Strings and Things: A Semantic Search Engine for news quotes using Named Entity Recognition.

Panos Kostakos
Center of Ubiquitus Computing
University of Oulu
Oulu, Finland
panos.kostakos@oulu.fi

***Abstract*—Emerging methods for content delivery such as quote-searching and entity-searching, enable users to quickly identify novel and relevant information from unstructured texts, news articles, and media sources. These methods have widespread applications in web surveillance and crime informatics, and can help improve intention disambiguation, character evaluation, threat analysis, and bias detection. Furthermore, quote-based and entity-based searching is also an empowering information retrieval tool that can enable non-technical users to gauge the quality of public discourse, allowing for more fine-grained analysis of core sociological questions. The paper presents a prototype search engine that allows users to search a news database containing quotes using a combination of *strings* and *things*. The ingestion pipeline, which forms the backend of the service, comprises of the following modules i) a crawler that ingests data from the GDELT Global Quotation Graph ii) a named entity recognition (NER) filter that labels data on the fly iii) an indexing mechanism that serves the data to an Elasticsearch cluster and iv) a user interface that allows users to formulate queries. The paper presents the high-level configuration of the pipeline and reports basic metrics and aggregations.**

***Index Terms*—Elasticsearch, search engine, Named Entity Recognition, quote-searching, crime informatics**

I. Introduction

Quotations by prominent personalities, institutions, office-holders, and organisations are often the source of great interest for researchers and investigators seeking to explore an issue in greater depths. Quotations can also help everyday internet users select the right platform for their daily entertainment and news consumption.

However, discovering who said what, when, and where is a very challenging task for both humans and machines, that can lead to misquotation [1], ambiguity [2], [3], and eventually lack of trust amongst the public [4]. Public statements given by elected officials, public figures and community leaders have a long-lasting socio-economic impact and are more likely to be widely reproduced in news and social media. Misquotations and spurious sayings are rampant in public and political discourse, especially in the online era where news and opinions are being generated, shared and reproduced at a staggering speed [1]. A testament to this is the rapid growth of disinformation and the urgent calls for regulatory action seeking to build more sustainable and trustworthy information sharing ecosystems [5].

Partly funded by European Commission grants CUTLER (770469) and PRINCE (815362), and by Academy of Finland 6Genesis Flagship (318927).

The article presents a prototype search engine[1] that allows users to query a database that contains quotations taken from news articles, using both strings and things. The implementation of the search engine has four core modules (Table 1). First, a module that develops and deploys a web crawler to ingest data from the GDELT Global Quotation Graph [6]. Second, a filter that augments the collected data by passing text fields through a default spaCY Entity Recognizer model [7]. Third, an Elasticsearch [8] cluster that indexes the annotated data and allows to run backend queries and aggregations. Finally, a module that setups the necessary web infrastructure, enabling requests from clients via a front-end using Flask [9] and NGINX [10]

TABLE I
TECHNOLOGY STACK USED TO BUILD THE SEARCH ENGINE

Tools	Description
Crawler	Custom crawler that pulls data from the GDELT API.
Analyser	Custom analyser/filter that applies a vanilla spaCY Entity Recognizer model on the GDELT data in near real-time.
Indexing	Custom Elasticsearch cluster that can handle high throughput indexing jobs.
Front-end	A custom application front-end using Flask and NGINX frameworks on Ubuntu to enable requests from clients.

II. Related concepts

The aim of this section is to provide a brief background about the key industry trends and developments in the field of quote-based searching. The section also outlines the core technologies and data sources used in developing our solution.

A. Quote-based Searching

In 2014 Yahoo! Inc. was granted a patent by the United States Patent and Trademark Office (USPTO) for a solution called "Quote-based search" [11] enabling users to use string queries for searching a *quote index*. The proposed system uses a database comprised of documents that include quotes. User-defined queries can return relevant quotes along with other attributes such as keywords, topics, and entities. Similarly,

[1] https://www.humcomp.ml

in 2017 the USPTO granted a patent to a team of Google engineers who developed a novel method for searching quotes that pays attention to the entities of a query using an entries knowledge graph [12]. The patent "Systems and methods for searching quotes of entities using a database" seeks to identify entities associated with a string query as well as develop a database to identify a set of quotes corresponding to a given search. It appears that this patent is currently used to retrieve quotes shown in the "knowledge panel" usually featured in the right hand side of the Google search results. In 2019 the solutions was updated with a continuation patent that focused on analysing audio content to identify quotations in audio and video files [13].

The GDELT Global Quotation Graph [6] is an open source project that has developed tools that scan news articles and then detect quoted statements from a vast number of sources with worldwide coverage. Overall, the Global Quotation Graph provides documents that include quotations in context taken from online news articles in 152 languages since January 1, 2020. The documents are freely available online and updated every minute, while the compressed files become available for crawling on a 15 minute heartbeat. The dataset contains JSONL documents that can include one or more quotations, and each row includes structured information extracted from a single article. Table 2 presents the key-value pairs found in the GDELT JSONL files. Next, we turn to review formal methods for indexing and searching text-based datasets.

TABLE II
GDELT GLOBAL QUOTATION GRAPH VARAIBLES

Key	***Description***
date	The date and time the article was last seen and indexed by GDELT.
url	The full URL of the article.
title	The title of the article.
lang	The human-readable name of the language the article is primarily written in. Articles with multiple languages will be listed under the primary language. Language detection is performed by CLD2 and will have a certain level of error.
quotes	An array containing one or more quotations identified in the article.
pre	A brief snippet of text preceding the quotation up to 100 characters to assist with speaker identification. This ranges in size based on the article language and linguistic queues and will typically be shorter than 100 characters.
quote	The actual quoted statement itself.
post	A brief snippet of text following the quotation up to 100 characters to assist with speaker identification. This ranges in size based on the article language and linguistic queues and will typically be shorter than 100 characters.

B. Quote indexing & Analytics with Elasticsearch

First released in 2010, Elasticsearch is a battle-tested distributed and open source search and analytics engine, which is built on Apache Lucene and can index both structured and unstructured data and texts [8]. At a high level, indexing documents in Elasticsearch requires the use of a schema (mappings) that essentially describes how fields within JSON documents should be handled, processed, stored and indexed. Thus, mappings are configured to correctly identify fields that should be indexed as text, numbers, dates, geo-locations, and date formats. While the default mapping of Elasticsearch is schema-less, users can install various plugins to customise this mapping with various analyzers and settings for indexing and searching more complex data [14]. With regards to text augmentation, the mapper-annotated-text plugin [15] developed by Mark Harwood, is a recent feature that can be used in conjunction with a named-entity recognition (NER) script, allowing for additional tokens to be injected into the token stream at the same position within a given sentence where the target text is annotated. For instance, using this markup feature, the sentence "Investors in APPLE rejoiced" will be enhanced and also include the following NER tags "Investors in [APPLE](APPLE&PERSON) rejoiced". Evidently, if integrated into a search engine service, this approach can provide added value for non-expert researchers. Take for example the problem of evaluating the cost of illicit drugs over time, a task that requires extensive searches, manual annotation, and technical skills. Using the proposed approach, this problem is easily broken down to the following string query *"cocaine + [MONEY]"*, returning results that mention *cocaine* in addition to other text that contain monetary or other financial information such ac the street value of drug seizures or cash and property seized.

While indexing large scale textual data using the mapper-annotated-text plugin [15] enables users to run concurrently queries that include both *strings* and *things*, this approach also enables novel and memory efficient ways to analyze large corpora. Specifically, Elasticsearch has built-in aggregations that can be used for both backend analytics and frontend services (e.g. search as you type features). With regards to text aggregation, one might be interested in identifying popular and/or significant terms within a given index/corpus. Aggregation functions for popular terms count how many times specific tokens appear in the index. Combined with the mapper-annotated-text plugin, a *simple terms aggregation* [16] can gauge the overall popularity of various named entities across documents. Additionally, the *significant text aggregation* [17] identifies unusual and interesting term occurrences in the data that can be used to suggest relevant terms/documents associated with a user-defined query (e.g. similar, but not the same). For instance, the significant text aggregation for the query "Bird flu" is able to return search terms such as "COVID-19", "Influenza", "H5N1" therefore promoting users to explore new search paths. The next section presents the core modules and cloud infrastructure of the ingest pipeline that was deployed to index data into Elasticsearch.

III. METHODS AND MODULES

Regarding cloud infrastructure, the pipeline was first tested and developed in a local installation of Elasticsearch 7.7, and was subsequently lunched as a cloud service on the CSC cPouta environment [18]. The latest version of the search engine was set up on top of 5 Ubuntu 20.04.1 LTS Virtual Machines running on an OpenStack implementation. All nodes

run Elasticsearch 7.7. and are identical in terms of configuration and settings (62GB RAM, 8 VCPU, 80GB Disk). Indexing was performed on a custom Elasticsearch cluster comprised of 2 master-eligible nodes, 3 data-eligible nodes, 1 dedicated ingestion node and 1 dedicated management node. Furthermore, Logstash [8], Prometheus [19] and Kibana [8] instances were used to monitor and visualise cluster metrics. The Kibana dashboard and the frontend are supported by an NGINX [10] reverse proxy exposed to a public IP , while REST requests are handled via a Flask application [9]. The ingest pipeline was implemented using a series of Python3 functions shown bellow:

- Index settings and Mapping: Using the Python Elasticsearch API [20] we first create an index and then configure the desired index settings, analyzers, and mapping. In the settings block, we import the analyzer_shingle with max/min size 2, along with the standard tokenizer in order to create word n-grams. In effect, using shingle we can convert a stream of tokens into new tokens by concatenating adjacent terms creating 2-grams. Subsequently, in the mapping block, we import the annotated_text [15] that enables us to inject the NER marked-up annotation tokens directly into our token stream. Eventually, the augmented data are added into the new fields of the type *keyword* and *annotated_text*. Fig 1. shows the annotated index of a document that matched a queried using a combination of strings and NER tags (bomb + FAC).
- Get entities: The detection of named entities is achieved through a vanilla implementation of the spaCy en_core_web_sm model [7]. Once the crawler starts ingesting data into the Elasticsearch cluster, a *get_entities* function is called that does all the NER heavy-lifting on indexing time. Specifically, the target text fields are passed through the spaCy model, which is an English multitask Convolutional Neural Network trained on the OntoNotes corpus [21]. Self-reported metrics suggest that the model is able to assign context-specific token vectors, POS tags, dependency parse and named entities with about 85% accuracy [21]. The complete list of the entity types included in the annotation task (and also available via the user interface) are shown in Table 3.
- Deduplication: Data pipelines that index data into Elasticsearch often take advantage of the auto generated id values that can be used either for Kibana visualisation and bucketing or for any other type data lineage and providence tasks. A key limitation, however, is that this default function will provide a unique document ID for two very similar of even identical records. News items scraped from the web are bound to have a large number of duplicates because of news sharing practices. Thus, this problem requires the integration of a deduplication mechanism into the pipeline, making sure that similar documents are not be stored multiple times in Elasticsearch with different _id values. While there are many available methods for this tasks, for simplicity we use a more native approach using hashes. In particular we deployed a custom python script that uses the MD5 algorithm to convert strings into hashes and calculate the sums between fields [22]. Then, the sums are added in a dictionary and searched through the hash of doc values to see if any duplicate hashes can been found.


```
{
  "_index" : "nerindex",
  "_type" : "_doc",
  "_id" : "oS9WwnQBIqEymKKPeA1L",
  "_score" : 9.197617,
  "_source" : {
    "data" : "2020-01-22T05:46:58Z",
    "url" : "https://www.sify.com/news/mangaluru-airport-bomb-suspect-arrested-news-national-ubwkMhhdejebf
      .html",
    "title" : "Mangaluru airport bomb suspect arrested",
    "lang" : "ENGLISH",
    "title_ner" : {
      "entities" : [ ],
      "annotated_text" : "Mangaluru airport bomb suspect arrested"
    },
    "pre" : "",
    "pre_ner" : {
      "entities" : [ ],
      "annotated_text" : ""
    },
    "quote" : "Aditya Rao, the prime suspect in placing an improvised explosive device (IED) or bomb at the
      Mangaluru international airport on Monday, was arrested after he surrendered at the DGP office,",
    "quote_ner" : {
      "entities" : [
        "Aditya Rao",
        "the Mangaluru international airport",
        "DGP"
      ],
      "annotated_text" : "[Aditya Rao](Aditya%20Rao&ORG), the prime suspect in placing an improvised explosive
        device (IED) or bomb at [the Mangaluru international airport]
        (the%20Mangaluru%20international%20airport&FAC) on Monday, was arrested after he surrendered at the
        [DGP](DGP&PERSON) office,"
    },
    "post" : " a police official told IANS here.",
    "post_ner" : {
      "entities" : [
        "IANS"
      ],
      "annotated_text" : " a police official told [IANS](IANS&ORG) here."
    }
  }
}
```


Fig. 1. Example results for search query “bomb + FAC”

- Front-end: Elasticsearch features can be shared with clients using a RESTful API over HTTP. To achieve this we used NGINX as a reverse proxy and created self-signed SSL certificates for HTTPS requests using OpenSSL [23]. The content of the quote index is exposed to the frontend [24] via a Flask application integrated with Elasticsearch API [20]. The frontend is a simple HTML page with a search bar where the user can create simple query strings [25] with special operators (e.g. bomb + car) that will match and return documents from the index. Alternatively, for semantic queries the user can formulate searches that include both strings and NER tags (e.g. bomb + FAC) and retrieve documents containing the keyword ’bomb’ together with contextual information linked to the FAC tag. A shown in Fig. 2 the semantic query “bomb + FAC” returns quotes from news articles that include both the term ’bomb’ as well as FAC entities such as Trump Tower, Times Square, Blue Route Mall and Riverside Mall. The results are ranked based on the default relevance *_score* value returned by Elasticsearch, which is based on the *DefaultSimilarity* algorithm in core Lucene [26]. Thus, the page ranking algorithm prioritises documents that include both *strings* and *things*, while penalises results that include only strings.

IV. Results

A. Data crawling and indexing

The web crawler collected a total of 30,000 unique JSONL files (total 86GB) between January-May 2020, containing

TABLE III
ENTITY TYPES INCLUDED IN THE QUOTE INDEX

Type	*Description*
PERSON	People, including fictional.
NORP	Nationalities or religious or political groups.
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states.
LOC	Non-GPE locations, mountain ranges, bodies of water.
PRODUCT	Objects, vehicles, foods, etc. (Not services.)
EVENT	Named hurricanes, battles, wars, sports events, etc.
LAW	Named documents made into laws.
LANGUAGE	Any named language.
PERCENT	Percentage, including ”%“.
MONEY	Monetary values, including unit.

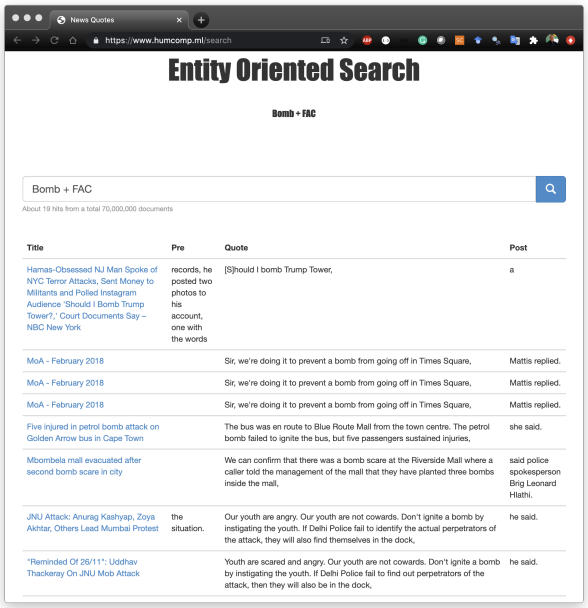

Fig. 2. Example results for the search query “bomb + FAC”. Source: https://www.humcomp.ml

structured information for approximately 70 million individual documents. In order to minimise the wastage of using unnecessary computing resources, the pipeline was configured to ingest a subset of the available data covering a 30 day period between approximately Jan-01 to February-01 2020. A total of 14,730,354 quotes/documents were retrieved and added into the quote index. About 1,200,000 documents were identified using the deduplication method described in the previous section. Given that the scope of the present work is to report the integration of various widely used and mature technologies for empowering non-technical users, the evaluation of the developed system was conducted using user-centered scenarios [27].

B. Aggregations

In order to empirically evaluate the ability of the search engine to retrieve novel content, we tested a set of queries and aggregations already discussed in Section II. We run a *simple term aggregation* [16] on the NER tags to identify the most popular terms in the “pre”, “quote”, and “post” fields (see the data model in Fig. 1). This helped identify some general characteristics of the index and better understand the news coverage captured in crawled documents. Subsequently, we applied a *significant text aggregation* [17] to assess any special characteristics in our documents.

Indicatively, we explored the distribution of the most popular words-tokens before and after the text field that contains the actual quote. As expected, the tokens that occur most frequently in the documents are function words (e.g. the, and, a, for) with low lexical meaning as well as noise crawled from raw HTML files. However, there are also numerous uses of gender pronouns (she/he, he said/she said) that might highlight gender bias. We investigated this further by looking at the text field before the quote, and found that the “he” utterances are just over 65,000 while the “she” utterances are about 40,000. Looking at the text field after the quote, we found that 194,000 instances of male pronouns and only 86,000 cases of female pronouns. These results indicate that female experts are grossly underrepresented in our index.

Similarly, because the quote index includes a large number of NER types, we can run aggregations to identify the most popular entities in the actual quote field. This can help us better understand what themes and topics are discussed in the quotes as well as the geographical coverage. The ten most popular entities are as follows: UN (105,594), Iran (74,603), American (44,392), Iraq (34,538), America (31,907), U.S. (31,184), Iranian (30,851), US (26,954), Soleimani (23,958), Trump (22,097), Americans (21,110), United States (20,613), Australia (17,588).

Next, we turned to evaluate the quality of the index (and the NER annotation) by applying significant text aggregations over the top 400 most relevant results against the global quote set. The significant text aggregation returns terms that can be used for suggesting similar content [17]. Given that our set includes NER tags, we queried the annotated_text field for the string “bomb” and requested to be evaluated against the background provided by the quote_ner.entities field containing the tagged entities. The results returned the following suggestions: bomb cyclone (1534.3), Bomb Bomb (644.4), BOMB-O-GRAM (276.1), bomb squad (276.1), Bomb Bay Gin (276.1), Iran (0.7), Islamic (0.2). The results indicate that the significant text aggregation returns entities across various NER tags such as location, products, GPE and LOC.

Finally, we run the same aggregation for the compound query “bomb + FEC”. This aggregation returns “more like this” results, and because we are including a NER tag in the search query, we expect to receive FEC locations where bomb-related events are previously discussed. The aggregation returns two suggestions: Delhi Police (11900.7), and Times Square (1270.5). We also run an aggregation using LOC instead of FEC and retrieved the following candidate terms involving locations: north coast, Camp Simba, Lamu County, Manda Bay, Al-Shabaab, Kenya, the Middle East, US, and

Iran.

C. Queries

We now turn to a more heuristic evaluation of the search engine. A set of compound plain text string queries were executed using the frontend interface of the application in an effort to simulate a potential use case scenario of a non-technical user interested in crime informatics.

More specifically, one plausible scenario involves an non-technical user (investigator) who explores the *modus operandi* of criminal groups using expert's opinions from open media sources. To retrieve relevant information, the user can search for "drugs + NORP", that will return results linking the string "drugs" with a series of ethnic groups, nationalities and religious or political groups. A second search might focus on identifying what places/infrastructure co-occur with keywords such as trafficking, cocaine or heroin. Desired results can be retrieved using the FAC tag along with any of the aforementioned search terms (e.g. heroin + FAC). In addition to key places and infrastructure, users can also explore the geographical location (LOC) or administrative jurisdictions associated with drug-related crime. Finally, using the NER tags PERCENT and MONEY, a user can find hits that discuss drug-related activities in conjunction with monetary values, quantitative measurements, and percentages. This search can provide longitudinal estimations of the street value of the recovered drugs.

V. Conclusions and Future work

The paper presents a high-level overview of a search engine that was build scratch, using a large corpus of news quotes provided by the GDELT project [6]. The search engine enables users to quickly drill through news articles and to identify quotations provided by experts and public figures. The added value of the proposed solution is based on the seamless integration of spaCY's NER model with Elasticsearch, allowing non-technical users to access novel information using a combination of strings and things. Furthermore, we have shown that aggregations and analytics can bring about interesting pathways for exploring fundamental sociological questions such as the reproduction of gender bias and stereotypes in the media. From the implementation side, there are numerous bottlenecks that need to be addresses. The search engine crawls raw data that have already been pre-processed and the quotations identified. While the work of the GDELT project has been extremely valuable, there is still room for improving the accuracy of the mechanism used in identifying quotes. In addition, the current version of the search engine implements the default NER model from the sapCy. The model has been reported to have an accuracy of about 80% when tested on a general web corpus [7]. Future iterations of the search engine will explore domain specific models using both static and contextualized word embeddings [28] in downstream tasks, that will increase the overall accuracy and quality of the user-defined queries.

References

[1] R. Keyes, *The quote verifier: Who said what, where, and when*. St. Martin's Griffin, 2007.

[2] P. Resnik, "Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language," *Journal of artificial intelligence research*, vol. 11, pp. 95–130, 1999.

[3] M. Mohamed and M. Oussalah, "Identifying and extracting named entities from wikipedia database using entity infoboxes," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 7, 2014.

[4] J. Turcotte, C. York, J. Irving, R. M. Scholl, and R. J. Pingree, "News recommendations from social media opinion leaders: Effects on media trust and information seeking," *Journal of Computer-Mediated Communication*, vol. 20, no. 5, pp. 520–535, 2015.

[5] K. Niklewicz, "Weeding out fake news: an approach to social media regulation," *European View*, vol. 16, no. 2, pp. 335–335, 2017.

[6] GDELT, "The global quotation graph." [Online]. Available: https://blog.gdeltproject.org/announcing-the-global-quotation-graph/

[7] SpaCy, "Named entity recognition." [Online]. Available: https://spacy.io/usage/linguistic-featuresnamed-entities

[8] Elasticsearch. [Online]. Available: https://www.elastic.co/elastic-stack

[9] Flask. [Online]. Available: https://flask.palletsprojects.com/en/1.1.x/

[10] NGINX. [Online]. Available: https://www.nginx.com/

[11] R. Blanco, M. P. Matthews, and P. Mika, "Quote-based search," Oct. 21 2014, uS Patent 8,868,558.

[12] E. Segalis, G. Chechik, Y. Matias, Y. Leviathan, and Y. Tzur , "Systems and methods for searching quotes of entities using a database," Aug. 8 2017, uS Patent 9,727,617.

[13] E. Segalis, G. Chechik, Y. Matias, Y. Leviathan, and Y. Tzur, "Systems and methods for searching quotes of entities using a database," Feb. 5 2019, uS Patent 10,198,508.

[14] Elasticsearch, "Elasticsearch mapping." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

[15] Elasticsearch, "Mapper annotated text plugin." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/plugins/current/mapper-annotated-text.html

[16] Elasticsearch, "Terms aggregation." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations-bucket-terms-aggregation.html

[17] Elasticsearch, "Significant text aggregation." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/master/search-aggregations-bucket-significanttext-aggregation.html

[18] CSC. It center for science, pouta cloud services. [Online]. Available: https://research.csc.fi/cloud-computing

[19] Prometheus. [Online]. Available: https://prometheus.io/

[20] Python elasticsearch client. [Online]. Available: https://elasticsearch-py.readthedocs.io/en/master/

[21] SpaCY, "Annotation specifications." [Online]. Available: https://spacy.io/api/annotation

[22] Elasticsearch, "How to find and remove duplicate documents in elasticsearch." [Online]. Available: https://www.elastic.co/blog/how-to-find-and-remove-duplicate-documents-in-elasticsearch

[23] OpenSSL. [Online]. Available: https://www.openssl.org/

[24] "Demo of the prototype entity oriented search engine." [Online]. Available: https://www.humcomp.ml

[25] Elasticsearch, "Simple query string query." [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-simple-query-string-query.html

[26] Lucene. Class tfidfsimilarity. [Online]. Available: https://lucene.apache.org/core/4_0_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

[27] D. Petrelli, M. Beaulieu, M. Sanderson, G. Demetriou, P. Herring, and P. Hansen, "Observing users, designing clarity: A case study on the user-centered design of a cross-language information retrieval system," *Journal of the American Society for Information Science and Technology*, vol. 55, no. 10, pp. 923–934, 2004.

[28] K. Ethayarajh, "How contextual are contextualized word representations? comparing the geometry of BERT, ELMo, and GPT-2 embeddings," in *Proceedings of the 2019 EMNLP and the 9th IJCNLP*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 55–65. [Online]. Available: https://www.aclweb.org/anthology/D19-1006