد

برای محاسبه وزن‌دهی Inc-ltc روی تمامی داکيومنت‌هایی که term‌های query را دارند حرکت می‌کنیم (در تابع search در فایل LNC\_LTC در پکیج ، وزن یک term در کوئری برابر می‌شود با:

$$df = \text{len}(\text{index.positional}[\text{token\_id}])$$

$$\text{weight\_term\_query} = (1 + \log(\text{count})) * \log(N/df) / \sqrt{\text{normalize\_query}}$$

و به ازای هر document برای محاسبه وزن term در document داریم:

$$\text{weight\_term\_doc} = (1 + \log(\text{tf\_document}) * 1 / \sqrt{\text{index.normalize\_doc}[\text{document}]})$$

سپس با ضرب داخلی این دو عدد در یکدیگر و مرتب کردن داکيومنت‌ها نتایج سرچ را به کاربر نشان می‌دهیم. آرایه normalize\_doc آرایه‌ای است که موقع اضافه کردن یک داکيومنت به نمایه محاسبه می‌شود و ذخیره می‌شود و در جستجوی مان از آن استفاده می‌کنیم.

برای جستجوی proximity نیز با توجه به سایز پنجره وارد شده، بر روی تمامی داکيومنت‌هایی که term‌های query را شامل می‌شوند حرکت می‌کنیم و چک می‌کنیم که آیا در description یا title آن document تمامی query‌ها در یک بازه به اندازه پنجره وجود دارند یا نه و لیستی از document‌های مناسب را پیدا می‌کنیم (در تابع search در proximity در پکیج search) و سپس تمامی آن‌ها را به جستجوی Inc\_ltc پاس می‌دهیم و نتایج را به کاربر نشان می‌دهیم.

**بارمندی:**

### 1. نمایش لیست اسناد مرتبط به ترتیب شباهت در جستجو ترتیب‌دار در فضای برداری tf-idf به روش Inc-ltc

با وارد کردن کوئری (و قبل از آن correct کردن آن) می‌توانیم بوسیله ویژگی Inc-ltc پیاده‌سازی شده نتایج سرچمان را مشاهده کنیم:

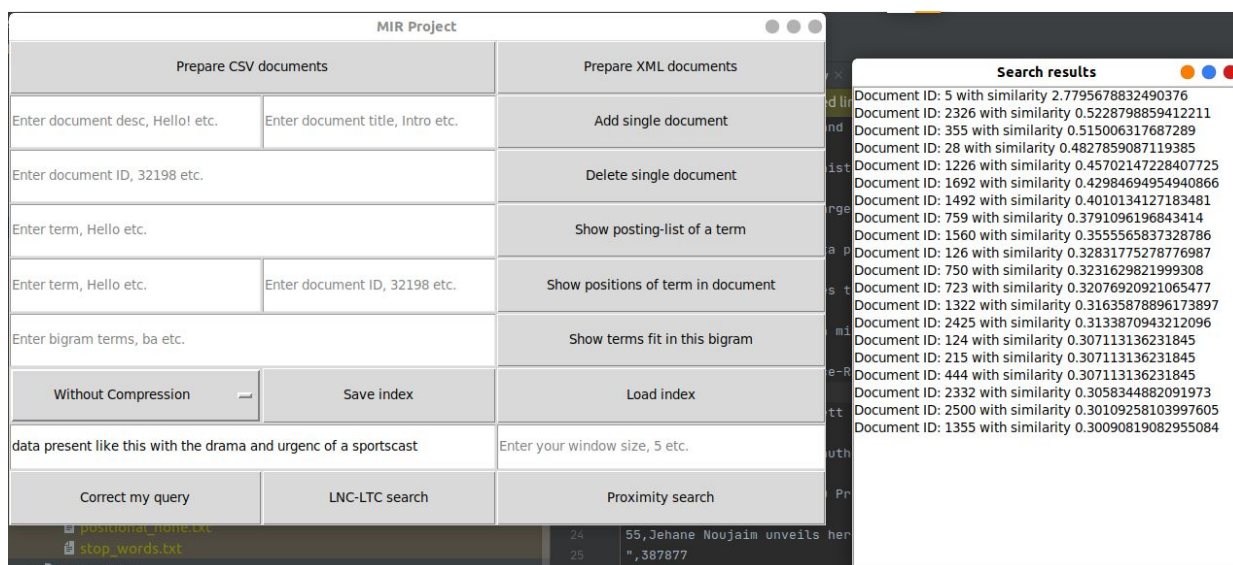
data presented like this. With the drama and urgency of a sportscaster		Enter your window size, 5 etc.
Correct my query	LNC-LTC search	Proximity search

و بعد از اصلاح کردن پرسمان داریم:

The screenshot shows the MIR Project application window. A dialog box titled "Corrected query" is open, displaying the original query, cleaned query, Jaccard distance (0), and edit distance (0). The main window contains several input fields and buttons for query correction, search, and index management.

Corrected query		are XML documents
Original query: data presented like this. With the drama and urgency of a sportscaster		
Cleaned query: data present like this with the drama and urgenc of a sportscast		
Enter	Corrected query: data present like this with the drama and urgenc of a sportscast	single document
Jaccard distance: 0		
Edit distance: 0		
Enter document ID, 32198 etc.		Delete single document
Enter term, Hello etc.		Show posting-list of a term
Enter term, Hello etc.	Enter document ID, 32198 etc.	Show positions of term in document
Enter bigram terms, ba etc.		Show terms fit in this bigram
Without Compression	Save index	Load index
data present like this with the drama and urgenc of a sportscast		Enter your window size, 5 etc.
Correct my query	LNC-LTC search	Proximity search

و بعد از زدن دکمه جستجوی Inc-ltc:



## 2. نمایش لیست اسناد مطابق با پرسمان و اندازه پنجره ورودی در جستجو proximity

با محدود کردن همان پرسمان در یک پنجره داریم:

data present like this with the drama and urgenc of a sportscast	14
Correct my query	Proximity search

که نتایج آن بدین شکل است:

