

Deduplication of Spanish Articles

Vyaas Shenoy

VNS170230

Ishan Sharma

IXS171130

Mavis Francia

MCF140030

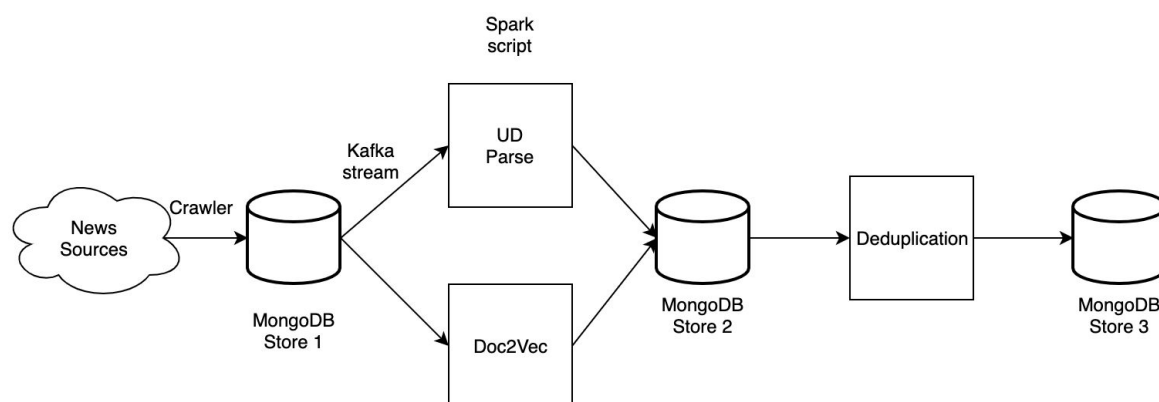
Tanushri Singh

TTS150030

Introduction

As part of a political event coding pipeline, we designed a Spark-based preprocessing tool that collects data by crawling a set of Spanish news websites on a daily basis, extracts the main content of the article and related metadata, processes the extracted content with udpipe, generates universal dependency parse for each sentences within the content, and stores the collected data and processed data in MongoDB. Then, we ran a deduplication algorithm we created using Doc2Vec on the entire document and computed the Jaccard similarity on the proper nouns, nouns, numbers, and verbs generated from the universal dependency parse tree.

Project Pipeline



News Sources

Initially, we used the list of news websites provided as part of the resources. However, these were not enough and resulted in 200-300 articles per day. To get more content, we used the following resources:

1. Google News: Manually browsing Google News allowed us to gather a list of Spanish news websites that were popular.
2. Similarsites.com: Similarsites.com is a free tool that shows up to 40 similar websites for a given website.
3. Bing News Search API: The biggest drawback with previous methods is that they require manual human work. Microsoft provides a News Search API that lets you get top news filtered by article and region. By iterating through several news items, we could discover unique websites publishing news in Spanish.

Using these 3 methods, we ended up with 120 Spanish news websites.

Data Collection

We started with News Please for crawling the websites. While News Please was able to gather old articles, it was not efficient at finding recent articles.

Our next consideration was scrapy but that requires manual work for each website. This was not a scalable solution when we consider more than a few websites and we decided not to use it.

We settled on using two crawlers that write to a common datastore. We use News Please to discover the articles and save them to disk. Then, a Python script puts them into Mongo DB. A second crawler uses RSS, an open web standard and writes directly to Mongo DB.

RSS stands for Really Simple Syndication and was created with the goal of allowing users and applications to access updates to online content in a standard way. A RSS feed is a simple XML document containing information about a few latest articles on the website. The standard does not decide the number of articles, and publishers are free to choose this number. With our crawler, we saw feeds with 10 to 100 articles.

These two crawlers complement each other quite well and have the following advantages and disadvantages:

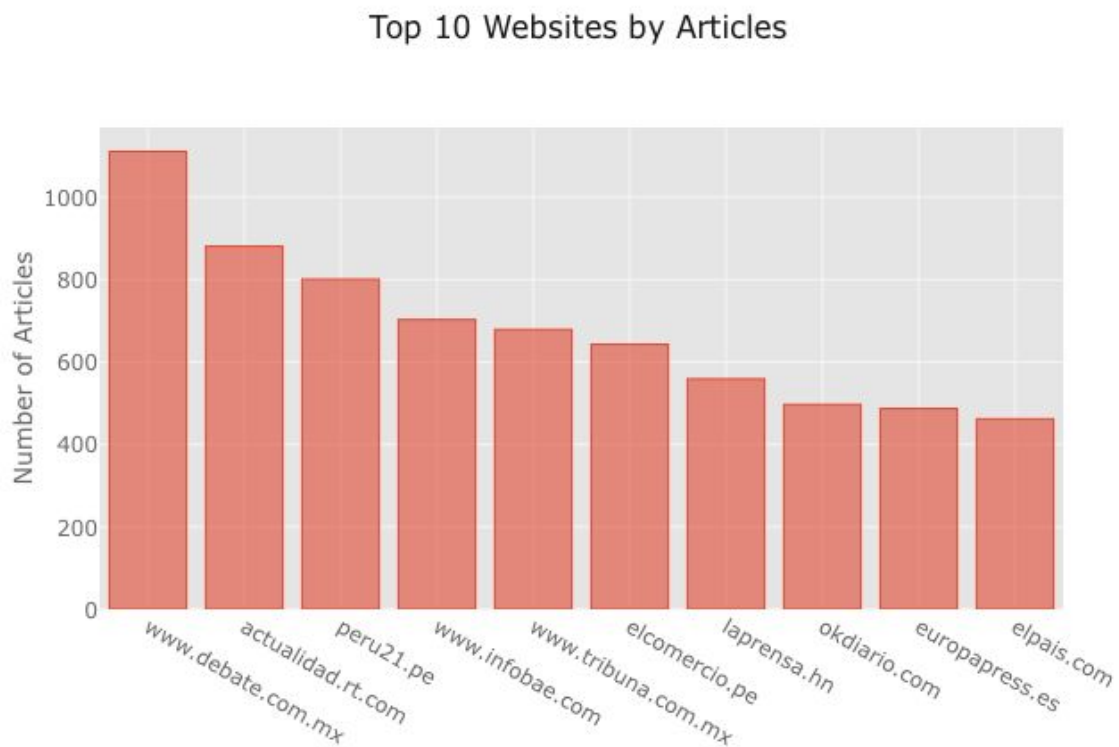
1. RSS Crawler can gather most recent articles easily. However, it needs to be run frequently (15-60 minute interval so that it does not miss any new articles).

2. News Please can gather recent as well as old articles but is more resource intensive. It tries to use the website's sitemap to speed up crawling but falls back to crawling recursively, which can take a lot of time. It can gather almost every article from a website but requires a huge amount of resources as well as time.

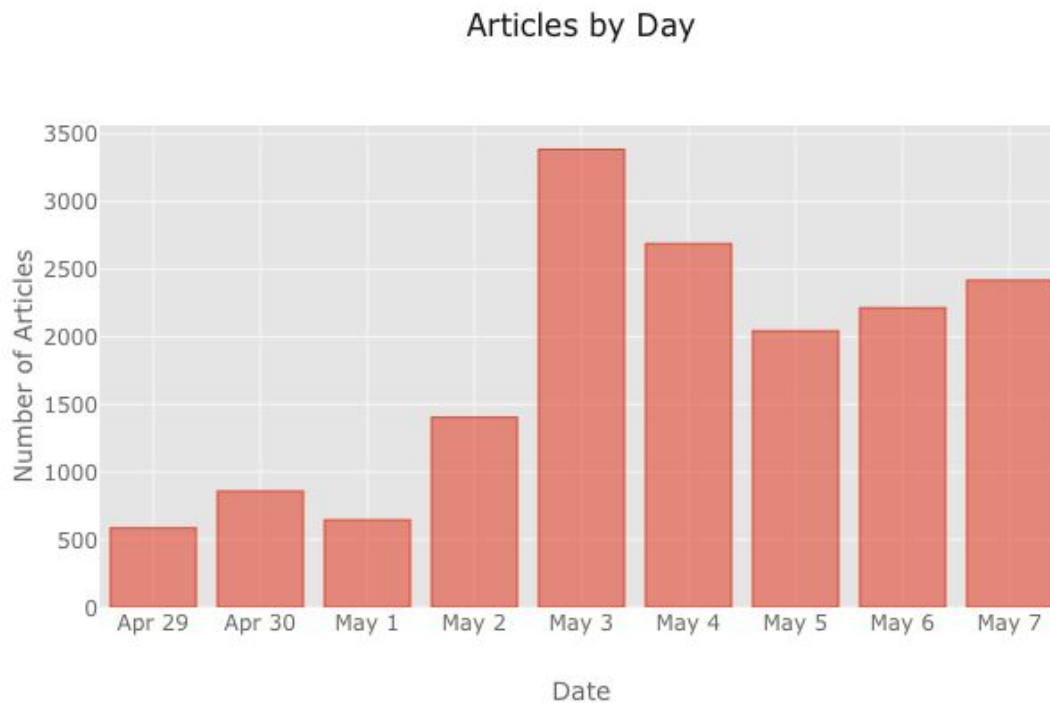
Due to News Please crawling a lot of older articles, we ended up with around 0.5 Million articles dating from 2001 to 2019. For April 29-May 7 interval, we had 300 articles per day.

After adding RSS, we were able to collect a total of 16701 articles for above interval, which is 1855 articles per day.

Top 10 websites by number of articles were:



Article distribution by day:



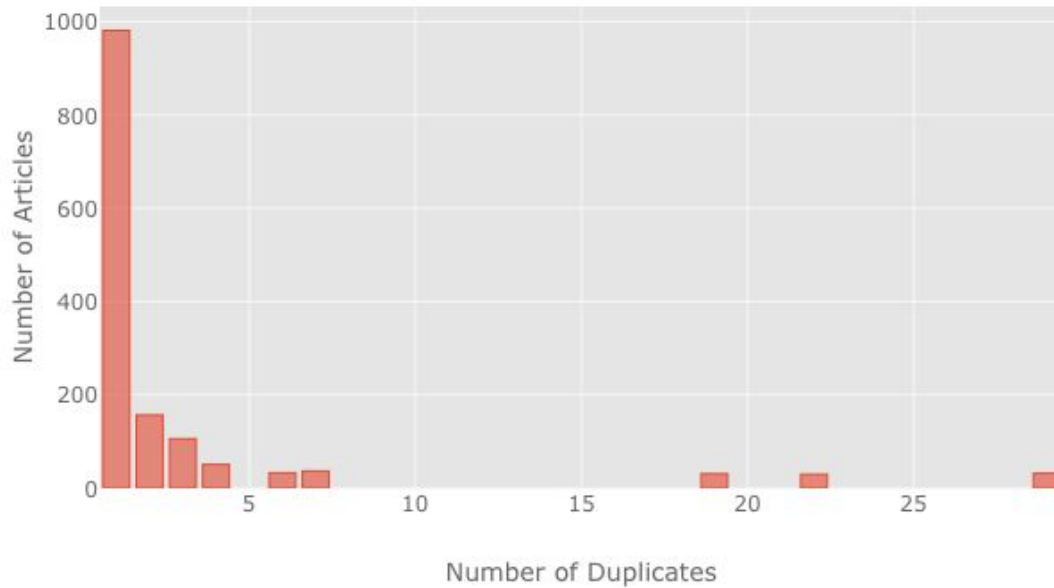
Doc2Vec

Doc2Vec is an algorithm used to produce word embeddings. It is similar to Word2Vec with the key difference being Doc2Vec encodes whole documents instead of individual words.

We trained the model on all of the articles from 2019 (78,154) to make sure that it had an extensive vocabulary. For each document, we put it through the same cleaning pipeline. Then we generate a vector for each document. After inferring vectors for each document, cosine similarity of current document was computed against all the documents for current day.

Upon experimenting, we found that a 60% was a good threshold for documents that were potential duplicates. If we set the threshold to 80%, lot of articles talking about similar topics but not having enough similar words were left out. For lower thresholds, articles with broad categories (e.g. bowling and skating) were marked as potential duplicates.

Distribution of Duplicate Articles (Doc2Vec)



Challenges

Crawler Design

News Please had date filters but they did not always work with the Recursive Crawler, which had to be used for around 25 websites. As a result, we had to filter a lot of news articles when writing to the database. We ended up collecting more than half a million articles, out of which only ~11000 were useful. News Please also failed for around ~45 sites that used infinite scrolling. Due to presence of infinite scrolling, it could not find next/previous links and terminated early.

We decided to use Scrapy but each scrapy crawler would need to be customized for each website and that would have taken a lot of manual work. Instead of using Scrapy, we created our own RSS crawler, which discovered whether the websites had RSS or not and then used news please in script mode (calling from Python script) to get individual articles.

RSS has the limitation of having only a few recent articles and we cannot get historical data. However, it lets us get the latest articles frequently.

Interconnecting MongoDB with Kafka and Spark

Since the web crawler wrote the news article data to MongoDB, we wrote a script that reads from MongoDB and publishes articles that span a certain date to a Kafka topic. (This script can be run daily to begin processing the articles for that particular day.) We also wrote a Spark streaming script that consumes the news articles in the Kafka topic and processes them (as described in the next section). This script also writes the processed data back to MongoDB.

Implementation of UD Parse

Implementing UD parse involved installing the `ufal.udpipe` library for Python and loading a pretrained model for the Spanish language (AnCora). We used Spark streaming to read the news article data from MongoDB (which was streamed through Kafka) and converted each stream batch to a Spark dataframe. Then using a user-defined function, we passed the text from each article to be parsed with UDPipe and wrote the parse results to its own column in the dataframe. Additionally, we scanned through the parsed results to collect parts-of-speech we considered important for content deduplication (proper nouns, nouns, numbers, and verbs) and stored those tokens as a dictionary in a separate column in the dataframe. Finally, we wrote the dataframe back out to MongoDB in a new collection. Performing the parse for our news articles published from April 29, 2019 onwards (around 15,800 articles) took approximately 4 hours.

Similarity measures for deduplication

The Doc2Vec model took 20 minutes for training a model on approximately 7500 articles. Training a Doc2Vec model on a large dataset is time consuming. After training the model, comparing two articles which are used for training the model does not require too much time but comparing two articles not in the training data is time consuming. The similarity computation using the Doc2Vec library is slow. Duplicate articles can have a similarity rating varying from 60% to 100%. We had to fix a lower bound on the similarity rating approximation to 60%. When implementing the final DeDuplication Algorithm using output from Doc2Vec as well as output from UDParse, a 10% cut off allowed the group to accumulate articles that are truly relevant.

Measuring Similarity using UD Parse output

We measured similarity between two articles' UD parse output by computing the Jaccard similarity between the sets of proper nouns, nouns, numbers, and verbs from each article. We only perform this comparison if two articles have a similarity rating above 60% for doc2vec

cosine similarity since this second method is mainly to refine the results from Doc2Vec. If two documents have a similarity above 60% for Doc2Vec and are at least 10% similar in Jaccard similarity, we consider them as duplicates.

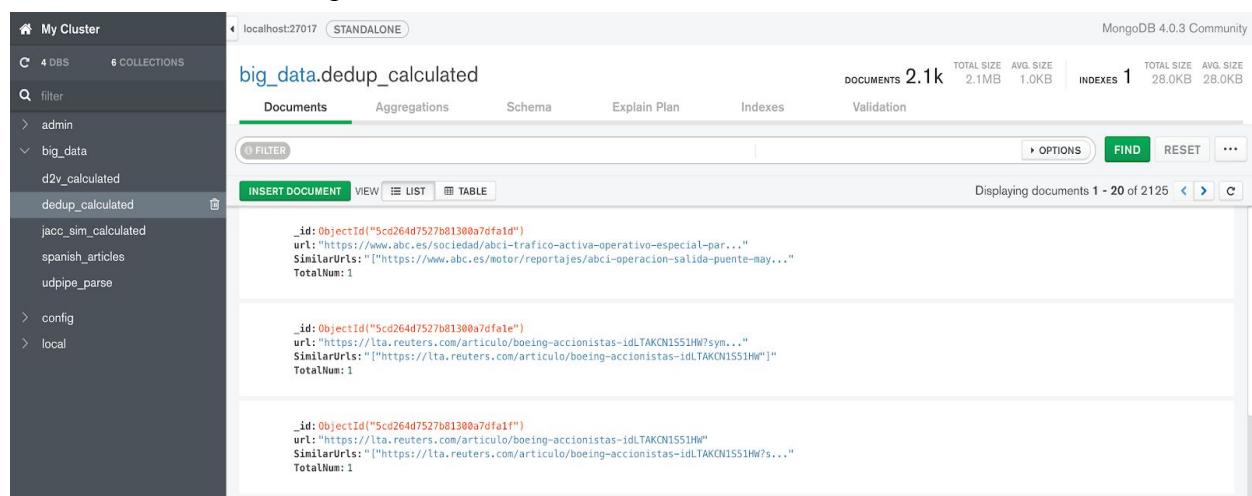
Results

Out of about 15,800 articles, we have identified 2125 articles that have one or more duplicates according to the similarity thresholds we have set. The total number of duplicates was 2683, which is approximately 17% of the dataset.

More details:

	Number of duplicates
Mean	56.65
50th percentile	7.00
75th percentile	77.50
Max	407

The 50th percentile count was 7, which means that we detected 7 or more duplicates for each article when we found duplicates.



The screenshot shows the MongoDB Compass interface for a cluster named 'My Cluster' at 'localhost:27017'. The selected database is 'big_data' and the collection is 'dedup_calculated'. The document statistics show 2.1k documents, 2.1MB total size, and 1 index. The document list shows three entries, each with a unique ID, a URL, similar URLs, and a 'TotalNum' of 1.

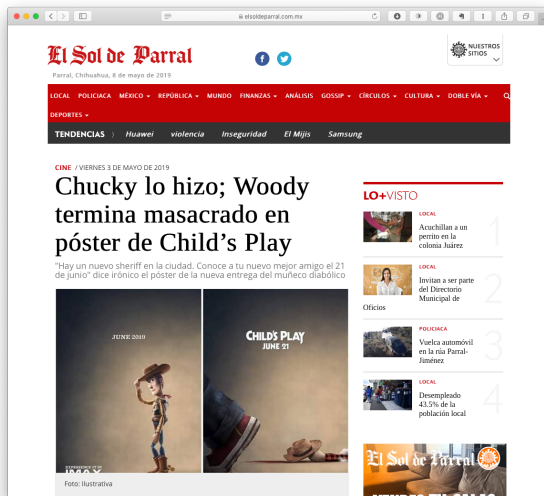
```
{ "_id": "ObjectId('5cd264d7527b81300a7dfa1d')",  
  "url": "https://www.abc.es/sociedad/abci-trafico-activa-operativo-especial-par...",  
  "SimilarURLs": ["https://www.abc.es/motor/reportajes/abci-operacion-salida-puente-may...",  
  "TotalNum": 1  
}
```

```
{ "_id": "ObjectId('5cd264d7527b81300a7dfa1e')",  
  "url": "https://lta.reuters.com/articulo/boeing-accionistas-idLTAKN1551Hw?sym...",  
  "SimilarURLs": ["https://lta.reuters.com/articulo/boeing-accionistas-idLTAKN1551Hw?...",  
  "TotalNum": 1  
}
```

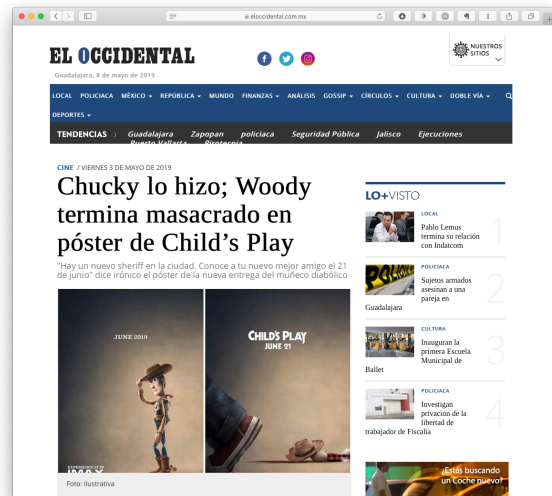
```
{ "_id": "ObjectId('5cd264d7527b81300a7dfa1f')",  
  "url": "https://lta.reuters.com/articulo/boeing-accionistas-idLTAKN1551Hw",  
  "SimilarURLs": ["https://lta.reuters.com/articulo/boeing-accionistas-idLTAKN1551Hw?...",  
  "TotalNum": 1  
}
```

Upon manual inspection of a small subset of data, we found that articles were relevant in most of the cases.

Some examples of matching articles from different websites:



<https://www.elsoldeparral.com.mx/cultura/cine/chucky-lo-hizo-woody-termina-masacrado-en-poster-de-childs-play-3546022.html>



<https://www.eloccidental.com.mx/cultura/cine/chucky-lo-hizo-woody-termina-masacrado-en-poster-de-childs-play-3546022.html>



https://www.abc.es/economia/abci-trump-anuncia-aumento-aranceles-sobre-productos-china-201905052024_noticia.html



<https://www.debate.com.mx/mundo/Trump-anuncia-aumento-de-aranceles-en-productos-importados-de-China-20190505-0113.html>

Future Scope

There are a few changes that can be made to the project to further improve the data collection and deduplication.

- **Data Collection:** Currently, News Please runs from command line which requires manual processing and it can't be run automatically from a Python script, which makes data collection far more tedious. Additionally, it doesn't support any heuristics for article length out of the box. This causes News Please to index content that is just video and audio with very less text. This content can be marked as duplicate by deduplication algorithm because of very few words available.
- **Considering Article Length:** Right now, one limitation of the designed algorithm is that it can match article with varying lengths. In the future, matching articles that are of similar length should be helpful.
- **Model Updates:** Doc2Vec models are limited by the fact that they cannot be updated with new documents easily. Selecting a different model that supports updating or using a batch process to continuously update data would increase the accuracy of deduplication.
- **Integrating Doc2Vec with Spark:** Currently, Doc2Vec is done outside of Spark. Integrating with Spark should allow calculation for Doc2Vec scores in parallel.
- **Use another similarity measure for UDPipe Output:** Currently, we are using Jaccard similarity for the output generated by UDPipe. It is not very accurate and can result in varying measures for similar articles. In the future, a measure like cosine similarity should help further improve the scope of the project.