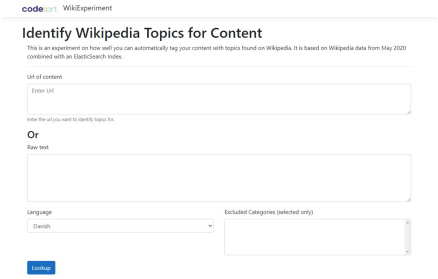
ALLAN THRAEN | 3 years ago | PDF |

Vision Demos & Prototypes CMS Information Retrieval Big Data ElasticSearch .NET Development

EXPERIMENTING WITH WIKIPEDIA TOPICS FOR CONTENT



Automatically tagging your content with topics from a known, well described topic base like Wikipedia can have many cool uses. You can organize your content, suggesting keywords and outbound links, not to mention that you can build up interest profiles of your visitors. These interest profiles can the be used to suggest appropriate content and keep your visitors engaged. Inspired by Episerver Content Intelligence and a couple of earlier projects I've done in the past, I decided to perform an experiment to see how far I could get with a DIY approach as opposed to the traditional cloud-based NLP/AI.

Background

Frequent readers of this blog will know that I have a many years old passion for search and information retrieval - and automatic text classification / clustering is a topic in the beautiful middle ground between intelligent content and information retrieval that I've always been deeply fascinated with.

In the past, I have for example used the StackOverflow dataset to auto-tag technical text (Auto Tagging Using Search) with a rather simple approach: Instead of doing a full AI neural network training or Natural Language Processing of text to train against certain categories, I simply index all the content in a good search engine (these days ElasticSearch is my weapon of choice) and then I do a "more-like-this" query with the content I want to identify topics for. That way I'm leveraging the TF-IDF algorithm in the search engine and letting it do the job for me, fast and efficient. Sure, it might not be as good as a properly trained AI - but the only thing that matters is really if it's 'Good enough' for the use I'm aiming at.

Earlier this year, Episerver launched Content Intelligence (formerly Idio) - and I've been happy to try it out and play around with it (Idio: First look on Content Analytics and Recommendations). In fact, you can probably see recommended content coming from it just at the bottom of this blog post. One of the things I actually liked with Idio was that it identifies topics from a set of well described topics. Described in a way that almost looks like english Wikipedia topics.

Now, Episerver Content Intelligence is a powerful, fully developed cloud based tool that I strongly encourage everybody to try (and I understand that for a limited time it's even possible to get a free trial of some of the awesome functionality). But I'm the kind of guy that always wants to build something himself - not for production use - but as an experiment to understand the potentials of new technology. In fact I once built my own compression-algorithm (called it AllanZip) to fully understand Huffman compression - but that's a story for another time. So, an evening last week I started on this little DIY Wikipedia Content Classification experiment. Consider it something in-between fan fiction and those youtube videos where people try to recreate their favorite fastfood dishes at home.

The build

I fairly quickly found a place online where I could download wikipedia search indexes as json files. They came as some extremely large gzipped files - but processing large files isn't all that tricky when you know how to(Reading very large gzipped json files in c#).

Then I indexed all text, titles, categories and such into my own ElasticSearch server (something every household should have).

It took a while, but I did it language by language. At the time of writing, I have Danish, Swedish, Norwegian, Finnish, Dutch, German, French, Spanish, Russian and of course English (which was by far the largest). My index has around 21M records and takes up approx 150 Gb on my server. It might seem like a lot - but not to ElasticSearch which runs smoothly (while the server also does a lot of other things).

After that, all that was left to do was to put together some clever queries and build a very basic UI to test it out.

In the UI you can either enter one (or more) urls, or some text. You will get the best result by copying and pasting in text - but if you are lazy you could also just use a url. The problem with urls is that it'll just fetch all text on the url - not worry about if it's the main content or not - so you might get some noise. If you enter multiple urls, it will pretend that a user has visited those urls and tries to identify the common themes to get the topics the user is interested in.

You should also remember to select which language you are analyzing, this sets which wikipedia language we'll be finding topics in.

Once you have identified some topics you can fine tune them by excluding categories of topics - as there can be a lot of noise in Wikipedia articles.

You can try it out here: https://codeartwiki.azurewebsites.net/

Example results

Here are the suggested topics found if I look up this blog post https://www.codeart.dk/blog/2020/3/free-online-course---c-and-.net-core-introduction/. It's a blog post I wrote about a free .NET Core/ C# course I made available a few months ago, teaching students a bunch of different tools and technologies while letting them build a game.

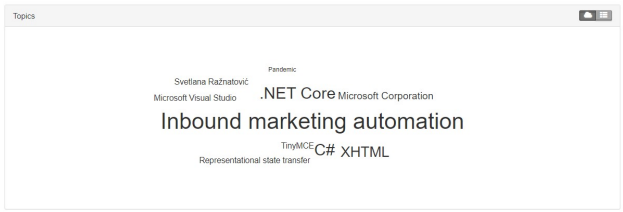
If I analyze it with my wiki experiment I get this:

Topics:

Topics:

ASP.NET Core	100 %
List of .NET libraries and frameworks	98 %
List of educational programming languages	98 %
Bootstrap curriculum	95 %
ASP.NET Razor	94 %
Component-based Scalable Logical Architecture	94 %
ProgramByDesign	92 %
FreeCodeCamp	92 %
.NET Framework	89 %
International Information Technology University	89 %
Hack Reactor	88 %
ZK (framework)	87 %
C1 CMS	87 %
Pre-assessment	86 %
Course Hero	86 %
Tynker	86 %
Virtual learning environment	85 %

And for comparison, if I look in Content Intelligence I get these topics:



As you can see I still have a lot more noise compared to the more cleanly selected topics below - But the main detected topic in the bottom screenshot "Inbound marketing automation" has absolutely nothing to do with the blog post topic - something that just goes to show - this is not an exact science :-)

Feel free to try out the search with some of your content - and ideally in different languages - and let me know what you think in the comments below!

Vision Demos & Prototypes | CMS | Information Retrieval | Big Data | ElasticSearch | .NET Development

RECENT POSTS

CodeArt ApS
Teknikerbyen 5, 2830 Virum, Denmark
Email: info@codeart.dk
Phone: +45 26 13 66 96
CVR: 39680688