A cartological approach to visualising term popularity in NLP

Figure 1: CartoGram takes away the heavy load of mapping key terminology.

Projects such as Optimus, PyGrams and optimising the ONS website search show that here, at the Data Science Campus, we are developing an expertise in the field of Natural Language Processing (NLP). However, after investing so much time developing complex and successful algorithms, we, like many others in the Data Science community often struggle to dissmeninate and share our findings in a user-friendly manner.

Current ways in the NLP community to visualise results include word clouds and force directive graphs, and whilst there have been efforts to improve these, we can perhaps utilise much more of the underlying data than the text alone. For PyGrams, we did just that by using the city and country information, and company details, held within patent databases. Our map-based visualisation tool is called CartoGram.

As you would expect, analysing when and where patents have been filled and granted is itself not novel, see Lens.org and Global Patent Explorer for example. However, what we have achieved here with CartoGram is the integration of NLP and cartography by analysing where and when *key terminology* has occured using patent applications. This allows us to analyse changes in key terminology over time, and also by taking the company information from the patent data, we can also view what terminology is related to which company.

To create a map using CartoGram, the original dataset must contain a free text field, a date field, city and

country information, and a catergorical field (such as company information for patents). Then, using PyGrams we generate a CSV report of the top *n* terms from the dataset. The free text field of the original dataset is then scoured for the top terms, and catergorical information and date, as well as its location, are saved for those that contain a top term.

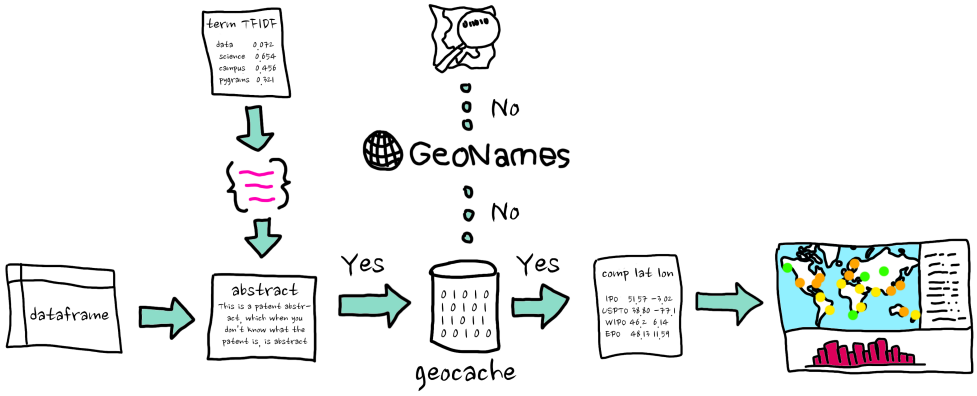

Figure 2: The process of CartoGram. A dataframe is fed in, comprising a free text field, a date field, city and country information, and a catergorical field (here company information). The top *n* terms report from PyGrams is then turned into a JSON file and data where key terms are found in abstracts is saved. City and country information is turned into decimeal degree coordinates using a geocache file (built using GeoNames and nominatim). A CSV file of the date, the latitude and longitude, and the key terms and the number of times they occured within the free text field is then visualised on a map using various JavaScript D3 libraries.

To convert the city and country information to latitude and longitude decimal coordinates, in order to plot in CartoGram, we use a a geocached file comprising a list of latitudes and longitudes for almost 50,000 cities and countries around the world. This geocached file was built using two open-source services, GeoNames and nominatim API. The GeoNames database can be saved locally, and therefore is the primary source of coordinate information. But where the location information is not held in the GeoNames database the nominatim API is called, which allows us to search OpenStreetMap's extensive places of interest dataset. This approach gave us a 99.5% success rate of converting city and country to latitude and longitude for 3.2 million patent applications. Furthermore, storing the information in a geocached file means that in the future, only patents whose cities are not currently held within the file will be searched for using GeoNames or nominatim, making the conversion more efficient.

When the entire dataset has been searched through for the top terms, a CSV file is saved where each row is a unique entry comprising the date, the latitude and longitude, and the key terms and the number of times they occured within the free text field. For the 3.2 million patents in the entire United States Patent and Trademark Office (USPTO) dataset this amounts to a 0.5 GB CSV file. Therefore, we need an efficient way of displaying this amount of information.

To do this, we use the D3 JavaScript library, which uses HTML, SVG, and CSS. First, we implement Crossfilter, which supports extremely fast interaction between coordinated views, even with datasets containing a million or more records. We took inspiration from Curran Kelleher's block so that we can combine the map with a bar chart. However, to reduce the number of markers displayed on the map and

improve loading time, we used Leaflet.markercluster. This plugin aggregates markers at a global scale, but then disagregates individual locations at a finer scale. Finally, we created a table showing the key terms, the term count and the company that term appears in the most. When clicking a term a full list of companies and term counts for each is displayed in a pop-up.

CartoGram is fully dynamic, whereby zooming in on the map will cause the map, bar chart and table to repopulate. Furthermore, the bar chart can be filtered by dragging a box over a particular time period. There is also a search box above the table to search by term, company and country code. CartoGram is efficient as data is only appended or removed when a change is made, instead of reloading the entire dataset for each dynamic change to the map, bar chart or table.

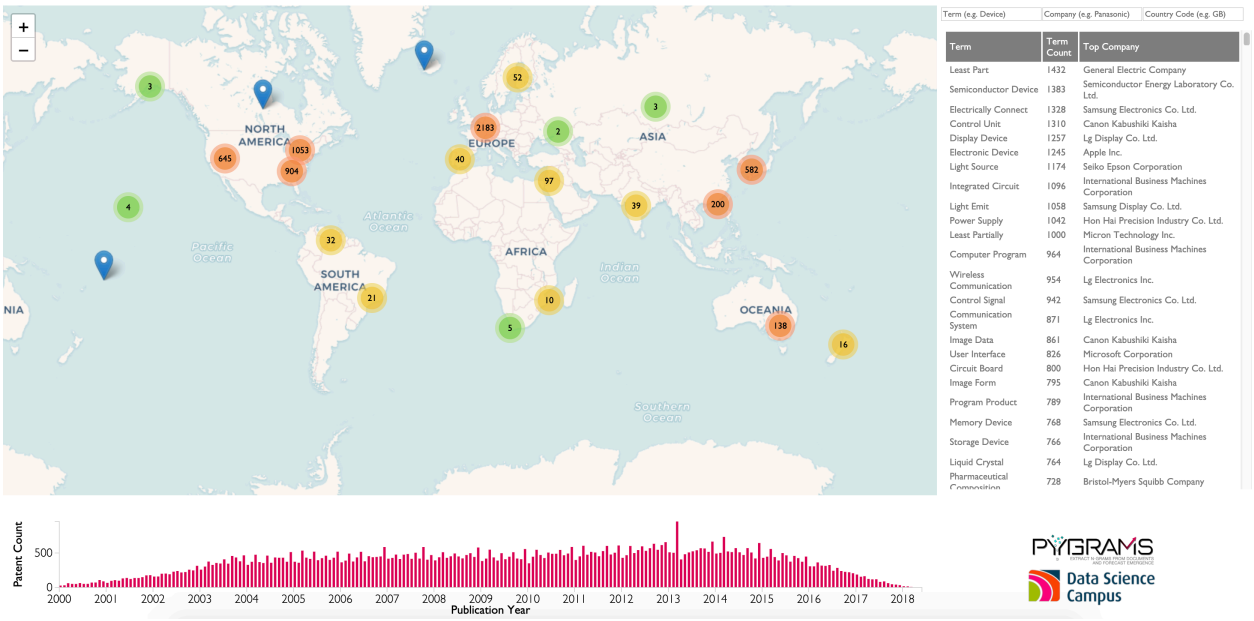

Figure 3: A screenshot of the 100,000 patents loaded into CartoGram.

Figure 3 shows an screenshot of CartoGram using a 100,000 subsample of the 3.2 million patent dataset. The map is available here. As shown, the most popular key term is 'least part', used 1,432 times and the most by 'General Electric Company'. By clicking the 2,183 circle above Europe the map zooms automatically to this region, and the data is disagregated to the next cluster level. 'Least part' is still the most popular key term, being used 434 times, but the top company is now 'Infineon Technologies Ag'.

Whilst CartoGram here is has been used for patents, it can be used for any dataset compatible with PyGrams, containing a free text field, date field, country and city field, and catergorical field. To find out more, visit our GitHub repository here.