Assignment 5

CS 734: Introduction to Information Retrieval

Fall 2017

Grant Atkins

Finished on December 16, 2017

1

Question

Answer

2

Question

```
10.6. Find two examples of document filtering systems
on the Web. How do they build a profile for your
information need? Is the system static or adaptive?
```

Answer

Two examples of document filtering systems on the web are Amazon and Medium. When buying tech accessories such as HDMI cables, keyboards, or mouses I usually go to Amazon. Amazon keeps track of the things I browse as well as what I previously bought and attempts to make recommendations as shown in Figure 1. This kind of filtering is an adaptive system. If I start looking at a specific book frequently, Amazon will eventually start recommending the same book if I haven't bought it, or books that are similar to the one I viewed.

Figure 1: Amazon recommended buys

Another example of a document filtering system is Medium, a popular article sharing website much like blogspot. I often enjoy reading tech related articles for machine learning, javascript, and much more on Medium. Medium keeps track of users viewing history and then when a user visits the home page of the website it offers them lists of articles to read. It has a "You might like" section where for each article it even says "Based on your interests." This shows that it has filtering in place that is adaptive to my viewing history and tries to find the best trending/matching documents for my reading. My results are shown in Figure 2. If I decide to start reading other topics a majority of the time then this filtering system will be updated over time on its own.

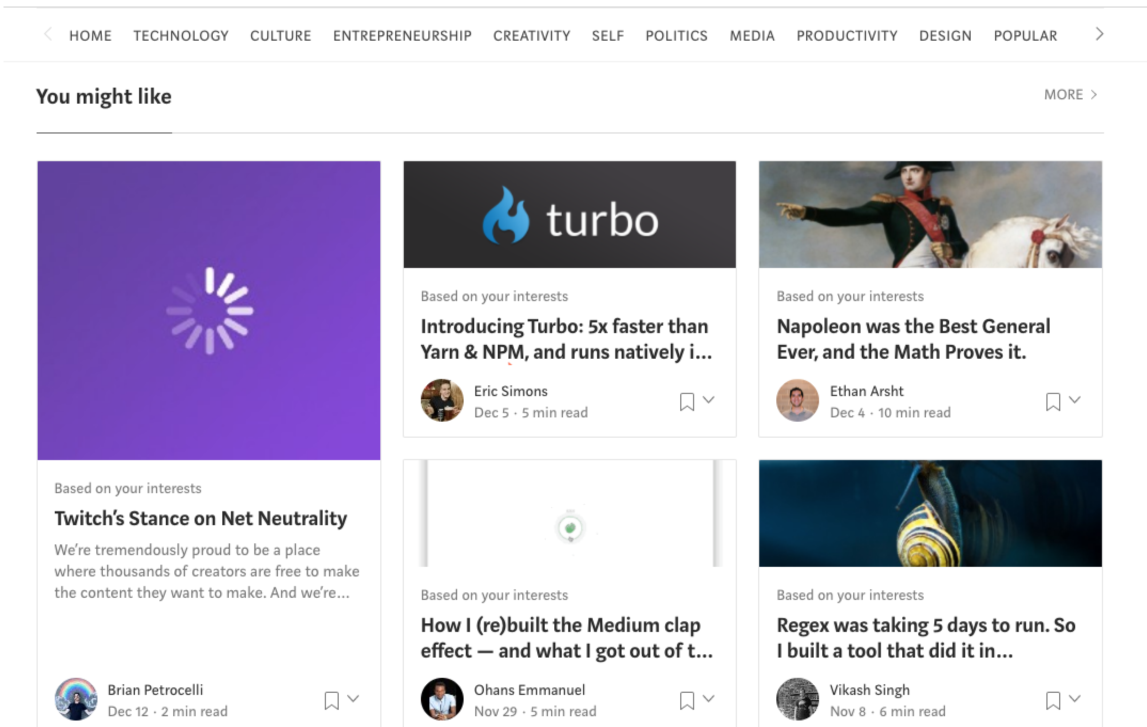

Figure 2: Medium recommended reads

3

Question

```
10.11. Suggest how the maximum and minimum resource
ranking scores, Rmax and Rmin, could be estimated
for a given query
```

Answer

A relatively simple way to find the maximum and minimum resource ranking scores is by sorting results. Once query results are returned and we have the resource scores for each estimate it should be relatively to see the worst ranked document and the highest rank document. This is also dependent on the window of results. Since resource selection in a general distributed search application involves first ranking the nodes using their representations, and then selecting the top k ranked nodes, or all nodes that score above some threshold value. If I searched something in a search engine we would be given their top k results, normally 10, and then we can determine that the Rmax is the first result return. Since k is 10 here then Rmin would be 10, but if k increased it would still be considered the max k.

The book also mentions using query likelihood of query term frequency. A query's likelihood could be compared to the terms returned in SERP results and then Rmin and Rmax can computed based on comparison of these values. Either way, the resources would be ranked showed that there is a max and min value in the results.

4

Question

```
11.5. How many papers dealing with term dependency
can you find in the SIGIR proceedings since 2000?
List their citations
```

Answer

To solve this problem I decided to use google scholar to find the citations. I searched on the terms "term dependency," "term dependencies," and "term dependent" with ACM SIGIR as the source and all publication later than 2000. For this query I got 129 results as shown in Figure 3. Whats interesting is when I went back later this result changed going up to 155 related articles, but regardless I kept what I originally had.

Google creates a query string that I could reuse to perform these queries, for example:

`https://scholar.google.com/scholar?start=0&q=%22term+dependency%22+%7C+%22term+dependencies%22+%7C+%22term+dependent%22+source:%22ACM+SIGIR%22&hl=en&as_sdt=0,47&as_ylo=2000`

One of the keys is "start" indicating an offset position for pagination. Since there were 10 results per page I created two scripts to download each page and then parse the results. The first script I wrote was done in nodejs using headless chrome, puppeteer.js, as shown in Listing 1. The pages downloaded pages are found in my assignment 5 github repository [1]. The other script was used to parse the document for titles, citation counts, and author names as shown in Listing 2. For this question I left out author names as it was difficult to parse utf-8 for latex.

The results are as follows:

Title	**Citations**
A Markov random field model for term dependencies	776
Incorporating term dependency in the DFR framework	56

Modeling higher-order term dependencies in information retrieval using query hypergraphs	54
Incorporating query term dependencies in language models for document retrieval	19
Modeling term dependencies with quantum language models for IR	21
Refining term weights of documents using term dependencies	9
Score-safe term-dependency processing with hybrid indexes	10
Utilizing phrase based semantic information for term dependency	0
Automatic P hrase Indexing for Document Retrieval: An Examination of Syntactic and Non-Syntactic Methods	169
Improving Search using Proximity-Based Statistics	1
Two-stage query segmentation for information retrieval	49

Dependence language model for information retrieval	294
Random walk term weighting for information retrieval	35
A comparison of various approaches for using probabilistic dependencies in language modeling	16
Latent concept expansion using markov random fields	226
Word embedding based generalized language model for information retrieval	49
Blog track research at TREC	56
Improving weak ad-hoc queries using wikipedia asexternal corpus	103
Modelling term dependence with copulas	8
A study of Poisson query generation model for information retrieval	52
A Markov random field model for term dependencies	776
Incorporating term dependency in the DFR framework	56

Modeling higher-order term dependencies in information retrieval using query hypergraphs	54
Incorporating query term dependencies in language models for document retrieval	19
Modeling term dependencies with quantum language models for IR	21
Refining term weights of documents using term dependencies	9
Score-safe term-dependency processing with hybrid indexes	10
Utilizing phrase based semantic information for term dependency	0
Automatic P hrase Indexing for Document Retrieval: An Examination of Syntactic and Non-Syntactic Methods	169
Improving Search using Proximity-Based Statistics	1
Robust ranking models via risk-sensitive optimization	41

An unsupervised topic segmentation model incorporating word order	28
Extending BM25 with multiple query operators	14
Taily: shard selection using the tail of score distributions	31
Sentiment diversification with different biases	19
An IR-based evaluation framework for web search query segmentation	15
To index or not to index: time-space trade-offs in search engines with positional ranking functions	17
Leveraging user interaction signals for web image search	5
Query representation for cross-temporal information retrieval	5
User variability and IR system evaluation	27
Ranking document clusters using markov random fields	13

Timestamp-based result cache invalidation for web search engines	27
Pseudo test collections for training and tuning microblog rankers	17
Terms over LOAD: Leveraging Named Entities for Cross-Document Extraction and Summarization of Events	9
Query-performance prediction: setting the expectations straight	6
A context-aware time model for web search	18
Optimizing positional index structures for versioned document collections	9
Learning for Efficient Supervised Query Expansion via Two-stage Feature Selection	2
[PDF][PDF] Introduction to Probabilistic Models for Information Retrieval	4
Finding topic words for hierarchical summarization	219

A comparison of sentence retrieval techniques	30
Beyond bags of words: Effectively modeling dependence and features in information retrieval	4
MRF based approach for sentence retrieval	6
Integrating word relationships into language models	184
Social annotation in query expansion: a machine learning approach	39
The seventeenth australasian document computing symposium	0
An exploration of proximity measures in information retrieval	245
A proximity language model for information retrieval	69
Structured retrieval for question answering	106
Exploiting term dependence while handling negation in medical search	21

Positional language models for information retrieval	198
Parameterized concept weighting in verbose queries	100
The role of knowledge in conceptual retrieval: a study in the domain of clinical medicine	73
Fielded sequential dependence model for ad-hoc entity retrieval in the web of data	33
Looking inside the box: Context-sensitive translation for cross-language information retrieval	16
Embedding-based Query Expansion for Weighted Sequential Dependence Retrieval Model	0
Integrating phrase inseparability in phrase-based model	10
A support vector method for optimizing average precision	591
Modeling multi-query retrieval tasks using density matrix transformation	4

Empirical development of an exponential probabilistic model for text retrieval: using textual analysis to build a better model	25
Building a web test collection using social media	3
Improving the estimation of relevance models using large external corpora	191
Discovering key concepts in verbose queries	262
Building and applying a concept hierarchy representation of a user profile	80
An improved markov random field model for supporting verbose queries	56
CRTER: using cross terms to enhance probabilistic information retrieval	32
Parameterized fielded term dependence models for ad-hoc entity retrieval from knowledge graph	10

Query term ranking based on dependency parsing of verbose queries	28
The score-distributional threshold optimization for adaptive binary classification tasks	69
Learning to reweight terms with distributed representations	35
Building simulated queries for known-item topics: an analysis using six european languages	97
Unsupervised query segmentation using clickthrough for information retrieval	46
Compact query term selection using topically related text	38
A frequency-based and a poisson-based definition of the probability of being informative	39
Positional relevance model for pseudo-relevance feedback	161
Learning for search result diversification	42
Exploring reductions for long web queries	74

Using statistical decision theory and relevance models for query-performance prediction	25
Exploiting semantics for improving clinical information retrieval	21
Modeling subset distributions for verbose queries	11
Axiomatic analysis for improving the log-logistic feedback model	8
Topic-centric classification of twitter user's political orientation	12
Sigir 2014 workshop on semantic matching in information retrieval	4
Discriminative models for information retrieval	314
Retrieval and feedback models for blog feed search	150
A ranking approach to target detection for automatic link generation	9
Flat vs. hierarchical phrase-based translation models for cross-language information retrieval	3

Non-compositional term dependence for information retrieval	7
Query performance prediction in web search environments	169
Building enriched document representations using aggregated anchor text	62
How good is a span of terms?: exploiting proximity to improve web retrieval	47
Query expansion using path-constrained random walks	25
An adaptive evidence weighting method for medical record search	14
Learning to efficiently rank	74
Exploiting proximity feature in bigram language model for information retrieval	6
Efficient cost-aware cascade ranking in multi-stage retrieval	3
Set-based model: A new approach for information retrieval	38
Modeling click-through based word-pairs for web search	3

Term Proximity Constraints for Pseudo-Relevance Feedback	0
Improving language estimation with the paragraph vector model for ad-hoc retrieval	15
Learning to respond with deep neural networks for retrieval-based human-computer conversation system	31
Find-similar: similarity browsing as a search tool	55
Linear discriminant model for information retrieval	118
Quote Recommendation in Dialogue using Deep Neural Network	2
Query term ranking based on search results overlap	1
Document retrieval using entity-based language models	15
Modeling Document Novelty with Neural Tensor Network for Search Result Diversification	10

An enhanced context-sensitive proximity model for probabilistic information retrieval	6
Efficient & Effective Selective Query Rewriting with Efficiency Predictions	0
A Probabilistic Model for Information Retrieval Based on Maximum Value Distribution	3
Using key concepts in a translation model for retrieval	8
Evaluating non-deterministic retrieval systems	3
A simple enhancement for ad-hoc information retrieval via topic modelling	4
Retrieval sensitivity under training using different measures	14
Copulas for information retrieval	19
DBpedia-Entity v2: A Test Collection for Entity Search	1
On the cost of phrase-based ranking	1
SPOT: Selecting occuPations frOm Trajectories	0

Entity query feature expansion using knowledge base links	84

```
// v0.12.0 puppeteer
const puppeteer = require('puppeteer');
const fs = require('fs');
const crypto = require('crypto');

var outputDir = './data/html/';

async function headless(url, offset) {

    var sleep_time = Math.floor(Math.random() * 15) + 1;
    await delay(sleep_time * 1000);

    const browser = await puppeteer.launch({
        ignoreHTTPSErrors: true,
        // headless: false,
    });
    const page = await browser.newPage();

    page.emulate({
        viewport: {
            width: 1024,
            height: 768,
        },
        // macbook pro
        userAgent: "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.94 Safari/537.36",
    });

    try {

        // timeout at 5 minutes (5 * 60 * 1000ms), network idle at 3 seconds
        await page.goto(url, {
            waitUntil: 'load',
            timeout: 300000,
        });

        // Get page content (html, xml, etc)
        const cont = await page.content();
```

```
        await fs.writeFile(outputDir + offset +".html", cont, function(err) {
            if (err) {
                return console.log(err);
            }
            console.log("The file was saved!");
        });

    } catch (e) {
        console.log("Failed with error:", e);
        process.exit(1);
    }

    browser.close();
}

function delay(time) {
   return new Promise(function(resolve) {
       setTimeout(resolve, time)
   });
}

// iterate through and save html
for(var i = 0; i < 13; i++){
    var temp = i;
    if (i != 0){
        temp = i * 10;
    }
    var url = "https://scholar.google.com/scholar?start=" + temp + "&q=%22term+dependency%22+%7C+%22term+dependencies%22+%7C+%22term+dependent%22+source:%22ACM+SIGIR%22&hl=en&as_sdt=0,47&as_ylo=2000";
    headless(url, temp).then(v => {
        // Once all the async parts finish this prints.
        console.log("Finished Headless");
        console.log(url);
    })
}
```

Listing 1: Headless chrome script to download rendered pages

```
from bs4 import BeautifulSoup
import os
import json


def parse_scholar():
    directory = "./data/html/"
    titles = []
```

```
    for filename in os.listdir(directory):
        with open(directory + filename) as f:
            print(filename)
            text = f.read()
            soup = BeautifulSoup(text, 'html.parser')
            # citations
            sections = soup.select("div.gs_ri")
            for s in sections:
                temp = {"title": "", "citations": "", "authors": ""}
                title = s.select_one("h3.gs_rt")
                # titles.append(title.text.strip())
                authors = s.select_one("div.gs_a")
                temp["title"] = title.text.strip()
                temp["authors"] = authors.text.strip()

                cite_count = s.select("div.gs_fl a")[2].text
                if not cite_count.startswith("Cited by"):
                    cite_count = "Cited by 0"
                cite_count = cite_count.split()[2]
                temp["citations"] = cite_count
                titles.append(temp)

    with open("./data/articles.json", 'w') as out:
        json.dump(titles, out, indent=4)

    format_latex(titles)


def format_latex(titles):
    for t in titles:
        print(t["title"].replace("&", "\\&") + " & " +
              t["citations"] + " \\\\ \\hline")


if __name__ == "__main__":
    parse_scholar()
```

Listing 2: Script to parse google scholar HTML pages

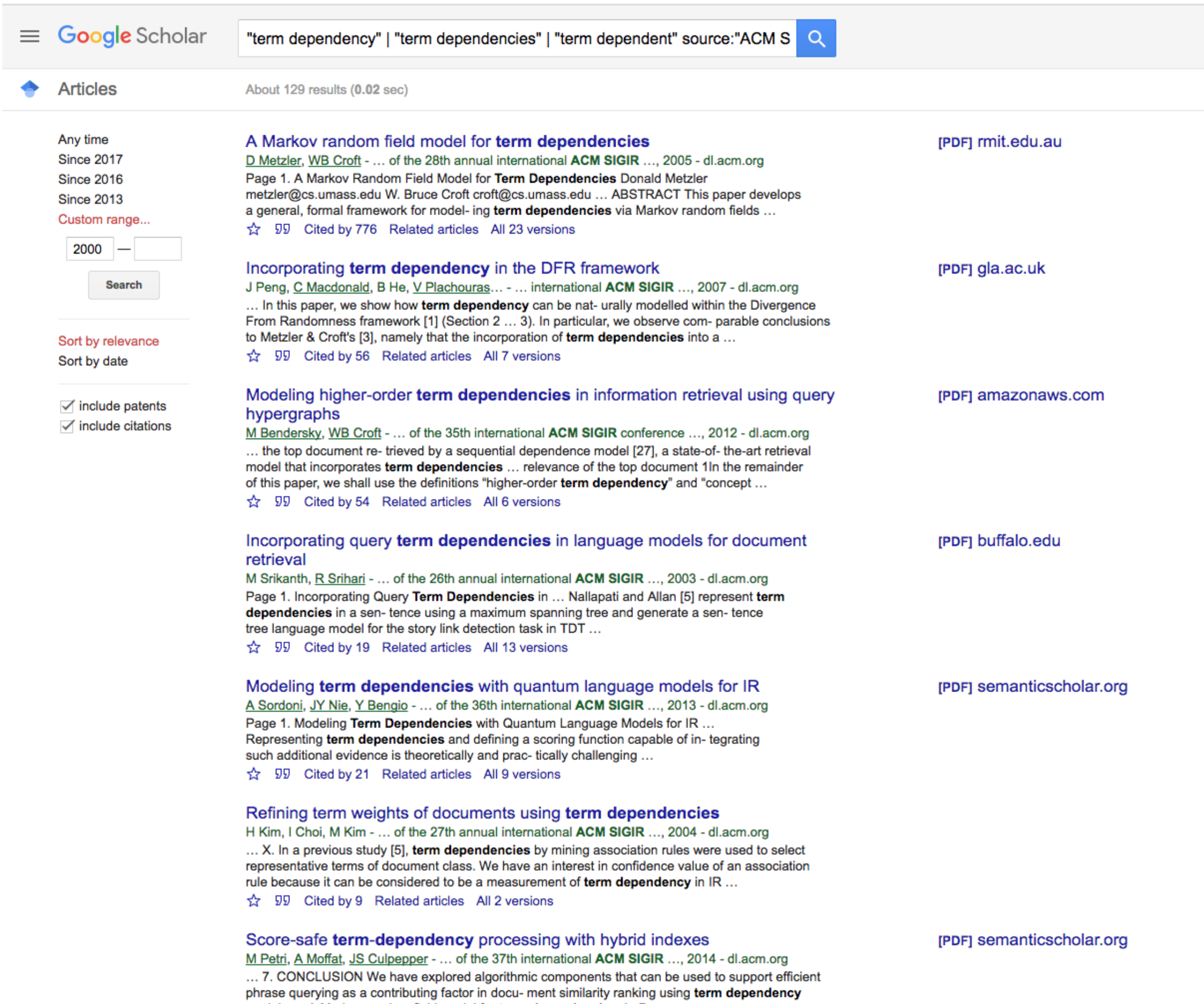

Figure 3: Google scholar results for term dependency articles later than 2000 for ACM SIGIR

5

Question

```
11.11. Look at a sample of images or videos that have been
tagged by users and separate the tags into three groups:
those you think could eventually be done automatically
by image processing and object recognition, those you think would
not be possible to derive by image processing, and spam. Also
decide which of the tags should be most useful for queries
related to those images. Summarize your findings.
```

Answer

To answer this question I decided to use Instagram to find images tagged by users. Instagram, much like Twitter, allows for users to tag photos and posts using a # symbol. Its apparent that many posts on these sites bring in spam tags to get higher views or for other reasons.

For this problem I first started with a base hashtag "#lamborghini" and got a lot of unexpected results as shown in Figure 4. Just from the top six results, only half of the results were lamborghini model cars. You can tell just from these results that the non lamborghini tags were probably just spam tags.

When choosing one of the images from result page, such as Figure 5, its seen that the result is actually lamborghini was a spam tag for this image. The tags used were:

- Ferrari
- PaganiHuaryaBC
- pagani
- supercars
- fast
- speed
- london
- omg
- khk

- harrods
- city
- nightlife
- lamborghini
- astonmartin
- bugatti
- like4like
- likeforfollow
- followforfollow
- follow4follow
- rich
- wealth
- earth
- lifestyle

A majority of these tags would end up in spam. There are, however, a few of the tags that could be derived from image processing such as: supercars, bugatti, and city. If a image processing is done, assuming it is accurate, then the tags that address other types of cars, such as Ferrari, PaganiHuaryaBC, lamborghini, astonmartin, and paganic, should not be grouped together with the actual descriptors of this image. Tags that could not be taken from image processing, or rather should not, are things like wealth and earth since the first one is opinionated and earth is too broad of a topic. Tags such as likeforlike, follow4follow, omg, speed, wealth, and a few others, are more of spam tags for this image. The lamborghini tag itself is also considered spam for Figure 5, as the car isn't even a lamborghini.

The tags that would be most useful for queries would be the objects actually located inside the image. For example, bugatti is a good descriptor of the car in the image as its exactly whats in the image and it is also considered a supercar. However, if we search for supercar then we'll get much broader search results. So if I had to choose tags from this image, I would probably only choose bugatti if I wanted concise results. Of course there are many tags that could be used for recommendations based off of this tag, such as rich, city, or supercar that could be recommended to users.

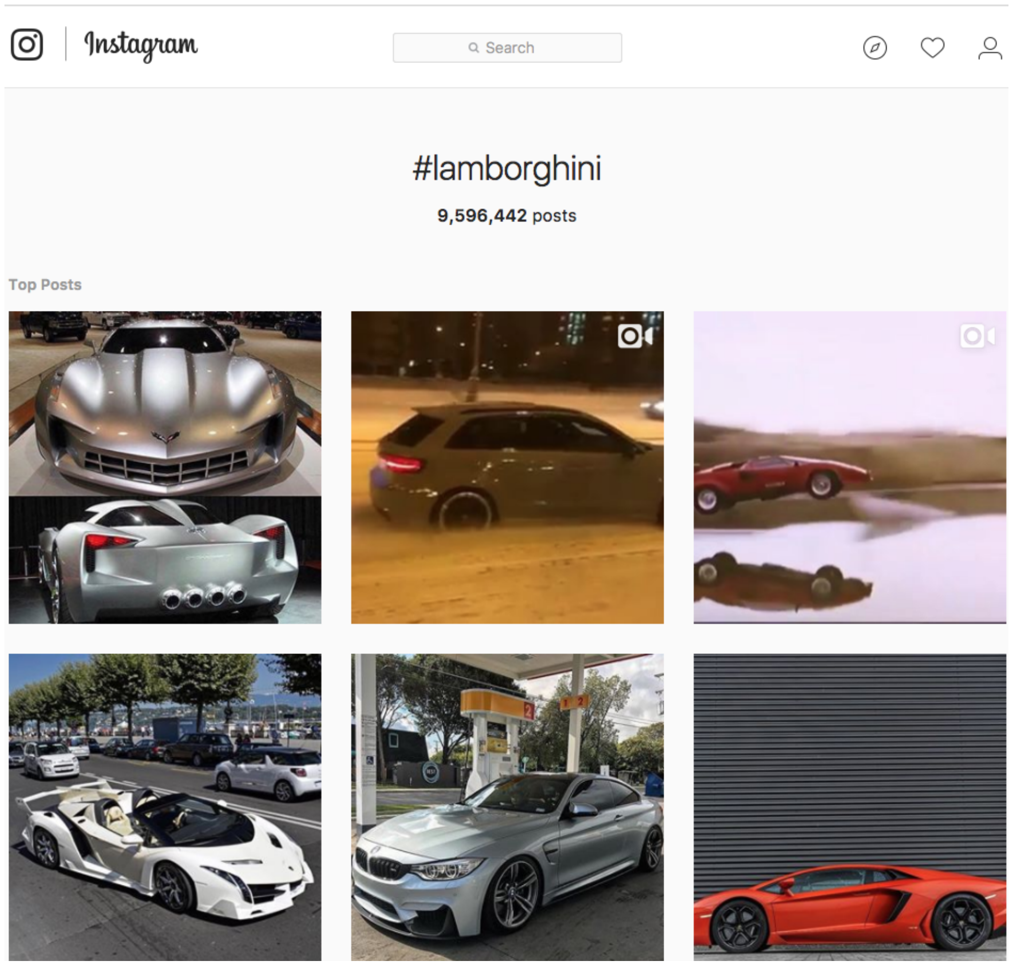

Figure 4: Results page from Instagram

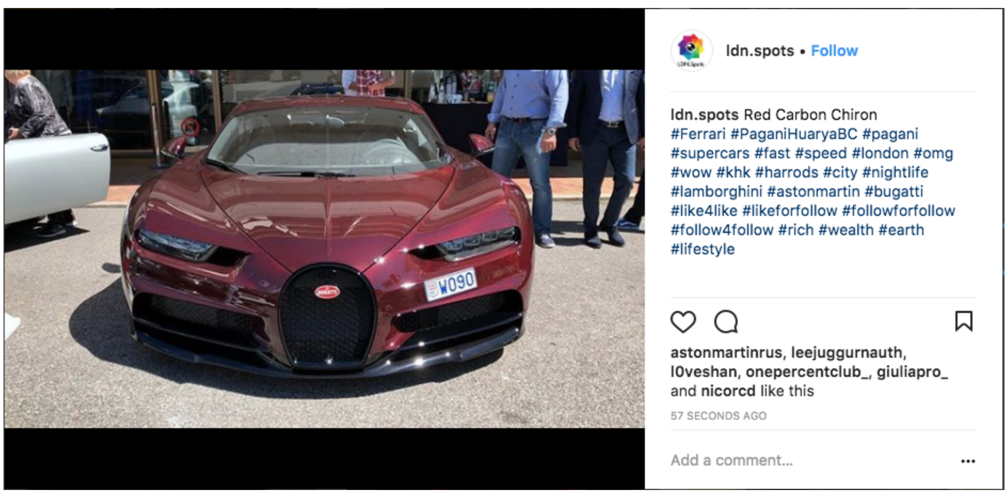

Figure 5: Selected result from Instagram for #lamborghini tag

References

[1] Atkins, Grant. “CS734 Assignment 5 Repository” Github. N.p., 15 December 2017. Web. 15 December 2017.https://github.com/grantat/cs834-f17/tree/master/assignments/A5.