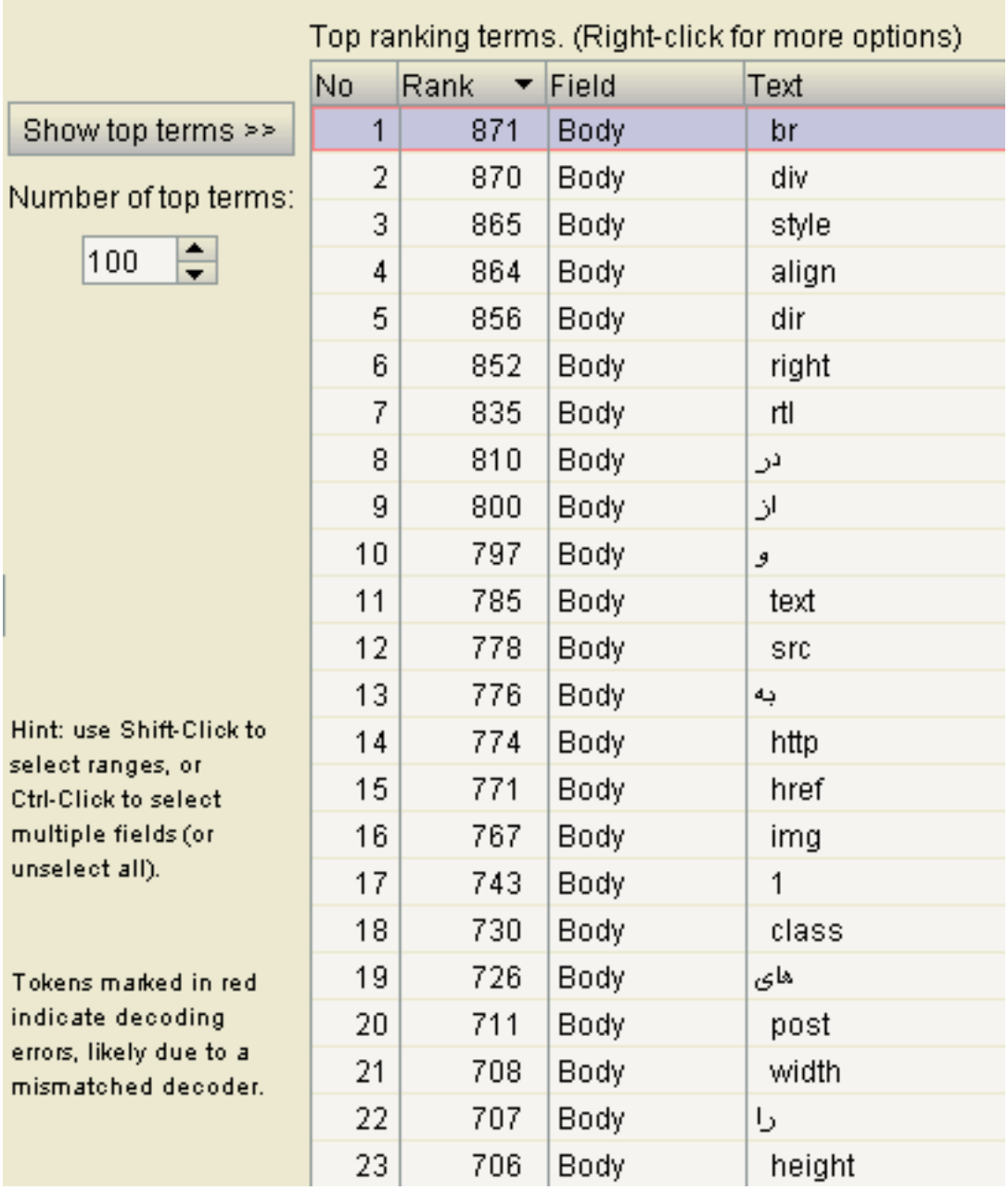
عنوان: **استفاده از Luke برای بهبود کیفیت جستجوی لوسین**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۵ ۲۳:۱۷
آدرس: www.dotnettips.info
گروه‌ها: Lucene.NET

برای آشنایی مقدماتی با لوسین می‌توان به مقالات فارسی ذیل مراجعه کرد:
راهنمای گام به گام Lucene در c#
کتابخانه جستجوی فارسی با Lucene.net
آشنایی با کتابخانه Lucene.NET – قسمت اول
آشنایی با کتابخانه Lucene.NET – قسمت دوم
معرفی کتابخانه‌ی مشهور و قدرتمند Lucene.net -- قسمت اول

به صورت خلاصه اگر نیاز به جستجوی سریع و پیشرفته‌ای بر روی حجم عظیمی از اطلاعات دارید، روش متداول `select * from table where field like something` توصیه نمی‌شود. بسیار کند است؛ مصرف CPU بالایی دارد. از ایندکس استفاده نمی‌کند. راه حل توصیه شده جهت برخورد با این نوع مسایل استفاده از full text search است. نگارش کامل SQL Server حاوی یک موتور FTS توکار هست . اگر از بانک اطلاعاتی خاصی استفاده می‌کنید که دارای موتور FTS نیست یا ... FTS مخصوص SQL Server به درد کار شما نمی‌خورد یا نیاز به سفارشی سازی دارد (مثلا امکان تعریف stop words فارسی (کلماتی مانند به، از، تا و امثال آن))، از موتور FTS جانبی دیگری به نام لوسین نیز می‌توان استفاده کرد.

در کنار این‌ها ابزاری برای آنالیز و کوئری گرفتن از فایل‌های ایندکس تهیه شده توسط لوسین نیز وجود دارد به نام Luke . برای نمونه اگر بانک اطلاعاتی سایت جاری را با لوسین به نحو متداولی ایندکس کنیم، در صفحه اول این برنامه، top ranking terms آن به شکل زیر ظاهر می‌شود:



No	Rank	Field	Text
1	871	Body	br
2	870	Body	div
3	865	Body	style
4	864	Body	align
5	856	Body	dir
6	852	Body	right
7	835	Body	rtl
8	810	Body	در
9	800	Body	از
10	797	Body	و
11	785	Body	text
12	778	Body	src
13	776	Body	به
14	774	Body	http
15	771	Body	href
16	767	Body	img
17	743	Body	1
18	730	Body	class
19	726	Body	های
20	711	Body	post
21	708	Body	width
22	707	Body	را
23	706	Body	height

در اینجا چون متون تهیه شده از نوع HTML هستند، تگ br در آن‌ها زیاد است و یا یک سری حروف و کلمات فارسی هم در صدر قرار دارند که بهتر است از لیست ایندکس حذف شوند. برای اینکار تنها کافی است یک hash table را به نحو زیر تعریف و به StandardAnalyzer لوسین ارسال کنیم:

```
var stopWords = new Hashtable();
stopWords.Add("br","br");
// ...
var analyzer = new StandardAnalyzer(Version.LUCENE_29, stopWords);
```

یا آقای عرب عامری برای حروف و کلمات فارسی که نباید ایندکس شوند، یک لیست نسبتا جامع را در اینجا تهیه کرده‌اند.
اینبار اگر stop words یاد شده را اعمال و مجددا ایندکس‌ها را تهیه کنیم به خروجی بهتری خواهیم رسید.
در کل حداقل از این لحاظ، لوسین نسبت به FTS توکار SQL Server مناسب‌تر به نظر می‌رسد.

نظرات خوانندگان

نویسنده: شهروز جعفری
تاریخ: ۱۳۹۱/۰۴/۲۶ ۰:۸

سلام
یه سری از این نمونه‌ها از کد زیر استفاده شده

```
////////////// Begin the new section
    QueryParser oParser = new QueryParser("Body", new StandardAnalyzer());
    string sTitle = "", sWriterID = "", finalQuery = "";

    sTitle = " AND (Title:" + titleTerm + ")";

    sWriterID = " AND (WriterID:" + writerID + ")";

    finalQuery = "(" + bodyTerm + sTitle + sWriterID + ")";
    hits = searcher.Search(oParser.Parse(finalQuery));
    ////////////// End of the new section
```

که به نظرم حرفه ای نیست.
در ضمن اگه میشه چند تا سایت برای استفاده از کتابخانه در Entity framework معرفی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۶ ۰:۴۷

کاری به EF نداره. به شکل یک سیستم مستقل بهش نگاه کنید. رکوردها از Db دریافت و به شکل document به لوسین اضافه خواهند شد. در همین حین index هم تشکیل می‌شود.
کوئری‌های آن دقیقا به همین شکلی هست که در بالا اومده و زبان آن SQL نیست .
البته پروژه LINQ به آن هم وجود دارد: (^)

نویسنده: رضا.ب
تاریخ: ۱۳۹۱/۰۴/۲۷ ۱۱:۵۴

برای full text search در زبان فارسی این 150 کلمه که آقا عرب‌عامری زحمت کشیدن اصلا کفایت نمیکند. زبان فارسی و دیکته‌اش به این نحو اصلا مناسب ایندکس کردن نیست و بیشتر نیاز به توسعه دارد.
شاید اگر بشه با ابزارهای توسعه‌ای که روی زبان فارسی کار شده - مثل ویراسباز ^ - خروجی بگیریم برای ایندکس کردن و در نتیجه تحلیل اونا به وسیله همین کتابخونه Luke یا Solr یا Nutch - که در وبلاگ آقای زبردست معرفی شده ^ - آسونتره.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۷ ۱۳:۵

کار آقای عرب عامری فراتر است از 150 کلمه ذکر شده. آنالیز فتحه، کسره، ی فارسی و عربی و غیره هم در آن لحاظ شده.

نویسنده: رضا.ب
تاریخ: ۱۳۹۱/۰۴/۲۸ ۰:۰

جستجو که در سایت همکنون هست، از همین کتابخانه استفاده میکنه یا توسعه داده شده؟
یه سوال شاید بی‌ربط. در آنالیز نتایج شاخص‌های لوسین ضریبی غیر از تعداد دفعات تکرار شاخص مد نظر هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۸ ۰:۲۶

- از کتابخانه اصلی lucene.net استفاده شده.
- توضیحات مفصلش رو می‌تونید اینجا مطالعه کنید.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۱۰ ۱۳:۳۶

مهندس پیشنیاز اجرای Luke در ویندوز 7 چی؟
این آدرس لینک کلی کامپوننت برای دانلود گذاشته

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۰ ۱۳:۳۹

- فایل‌های jar نیاز به موتور اجرایی جاوا (JRE) دارند.
- luke نسخه دات نتی هم دارد.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۱۲ ۱۵:۲۵

واقعا کتابخانه سریعی هستش برای کاری که من داشتم نزدیک به 16 هزار رکود 10 فیلدی رو توی لوسین ایندکس کرد که برای دریافت نتایجش زمان خیلی کمتری رو گرفت.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۱۲ ۱۵:۲۹

ممنون برای من خیلی مورد استفاده بوده

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۱۲ ۱۵:۳۷

در مورد این تصویر کمی توضیح میدین که به کدوم بخش لوسین مربوطه ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۲ ۱۷:۱۱

این عدد درصد نزدیک بودن جواب به جستجوی انجام شده است (رتبه جستجوی لوسین). فرمول محاسبه آن به صورت زیر است:

```
var hits = searcher.Search(query, 10).ScoreDocs;
var scoreNorm = 100.0f / hits.GetMaxScore();
foreach (var scoreDoc in hits)
{
   var resultScore = scoreNorm * scoreDoc.score;
```

```
}
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۲ ۱۷:۲۶

بله. اینقدر باکیفیت است که RavenDB برای سیستم جستجوی خودش از لوسین به صورت توکار استفاده می‌کنه.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۱۳ ۲۳:۵

آیا برای صفحه بندی معادلی برای skip و take وجود داره، با توجه به جستجویی که من انجام دادم (+ و + و + و + و + و + و + و +) نمونه‌ای دال بر وجود یک همچین امکانی ندیدم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۳ ۲۳:۳۵

- سؤال شما مرتبط با بحث Luke نیست.
- و ... نداره. هر بار باید جستجو کنید. بعد روی نتیجه حاصل صفحه بندی رو انجام بدید. یا می‌تونید نتیجه جستجوی خاص انجام شده رو کش کنید تا سربار جستجوی مجدد حذف شود. (هرچند اینقدر سریع است که نیازی به کش نیست)

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۱۴ ۱۹:۴۴

بابت پاسختون ممنونم

عنوان: **استفاده از لوسین برای برجسته سازی عبارت جستجو شده در نتایج حاصل**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۷ ۱۲:۳۸
آدرس: www.dotnettips.info
گروه‌ها: Lucene.NET

قسمت جستجوی سایت جاری رو با استفاده از لوسین بازنویسی کردم. خلاصه‌ای از نحوه انجام این‌کار رو در ادامه ملاحظه خواهید کرد:

1) دریافت کتابخانه‌های لازم

نیاز به کتابخانه‌های Lucene.NET و همچنین Lucene.Net Contrib است که هر دو مورد را به سادگی توسط NuGet می‌توانید دریافت و نصب کنید.
Highlighter استفاده شده، در کتابخانه Lucene.Net Contrib قرار دارد. به همین جهت این مورد را نیز باید جداگانه دریافت کرد.

2) تهیه منبع داده

در اینجا جهت سادگی کار فرض کنید که لیستی از مطالب را به فرمت زیر دراختیار داریم:

```
public class Post
{
    public int Id { set; get; }
    public string Title { set; get; }
    public string Body { set; get; }
}
```

تفاوتی نمی‌کند که از چه منبع داده‌ای استفاده می‌کنید. آیا قرار است یک سری فایل متنی ساده موجود در یک پوشه را ایندکس کنید یا تعدادی رکورد بانک اطلاعاتی؛ از NHibernate استفاده می‌کنید یا از Entity framework و یا از ADO.NET. کتابخانه Lucene مستقل است از منبع داده مورد استفاده و تنها اطلاعاتی با فرمت شیء Document معرفی شده به آن‌را می‌شناسد.

3) تبدیل اطلاعات به فرمت Lucene.NET

همانطور که عنوان شد نیاز است هر رکورد از اطلاعات خود را به شیء Document نگاشت کنیم. نمونه‌ای از اینکار را در متد ذیل مشاهده می‌نمائید:

```
static Document MapPostToDocument(Post post)
{
    var postDocument = new Document();
    postDocument.Add(new Field("Id", post.Id.ToString(), Field.Store.YES, Field.Index.NOT_ANALYZED));
    postDocument.Add(new Field("Title", post.Title, Field.Store.YES, Field.Index.ANALYZED,
Field.TermVector.WITH_POSITIONS_OFFSETS));
    postDocument.Add(new Field("Body", post.Body, Field.Store.YES, Field.Index.ANALYZED,
Field.TermVector.WITH_POSITIONS_OFFSETS));
    return postDocument;
}
```

این متد وهله‌ای از شیء Post را دریافت کرده و آن‌را تبدیل به یک سند Lucene می‌کند.
کار با ایجاد یک وهله از شیء Document شروع شده و سپس اطلاعات به صوت فیلدهایی به این سند اضافه می‌شوند.

توضیحات آرگومان‌های مختلف سازنده کلاس Field:
- در ابتدا نام فیلد مورد نظر ذکر می‌گردد.
- سپس مقدار متناظر با آن فیلد، به صورت رشته باید معرفی شود.
- آرگومان سوم آن مشخص می‌کند که اصل اطلاعات نیز علاوه بر ایندکس شدن باید در فایل‌های Lucene ذخیره شوند یا خیر. توسط Field.Store.YES مشخص می‌کنیم که بله؛ علاقمندیم تا اصل اطلاعات نیز از طریق Lucene قابل بازیابی باشند. این مورد جهت نمایش سریع نتایج جستجوها می‌تواند مفید باشد. اگر قرار نیست اطلاعاتی را از این فیلد خاص به کاربر نمایش دهید می‌توانید از گزینه Field.Store.NO استفاده کنید. همچنین امکان فشرده سازی اطلاعات ذخیره شده با انتخاب گزینه

Field.Store.COMPRESS نیز میسر است.
- توسط آرگومان چهارم آن تعیین خواهیم کرد که اطلاعات فیلد مورد نظر ایندکس شوند یا خیر. مقدار Field.Index.NOT_ANALYZED سبب عدم ایندکس شدن فیلد Id می‌شوند (چون قرار نیست روی id در قسمت جستجوی عمومی سایت، جستجویی صورت گیرد). به کمک مقدار Field.Index.ANALYZED، مقدار معرفی شده، ایندکس خواهد شد.
- پارامتر پنجم آن‌را جهت سرعت عمل در نمایان سازی/برجسته کردن و highlighting عبارات جستجو شده در متن‌های یافت شده معرفی کرده‌ایم. الگوریتم‌های متناظر با این روش در فایل‌های Lucene.Net Contrib قرار دارند.

یک نکته

اگر اطلاعاتی را که قرار است ایندکس کنید از نوع HTML می‌باشند، بهتر است تمام تگ‌های آن‌را پیش از افزودن به لوسین حذف کنید. به این ترتیب نتایج جستجوی دقیق‌تری را می‌توان شاهد بود. برای این منظور می‌توان از متد ذیل کمک گرفت:

```
public static string RemoveHtmlTags(string text)
{
    return string.IsNullOrEmpty(text) ? string.Empty : Regex.Replace(text, @"<(.|\n)*?>",
string.Empty);
}
```

4) تهیه Full text index به کمک Lucene.NET

تا اینجا توانستیم اطلاعات خود را به فرمت اسناد لوسین تبدیل کنیم. اکنون ثبت و تبدیل آن‌ها به فایل‌های Full text search لوسین به سادگی زیر است:

```
static readonly Lucene.Net.Util.Version _version = Lucene.Net.Util.Version.LUCENE_29;
public static void CreateIdx(IEnumerable<Post> dataList)
{
    var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
    var analyzer = new StandardAnalyzer(_version);
    using (var writer = new IndexWriter(directory, analyzer, create: true, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
    {
         foreach (var post in dataList)
        {
            writer.AddDocument(MapPostToDocument(post));
        }

        writer.Optimize();
        writer.Commit();
        writer.Close();
        directory.Close();
    }
}
```

ابتدا محل ذخیره سازی فایل‌های full text search مشخص می‌شوند. سپس آنالیز کننده اطلاعات باید معرفی شود. در ادامه به کمک این اطلاعات، شیء IndexWriter ایجاد و مستندات لوسین به آن اضافه می‌شوند. در آخر، این اطلاعات بهینه سازی شده و ثبت نهایی صورت خواهد گرفت.
ذکر version در اینجا ضروری است؛ از این جهت که اگر ایندکسی با فرمت مثلا LUCENE_29 تهیه شود ممکن است با نگارش بعدی این کتابخانه سازگار نباشد و در صورت ارتقاء، نتایج جستجوی انجام شده، کاملا بی‌ربط نمایش داده شوند. با ذکر صریح نگارش، دیگر این اتفاق رخ نخواهد داد.

نکته

StandardAnalyzer توکار لوسین، امکان دریافت لیستی از واژه‌هایی که نباید ایندکس شوند را نیز دارا است. اطلاعات بیشتر در اینجا .

5) به روز رسانی ایندکس‌ها

به کمک سه متد ذیل می‌توان اطلاعات ایندکس‌های موجود را به روز یا حذف کرد:

```
public static void UpdateIndex(Post post)
{
       var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
       var analyzer = new StandardAnalyzer(_version);
       using (var indexWriter = new IndexWriter(directory, analyzer, create: false, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
       {
           var newDoc = MapPostToDocument(post);

            indexWriter.UpdateDocument(new Term("Id", post.Id.ToString()), newDoc);
            indexWriter.Commit();
            indexWriter.Close();
            directory.Close();
        }
}

public static void DeleteIndex(Post post)
{
        var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
        var analyzer = new StandardAnalyzer(_version);
        using (var indexWriter = new IndexWriter(directory, analyzer, create: false, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
        {
            indexWriter.DeleteDocuments(new Term("Id", post.Id.ToString()));
            indexWriter.Commit();
            indexWriter.Close();
            directory.Close();
         }
}

public static void AddIndex(Post post)
{
     var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
     var analyzer = new StandardAnalyzer(_version, getStopWords());
     using (var indexWriter = new IndexWriter(directory, analyzer, create: false, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
     {
          var searchQuery = new TermQuery(new Term("Id", post.Id.ToString()));
          indexWriter.DeleteDocuments(searchQuery);

           var newDoc = MapPostToDocument(post);
           indexWriter.AddDocument(newDoc);
           indexWriter.Commit();
           indexWriter.Close();
           directory.Close();
       }
}
```

تنها نکته مهم این متدها، استفاده از متد IndexWriter با پارامتر create مساوی false است. به این ترتیب فایل‌های موجود بجای از نو ساخته شدن، به روز خواهند شد.
محل فراخوانی این متدها هم می‌تواند در کنار متدهای به روز رسانی اطلاعات اصلی در بانک اطلاعاتی برنامه باشند. اگر رکوردی اضافه یا حذف شده، ایندکس متناظر نیز باید به روز شود.

6) جستجو در اطلاعات ایندکس شده و نمایش آن‌ها به همراه نمایان/برجسته سازی عبارات جستجو شده

قسمت نهایی کار با لوسین و اطلاعات ایندکس‌های تهیه شده، کوئری گرفتن از آن‌ها است. متدهای کامل مورد نیاز را در ذیل مشاهده می‌کنید:

```
public static void Query(string term)
{
     var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
     using (var searcher = new IndexSearcher(directory, readOnly: true))
     {
          var analyzer = new StandardAnalyzer(_version);
          var parser = new MultiFieldQueryParser(_version, new[] { "Body", "Title" }, analyzer);
          var query = parseQuery(term, parser);
          var hits = searcher.Search(query, 10).ScoreDocs;
```

```
        if (hits.Length == 0)
        {
             term = searchByPartialWords(term);
             query = parseQuery(term, parser);
             hits = searcher.Search(query, 10).ScoreDocs;
         }

         FastVectorHighlighter fvHighlighter = new FastVectorHighlighter(true, true);
         foreach (var scoreDoc in hits)
         {
             var doc = searcher.Doc(scoreDoc.doc);
             string bestfragment = fvHighlighter.GetBestFragment(
                             fvHighlighter.GetFieldQuery(query),
                             searcher.GetIndexReader(),
                             docId: scoreDoc.doc,
                             fieldName: "Body",
                             fragCharSize: 400);
              var id = doc.Get("Id");
              var title = doc.Get("Title");
              var score = scoreDoc.score;
              Console.WriteLine(bestfragment);
          }

          searcher.Close();
          directory.Close();
    }
}

private static Query parseQuery(string searchQuery, QueryParser parser)
{
    Query query;
     try
     {
         query = parser.Parse(searchQuery.Trim());
     }
     catch (ParseException)
     {
         query = parser.Parse(QueryParser.Escape(searchQuery.Trim()));
     }
     return query;
}

private static string searchByPartialWords(string bodyTerm)
{
    bodyTerm = bodyTerm.Replace("*", "").Replace("?", "");
    var terms = bodyTerm.Trim().Replace("-", " ").Split(' ')
                            .Where(x => !string.IsNullOrEmpty(x))
                            .Select(x => x.Trim() + "*");
    bodyTerm = string.Join(" ", terms);
    return bodyTerm;
}
```

توضیحات:

اکثر سایت‌ها را که بررسی کنید، جستجوی بر روی یک فیلد را توضیح داده‌اند. در اینجا نحوه جستجو بر روی چند فیلد را به کمک MultiFieldQueryParser ملاحظه می‌کنید.

نکته‌ی مهمی را هم که در اینجا باید به آن دقت داشت، حساس بودن لوسین به کوچکی و بزرگی نام فیلدهای معرفی شده است و در صورت عدم رعایت این مساله، جستجوی شما نتیجه‌ای را دربر نخواهد داشت.

در ادامه برای parse اطلاعات، از متد کمکی parseQuery استفاده شده است. ممکن است به ParseException بخاطر یک سری حروف خاص بکارگرفته شده در عبارات مورد جستجو برسیم. در اینجا می‌توان توسط متد QueryParser.Escape، اطلاعات دریافتی را اصلاح کرد.

سپس نحوه استفاده از کوئری تهیه شده و متد Search را ملاحظه می‌کنید. در اینجا بهتر است تعداد رکوردهای بازگشت داده شده را تعیین کرد (به کمک آرگومان دوم متد جستجو) تا بی‌جهت سرعت عملیات را پایین نیاورده و همچنین مصرف حافظه سیستم را نیز بالا نبریم.

ممکن است تعداد hits یا نتایج حاصل صفر باشد؛ بنابراین بد نیست خودمان دست به کار شده و به کمک متد searchByPartialWords، ورودی کاربر را بر اساس زبان جستجوی ویژه لوسین اندکی بهینه کنیم تا بتوان به نتایج بهتری دست یافت.

در آخر نحوه کار با ScoreDocs یافت شده را ملاحظه می‌کنید. اگر محتوای فیلد را در حین ایندکس سازی ذخیره کرده باشیم، به کمک متد doc.Get می‌توان به اطلاعات کامل آن نیز دست یافت.

همچنین نکته دیگری را که در اینجا می‌توان ملاحظه کرد استفاده از FastVectorHighlighter می‌باشد. به کمک این Highlighter ویژه می‌توان نتایج جستجو را شبیه به نتایج نمایش داده شده توسط موتور جستجوی گوگل درآورد. برای مثال اگر شخصی ef code first را جستجو کرد، توسط متد GetBestFragment، بهترین جزئی که شامل بیشترین تعداد حروف جستجو شده است، یافت گردیده و همچنین به کمک تگ‌های B، ضخیم نمایش داده خواهند شد.

نظرات خوانندگان

نویسنده: Humid
تاریخ: ۱۳۹۱/۰۵/۰۵ ۱:۵۰

سلام و خسته نباشید. من از طریق سایت شما با این کتابخونه آشنا شدم. دارم باهاش کار می‌کنم اما یه مشکلی باهاش دارم. و اون اینه که قبلا توی سرچ من اگر کلمه "ما" رو سرچ می‌کردم 2600 تا نتیجه برمیگردوند اما الان 20 تا. چرا؟
و سوال بعد من اینه که چطور می‌تونم ایندکس کلمه پیدا شده توی متن رو پیدا کنم؟ چون نرم افزار خواسته شده رو می‌خوان مثه نرم افزار نور باشه.
ممنون میشم راهنماییم کنید.
تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۵ ۸:۲۳

- احتمالا به عنوان stopword معرفی شده. این نوع کلمات ایندکس نخواهند شد. کلمه «ما» ارزش جستجو ندارد مانند «از»، «و»، «به»، «تا» و امثال آن.
- در مطلب فوق به قسمت ذیل دقت کنید. این Id همان Id واقعی یک رکورد در دیتابیس است که به عنوان یک سند لوسین ثبت شده:

```
var id = doc.Get("Id");
```

نویسنده: Humid
تاریخ: ۱۳۹۱/۰۵/۰۵ ۱۱:۱۷

ممنون آقای نصیری اما در زبان عربی "ما" یک کلمه تقریبا مهم هست. چطور می‌تونم اون رو از توی stopword خارج کنم.
سوال دیگه اینکه من با سرچ قبلیم که اول یه select میزدم با لایک و رکوردهایی که اون کلمه توش بود و پیدا میکردم و با استفاده از سرچ حرف به حرف در میاوردم کلمه رو. اما الان مثلا اگر "حسین" رو سرچ کنم با لوسین 10 نتیجه و با سرچ قبلی 31 نتیجه میده. چطور می‌تونم نزدیک کنم به نتیجه واقعی؟ البته در سرچ با لوسین از کاراکتر * استفاده هم کردم فرقی نکرد.

چطور می‌تونم در مبحث اِعراب‌های کلمات عربی از لوسین استفاده کنم؟ آیا از زبان عربی و اِعراب گذاری‌ها پشتیبانی می‌شه در این کتابخانه؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۵ ۱۱:۳۴

- 10 نتیجه احتمالا به تنظیم زیر مرتبط است:

```
searcher.Search(query, 10)
```

در اینجا فقط 10 نتیجه بازگشت داده می‌شود (پارامتر دوم ذکر شده).

- در مورد اعراب زبان عربی به صورت پیش فرض خیر. اما اگر به کدهای فوق دقت کرده باشید از یک StandardAnalyzer توکار استفاده شده. این مورد یک سری تنظیمات ابتدایی را به همراه دارد. اگر کارکرد آن مورد قبول شما نیست می‌تونید خودتون یک Analyzer سفارشی رو توسعه بدید.
برای مثال یک نمونه سورس باز رو اینجا می‌تونید پیدا کنید که مباحث اعراب گذاری، ی و ک فارسی و عربی، یک سری stopword

فارسی و مسایل دیگر را هم لحاظ کرده.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۱:۵۸

سلام آقای نصیری،
دو تا سوال داشتم ازتون:
-1 از این توابع مثلاً باید موقع درج مقاله(پست یا ..) در بانک اطلاعاتی برای ایندکس جدید استفاده کرد و موقع ویرایش و حذف مقاله(پست یا ..) هم از توابع معرفی شده متناسب استفاده کرد؟

-2 جستجوی پیشرفته به چه صورت هستش؟مثلاً تاریخ درج مقاله از ... تا ... ،نام نویسنده ،کلمه کلیدی و ... (که هر کدام از این موارد میتونه به صورت اختیاری و یا اجباری باشه).

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۲:۲

- بله. نیاز است مدام این ایندکس را به روز نگه داشت.
- برای این موارد متداول از تاریخ تا تاریخ، از همان SQL معمولی استفاده کنید. هر جایی که امکان تعریف ایندکس و کوئری‌های SQL ایی که از ایندکس استفاده می‌کنند، وجود دارد، روش‌های متداول SQLایی بهینه‌ترین روش‌ها هستند. هدف در اینجا، full text search است بر روی انبوهی text. جستجوی بسیار سریع روی فیلدهای ایندکس نشده حجیم متنی با کیفیتی بالا. این هدف full text search است. چیزی مثل جستجوی گوگل.
در غیر اینصورت نیاز خواهید داشت از عبارات sql به همراه like استفاده کنید که ... بسیار کند هستند؛ چون باید کل جداول و بانک اطلاعاتی را هربار اسکن کنند و در حالت استفاده از like از ایندکس استفاده نمی‌شود.

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۲:۴۱

-منظورتون از "روش‌های متداول SQLایی بهینه‌ترین روش‌ها هستند" چیه؟

-پس در جستجوهای پیشرفته باید از روش معمول(like بدون استفاده از full text search) استفاده کرد و در جستجوهای تک کلمه ای مثل همین سایت باید از lucene(یا full text search) استفاده کرد،درسته؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۶ ۲۳:۹

- اگر کوئری SQL شما از ایندکس استفاده می‌کند نیازی به روش‌های full text search ندارید و موتورهای بانک اطلاعاتی به اندازه کافی برای مدیریت این نوع موارد سریع و بهینه هستند.
- جستجوی این سایت و یا full text search تک کلمه‌ای نیست. می‌تونید جمله هم بنویسید. کلا برای بهبود سرعت، کاهش مصرف CPU و حافظه کوئری‌های SQL ایی که از like استفاده می‌کنند، روش full text search پیشنهاد می‌شود.
استفاده از like در عبارات SQL روش بهینه‌ای نیست چون هربار full table scan صورت می‌گیرد (تصور کنید 100 نفر در حال جستجوی مطالبی در سایت هستند. در این حالت مصرف CPU، استهلاک هارد و مصرف بالای حافظه را درحین اسکن کامل جداول بانک اطلاعاتی درنظر بگیرید)

نویسنده: عرفان
تاریخ: ۱۳۹۱/۰۶/۲۷ ۰:۰

"برای این موارد متداول از تاریخ تا تاریخ، از همان SQL معمولی استفاده کنید "
منظورتون اینه که تو جستجوهای پیشرفته باید از روش معمول و like(یعنی بدون استفاده از full text search و lucene)استفاده بشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۷ ۰:۵

برای جستجوی تاریخ از like استفاده نمی‌شود. like معادل متد الحاقی Contains در LINQ to EF است + توضیح دادم چرا like مناسب نیست.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۷/۱۲ ۱۵:۲۳

البته با تلفیقی از جستجوی لوسین و کوئری روی داده‌های رنج دار مثل **از تاریخ تا تاریخ** میتوان نتیجه ای سریعتری بدست آورد چون میتوان شناسه آبجکت را در لوسین ذخیره کرد و در زمانی که کوئری دریافت شد بقیه کارها را انجام داد یا تاریخ را هم لوسین ذخیره کرد و روی آبجکت‌های گرفته شده کار کرد یا که با توجه به شناسه‌های بدست آمده از بانک کوئری گرفت

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۱/۰۹/۰۷ ۸:۴۵

البته من تست نکردم ولی شاید بتوان با توجه به این که میتوان تاریخ را بصورت عدد ذخیره کرد و نیز با استفاده از عبارات با قاعده تمام وظایف جستجو را به لوسین سپرد.

نویسنده: محسن عباس آبادعربی
تاریخ: ۱۳۹۱/۰۹/۲۰ ۱۷:۴۳

سلام؛
من تازه mvc رو شروع کردم در پروژه ای نیاز به استفاده از لوسین دارم خواهش میکنم یک نمونه برنامه که با استفاده از لوسین می‌باشد را در سایت قرار دهید
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۲۰ ۱۷:۴۶

این دو برچسب را در سایت دنبال کنید:
برچسب مطالب مرتبط با لوسین
برچسب اشتراک‌گذاری‌های مرتبط با لوسین

نویسنده: حسین غلامی
تاریخ: ۱۳۹۱/۱۰/۱۱ ۰:۱۸

سلام؛
در قسمت FastVectorHighlighter متدی با نام () GetIndexReader شناخته شده نیست.
آیا از ورژن لوسین است؟

نویسنده: حسین غلامی
تاریخ: ۱۳۹۱/۱۰/۱۱ ۰:۲۲

در قسمت متد Query اگر ما خواسته باشیم اطلاعات رو در قالب همون شی (مثلا post) برگردونیم ، چطور باید این کار رو انجام داد؟
مثلا قسمت جستجوی همین سایت.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۱ ۰:۲۵

بله. در نگارش جدید یک سری متدهای Get دار به خاصیت تبدیل شدن. مثلا متد GetIndexReader تبدیل شده به خاصیتی به نام IndexReader و مواردی از این دست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۱ ۰:۲۸

سورس قسمتی از جستجوی سایت در دسترس است . ولی در کل مقادیری رو که در ایندکس ذخیره کردید به صورت زیر قابل بازیابی است. پس از آن نگاشت نهایی را خودتان باید انجام دهید.

```
var prop = doc.Get("prop_name");
```

نویسنده: M.Q
تاریخ: ۱۳۹۲/۰۲/۲۱ ۱۶:۴۲

با سلام

من میخوام توی lucene از عبارات منطقی and و or استفاده کنم ولی هرچی تست می‌کنم جواب دقیق و کاملی نمیگیرم (جستجو انجام میشه ولی مثلا از 3 آیتمی که حتما باید برگردونه 2 تاشو بر میگردونه).

آیا عبارات منطقی رو میشه در lucene استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۱ ۱۷:۰

بله. بهترین مرجع برای این مسایل « Apache Lucene - Query Parser Syntax » است و همچنین اینجا برای مثال‌های بیشتر. به علاوه کلاس BooleanQuery هم برای اینکار وجود دارد.

نویسنده: آیمو
تاریخ: ۱۳۹۲/۰۸/۱۵ ۱۸:۱۵

سلام؛ من پروژه لوسین رو که شما ضمیمه کرده بودین توبخش استفاده از AutoComplete Jquery هم دانلود کردم و عین همونا رو پیاده کردم و همه چیز داره خوب کار میکنه. منتها شما اونجا چند تا post رو یک جا به لوسین دادین تا ایندکس کنه و لوسین هم برای همشون یه فایل میسازه . اما من که هر چند وقت یه بار تو سایت یه مطلبو ایندکس میکنم برای هر کدوم یه فایل ساخته و خب اگه تعداد مطلبام زیاد باشه این همینجور برای همشون تو دایرکتوری خودش فایل‌های یک کیلو بایتی میسازه .ِایا این درسته؟ نمیدونم مشکل از کجاست! اگه میشه راهنمایی کنین....

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۵ ۲۰:۴۱

Too many open files

نویسنده: آیمو
تاریخ: ۱۳۹۲/۰۸/۱۶ ۱۲:۱

ممنون به خاطر پاسختون . اون لینکو نگاه کردم و ایتم هایی که گفته بود رو من رعایت کرده بودم . اما همین طور فایل اضاف می‌کرد . بعد اومدم ببینم تا چند تا این همینجوری میخواد فایل بسازه ... یه برنامه ساده نوشتم و توش همون درخواست رو با httpClient بهش میدادم و و اونم می‌ساخت . جالب اینجاست که تا مثلا 30 تا ایتم فایل با پسوندها مختلف می‌ساخت بعد یه فایل Write Lock می‌ساخت .همشونو با هم یکی میکرد با یه اسمه جدید و بعد باز شروع به ایندکس کردن میکرد .من با این برنامه چیزی حدود 10 هزار درخواست رو براش فرستادم و اونم ایندکس کرد مثله برق بدونه اینکه تعداد فایل‌ها از ماکزیمم 50 تا ایتم بیشتر بشه . من بی خود نگران بود . دستتون درد نکنه

نویسنده: جهش
تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۰:۵۰

سلام
از تابع **CreateIdx** کجا باید استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۱:۰

چون یک لیست را قبول می‌کند، نیاز است یکبار برای ایندکس کردن لیستی از اسناد موجود ایندکس نشده اجرا شود (در پنل مدیریتی برنامه مثلا). در سایر موارد (مانند افزوده شدن، به روز رسانی یا حذف یک رکورد) از مورد 5 استفاده کنید.

عنوان: **نحوه استفاده صحیح از لوسین در ASP.NET**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۹ ۹:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, Design patterns, Lucene.NET

بر مبنای پیاده سازی متداولی که در n هزار سایت اینترنتی می‌توان یافت، نحوه کار با جستجوگر لوسین حدودا به این شکل است:

```
var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory + "\\LuceneIndex"));
using (var searcher = new IndexSearcher(directory, readOnly: true))
{
      //do something ...

       searcher.Close();
       directory.Close();
}
```

و ... اینکار به این شکل غلط است!
مطابق مستندات رسمی لوسین، این کتابخانه thread-safe است. به این معنا که در آن واحد چندین و چند کاربر می‌توانند از یک وهله از شیءهای Reader و Searcher استفاده کنند و نباید به ازای هر جستجو، یکبار این اشیاء را ایجاد و تخریب کرد. البته در اینجا تنها یک Writer در آن واحد می‌تواند مشغول به کار باشد.
مشکلاتی که به همراه باز و بسته کردن بیش از حد IndexSearcher وجود دارد، مصرف بالای حافظه است (به ازای هر کاربر مراجعه کننده، یکبار باید ایندکس‌ها در حافظه بارگذاری شوند) و همچنین تاخیر اولیه این بارگذاری و کندی آن‌را نیز باید مدنظر داشت.

نتیجه گیری:
برای کار با جستجوگر لوسین نیاز است از الگوی Singleton استفاده شود و تنها یک وهله از این اشیاء بین تردهای مختلف به اشتراک گذاشته شود.

نظرات خوانندگان

نویسنده: حسین غلامی
تاریخ: ۱۳۹۱/۱۰/۱۱ ۱:۲۹

آقای نصیری میشه نمونه ای رو با استفاده از این الگو ، مثال بزنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۱۱ ۱۰:۹

مراجعه کنید به مثال Auto Complete .

عنوان: **یافتن «مطالب مرتبط» توسط لوسین**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۳۱ ۱۷:۲۵
آدرس: www.dotnettips.info
گروه‌ها: Lucene.NET

موتور لوسین علاوه بر فراهم آوردن امکان جستجوی سریع بر روی متون حجیم، امکان یافتن مطالبی مشابه یا مرتبط با مطلبی خاص را نیز فراهم می‌کند. نمونه آن‌را شاید در بعضی از انجمن‌ها یا وبلاگ‌ها دیده باشید که در ذیل مطلب جاری، چندین لینک را به مطالبی مشابه نیز نمایش می‌دهند. در ادامه نحوه استفاده از این قابلیت را در لوسین بررسی خواهیم کرد.

یافتن شماره سند متناظر لوسین

همان مثال « استفاده از لوسین برای برجسته سازی عبارت جستجو شده در نتایج حاصل » را در نظر بگیرید. در ابتدا نیاز است شماره یک مطلب را تبدیل به شماره سند لوسین کنیم. برای مثال ممکن است Id یک مطلب 1000 باشد، اما شماره سند متناظر آن در لوسین 800 ثبت شده باشد. بنابراین جستجوی ذیل الزامی است:

```
static readonly Lucene.Net.Util.Version _version = Lucene.Net.Util.Version.LUCENE_29;
static readonly IndexSearcher _searcher = new IndexSearcher(@"c:\path\idx", readOnly: true);
private static int GetLuceneDocumentNumber(int postId)
{
    var analyzer = new StandardAnalyzer(_version);
    var parser = new QueryParser(_version, "Id", analyzer);
    var query = parser.Parse(postId.ToString());
    var doc = _searcher.Search(query, 1);
    if (doc.totalHits == 0)
    {
        return 0;
    }
    return doc.scoreDocs[0].doc;
}
```

در اینجا بر اساس شماره یک مطلب، کوئری متناظر با آن تشکیل شده و جستجویی بر روی اسناد ثبت شده در ایندکس‌های لوسین صورت می‌گیرد. اگر اطلاعاتی یافت شد، شماره سند متناظر بازگشت داده می‌شود.
از این جهت به شماره سند یاد شده نیاز داریم که قرار است مطالب مرتبط با کل این سند را بیابیم.

ساختن کوئری‌های MoreLikeThis

امکانات یافتن مطالب مشابه یک مطلب در اسمبلی Lucene.Net.Contrib.Queries.dll قرار دارد. بنابراین در اینجا نیاز به فایل‌های پروژه Lucene.Net Contrib وجود دارد.
پس از یافتن شماره سند متناظر با یک مطلب، اکنون نوبت به ساخت کوئری‌های پیشرفته MoreLikeThis است که نحوه انجام تنظیمات آن‌را در ذیل مشاهده می‌کنید:

```
private static Query CreateMoreLikeThisQuery(int postId)
{
    var docNum = GetLuceneDocumentNumber(postId);
    if (docNum == 0)
            return null;

     var analyzer = new StandardAnalyzer(_version);
     var reader = _searcher.GetIndexReader();

     var moreLikeThis = new MoreLikeThis(reader);
     moreLikeThis.SetAnalyzer(analyzer);
     moreLikeThis.SetFieldNames(new[] { "Title", "Body"});
     moreLikeThis.SetMinDocFreq(1);
     moreLikeThis.SetMinTermFreq(1);
     moreLikeThis.SetBoost(true);

     return moreLikeThis.Like(docNum);
```

```
}
```

در اینجا فیلدهایی که قرار است در جستجو حضور داشته باشند توسط متد SetFieldNames معرفی می‌شوند. توسط متد SetMinDocFreq مشخص می‌کنیم که واژه‌های مشابه و مرتبط باید حداقل در چند سند ظاهر شده باشند. همچنین توسط متد SetMinTermFreq تعیین می‌گردد که یک واژه باید چندبار در این اسناد وجود داشته باشد. متد SetBoost سبب می‌شود که آنالیز بهتری بر اساس رتبه بندی‌های حاصل صورت گیرد.

نمایش مطالب مرتبط توسط کوئری MoreLikeThis

پس از این تنظیمات، متد moreLikeThis.Like، یک شیء Query را در اختیار ما قرار خواهد داد. از اینجای کار به بعد همانند سایر مطالب مشابه است. بر اساس این کوئری، جستجویی صورت گرفته و سپس اطلاعات یافت شده نمایش داده می‌شود:

```
private static void ShowMoreLikeThisPostItems(int postId)
{
     var query = CreateMoreLikeThisQuery(postId);
     if (query == null)
               return;

      var hits = _searcher.Search(query, n: 10);
      foreach (var item in hits.scoreDocs)
      {
           var doc = _searcher.Doc(item.doc);
           var id = doc.Get("Id");
           var title = doc.Get("Title");
           Console.WriteLine(title);
       }
}
```

نظرات خوانندگان

نویسنده: دل محسن
تاریخ: ۱۳۹۳/۰۷/۰۸ ۱۰:۴۸

سلام؛ میشه در Lucene جستجویی شبیه sql انجام داد؟ منظور اینکه همزمان item1 رو داخل field1 و item2 رو در field2 و.... جستجو کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۰۸ ۱۰:۵۳

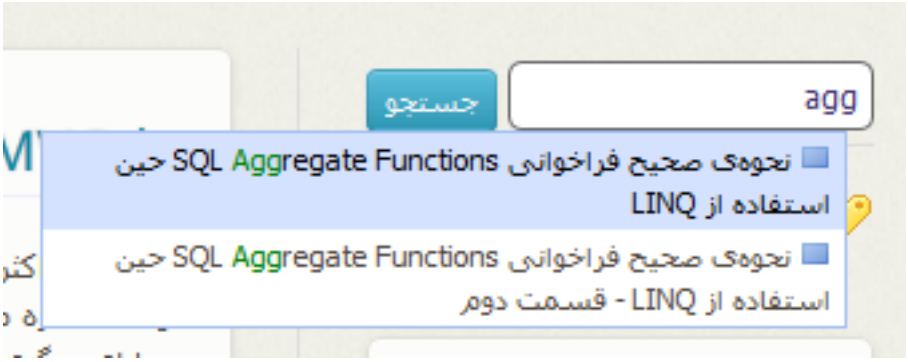
بله. زبان مخصوص خودش را دارد: Query Parser Syntax و Lucene Query Syntax
ضمنا بر همین مبنا LINQ to Lucene هم طراحی شده: ^ و ^

عنوان: **نحوه اضافه کردن Auto-Complete به جستجوی لوسین در ASP.NET MVC و Web forms**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۰۶ ۹:۵۸
آدرس: www.dotnettips.info
گروه‌ها: Lucene.NET, ASP.Net, MVC

پیشنیازها:

چگونه با استفاده از لوسین مطالب را ایندکس کنیم؟
چگونه از افزونه jQuery Auto-Complete استفاده کنیم؟
نحوه استفاده صحیح از لوسین در ASP.NET

اگر به جستجوی سایت دقت کرده باشید، قابلیت ارائه پیشنهاداتی به کاربر توسط یک Auto-Complete به آن اضافه شده‌است. در مطلب جاری به بررسی این مورد به همراه دو مثال Web forms و MVC پرداخته خواهد شد.



قسمت عمده مطلب جاری با پیشنیازهای یاد شده فوق یکی است. در اینجا فقط به ذکر تفاوت‌ها بسنده خواهد شد.

الف) دریافت لوسین

از طریق NuGet آخرین نگارش را دریافت و به پروژه خود اضافه کنید. همچنین Lucene.NET Contrib را نیز به همین نحو دریافت نمائید.

ب) ایجاد ایندکس

کدهای این قسمت با مطلب برجسته سازی قسمت‌های جستجو شده، یکی است:

```
using System.Collections.Generic;
using System.IO;
using Lucene.Net.Analysis.Standard;
using Lucene.Net.Documents;
using Lucene.Net.Index;
using Lucene.Net.Store;
using LuceneSearch.Core.Model;
using LuceneSearch.Core.Utils;

namespace LuceneSearch.Core
{
    public static class CreateIndex
    {
        static readonly Lucene.Net.Util.Version _version = Lucene.Net.Util.Version.LUCENE_30;

        public static Document MapPostToDocument(Post post)
        {
            var postDocument = new Document();
            postDocument.Add(new Field("Id", post.Id.ToString(), Field.Store.YES,
Field.Index.NOT_ANALYZED));
            var titleField = new Field("Title", post.Title, Field.Store.YES, Field.Index.ANALYZED,
Field.TermVector.WITH_POSITIONS_OFFSETS);
```

```
            titleField.Boost = 3;
            postDocument.Add(titleField);
            postDocument.Add(new Field("Body", post.Body.RemoveHtmlTags(), Field.Store.YES,
Field.Index.ANALYZED, Field.TermVector.WITH_POSITIONS_OFFSETS));
            return postDocument;
        }

        public static void CreateFullTextIndex(IEnumerable<Post> dataList, string path)
        {
            var directory = FSDirectory.Open(new DirectoryInfo(path));
            var analyzer = new StandardAnalyzer(_version);
            using (var writer = new IndexWriter(directory, analyzer, create: true, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
            {
                foreach (var post in dataList)
                {
                    writer.AddDocument(MapPostToDocument(post));
                }

                writer.Optimize();
                writer.Commit();
                writer.Close();
                directory.Close();
            }
        }
    }
}
```

تنها تفاوت آن اضافه شدن titleField.Boost = 3 می‌باشد. توسط Boost به لوسین خواهیم گفت که اهمیت عبارات ذکر شده در عناوین مطالب، بیشتر است از اهمیت متون آن‌ها.

ج) تهیه قسمت منبع داده Auto-Complete

```
namespace LuceneSearch.Core.Model
{
    public class SearchResult
    {
        public int Id { set; get; }
        public string Title { set; get; }
    }
}
```

```
using System.Collections.Generic;
using System.IO;
using Lucene.Net.Index;
using Lucene.Net.Search;
using Lucene.Net.Store;
using LuceneSearch.Core.Model;
using LuceneSearch.Core.Utils;

namespace LuceneSearch.Core
{
    public static class AutoComplete
    {
        private static IndexSearcher _searcher;

        /// <summary>
        /// Get terms starting with the given prefix
        /// </summary>
        /// <param name="prefix"></param>
        /// <param name="maxItems"></param>
        /// <returns></returns>
        public static IList<SearchResult> GetTermsScored(string indexPath, string prefix, int maxItems
= 10)
        {
            if (_searcher == null)
                _searcher = new IndexSearcher(FSDirectory.Open(new DirectoryInfo(indexPath)), true);

            var resultsList = new List<SearchResult>();
            if (string.IsNullOrWhiteSpace(prefix))
                return resultsList;
```

```
            prefix = prefix.ApplyCorrectYeKe();

            var results = _searcher.Search(new PrefixQuery(new Term("Title", prefix)), null, maxItems);
            if (results.TotalHits == 0)
            {
                results = _searcher.Search(new PrefixQuery(new Term("Body", prefix)), null, maxItems);
            }

            foreach (var doc in results.ScoreDocs)
            {
                resultsList.Add(new SearchResult
                {
                    Title = _searcher.Doc(doc.Doc).Get("Title"),
                    Id = int.Parse(_searcher.Doc(doc.Doc).Get("Id"))
                });
            }

            return resultsList;
        }
    }
}
```

توضیحات:

برای نمایش Auto-Complete نیاز به منبع داده داریم که نحوه ایجاد آن‌را در کدهای فوق ملاحظه می‌کنید. در اینجا توسط جستجوی سریع لوسین و امکانات PrefixQuery آن، به تعدادی مشخص (maxItems)، رکوردهای یافت شده را بازگشت خواهیم داد. خروجی حاصل لیستی است از SearchResultها شامل عنوان مطلب و Id آن. عنوان را به کاربر نمایش خواهیم داد؛ از Id برای هدایت او به مطلبی مشخص استفاده خواهیم کرد.

د) نمایش Auto-Complete در ASP.NET MVC

```
using System.Text;
using System.Web.Mvc;
using LuceneSearch.Core;
using System.Web;

namespace LuceneSearch.Controllers
{
    public class HomeController : Controller
    {
        static string _indexPath = HttpRuntime.AppDomainAppPath + @"App_Data\idx";

        public ActionResult Index(int? id)
        {
            if (id.HasValue)
            {
                //todo: do something
            }
            return View(); //Show the page
        }

        public virtual ActionResult ScoredTerms(string q)
        {
            if (string.IsNullOrWhiteSpace(q))
                return Content(string.Empty);

            var result = new StringBuilder();
            var items = AutoComplete.GetTermsScored(_indexPath, q);
            foreach (var item in items)
            {
                var postUrl = this.Url.Action(actionName: "Index", controllerName: "Home", routeValues:
new { id = item.Id }, protocol: "http");
                result.AppendLine(item.Title + "|" + postUrl);
            }

            return Content(result.ToString());
        }
    }
}
```

```
@{
```

```
    ViewBag.Title = "جستجو";
    var scoredTermsUrl = Url.Action(actionName: "ScoredTerms", controllerName: "Home");
    var bulletImage = Url.Content("~/Content/Images/bullet_shape.png");
}
<h2>
    جستجو</h2>

<div align="center">
    @Html.TextBox("term", "", htmlAttributes: new { dir = "ltr" })
    <br />
    را وارد نمائید lu جهت آزمایش
</div>

@section scripts
{
    <script type="text/javascript">
        EnableSearchAutocomplete('@scoredTermsUrl', '@bulletImage');
    </script>
}
```

```
function EnableSearchAutocomplete(url, img) {
    var formatItem = function (row) {
        if (!row) return "";
        return "<img src='" + img + "' /> " + row[0];
    }

    $(document).ready(function () {
        $("#term").autocomplete(url, {
            dir: 'rtl', minChars: 2, delay: 5,
            mustMatch: false, max: 20, autoFill: false,
            matchContains: false, scroll: false, width: 300,
            formatItem: formatItem
        }).result(function (evt, row, formatted) {
            if (!row) return;
            window.location = row[1];
        });
    });
}
```

توضیحات:

- ابتدا ارجاعاتی را به jQuery، افزونه Auto-Complete و اسکریپت سفارشی تهیه شده، در فایل layout پروژه تعریف خواهیم کرد. در اینجا سه قسمت را مشاهده می‌کنید: کدهای کنترلر، View متناظر و اسکریپتی که Auto-Complete را فعال خواهد ساخت.
- قسمت مهم کدهای کنترلر، دو سطر زیر هستند:

```
result.AppendLine(item.Title + "|" + postUrl);
return Content(result.ToString());
```

مطابق نیاز افزونه انتخاب شده در مثال جاری، فرمت خروجی مدنظر باید شامل سطرهایی حاوی متن قابل نمایش به همراه یک Id (یا در اینجا یک آدرس مشخص) باشد. البته ذکر این Id اختیاری بوده و در اینجا جهت تکمیل بحث ارائه شده است.
return Content هم سبب بازگشت این اطلاعات به افزونه خواهد شد.

- کدهای View متناظر بسیار ساده هستند. تنها نام TextBox تعریف شده مهم می‌باشد که در متد جاوا اسکریپتی EnableSearchAutocomplete استفاده شده است. به علاوه، نحوه مقدار دهی آدرس دسترسی به اکشن متد ScoredTerms نیز مهم می‌باشد.
- در متد EnableSearchAutocomplete نحوه فراخوانی افزونه autocomplete را ملاحظه می‌کنید.

جهت آن، به راست به چپ تنظیم شده است. با 2 کاراکتر ورودی فعال خواهد شد با وقفه‌ای کوتاه. نیازی نیست تا انتخاب کاربر از لیست ظاهر شده حتما با عبارت جستجو شده صد در صد یکی باشد. حداکثر 20 آیتم در لیست ظاهر خواهند شد. اسکرول بار لیست را حذف کرده‌ایم. عرض آن به 300 تنظیم شده است و نحوه فرمت دهی نمایشی آن‌را نیز ملاحظه می‌کنید. برای این منظور از متد formatItem استفاده شده است. آرایه row در اینجا در برگیرنده اعضای Title و Id ارسالی به افزونه است. اندیس صفر آن به عنوان دریافتی اشاره می‌کند.

همچنین نحوه نشان دادن عکس العمل به عنصر انتخابی را هم ملاحظه می‌کنید (در متد result مقدار دهی شده).
window.location را به عنصر دوم آرایه row هدایت خواهیم کرد. این عنصر دوم مطابق کدهای اکشن متد تهیه شده، به آدرس یک صفحه اشاره می‌کند.

ه) نمایش Auto-Complete در ASP.NET WebForms

قسمت عمده مطالب فوق با وب فرم‌ها نیز یکی است. خصوصا توضیحات مرتبط با متد EnableSearchAutocomplete ذکر شده.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
Inherits="LuceneSearch.WebForms.Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>جستجو</title>
    <link href="Content/Site.css" rel="stylesheet" type="text/css" />
    <script src="Scripts/jquery-1.7.1.min.js" type="text/javascript"></script>
    <script src="Scripts/jquery.autocomplete.js" type="text/javascript"></script>
    <script src="Scripts/custom.js" type="text/javascript"></script>
</head>
<body dir="rtl">
    <h2>
        جستجو</h2>
    <form id="form1" runat="server">
    <div align="center">
        <asp:TextBox runat="server" dir="ltr" ID="term"></asp:TextBox>
        <br />
        را وارد نمائید lu جهت آزمایش
    </div>
    </form>
    <script type="text/javascript">
        EnableSearchAutocomplete('Search.ashx', 'Content/Images/bullet_shape.png');
    </script>
</body>
</html>
```

```
using System.Text;
using System.Web;
using LuceneSearch.Core;

namespace LuceneSearch.WebForms
{
    public class Search : IHttpHandler
    {
        static string _indexPath = HttpRuntime.AppDomainAppPath + @"App_Data\idx";

        public void ProcessRequest(HttpContext context)
        {
            string q = context.Request.QueryString["q"];
            if (string.IsNullOrWhiteSpace(q))
            {
                context.Response.Write(string.Empty);
                context.Response.End();
            }

            var result = new StringBuilder();
            var items = AutoComplete.GetTermsScored(_indexPath, q);
            foreach (var item in items)
            {
                var postUrl = "Default.aspx?id=" + item.Id;
                result.AppendLine(item.Title + "|" + postUrl);
            }

            context.Response.ContentType = "text/plain";
            context.Response.Write(result.ToString());
            context.Response.End();
        }

        public bool IsReusable
        { get { return false; } }
    }
}
```

در اینجا بجای Controller از یک Generic handler استفاده شده است (Search.ashx).

```
result.AppendLine(item.Title + "|" + postUrl);
context.Response.Write(result.ToString());
```

در آن، عنوان مطالب یافت شده به همراه یک آدرس مشخص، تهیه و در Response نوشته خواهند شد.

کدهای کامل مثال فوق را از اینجا می‌توانید دریافت کنید:

LuceneSearch.zip

همچنین باید دقت داشت که پروژه MVC آن از نوع MVC4 است (VS2010) و فرض براین می‌باشد که IIS Express 7.5 را نیز پیشتر نصب کرده‌اید.

کلمه عبور فایل: dotnettips91

نظرات خوانندگان

نویسنده: محسن
تاریخ: ۱۳۹۱/۱۰/۰۷ ۱۲:۵۹

ممنون از این مطلب مفید. آیا با استفاده روش فوق میشه چند جدول رو با هم ایندکس کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۰۷ ۱۳:۳

بله. در اینجا مشکلی با ثبت اطلاعات از چندین جدول مختلف، در یک ایندکس وجود ندارد. برای مدیریت جستجوی بهتر روی آن‌ها یک روش این است که در متد MapPostToDocument ، فیلد دیگری را به نام مثلا TableName اضافه کنید تا بشود در حین جستجو از آن استفاده کرد.

نویسنده: محمد حبیبی
تاریخ: ۱۳۹۱/۱۰/۲۵ ۱۴:۴

من یک تکست باکس ایجاد کردم که از همین Autocomplete شما استفاده میکنه. منتها یک دکمه جستجو هم قرار دادم که با کلیک کردن روی اون دوباره از لوسین استفاده میکنه و چندین رکورد رو هم بازگشت میده.
اما چون باید یک سری رشته که نتیجه‌ی لوسین هست رو برگردونم از return Content استفاده میکنم ولی نتایج رو در یک صفحه‌ی جدید و بدون Master Page میاره.
نحوه‌ی نمایش نتایج اطلاعات لوسین در یک View که حاوی یک Master Page هم هست چطوریه؟
مثلا نتایج زیر همون تکست باکس نمایش داده بشه.
آیا باید نوع اکشن چیز دیگه ای باشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۵ ۱۴:۲۱

نتیجه جستجوی لوسین مثلا در مطلب فوق لیستی از SearchResultها است. همین لیست را در اکشن متد بازگشت دهید. سپس در View متناظر یک حلقه درست کرده و حاصل را به صورت دلخواهی فرمت کنید.
علت استفاده از return Content در مثال بالا، نیاز افزونه جی‌کوئری استفاده شده به خروجی ساده متنی است. در حالت‌های دیگر از return View معمولی استفاده کنید.

نویسنده: مرتضی
تاریخ: ۱۳۹۱/۱۱/۰۵ ۱۲:۴۵

سلام آقای نصیری ممنون از اینکه اطلاعتتون رو در اختیار دیگران قرار میدید بنده از کدهای بالا تویه سایتم استفاده کردم و جستجوی سایت مثل ساعت داره کار میکنه فقط تویه ظاهرش به مشکل برخوردم بنده میخوام اون box رو که اطلاعات توش نمایش داده میشه از سمت راست تکس باکس تراز بشه مثل همون عکسی که بالا گذاشتید از بنده از سمت چپ تراز شده و یه سوال دیگه, اون عکس loading رو هنگام جستجو نمایش نمیده آدرس دهی اون هم درسته البته تکس باس بنده position:absolute هستش نمیدونم ماله اینه که نمایش نمیده یا مثلا میره یه قسمت دیدگی از صفحه. ممنون میشم راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۵ ۱۳:۱

این موارد را باید با اصلاح اسکریپت یا css مربوط به auto-complete مدیریت کنید.
مثلا محل قرارگیری منوی بازشونده به صورت زیر مقدار دهی شده. این را در فایل jquery.autocomplete.js یافته و اصلاح کنید:

```
left: offset.left - options.width + 125
```

یا تصویر loading در css به نحو زیر تعیین شده:

```
.ac_loading
{
background: white url('Images/indicator.gif') left center no-repeat;
}
```

نویسنده: امیر
تاریخ: ۱۳۹۱/۱۱/۰۹ ۱۷:۴۲

سلام؛ من یه مشکلی دارم وقتی بعد از جستجو گزینه ای رو انتخاب می‌کنی صفحه refresh می‌شه و متن انتخاب شده پاک می‌شه میشه لطفا منو راهنمایی کنید مرسی

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۹ ۱۸:۶

بله. به همین نحو طراحی شده. زمانیکه یک گزینه انتخاب می‌شود، سطر زیر، کاربر را به صفحه متناظر هدایت می‌کند:

```
window.location = row[1];
```

این سطر در سمت کلاینت، مساوی Response.Redirect سمت سرور است. می‌تونید اینجا متن انتخابی رو به صورت مثلا یک کوئری استرینگ تعریف کنید و بعد در صفحه‌ای دیگر دریافت و نمایش بدید.

نویسنده: محسن.د
تاریخ: ۱۳۹۱/۱۱/۱۷ ۱:۵

بسیار عالی بود . تنها یک مشکل برای من در ارتباط با کدهای آموزشی که برای استفاده از لوسین در اینترنت پیدا کردم وجود داره و اون هم اینکه در اونها کلاس و توابع به صورت استاتیک تعریف و در کنترلرها و یا کلاس‌های لایه سرویس فراخوانی شدن .
این مسئله باعث اشکال در نوشتن آزمون واحد برای کنترلرها و یا متدها نمیشه ؟

آیا در استاتیک معرفی کردن کلاس و توابع علت و مزیت خاصی وجود داره ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۱۷ ۸:۵۵

- اگر از لایه سرویس استفاده می‌کنید که نهایتا با یک سری اینترفیس در کنترلرها کار خواهید کرد.
- بله. توضیح دادم در مطلب « نحوه استفاده صحیح از لوسین در ASP.NET ».

نویسنده: میهمان
تاریخ: ۱۳۹۱/۱۱/۲۱ ۲۰:۱۵

با سلام
امکانش هست که یه توضیح هم در مورد نحوه کار مستقیم با دیتابیس بدین؟
کدام قسمت‌ها باید تغییر کند و کدام قسمت‌ها باید حذف و اضافه شوند
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۱ ۲۰:۳۰

لطفا اولین پیشنیاز عنوان شده را مطالعه کنید.

نویسنده: میهمان
تاریخ: ۱۳۹۱/۱۱/۲۲ ۰:۱۷

با سلام مجدد؛ تمامی پیشنیازهای این مقاله را مطالعه کردم و خط به خط کدهای پروژه ای را که زحمت کشیدید و تهیه کرده اید را بررسی کردم. شما در اولین پیشنیاز عنوان شده فرموده بودید " کتابخانه Lucene مستقل است از منبع داده مورد استفاده و تنها اطلاعاتی با فرمت شیء Document معرفی شده به آن‌را می‌شناسد. " و در پروژه، در ابتدا کل داده‌ها را به یک document تبدیل کردید و سپس آنها را به IndexWriter اضافه کردید. و سپس جستجوهای خود را روی این دیکشنری انجام می‌دادید.
حالا سوال من این است: من در ابتدا باید تمامی اطلاعات را از طریق عنوان شده به یک فایل دیکشنری تبدیل کنم و هر مطلب جدید که اضافه میشود ، به این دیکشنری نیز اضافه شود ؟ اگر تعداد رکودها بالای میلیون باشد؟ راهی وجود ندارد که ما مستقیم به خود دیتابیس کار کنیم ؟ و یا نه من ابتدا باید روی دیتابیس کلمه مورد نظر را جستجو کتم و پس از یافتن آن، آنرا به دیکشنری اضافه کنم ؟ بهترین راه چیست که برای داده‌های بالا جوابگو باشد
با تشکر فراوان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۲۲ ۰:۵۱

- لوسین مستقل است از بانک اطلاعاتی. همچنین یکبار باید این ایندکس را تهیه کنید. اگر تعداد رکوردهای شما بالا است، فقط همان بار اول است که کار تهیه زمانبر خواهد بود. برای دفعات بعد در حد اضافه کردن چند سند لوسین به آن یا به روز رسانی و حذف است و کار دیگری ندارد.
- پس از تهیه ایندکس، جستجوی لوسین کاری به بانک اطلاعاتی شما ندارد. بر روی ایندکس خودش انجام می‌شود و نیازی به جستجوی مجدد در بانک اطلاعاتی شما نیست. یک سیستم مستقل است.
این روش متداول کار با لوسین است و حالت دیگری هم ندارد. این مستقل بودن هم یک مزیت است. برای مثال SQL Server CE یا خیلی از بانک‌های اطلاعاتی دیگر Full Text Search توکار ندارند. اینجا لوسین خوب جواب می‌ده.
ضمن اینکه من در یک دمو استفاده از لوسین برای ایندکس کردن کل اطلاعات ویکی‌پدیا رو دیدم. تهیه ایندکس آن یک روز کار برده بوده (با توجه به حجم اطلاعات بالای ویکی پدیا)، اما جستجوی آن فوق العاده سریع و با کیفیت بود. این ویدیو رو در اینجا می‌تونید مشاهده کنید:
Full-text search with Lucene and neat things you can do with it

نویسنده: morteza
تاریخ: ۱۳۹۱/۱۲/۰۱ ۲۲:۳۰

سلام آقای نصیری میخواستم ببینم آیا شما اطلاعی از نحوه غیر فعال کردم اینتر توبه این پلاگین دارید بنده میخوام وقتی کاربر اینتر رو فشار داد جستجوی معمولی انجام بشه و اگه خواست از مواردی که توسط autocomplate براش آمده توسط موس یکی رو انتخاب کنه و به صفحه مربوطه هدایت بشه ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۰۱ ۲۲:۴۹

کلید enter با case KEY.RETURN در این افزونه جی‌کوئری مدیریت می‌شود.

نویسنده: علیرضا پایدار
تاریخ: ۱۳۹۱/۱۲/۲۰ ۱۰:۵۵

یک سوال خدمت شما داشتم اینکه :

وقتی بخوایم از 2 جدول متفاوت جستجو کنیم به نظر شما اطلاعات هر جدول را جداگانه ایندکس کنیم(منظورم همان کاری که متد CreateFullTextIndex شما انجام میده) یا خیر؟
البته منظور دو جدولی که با هم رابطه دارند.

چندتا لینک در مورد لوسین:

http://www.thebestcsharpprogrammerintheworld.com/blogs/how-to-create-and-search-a-lucene-net-index-in-4-simple-steps-using-c-sharp-step-1.aspx

http://www.codeproject.com/Articles/272309/Lucene-Search-Programming

http://www.codeproject.com/Articles/320219/Lucene-Net-ultra-fast-search-for-MVC-or-WebForms

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۲۰ ۱۱:۳۲

من همه رو داخل یک ایندکس ثبت می‌کنم. فقط یک فیلد اضافه‌تر به نام «نام جدول» مورد نظر نیاز هست تا بشود روی آن کوئری خاص گرفت یا اینکه کلا روی تمام رکوردها جستجو کرد به یکباره.

نویسنده: مهمان
تاریخ: ۱۳۹۲/۰۳/۱۹ ۷:۵۵

ضمن خسته نباشید
اگر نیاز داشته باشیم که بعد از انتخاب گزینه کاربر به صفحه دیگه هدایت نشه و فقط آیتم انتخابی در تکست باکس بمونه باید چه کاری انجام داد. مرسی

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۹ ۱۷:۱۳

سطر window.location رو حذف کنید.

نویسنده: imo0
تاریخ: ۱۳۹۲/۰۷/۲۷ ۱۰:۵۵

سلام؛ تویه اولین پیشنیازها که کار با لوسین رو گفته بودین توابعی مثله SearchByPartialWords و Query و غیره ایجاد شده ولی تو اینجا یه جور دیگه جستجو کردین توی ایندکس‌ها . میخواستم بدونم او توابع رو مگه واسه سرچ تو ایندکس‌ها ننوشتین؟ تازه اینجا شما دارین متن Title رو میفرستین سمت کاربر . این highlight کردن عبارت جستجو شدش پس چی شد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۲۷ ۱۱:۳۵

- این هم یک روش دیگر هست. از آن روش‌های ذکر شده در پیشنیازها هم می‌توانید استفاده کنید.
- highlight کردن عبارت جستجو شد ه در عنوان بازگشت داده شده، توسط افزونه auto-complete انجام می‌شود؛ خودکار است.

نویسنده: imo0
تاریخ: ۱۳۹۲/۰۷/۲۷ ۱۱:۴۶

تو اون تابع Query که نوشتین یه دونه FastVectorHighlighter ایجاد کردین . اگه highlight توسط auto-complete خودکار انجام میشه پس نقش این چیه؟ میشه بگین کدوم روش بهتره ؟ یا فرقی نمیکنه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۲۷ ۱۱:۵۲

دو بحث وجود داره:
- برجسته سازی قسمتی از عبارت جستجو شده در لیست نمایش داده شده توسط افزونه auto-complete. این مورد خودکار است و توسط افزونه انجام می‌شود.
- برجسته سازی قسمتی از عبارت جستجو شده در نتایج یک جستجوی کامل بعدی که قرار است highlight آن توسط ما با کدنویسی خاصی انجام شود. مراجعه کنید به این مطلب برای توضیحات بیشتر.

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۳/۰۲/۲۲ ۱۵:۵۶

با سلام
من از لوسین در asp.net web form استفاده کردم و کاملا هم راضی هستم . فقط یه مشکلی داشتم. اگه بخوام غیر از عبارت مورد نظر برای جستجو یه مقدار دیگه رو هم بفرستم باید چکار کنم؟
مثلا یه Dropdown داشته باشیم وبخواییم مثلا مقدار categoryid رو هم پاس بدیم تا به query مربوطه اضافه بشه.من dropdown رو اضافه کردم اما مقدارش رو نمیشه در هندلر مربوط به Search دریافت کرد مثل q.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۲۲ ۱۷:۲۶

افزونه‌ی مورد استفاده در آن به نام jquery.autocomplete.js سورسش پیوست هست. ajax را در آن جستجو کنید به مورد ذیل خواهید رسید:

```
$.ajax({
    //...
    data: $.extend({
             q: lastWord(term),
             limit: options.max
             }, extraParams),
       //...
});
```

در اینجا q مشخص است و extraParams آن متغیر. بنابراین برای ارسال اطلاعات اضافی باید extraParams را مانند کدهای ذیل مقدار دهی کرد. نحوه‌ی دریافت عناصر آن در سمت سرور، مانند نحوه‌ی دریافت q است.

```
$(document).ready(function () {
        $("#term").autocomplete(url, {
            // .....
            extraParams: { id: 12, xyz: "test" }
        }).result(function (evt, row, formatted) {
            // .....
        });
    });
```

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۳/۰۳/۰۷ ۱۸:۱۱

اگر بخوام لیستی که برای نتیجه جستجو باز میشه رو Custom کنم مثلاً یه عکس بهش اضافه کنم یا یه جدول ساخته شده با HTML رو توش نشون بدم با یه سری جزئیات ، آیا امکان پذیره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۰۷ ۱۹:۳۰

نحوه سفارشی سازی هر آیتم لیست را در مطلب فوق، در متد formatItem می‌توانید مشاهده کنید.

نویسنده: علیرضا
تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۰:۵۸

با سلام؛ تکست باکسی که می‌خام جستجو از طریق اون انجام بشه در یک مستر پیج قرار داره و زمانی که صفحه و مستر پیج در یک پوشه قرار دارند مشکلی نیست ولی اگر صفحه در یک پوشه دیگر قرار گیرد جواب نمی‌ده. توضیح اینکه از متد ResolveClientUrl هم استفاده کردم ولی جواب نداد :

```
EnableSearchAutocomplete('<%=ResolveClientUrl( "~/Handlers/find.ashx")%>',
'<%=ResolveClientUrl("Images/bullet_shape.png")%>');
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۱:۱۴

- نحوه استفاده از افزونه Firebug برای دیباگ برنامه‌های ASP.NET مبتنی بر jQuery
- بررسی کنید آیا ID جعبه متنی در آن صفحه، term است یا چیز دیگری ؟ اگر تغییر کرده، ClientIDMode=Static بهتر است تنظیم شود.

نویسنده: علیرضا
تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۸:۳۱

با تشکر از پاسخ شما ، id درست بود و با فایرباگ هم بررسی کردم و خطای زیر رو می‌داد :

```
TypeError: $(...).autocomplete is not a function
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۰۳ ۱۸:۵۴

این خطا یعنی اسکریپت‌های اصلی به صفحه پیوست نشده‌اند. برای تعریف آن‌ها هم از ResolveClientUrl استفاده کنید:

```
<script type="text/javascript" src='<%: ResolveClientUrl("~/path....") %>'></script>
```

عنوان: **نحوه اضافه کردن قابلیت غلط گیر املایی شبیه به جستجوی گوگل توسط لوسین**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۰۶ ۲۱:۱۰
آدرس: www.dotnettips.info
گروه‌ها: Lucene.NET

پیشنیاز:
چگونه با استفاده از لوسین مطالب را ایندکس کنیم؟

مقدمه

اگر به جستجوی سایت دقت کرده باشید، قابلیتی تحت عنوان پیشنهاد «عبارات مشابه» به آن اضافه شده است:

این مورد بر اساس ماژول غلط یاب املایی لوسین تهیه شده و بسیار شبیه به "did you mean" جستجوی گوگل است. در ادامه به نحوه پیاده سازی آن خواهیم پرداخت.

کتابخانه‌های مورد نیاز

علاوه بر کتابخانه لوسین ، نیاز به دریافت پروژه Contrib آن نیز می‌باشد تا بتوان از اسمبلی Lucene.Net.Contrib.SpellChecker.dll موجود در آن استفاده کرد.

نحوه کار با غلط یاب املایی لوسین

خلاصه کار با غلط یاب املایی لوسین همین چند سطر ذیل است:

```
var indexReader = IndexReader.Open(FSDirectory.Open(indexPath), readOnly: true);

// Create the SpellChecker
var spellChecker = new SpellChecker.Net.Search.Spell.SpellChecker(FSDirectory.Open(indexPath +
"\\Spell"));

// Create SpellChecker Index
spellChecker.ClearIndex();
spellChecker.IndexDictionary(new LuceneDictionary(indexReader, "Title"));
spellChecker.IndexDictionary(new LuceneDictionary(indexReader, "Body"));
```

```
//Suggest Similar Words
var results = spellChecker.SuggestSimilar(term, number, null, null, true);
```

کار بر اساس یک ایندکس از پیش موجود لوسین شروع می‌شود. در اینجا فرض شده است که این ایندکس در پوشه indexPath قرار دارد.

در ادامه شیء spellChecker را آغاز خواهیم کرد. بهتر است پوشه تولید فایل‌های آن با پوشه ایندکس اصلی یکسان نباشد. اگر یکسان درنظر گرفته شود، تمام مداخل جدید به ایندکس موجود اضافه خواهند شد که می‌تواند سرعت جستجوی معمولی را کاهش دهد.

سپس کار تهیه ایندکس جدید غلط یاب املایی، شروع خواهد شد. متد spellChecker.ClearIndex، اطلاعات موجود در ایندکسی قدیمی را حذف کرده و سپس spellChecker.IndexDictionary، فیلدهایی را که نیاز داریم در تهیه غلط یاب املایی حضور داشته باشند، مشخص می‌کند.

همانطور که ملاحظه می‌کنید ایندکس جدید تهیه شده، بر اساس بانک اطلاعاتی واژه‌های موجود در ایندکس اصلی برنامه که توسط indexReader معرفی شده، تهیه می‌شود. برای نمونه در تصویر ابتدای مطلب جاری، واژه‌های پیشنهادی، واژه‌هایی هستند که پیشتر یکبار تایپ شده و در بانک اطلاعاتی برنامه موجود بوده‌اند.

و در آخر برای استفاده از امکانات تهیه شده، تنها کافی است متد spellChecker.SuggestSimilar را فراخوانی کنیم (در زمانیکه جستجوی اصلی سایت نتیجه‌ای را ارائه نداده است). حاصل لیستی از واژه‌های مشابه است.

نظرات خوانندگان

نویسنده: علی رادمان فر
تاریخ: ۱۳۹۱/۰۹/۰۷ ۱۶:۵

Spell Checker فارسی هم با این امکانات وجود داره؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۰۷ ۱۶:۲۹

با فارسی هم کار می‌کنه: (^ ، ^ و یا ^)

نویسنده: بافکر
تاریخ: ۱۳۹۳/۰۲/۳۱ ۱۷:۷

با سلام

حالتی هست که اینجا هم بشه اینسرت، دیلیت و آپدیت کرد؟ (همانند ایندکس اصلی)

نویسنده: محسن تقی پور
تاریخ: ۱۳۹۳/۰۶/۱۳ ۱۴:۱۳

با سلام
ببخشید یه سوال داشتم میخواستم بدونم Create SpellChecker Index رو چه مواقعی باید فراخوانی کنم ؟
موقعی که خبر درج شد و ادیت شد و پاک شد باید فراخوانی بشه ؟ یا تو یه بازه زمانی خواص مثلا هر 24 ساعت ایجاد بشه؟
میخواستم بدونم ایا میشه این ایندکس‌ها رو اپدیت کرد یا اینکه هر دفعه تو ایندکس اصلیم تغییری ایجاد شد باید از اول ساخته شه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۱۳ ۱۴:۲۲

این مورد خاص SpellChecker قابلیت به روز رسانی ندارد. هر بار باید از صفر انجام شود. روزی یکبار کافی است.

عنوان: **استفاده از لوسین برای انجام محاسبات آماری بر روی متون**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۹/۱۹ ۱۳:۳۳
آدرس: www.dotnettips.info
برچسب‌ها: Lucene.NET

احتمالا یک سری از کارهای اینفوگرافیک مانند tags cloud و words cloud را دیده‌اید. برای مثال در یک سخنرانی خاص، سخنران بیشتر از چه واژه‌هایی استفاده کرده است و سپس ترسیم درشت‌تر واژه‌هایی با تکرار بیشتر در یک تصویر نهایی. محاسبات آماری این نوع بررسی‌ها را توسط لوسین نیز می‌توان انجام داد که در ادامه به نحوه انجام آن خواهیم پرداخت.

بررسی آماری واژه‌های بکار رفته در شاهنامه

مرحله اول: ایجاد ایندکس

```
using System;
using System.Collections.Generic;
using System.IO;
using Lucene.Net.Analysis.Standard;
using Lucene.Net.Documents;
using Lucene.Net.Index;
using Lucene.Net.Store;

namespace ShaahnamehAnalysis
{
    public static class CreateIndex
    {
        static readonly Lucene.Net.Util.Version _version = Lucene.Net.Util.Version.LUCENE_CURRENT;

        static HashSet<string> getStopWords()
        {
            var result = new HashSet<string>();
            var stopWords = new[]
            {
                "به",
                "با",
                "از",
                "تا",
                "و",
                "است",
                "هست",
                "هستم",
                "هستیم",
                "هستید",
                "هستند",
                "نیست",
                "نیستم",
                "نیستیم",
                "نیستند",
                "اما",
                "یا",
                "این",
                "آن",
                "اینجا",
                "آنجا",
                "بود",
                "باد",
                "برای",
                "که",
                "دارم",
                "داری",
                "دارد",
                "داریم",
                "دارید",
                "دارند",
                "چند",
                "را",
                "ها",
                "های",
                "می",
                "هم",
                "در",
                "باشم",
```

```
"باشی",
"باشد",
"باشیم",
"باشید",
"باشند",
"اگر",
"مگر",
"بجز",
"جز",
"الا",
"اینکه",
"چرا",
"کی",
"چه",
"چطور",
"چی",
"چیست",
"آیا",
"چنین",
"اینچنین",
"نخست",
"اول",
"آخر",
"انتها",
"صد",
"هزار",
"میلیون",
"ملیون",
"میلیارد",
"ملیارد",
"یکهزار",
"تریلیون",
"تریلیارد",
"میان",
"بین",
"زیر",
"بیش",
"روی",
"ضمن",
"همانا",
"ای",
"بعد",
"پس",
"قبل",
"پیش",
"هیچ",
"همه",
"واما",
"شد",
"شده",
"شدم",
"شدی",
"شدیم",
"شدند",
"یک",
"یکی",
"نبود",
"میکند",
"میکنم",
"میکنیم",
"میکنید",
"میکنند",
"میکنی",
"طور",
"اینطور",
"آنطور",
"هر",
"حال",
"مثل",
"خواهم",
"خواهی",
"خواهد",
"خواهیم",
"خواهید",
"خواهند",
"داشته",
"داشت",
"داشتی",
"داشتم",
"داشتیم",
"داشتید",
```

```
                "داشتند",
                "آنکه",
                "مورد",
                "کنید",
                "کنم",
                "کنی",
                "کنند",
                "کنیم",
                "نکنم",
                "نکنی",
                "نکند",
                "نکنیم",
                "نکنید",
                "نکنند",
                "نکن",
                "بگو",
                "نگو",
                "مگو",
                "بنابراین",
                "بدین",
                "من",
                "تو",
                "او",
                "ما",
                "شما",
                "ایشان",
                "ی",
                "ـ",
                "هایی",
                "خیلی",
                "بسیار",
                "1",
                "بر",
                "1",
                "شود",
                "کرد",
                "کرده",
                "نیز",
                "خود",
                "شوند",
                "اند",
                "داد",
                "دهد",
                "گشت",
                "ز",
                "گفت",
                "آمد",
                "اندر",
                "چون",
                "بد",
                "چو",
                "همی",
                "پر",
                "سوی",
                "دو",
                "گر",
                "بی",
                "گرد",
                "زین",
                "کس",
                "زان",
                "جای",
                "آید"
            };

            foreach (var item in stopWords)
                result.Add(item);

            return result;
        }

        public static void CreateShaahnamehIndex(string file = "shaahnameh.txt")
        {
            var directory = FSDirectory.Open(new DirectoryInfo(Environment.CurrentDirectory +
"\\LuceneIndex"));
            var analyzer = new StandardAnalyzer(_version, getStopWords());
            using (var writer = new IndexWriter(directory, analyzer, create: true, mfl:
IndexWriter.MaxFieldLength.UNLIMITED))
            {
                var section = string.Empty;
                foreach (var line in File.ReadAllLines(file))
```

```
                {
                    int result;
                    if (int.TryParse(line, out result))
                    {
                        var postDocument = new Document();
                        postDocument.Add(new Field("Id", result.ToString(), Field.Store.YES,
Field.Index.NOT_ANALYZED));
                        postDocument.Add(new Field("Body", section, Field.Store.YES,
Field.Index.ANALYZED, Field.TermVector.WITH_POSITIONS_OFFSETS));
                        writer.AddDocument(postDocument);
                        section = string.Empty;
                    }
                    else
                        section += line;
                }

                writer.Optimize();
                writer.Commit();
                writer.Close();
                directory.Close();
            }
        }
    }
}
```

با ایجاد ایندکس‌های لوسین پیشتر در این سایت آشنا شده‌اید . روش کار نیز همانند سابق است. اطلاعات خود را، به هر فرمتی که تهیه شده باید تبدیل به اشیاء Document لوسین کرد. برای مثال در اینجا فقط یک فایل txt داریم که تشکیل شده است از تمام صفحات. به ازای هر صفحه، یک شیء Document تهیه و نوشته خواهد شد. همچنین در تهیه ایندکس از یک سری از واژه‌های بسیار متداول مانند «از»، «به»، «اندر»، (stopWords) صرفنظر شده است.

مرحله دوم: ایجاد ابر واژه‌ها

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using Lucene.Net.Index;
using Lucene.Net.Store;

namespace ShaahnamehAnalysis
{
    [DebuggerDisplay("{Frequency}, {Text}")]
    public class Tag
    {
        public string Text { set; get; }

        /// <summary>
        /// The frequency of a term is defined as the number of
        /// documents in which a specific term appears.
        /// </summary>
        public int Frequency { set; get; }
    }

    public static class WordsCloud
    {
        /// <summary>
        /// Create Words Cloud
        /// </summary>
        /// <param name="threshold">every term that appears in more than x Body</param>
        public static IList<Tag> Create(int threshold = 200)
        {
            var path = Environment.CurrentDirectory + "\\LuceneIndex";

            var results = new List<Tag>();
            var field = "Body";

            IndexReader indexReader = IndexReader.Open(FSDirectory.Open(path ), true);

            var termFrequency = indexReader.Terms();
            while (termFrequency.Next())
            {
                if (termFrequency.DocFreq() >= threshold && termFrequency.Term.Field == field)
                {
```

```
                    results.Add(new Tag { Text = termFrequency.Term.Text, Frequency =
termFrequency.DocFreq() });
                }
            }
            return results.OrderByDescending(x => x.Frequency).ToList();
        }
    }
}
```

پس از اینکه ایندکس لوسین تهیه شد، می‌توان به مداخل موجود در آن توسط متد indexReader.Terms دسترسی یافت.
نکته جالب آن فراهم بودن DocFreq هر واژه ایندکس شده است (فرکانس تکرار واژه؛ تعداد اشیاء Document ایی که واژه مورد نظر در آن‌ها تکرار شده است). برای مثال در اینجا اگر واژه‌ای 200 بار یا بیشتر در صفحات مختلف شاهنامه تکرار شده باشد، به عنوان یک واژه پر اهمیت انتخاب شده و به ابر واژه‌های نهایی اضافه می‌گردد.

مرحله سوم: استفاده از نتایج

```
using System;
using System.Diagnostics;
using System.IO;
using System.Linq;

namespace ShaahnamehAnalysis
{
    class Program
    {
        static void Main(string[] args)
        {
            CreateIndex.CreateShaahnamehIndex();
            var wordsCloudList = WordsCloud.Create();

            var data = wordsCloudList.Select(x => x.Text + ", " + x.Frequency)
                                     .Aggregate((s1, s2) => s1 + Environment.NewLine + s2);
            var output = "ShaahnamehAnalysis.txt";
            File.WriteAllText(output, data);
            Process.Start(output);
        }
    }
}
```

که نتیجه 15 مورد اول آن به صورت زیر است:

واژه | فرکانس
شاه, 1191
دل, 1088
سر, 1070
کار, 840
لشکر, 801
تخت, 755
روز, 745
ایران, 740
جهان, 724
مرد, 660
دست, 630
تاج, 623
نزدیک, 623
گیتی, 585
راه, 584

فایل‌های کامل این مثال را از اینجا می‌توانید دریافت کنید:

ShaahnamehAnalysis.zip

عنوان: **نگاهی به Latent Semantic Indexing**
نویسنده: حامد خسروجردی
تاریخ: ۱۳۹۳/۰۳/۰۹ ۱۲:۵۵
آدرس: www.dotnettips.info
گروه‌ها: Lucene.NET, Index, Full Text Search, search

مقدمه ای بر Latent Semantic Indexing

هنگامیکه برای اولین بار، جستجو بر مبنای کلمات کلیدی (keyword search) بر روی مجموعه‌ای از متون، به دنیای بازیابی اطلاعات معرفی شد شاید فقط یک ذهنیت مطرح می‌شد و آن یافتن لغت در متن بود. به بیان دیگر در آن زمان تنها بدنبال متونی می‌گشتیم که دقیقا شامل کلمه کلیدی مورد جستجوی کاربر باشند. روال کار نیز بدین صورت بود که از دل پرس و جوی کاربر، کلماتی بعنوان کلمات کلیدی استخراج می‌شد. سپس الگوریتم جستجو در میان متون موجود بدنبال متونی می‌گشت که دقیقا یک یا تمامی کلمات کلیدی در آن آمده باشند. اگر متنی شامل این کلمات بود به مجموعه جواب‌ها اضافه می‌گردید و در غیر این صورت حذف می‌گشت. در پایان جستجو با استفاده از الگوریتمی، نتایج حاصل رتبه بندی می‌گشت و به ترتیب رتبه با کاربر نمایش داده می‌شد.

نکته مهمی که در این روش دیده می‌شود اینست که متون به تنهایی و بدون در نظر گرفتن کل مجموعه پردازش می‌شدند و اگر تصمیمی مبنی بر جواب بودن یک متن گرفته می‌شد، آن تصمیم کاملا متکی به همان متن و مستقل از متون دیگر گرفته می‌شد. در آن سال‌ها هیچ توجهی به وابستگی موجود بین متون مختلف و ارتباط بین آنها نمی‌شد که این مسئله یکی از عوامل پایین بودن دقت جستجوها بشمار می‌رفت.

در ابتدا بر اساس همین دیدگاه الگوریتم‌ها و روش‌های اندیس گذاری (indexing) پیاده سازی می‌شدند که تنها مشخص می‌کردند یک لغت در یک سند (document) وجود دارد یا خیر. اما با گذشت زمان محققان متوجه ناکارآمدی این دیدگاه در استخراج اطلاعات شدند. به همین دلیل روشی بنام Latent Semantic Indexing که بر پایه Latent Semantic Analysis بنا شده بود به دنیای بازیابی و استخراج اطلاعات معرف شد. کاری که این روش انجام می‌داد این بود که گامی را به مجموعه مراحل موجود در پروسه اندیس گذاری اضافه می‌کرد. این روش بجای آنکه در اندیس گذاری تنها یک متن را در نظر بگیرد و ببیند چه لغاتی در آن آورده شده است، کل مجموعه اسناد را با هم و در کنار یکدیگر در نظر می‌گرفت تا ببیند که چه اسنادی لغات مشابه با لغات موجود در سند مورد بررسی را دارند. به بیان دیگر اسناد مشابه با سند فعلی را به نوعی مشخص می‌نمود.

بر اساس دیدگاه LSI اسناد مشابه با هم، اسنادی هستند که لغات مشابه یا مشترک بیشتری داشته باشند. توجه داشته باشید تنها نمی‌گوییم لغات مشترک بیشتری بلکه از واژه لغات مشابه نیز استفاده می‌کنیم. چرا که بر اساس LSI دو سند ممکن است هیچ لغت مشترکی نداشته باشند (یعنی لغات یکسان نداشته باشند) اما لغاتی در آنها وجود داشته باشد که به لحاظی معنایی و مفهومی هم معنا و یا مرتبط به هم باشند. بعنوان مثال لغات شش و ریه دو لغت متفاوت اما مرتبط با یکدیگر هستند و اگر دو لغات در دوسند آورده شوند می‌توان حدس زد که ارتباط و شباهتی معنایی بین آنها وجود دارد. به روش هایی که بر اساس این دیدگاه ارائه می‌شوند روش‌های جستجوی معنایی نیز گفته می‌شود. این دیدگاه مشابه دیدگاه انسانی در مواجهه با متون نیز است. انسان هنگامی که دو متن را با یکدیگر مقایسه می‌کند تنها بدنبال لغات یکسان در آن‌ها نمی‌گردد بلکه شباهت‌های معنایی بین لغات را نیز در نظر می‌گیرد این اصل و نگرش پایه و اساس الگوریتم LSI و همچنین حوزه ای از علم بازیابی اطلاعات بنام مدل سازی موضوعی (Topic Modeling) می‌باشد.

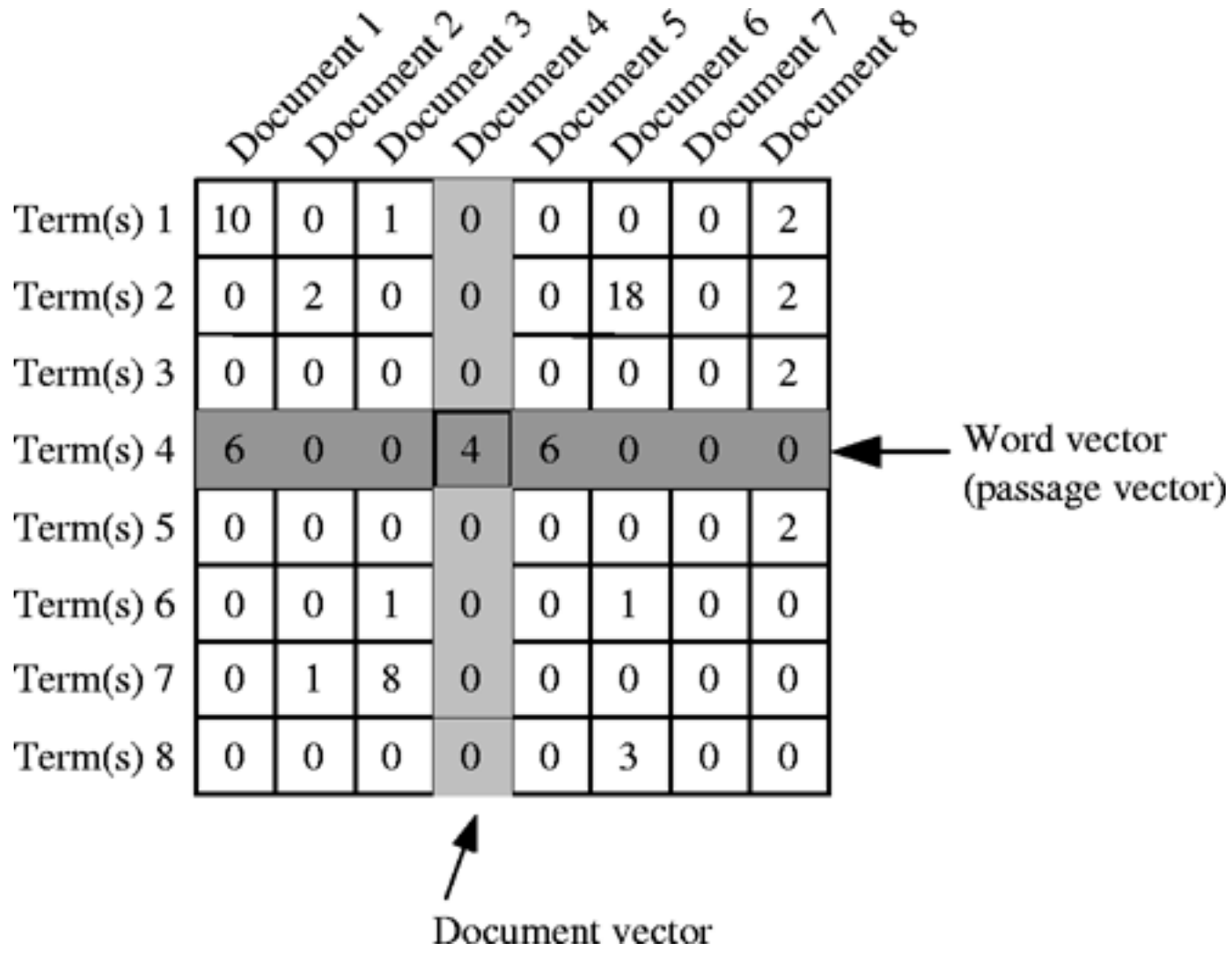
هنگامیکه شما پرس و جویی را بر روی مجموعه ای از اسناد (که بر اساس LSI اندیس گذاری شده‌اند) اجرا می‌کنید، موتور جستجو ابتدا بدنبال لغاتی می‌گردد که بیشترین شباهت را به کلمات موجود در پرس و جوی شما دارند. بعبارتی پرس و جوی شما را بسط می‌دهد (query expansion)، یعنی علاوه بر لغات موجود در پرس و جو، لغات مشابه آنها را نیز به پرس و جوی شما می‌افزاید. پس از بسط دادن پرس و جو، موتور جستجو مطابق روال معمول در سایر روش‌های جستجو، اسنادی که این لغات (پرس و جوی بسط داده شده) در آنها وجود دارند را بعنوان نتیجه به شما باز می‌گرداند. به این ترتیب ممکن است اسنادی به شما بازگردانده شوند که لغات پرس و جوی شما در آنها وجود نداشته باشد اما LSI بدلیل وجود ارتباطات معنایی، آنها را مشابه و مرتبط با جستجو تشخیص داده باشد. توجه داشته باشید که الگوریتم‌های جستجوی معمولی و ساده، بخشی از اسناد را که مرتبط با پرس و جو هستند، اما شامل لغات مورد نظر شما نمی‌شوند، از دست می‌دهد (یعنی کاهش recall).

برای آنکه با دیدگاه LSI بیشتر آشنا شوید در اینجا مثالی از نحوه عملکرد آن می‌زنیم. فرض کنید می‌خواهیم بر روی مجموعه ای از اسناد در حوزه زیست شناسی اندیس گذاری کنیم. بر مبنای روش LSI چنانچه لغاتی مانند کروموزم، ژن و DNA در اسناد زیادی در کنار یکدیگر آورده شوند (یا بعبارتی اسناد مشترک باهم زیادی داشته باشند)، الگوریتم جستجو چنین برداشت می‌کند که به احتمال زیاد نوعی رابطه معنایی بین آنها وجود دارد. به همین دلیل اگر شما پرس و جویی را با کلمه کلیدی "کروموزوم" اجرا نمایید، الگوریتم علاوه بر مقالاتی که مستقیما واژه کروموزوم در آنها وجود دارد، اسنادی که شامل لغات "DNA" و "ژن" نیز باشند را بعنوان نتیجه به شما باز خواهد گرداند. در واقع می‌توان گفت الگوریتم جستجو به پرس و جوی شما این دو واژه را نیز اضافه می‌کند که

همان بسط دادن پرس و جوی شما است. دقت داشته باشید که الگوریتم جستجو هیچ اطلاع و دانشی از معنای لغات مذکور ندارد و تنها بر اساس تحلیل‌های ریاضی به این نتیجه می‌رسد که در بخش‌های بعدی چگونگی آن را برای شما بازگو خواهیم نمود. یکی از برتری‌های مهم LSI نسبت به روش‌های مبتنی بر کلمات کلیدی (keyword based) این است که در LSI، ما به recall بالاتری دست پیدا می‌کنیم، بدین معنی که از کل جواب‌های موجود برای پرس و جوی شما، جواب‌های بیشتری به کاربر نمایش داده خواهند شد.
یکی از مهمترین نقاط قوت LSI اینست که این روش تنها متکی بر ریاضیات است و هیچ نیازی به دانستن معنای لغات یا پردازش کلمات در متون ندارد. این مسئله باعث می‌شود بتوان این روش را بر روی هر مجموعه متنی و با هر زبانی بکار گرفت. علاوه بر آن می‌توان LSI را بصورت ترکیبی با الگوریتم‌های جستجوی دیگر استفاده نمود و یا تنها متکی بر آن موتور جستجویی را پیاده سازی کرد.

نحوه عملکرد Latent Semantic Indexing

در روش LSI مبنا وقوع همزمان لغات در اسناد می‌باشد. در اصطلاح علمی به این مسئله word co-occurrence گفته می‌شود. به بیان دیگر LSI بدنبال لغاتی می‌گردد که در اسناد بیشتری در با هم آورده می‌شوند. پیش از آنکه وارد مباحث ریاضی و محاسباتی LSI شویم بهتر است کمی بیشتر در مورد این مسوله به لحاظ نظری بحث کنیم. **لغات زائد**
به نحوه صحبت کردن روز مره انسان‌ها دقت کنید. بسیاری از واژگانی که در طول روز و در محاوره‌ها از انها استفاده می‌کنیم، تاثیری در معنای سخن ما ندارند. این مسئله در نحوه نگارش ما نیز صادق است. خیلی از لغات از جمله حروف اضافه، حروف ربط، برخی از افعال پر استفاده و غیره در جملات دیده می‌شوند اما معنای سخن ما در آنها نهفته نمی‌باشد. بعنوان مثال به جمله "جهش در ژن‌ها می‌تواند منجر به بیماری سرطان شود" درقت کنید. در این جمله لغاتی که از اهمیت بالایی بر خوردار هستند و به نوعی بار معنایی جمله بر دوش آنهاست عبارتند از "جهش"، "ژن"، بیماری" و "سرطان". بنابراین می‌توان سایر لغات مانند "در"، "می تواند" و "به" را حذف نمود. به این لغات در اصطلاح علم بازیابی اطلاعات (Information Retrieval) لغات زائد (redundant) گفته می‌شود که در اکثر الگوریتم‌های جستجو یا پردازش زبان طبیعی (natural language processing) برای رسیدن به نتایج قابل قبول باید حذف می‌شوند.روش LSI نیز از این قاعده مستثنی نیست. پیش از اجرای آن بهتر است این لغات زائد حذف گردند. این مسئله علاوه بر آنکه بر روی کیفیت نتایج خروجی تاثیر مثبت دارد، تا حد قابل ملاحظه ای کار پردازش و محاسبات را نیز تسهیل می‌نماید.
مدل کردن لغات و اسناد
پس از آنکه لغات اضافی از مجموعه متون حذف شد باید بدنبال روشی برای مدل کردن داده‌های موجود در مجموعه اسناد بگردیم تا بتوان کاربر پردازش را با توجه به آن مدل انجام داد. روشی که در LSI برای مدلسازی بکار گرفته می‌شود استفاده از ماتریس لغت – سند (term-document matrix) است. این ماتریس یک گرید بسیار بزرگ است که هر سطر از آن نماینده یک سند و هر ستون از ان نماینده یک لغت در مجموعه متنی ما می‌باشد(البته این امکان وجود دارد که جای سطر و ستون‌ها عوض شود). هر سلول از این ماتریس بزرگ نیز به نوعی نشان دهنده ارتباط بین سند و لغت متناظر با آن سلول خواهد بود. بعنوان مثال در ساده‌ترین حات می‌توان گفت که اگر لغتی در سند یافت نشد خانه متناظر با انها در ماتریس لغت – سند خالی خواهد ماند و در غیر این صورت مقدار یک را خواهد گرفت. در برخی از روش‌ها سلول‌ها را با تعداد دفعات تکرار لغات در اسناد متناظر پر می‌کنند و در برخی دیگر از معیارهای پیچیده‌تری مانند tf*idf استفاده می‌نمایند. شکل زیر نمونه از این ماتریس‌ها را نشان می‌دهد :



برای ایجاد چنین ماتریسی باید تک تک اسناد و لغات موجود در مجموعه متنی را پردازش نمود و خانه‌های متناظر را در ماتریس لغت – سند مقدار دهی نمود.خروجی این کار ماتریسی مانند ماتریس شکل بالا خواهد شد (البته در مقیاسی بسیار بزرگتر) که بسیاری از خانه‌های ان صفر خواهند بود (مانند آنچه در شکل نیز مشاهده می‌کنید). به این مسئله تنک بودن (sparseness) ماتریس گفته می‌شود که یکی از مشکلات استفاده از مدل ماتریس لغت – سند محسوب می‌شود.

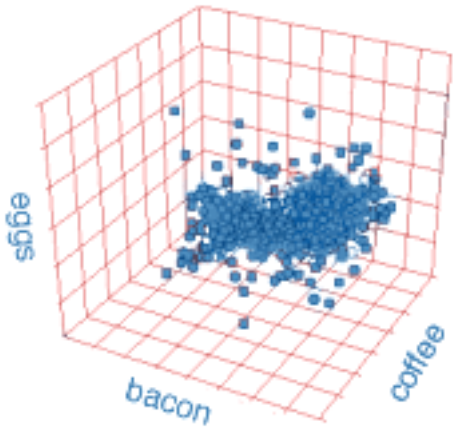
این ماتریس، بازتابی از کل مجموعه متنی را به ما می‌دهد. بعنوان مثال اگر بخواهیم ببینیم در سند i چه لغاتی وجود دارد، تنها کافی است به سراغ سطر iام از ماتریس برویم (البته در صورتی که ماتریس ما سند – لغت باشد) وآن را بیرون بکشیم. به این سطر در اصطلاح بردار سند (document vector) گفته می‌شود. همین کار را در مورد لغات نیز می‌توان انجام داد. بعنوان مثال با رفتن به سراغ ستون j ام می‌توان دریافت که لغت j ام در چه اسنادی آورده شده است. به ستون j ام نیز در ماتریس سند – لغت، بردار لغت (term vector) گفته می‌شود. توجه داشته باشید که این بردارها در مباحث و الگوریتم‌های مربوط به بازیابی اطلاعات و پردازش زبان طبیعی بسیار پر کاربرد می‌باشند.

با داشتن ماتریس لغت – سند می‌توان یک الگوریتم جستجو را پیاده سازی نمود. بسیاری از روش‌های جستجویی که تا کنون پیشنهاد شده اند نیز بر پایه چنین ماتریس هایی بنا شده اند. فرض کنید می‌خواهیم پرس و جویی با کلمات کلیدی "کروموزوم‌های انسان" اجرا کنیم. برای این منظور کافیست ابتدا کلمات کلیدی موجود در پرس و جو را استخراج کرده (در این مثال کروموزوم و انسان دو کلمه کلیدی ما هستند) و سپس به سراغ بردارهای هر یک برویم. همانطور که گفته شد با مراجعه به سطر یا ستون مربوط به لغات می‌توان بردار لغت مورد نظر را یافت. پس از یافتن بردار مربوط به کروموزوم و انسان می‌توان مشخص کرد که این لغات در چه اسناد و متونی اورده شده اند و آنها را استخراج و به کاربر نشان داد. این ساده‌ترین روش جستجو بر مبنای کلمات کلیدی می‌باشد. اما دقت داشته باشید که هدف نهایی در LSI چیزی فراتر از این است. بنابراین نیاز به انجام عملیاتی دیگر بر روی این ماتریس می‌باشد که بتوانیم بر اساس آن ارتباطات معنایی بین لغات و متون را تشخیص دهیم. برای این منظور LSI ماتری لغت – سند را تجزیه (decompose) می‌کند. برای این منظور نیز از تکنیک Singular Value Decomposition استفاده می‌نماید. پیش از

پرداختن به این تکنیک ابتدا بهتر است کمی با فضای برداری چند بعدی (multi-dimensional vector space) آشنا شویم. برای این منظور به مثال زیر توجه کنید. **مثالی از فضای چند بعدی**

فرض کنید قصد دارید تحقیقی در مورد اینکه مردم چه چیز هایی را معمولا برای صبحانه خود سفارش می‌دهند انجام دهید. برای این منظور در یک روز شلوغ به رستورانی در اطراف محل زندگی خود می‌روید و لیست سفارشات صبحانه را می‌گیرید. فرض کنید از بین اقلام متعدد، تمرکز شما تنها بر روی تخم مرغ (egg)، قهوه (coffee) و بیکن (bacon) است. در واقع قصد دارید ببینید چند نفر در سفارش خود این سه قلم را باهم درخواست کرده اند. برای این منظور سفارشات را تک تک بررسی می‌کنید و تعداد دفعات را ثبت می‌کنید.

پس از آنکه کار ثبت و جمع آوری داده‌ها به پایان رسید می‌توانید نتایج را در قالب نموداری نمایش دهید. یک روش برای اینکار رسم نموداری سه بعدی است که هر بعد آن مربوط به یکی از اقلام مذکور می‌باشد. بعنوان مثال در شکل زیر نموداری سه بعدی را که برای این منظور رسم شده است مشاهده می‌کنید. همانطور که در شکل نشان داده شده است محود x مربوط به "bacon"، محور y مربوط به "egg" و محور z نیز مربوط به "coffee" می‌باشد. از آنجایی که این نمودار سه بعدی است برای مشخص کردن نقاط بر روی آن به سه عدد (x ,y ,z) نیاز مندیم. حال اطلاعات جمع اوری شده از صورت سفارشات را یکی یکی بررسی می‌کنیم و بر اساس تعداد دفعات سفارش داده شدن این سه قلم نقطه ای را در این فضای سه بعدی رسم می‌کنیم. بعنوان مثال اگر در سفارشی 2 عدد تخم مرغ و یک قهوه سفارش داده شد بود، این سفارش با (0, 2, 1) در نمودار ما نمایش داده خواهد شد. به این ترتیب می‌توان محل قرار گرفتن این سفارش در فضای سه بعدی سفارشات صبحانه را یافت. این کار را برای تمامی سفارشات انجام می‌دهیم تا سر انجام نموداری مانند نمودار زیر بدست آید.



دقت داشته باشید که اگر از هریک از نقطه آغازین نمودار (0, 0, 1) خطی را به هر یک از نقاط رسم شده بکشید، بردار هایی در فضای "bacon-eggs-coffee"بدست خواهد آمد. هر کدام از این بردارها به ما نشان می‌دهند که در یک صبحانه خاص بیشتر از کدام یک از این سه قلم درخواست شده است. مجموع بردارها در کنار یکدیگر نیز می‌توانند اطلاعات خوبی راجع به گرایش و علاقه مردم به اقلام مذکور در صبحانه‌های خود به ما دهد. به این نمودار نمودار فضای بردار (vector – space) می‌گویند.

حالا وقت آن است که مجددا به بحث مربوط به بازیابی اطلاعات (information retrieval) باز گردیم. همانطور که گفتیم اسناد در یک مجموعه را می‌توان در قالب بردار هایی بنام Term – vector نمایش داد. این بردارها مشابه بردار مثال قبل ما هستند. با این تفاوت که به جای تعداد دفعات تکرار اقلام موجود در صبحانه افراد، تعداد دفعات تکرار لغات را در یک سند در خود دارند. از نظر اندازه نیز بسیار بزرگتر از مثال ما هستند. در یک مجموعه از اسناد ما هزاران هزار لغت داریم که باید بردارهای ما به اندازه تعداد کل لغات منحصر به فرد ما باشند. بعنوان مثال اگر در یک مجموعه ما هزار لغات غیر تکراری داریم بردارهای ما باید هزار بعد داشته باشند. نموداری که اطلاعات را در ان نمایش خواهیم داد نیز بجای سه بعد (در مثال قبل) می‌بایست هزار بعد (یا محور) داشته باشد که البته چنین فضایی قابل نمایش نمی‌باشد.

به مثال صبحانه توجه کنید. همانطور که می‌بینید برخی از نقاط بر روی نمودار نسبت به بقیه به یکدیگر نز دیکتر هستند و ابری از نقاط را در قسمتی از نمودار ایجاد کردند. این نقاط نزدیک به هم باعث می‌شوند که بردارهای آنها نیز با فاصله نزدیک به هم در

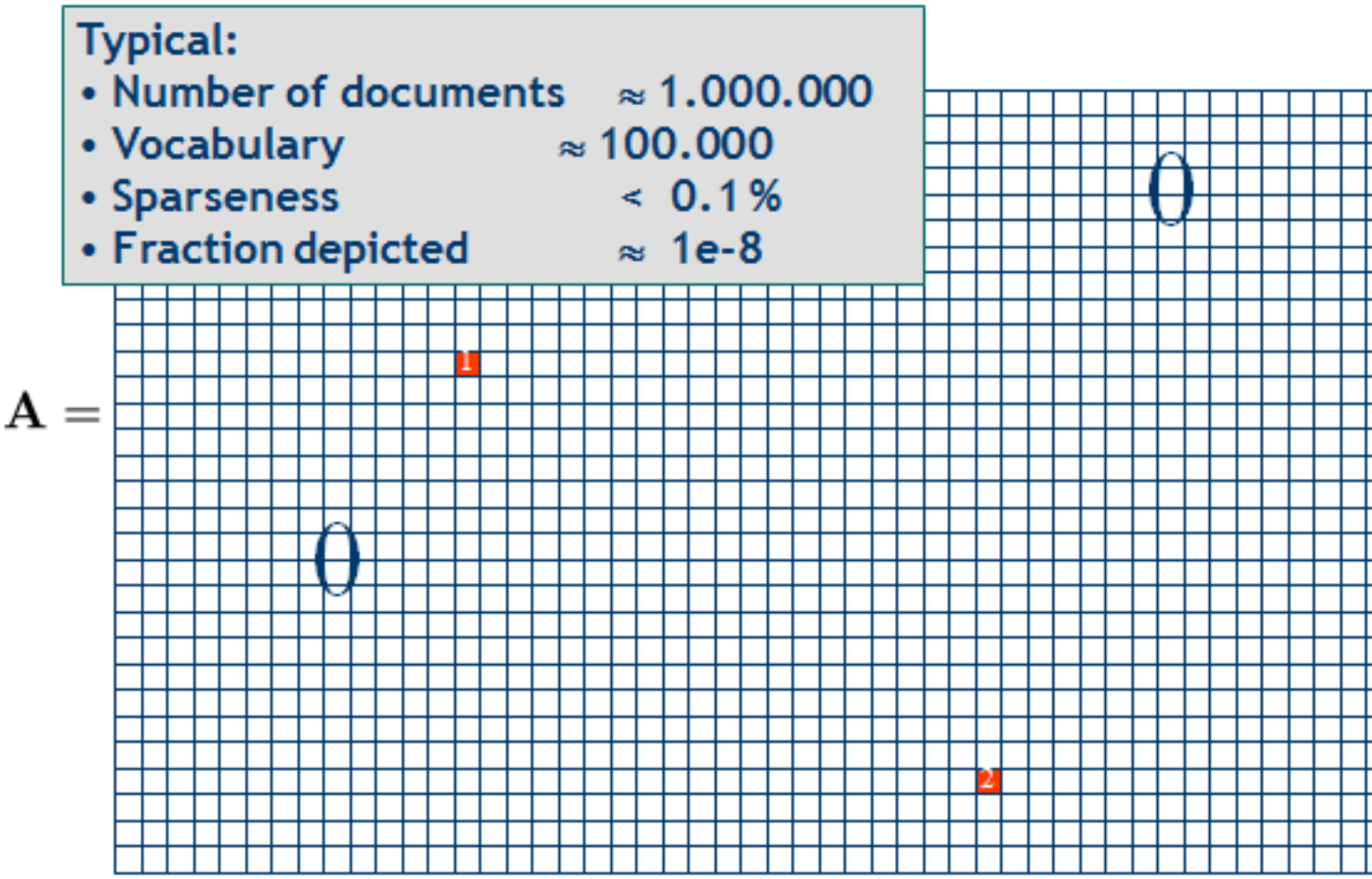
فضای برداری مثال ما قرار گیرند. علت نزدیک بودن این بردارها اینست که تعداد دفعات تکرار bacon، eggs و coffee در انها مشابه به هم بوده است. بنابراین می‌توان گفت که این نقاط (یا سفارشات مربوط به انها) به یکدیگر شبیه می‌باشند. در مورد فضای برداری مجموعه از اسناد نیز وضع به همین ترتیب است. اسنادی که لغات مشترک بیشتری با یک دیگر دارند بردارهای مربوط به انها در فضای برداری در کنار یکدیگر قرار خواهند گرفت. هر چه این مشترکات کمتر باشد منجر به فاصله گرفتن بردارها از یکدیگر می‌گردد. بنابراین می‌بینید که با داشتن فضای برداری و مقایسه بردارها با یکدیگر می‌توان نتیجه گرفت که دو سند چقدر به یکدیگر شباهت دارند.

در بسیاری از روش‌های جستجو از چنین بردار هایی برای یافتن اسناد مرتبط به پرس و جوی کاربران استفاده می‌کنند. برای ان منظور تنها کافی اس پرس و جوی کاربر را بصورت برداری در فضای برداری مورد نظر نگاشت دهیم و سپس بردار حاصل را با بردارهای مربوط به اسناد مقایسه کنیم و در نهایت آنهایی که بیشترین شباهت را دارند باز به کاربر بازگردانیم. این روش یکی از ساده‌ترین روش‌های مطرح شده در بازیابی اطلاعات است.

خوب حالا بیایید به Latent Semantic Indexing باز گردیم. روش LSI برمبنای همین فضای برداری عمل می‌کند با این تفاوت که فضای برداری را که دارای هزاران هزار بعد می‌باشد به فضای کوچکتری با ابعاد کمتر (مثلا 300 بعد) تبدیل می‌کند. به این کار در اصطلاح عملی کاهش ابعاد (dimensionality reduction) گفته می‌شود. دقت داشته باشید که هنگامیکه این عمل انجام می‌گیرد لغاتی که شباهت و یا ارتباط زیادی به لحاظ معنایی با یکدیگر دارند بجای اینکه هریک در قالب یک بعد نمایش داده شوند، همگی بصورت یک بعد در می‌آیند. بعنوان مثال لغات کروموزم و ژن از نظر معنایی با یکدیگر در ارتباط هستند. در فضای برداری اصلی این دو لغت در قالب دو بعد مجزا نمایش داده می‌شوند اما با اعمال کاهش ابعاد به ازای هر دوی آنها تنها یک بعد خواهیم داشت.
مزیت این کار اینست که اسنادی که لغات مشترکی ندارند اما به لحاظ معنایی با یکدیگر ارتباط دارند در فاضی برداری کاهش یافته نزدیکی بیشتری به یکدیگر خواهند داشت.

روش‌های مختلفی برای اعمال کاهش ابعاد وجود دارد. در LSI از روش Singular Value Decompistion استفاده می‌شود که در بحث بعدی در مورد آن صحبت خواهیم نمود. **Singular Value Decomposition**

پیشتر گفتیم که در LSI برای مدل کردن مجموعه اسناد موجود از ماتریس بزرگی بنام ماتریس لغت – سند استفاده می‌شود. این ماتریس در واقع نمایشی از مدل فضای برداری است که در بخش قبلی به آن اشاره شد. دقت داشته باشید که ما در دنیای واقعی در یک سیستم بزرگ تقریبا چیزی در حدود یک ملیون سند داریم که در مجموع این اسناد تقریبا صد هزار لغت غیر تکراری و منحصر به فرد یافت می‌شود. بنابراین می‌توان گفت میزان تنک بودن ماتریس ما تقریبا برابر با 0.1 درصد خواهد بود. یعنی از کل ماتریس تنها 0.1 درصد آن دارای اطلاعات است و اکثر سلول‌های ماتریس ما خالی می‌باشد. این مسئله را در شکل زیر می‌توانید مشاهده کنید.



در Latent Semantic Indexing با استفاده از روش Singular Value Decomposition این ماتریس را کوچک می‌کنند. به بیان بهتر تقریبی از ماتریس اصلی را ایجاد می‌کنند که ابعاد کوچکتری خواهد داشت. این کار مزایایی را بدنبال دارد. اول آنکه سطرها و ستون هایی (لغات و اسناد) که اهمیت کمی در مجموعه اسناد ما دارند را حذف می‌کند. علاوه بر آن این کار باعث می‌شود که ارتباطات معنایی بین لغات هم معنی یا مرتبط کشف شود. یافتن این ارتباطات معنایی بسیار در پاسخ به پرس و جوها مفید خواهد بود. چرا که مردم معمولا در پرس و جوهای خود از دایره لغات متفاوتی استفاده می‌کنند. بعنوان مثال برای جستجو در مورد مطالب مربوط به ژن‌های انسان برخی از واژه کروموزوم و برخی دیگر از واژه ژنوم و دیگران ممکن است از واژگان دیگری استفاده نمایند. این مسئله مشکلی را در جستجو بنام عدم تطبیق کلمات کلیدی (mismatch problem) بوجود می‌اورده که با اعمال SVD بر روی ماتریس سند – لغت این مشکل برطرف خواهد شد.

توجه داشته باشید که SVD ابعاد بردارهای لغات و سند را کاهش می‌دهد. بعنوان مثال بجای آنکه یک سند در قالب صد هزار بعد (که هر بعد مربوط به یک لغت می‌باشد) نمایش داده شود، بصورت یک بردار مثلا 150 بعدی نمایش داده خواهد شد. طبیعی است که این کاهش ابعاد منجر به از بین رفتن برخی از اطلاعات خواهد شد چرا که ما بسیاری از ابعاد را با یکدیگر ادغام کرده ایم. این مسئله شاید در ابتدا مسئله ای نا مطلوب به نظر آید اما در اینجا نکته ای در آن نهفته است. دقت داشته باشید که آنچه از دست می‌رود اطلاعات زائد (noise) می‌باشد. از بین رفتن این اطلاعات زائد منجر می‌شود تا ارتباطات پنهان موجود در مجموعه اسناد ما نمایان گردند. با اجرای SVD بر روی ماتریس، اسناد و لغات مشابه، مشابه باقی می‌مانند و انهایی که غیر مشابه هستند نیز غیر مشابه باقی خواهد ماند. پس ما از نظر ارتباطات بین اسناد و لغات چیزی را از دست نخواهیم داد.

در مباحث بعدی در مورد چگونگی اعمال SVD و همچنین نحوه پاسخگویی به پرس و جوها مطالب بیشتری را برای شما عزیزان خواهیم نوشت.

موفق و پیروز باشید.

نظرات خوانندگان

نویسنده: محمد رضا
تاریخ: ۱۳۹۳/۰۳/۱۰ ۱۰:۲۴

تشکر می‌کنم از مطلب مفیدتون
در این بازه منابعی دارید معرفی کنید ؟ بی صبرانه منظر بخش بعدی هستیم.
ممنون

نویسنده: حامد خسروجردی
تاریخ: ۱۳۹۳/۰۳/۱۴ ۲۱:۱۰

سلام دوست عزیز. از اونجایی که این روش سالهای زیادی است معرفی شده و مورد استفاده قرار گرفته (از اواخر دهه 90 میلادی) مقالات و منابع زیادی تو این حوزه منتشر شده تا بحال و بر روی اینترنت هم موجود است. ولی برای شروع می‌تونید سری به این لینک‌ها بزنید :

لینک زیر بطور آکادمیک توضیحاتی را در مورد Latent Semantic Analysis ارائه میده:
An introduction To Latent Semantic Analysis

این لینک مربوط به دانشگاه استندفورد هستش و واقعا یه مرجع عالی در مورد روش‌های مختلف بازیابی اطلاعات (Information Retrieval) هستش که اگر علاقه به سایر حوزه‌ها تو این زمینه دارید می‌تونید بعنوان یه مرجع خوب ازش استفاده کنید : Latent semantic indexing

اگر هم شرحی عامیانه‌تر از این مقوله می‌خواهید می‌تونید به این لینک سری بزنید : LATENT SEMANTIC INDEXING

نویسنده: محسن
تاریخ: ۱۳۹۳/۰۳/۲۱ ۱۲:۲۳

سلام
ممنون از مقاله جالبتون
آیا برنامه پیاده سازی شده ای هم وجود داره؟
نسخه ایرانی یا خارجی؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۲۱ ۱۲:۵۶

Semantic Search جزو تازه‌های SQL Server 2012 است (البته این مورد خاص، زبان‌های محدودی را پشتیبانی می‌کند).