ALLAN THRAEN | 6 years ago | PDF |

CMS Vision Demos & Prototypes Tips and Tricks Information Retrieval ElasticSearch AI .NET Development

AUTO TAGGING USING SEARCH

You don't always have to go the full AI route to get AI like results. In this blog post I'll describe an approach I've used several times (and for multiple purposes) with pretty decent results. Instead of classification algorithms, deep learning or neural networks I'll just simply query my favorite search engine.

As I tend to often do, I'll start this blog post with a personal anecdote. AI in combination with text/content analysis has always been a bit of a passion of mine. Back, what feels like a lifetime ago, I was working in Mondosoft (dotcom search vendor) and we had 2 master students come in to do their master thesis with us about automatic text classification (into predefined categories). Being a search vendor we had always wanted to do more for our customers than 'just search'. Now, these guys were some of the greatest technical minds I have worked with (and I've worked with a lot!) and they both went on to pursue what turned out to be great careers (you know who you are :-)) - and they did a great job over those 6 months, taking a variety of the best known AI algorithms and running them against big collections of text to classify them. When you are trying to teach computers how to do a task like that, that usually requires human intelligence, the key thing is of course to find a good way to evaluate if they are doing a decent job. And if they are at least doing something like what a human would have done. So, I assisted in setting up a small web site to test it. We basically asked a bunch of humans to have a look at a short text and put it into a number of categories. Then, we asked the various algorithms to do the same.

Just for fun, I decided to make a 'dumb baseline' to compare against. I took our search engine, Mondosearch, extracted the highest weighted words from the document being evaluated, did a search for those words in the entire corpus (except the document itself) and looked at the best results that had already been categorized (the training set) to see which category was most common there, and returned that. Easy peasy and very simple.

The funny thing was that in the overall evaluation, comparing to human classification my dumb approach didn't that that bad at all. Of course - some of the best AI approaches yielded better results - but not by a lot. I don't remember the exact numbers - but when you looked at the percentage of times the algorithms picked the same catagory as the humans, my dumb approach was only ~5% lower than the best algorithms! And then of course you have to start thinking about where is the 'good-enough-cutoff-point' :-)

Looking at StackExchange data once again

A favorite AI/BigData challenge site I enjoy, Kaggle.com, some years ago had a competition where you should make an algorithm that could suggest tags for new StackExchange questions, based on the knowledge trained from all the previous. Co-incidentally at the time of that competition I was giving a talk in the Netherlands about Episerver Find (that was recently introduced in the market at that time). As a little example in my 45 min. talk I coded a solution to that competition using a pre-made index of StackExchange questions and their tags, I had. It was more of a joke, so I never measured it's accuracy or submitted the solution obviously. But it was one more time where this technology actually worked.

Since I in a few recent posts, again explored the great dataset of questions from StackExchange - and even added them to ElasticSearch with good results, I figured the time was right to share this approach with you all.

My motivation for this post is of course not related to the fact that ElasticSearch (ESTC) went public through a successful IPO last friday and I instantly decided to invest :-) But I digress...

I added a bit more data to my index, so I have now have around 6 million questions indexed. Now, I can then basically just query it with a MoreLikeThis query, get the top 30 results and aggregate which tags are most common. Why 30 you ask? Well - it turned out to provide better results than with 100 (Which more shows off the 'most common tags').

Here is the code for the query:

```
private string[] TagsForDocument(string text)
                                    {
            var res = GetClient().Search<Question>(s => s.Query(q =>
                  q.MoreLikeThis(mlt =>
                      mlt.Like(l =>
                        l.Text(text)
                      )
                      .MinimumShouldMatch(1)
                      .MinTermFrequency(1)
                      .Fields(fi => fi.Field(f => f.Body).Field(f => f.Title))
                  ))
                  .Take(30)
                );
            var otherTags = res.Documents.SelectMany(d => d.TagList)
                .GroupBy(g => g).Select(t => new { Tag = t.Key, Count = t.Count() })
                .OrderByDescending(t => t.Count).Take(10).Select(t => t.Tag).ToList();
            return otherTags.ToArray();
        }
```

It turns out, that it can also be used to suggest tags for technical blog posts - umm - I wonder where I can find a use for that.

The results

To get an indication about how well this was working out, I tested it against some questions not in the index (a test set).

I ran through 500 questions and checked to see if the actual tags used for the posts where in the top 10 suggested tags the algorithm had come up with. All of the tags where matched in 318 times out of the 500. Keeping in mind that human tagging is not perfect either, this was pretty good.

All in all, I mathed up 1226 tags to the suggested, and only had 248 tags that were not in the top 10 suggested!

Below, you can try it yourself! Either provide a url to a technical blog post, or copy and paste in some text you want to see tag suggestions for.

CMS Vision Demos & Prototypes Tips and Tricks Information Retrieval ElasticSearch AI .NET Development

Storage Performance Aftermath - ElasticSearch Joins the Fight

Kaggle Competitions

CodeArt ApS

Teknikerbyen 5, 2830 Virum, Denmark

Email: info@codeart.dk

Phone: +45 26 13 66 96

CVR: 39680688