

# Geo temporal search

Project team: Yusuf Bhabhrawala (ynb2)

<b>Project Overview</b>	<b>2</b>
<b>Problem definition</b>	<b>4</b>
<b>Classifying geo-temporal context</b>	<b>5</b>
Defining time dimension	5
Defining location dimension	5
Extracting time and location dimension from documents	6
Validating Geo-Temporal Vector	6
<b>Implementation Approach</b>	<b>7</b>
Scope	7
Data Corpus	7
Geoparsing	7
Edinburg Geoparser	7
CLIFF-CLAVIN and CRF Classifier	8
Project Setup	9
<b>References</b>	<b>11</b>

# Project Overview

Time and location provide lot of context in search. If I am searching for a restaurant say in Google maps, it provides me the the list based on my location. There is another dimension which is time which is generally ignored. For example if I am searching for say some event in history, then not only are documents related to events important but other events during same time and location are also important in understanding that event.

Another example is if I am researching a person say 'Elon Musk'. When doing this search on Google, it only highlights the news worthy and current articles on Elon Musk. But I might be interested in reading about Elon Musk in the context of say Bay Area or Los Angeles and say 10 years back when he started SpaceX or 15 years back when he started X. Given that context, the relevant information seemed completely not accessible. There is no good interface to do the same as well.

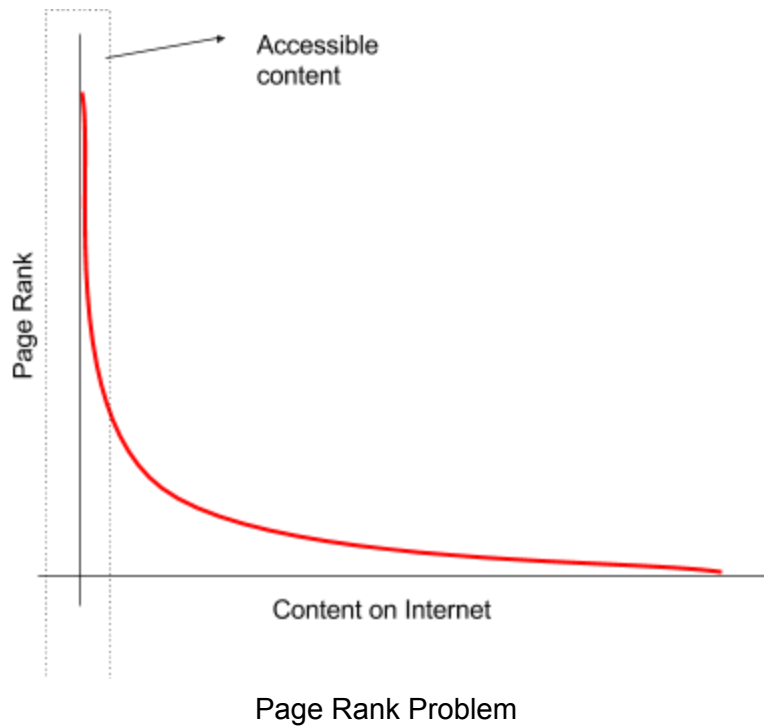
Yet another use could be in tourism where people can go back in time and see the chatter happening around some interesting events at locations of historical significance. Twitter feeds during the time of certain event from its location would be on interest many years later.

The project is to focus on categorizing text on location and time and feeding that into search.

## **Google page rank problem**

Google organizes content using its page-rank algorithm. The main problem with this approach is that content that is SEO optimized filters out at the top and pretty much hides all other content.

Graph:



What we end up consuming because of this approach is very specific content built for sharing and generally not very knowledgeable.

In general any algorithm which tries to categorize and optimize ranking will lead to such results eventually.

The internet as a result has become non discoverable, non explorable.

One good way to organize information is by adding physical constraints to its spread. This can typically be done by adding physical bounds to where it can be accessed. For this proof-of-concept, those physical bounds are time and space. All content is created somewhere. It might be a document on mathematics, but the author still has to write it at some physical location. Also, all content is written at some time. Thus a document on the same topic can be quite different depending if it was written today vs say 100 years back.

Exploring content within these two dimensions create really interesting results as seen in the proof-of-concept of this project. It allows exploring and analyzing content in a way which matters in my view. It highly discourages any possibility of SEO and pushing content to the users and encourages sharing of context along with content.

## Problem definition

Geo temporal search involves tagging the documents with their appropriate time and location and providing this information in the context of a geo temporal browser.

The traditional search approach uses the construct of a query to retrieve relevant documents. Here, the query term can be totally optional and user might just want to view the relevant documents in the context of time and location.

Thus the problem involves placing each document in time-space dimension and determining relevant documents based on the appropriate use context.

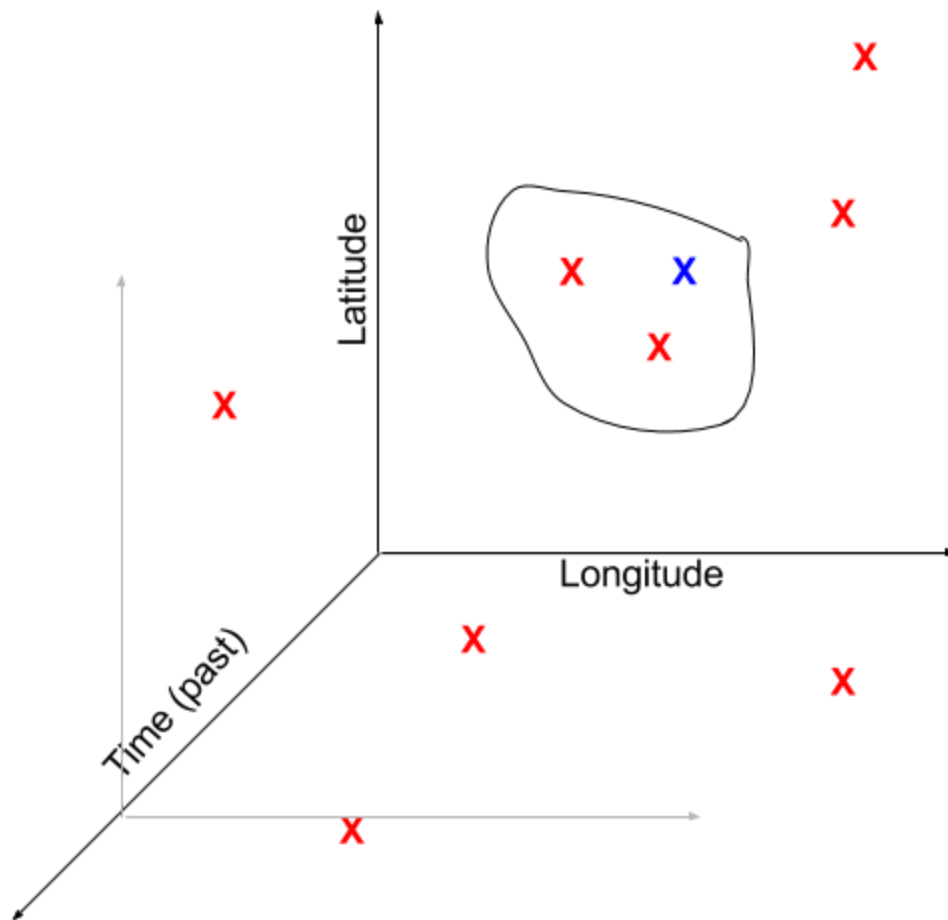


Figure 1: Geo temporal vector space of all documents

## Classifying geo-temporal context

A document can have a geo-temporal context in one of the following two ways -

1. The document was created in a certain geo-temporal vector space
2. The document refers to a certain geo-temporal vector space

All documents would have the #1 context above but not all would have #2. For example, though this document refers to Abraham Lincoln and 1920 somewhere, it does not refer to any geo-temporal context.

## Defining time dimension

Time dimension for documents is generally not absolute. A document could be related to an era, century, decade, year, month, day or absolute time. Thus the time dimension is entirely contextual. The same would apply to the query term as well. The query needs to provide the time in the context of its scale.

Thus I might be interested in one of few things:

1. What happened on 20th January 1920, say
2. What happened in 1920 AD
3. What happened in 19th century

The scale of time thus has to be capture and the context determined based on the content of the document.

## Defining location dimension

Location dimension in a similar manner has a large scale, though much better defined then time. For example, I as a user might be interested in one of following:

1. What happened right here, at the intersection of this cross street
2. What happened nearby, say in 5 mile radius
3. What happenend in this city
4. What happened in on this side of the world
5. What happened in this country, or whatever entity existed at the corresponding time

## Extracting time and location dimension from documents

The hard part is extracting time and location dimension with their relevant scale from the documents in a reliable manner. Some background knowledge of history would be required to make sense of the context. For example, if a document is talking about say, "Abraham Lincoln gave the following speech today as part of his Swear-in ceremony", some knowledge of "Abraham Lincoln" would be required to determine the dimension data.

## Validating Geo-Temporal Vector

Also, it would be of much interest to determine the dimensions not solely based on the content of the document. This would help in preventing historical spamming, where someone would write articles with past create dates and keywords to engineer their location on the geo-temporal vector space.

Few ideas of implementing such a validation:

- If a document in the past refer to a document in the future, it is clearly not consistent
- If crawler history can determine based on crawl history if a document was updated for geo-temporal shifting
- Use available knowledge base to validate the vector space
- Use feedback mechanism from users to validate or invalidate the meta data
- Use source reliability score based on above validation, to establish appropriate trust

# Implementation Approach

## Scope

This project is limited to a proof of concept of organizing information in a geo-temporal space and navigating that content via a simple interface. Since both these dimension can get really complex, we limit the time dimension to date and space dimension to city. This simplifies geo tagging content and allows easy visualization using Google Maps tools.

## Data Corpus

For the content parsing I picked data where date information was easy available. I was able to find large collection of archived blogs (about 700,000 blog articles) (3). This data was available as badly formed XML documents, with one document per blog site. Thus it had to be parsed and cleaned before it could be used. The data provided date information which forms the first dimension of this project.

## Geoparsing

The second part was to determine the space dimension. I researched a bunch of tools online for geoparsing. The initial plan was to go ahead with Edinburg Geoparser. Later I ended up using the CLIFF-CLAVIN parser (4).

## Edinburg Geoparser

This Geoparser was simple to use. The installation steps include the following -

1. Download the Geoparser from -  
<https://www.inf.ed.ac.uk/research/isdd/admin/package?view=1&id=187>
2. Run the pipeline using the provided scripts. Example:  

```
cat content.txt | ./run -t plain -g geonames-local -o ../out 2999
```
3. Change the gazetteer as required

The drawback of using this approach for geoparsing my blog content was that all the API calls were made to an externally hosted database and server of the Edinburg geoparser. This was not very scalable for the large dataset that I needed to process.

So the next step here was to setup the Geoparser locally. The documentation provides all the instructions to do that. I was able to setup the database but not make it fully working.

This document here describes the internal flow of the Geoparser -

<http://groups.inf.ed.ac.uk/geoparser/documentation/v1.1/html/overview.html> . I will not cover the details here, but one of the problem with this approach is it will was a simple text search and matching against the Gazetteer database that was selected.

This did not provide very good results in the initial test. For eg. if a blog was talking about some person *Tracy*, since *Tracy* is also the name of a place, it would match it against that place and all the place in the world with that name and provide those results.

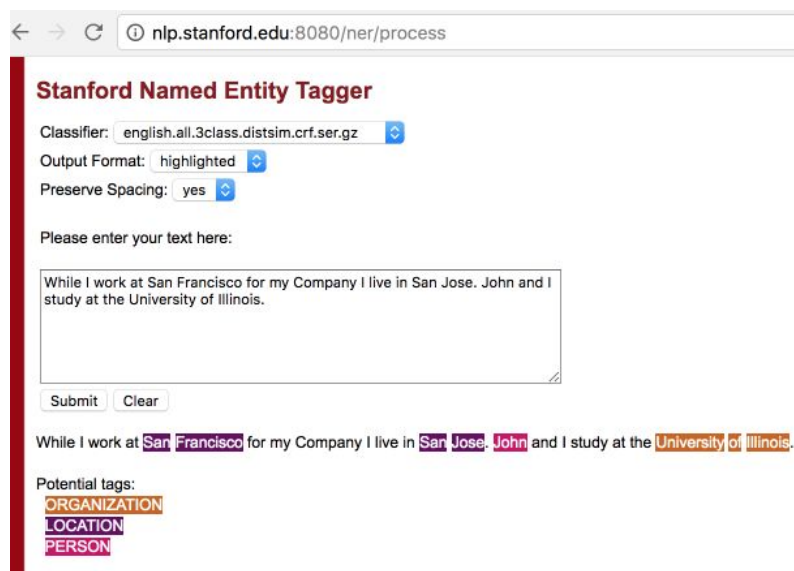
Thus the resulting geo-data had too much noise and not very useful. After much work and experimentation I looked for an alternate way of geo-parsing my data.

## CLIFF-CLAVIN and CRF Classifier

CLIFF-CLAVIN was built to parse out new articles and differentiates between people, organization and places. One neat feature about the places was that it would prioritize the places according to the document and provide a focused list of place at city level, which perfectly met my needs of this proof of concept.

CLIFF is a Java based web service. It rely's on Stanford's Named Entity Recognizer (5) to extract strings that might be people or places from the articles. Standord NER is also known as CRFClassifier. It provides a general implemenation of linear chain Conditional Random Field(CRF) sequence models.

The classifier identifies organization, person and location from a given text. The following example depicts that -



The screenshot shows the Stanford Named Entity Tagger web interface. The browser address bar displays 'nlp.stanford.edu:8080/ner/process'. The page title is 'Stanford Named Entity Tagger'. Below the title, there are three dropdown menus: 'Classifier' set to 'english.all.3class.distsim.crf.ser.gz', 'Output Format' set to 'highlighted', and 'Preserve Spacing' set to 'yes'. A text input area contains the sentence: 'While I work at San Francisco for my Company I live in San Jose. John and I study at the University of Illinois.' Below the input area are 'Submit' and 'Clear' buttons. The output shows the same sentence with entities highlighted: 'While I work at San Francisco for my Company I live in San Jose. John and I study at the University of Illinois.' Below the output, a section titled 'Potential tags:' lists three categories: 'ORGANIZATION' (highlighted in orange), 'LOCATION' (highlighted in purple), and 'PERSON' (highlighted in pink).



CLIFF-CLAVIN can be set up locally using various options described here - <https://cliff.mediacloud.org/> . I installed it using Vagrant and Virtualbox.

To install using Vagrant:

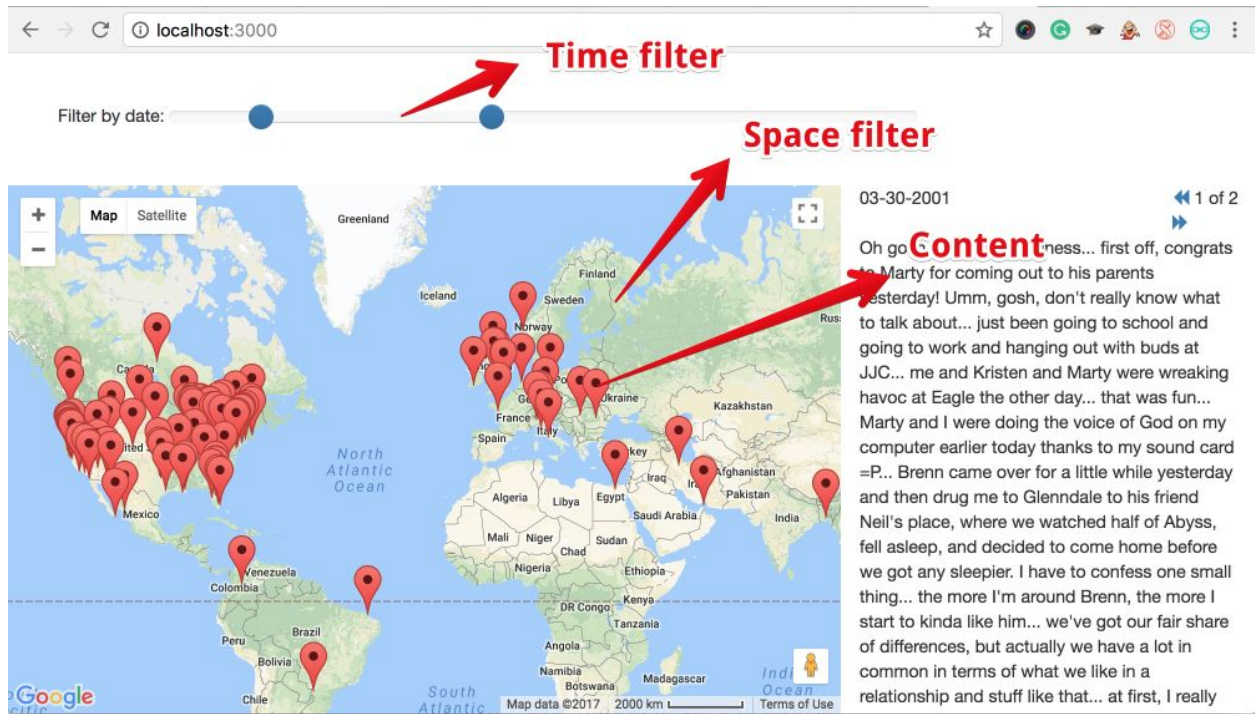
1. Clone this repository here at - <https://github.com/mitmedialab/CLIFF-up>
2. Install Virtualbox and Vagrant on your local machine
3. From the repo home, run the command *vagrant up*
  - a. This downloads the default VM image and begins installing all the dependencies required to run CC.
4. As part of the setup it primarily does 2 things
  - a. It install Java and set's up Tomcat server
  - b. It downloads the Geonames gazetteer from geonames.org
  - c. It builds a Lucene index of place names
  - d. It clones CLAVIN.git which is the CRF Classifier
  - e. It provides the startup scripts to start the web API endpoint
5. Once vagrant is set up, we would need to *vagrant ssh* and start the server manually using `sudo /home/vagrant/apache-tomcat-7.0.59/bin/startup.sh`
6. The API can then be access from the host computer via a GET request.
  - a. Example: `http://localhost:8999/cliff-2.3.0/parse/text?q=It's weird. I hate eating in front of other people`

## Project Setup

The project architecture involves 3 main parts:

1. Parse the blogs dataset into separate blogs items with their metadata
2. Setup the CLIFF-CLAVIN Geoparser to Geoparse each blog item
3. Process every blog item against CLIFF-CLAVIN Geoparser and tag the location information against the dataset
4. Upload the dataset to a local MySQL database for access by the visualization tool
5. Setup the visualization application using Node-express. Clone the project repository (6) and *npm install* followed by *node app.js* . This sets up the localhost visualization tool to browser the blogs data on the geo-temporal interface.

The resulting project visualization looks something as below:



## References

1. Hypercities - <http://www.hypercities.com/>
2. Context aware text retrieval - <https://ris.utwente.nl/ws/portalfiles/portal/5387712>
3. Blogger Dataset: J. Schler, M. Koppel, S. Argamon and J. Pennebaker (2006). Effects of Age and Gender on Blogging in *Proceedings of 2006 AAAI Spring Symposium on Computational Approaches for Analyzing Weblogs*
4. D'Ignazio, C., Bhargava, R., Zuckerman, E., & Beck, L. (2014). CLIFF-CLAVIN: Determining geographic focus for news. In NewsKDD: Data Science for News Publishing, at KDD 2014. New York, NY, USA.
5. Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370.  
<http://nlp.stanford.edu/~manning/papers/gibbscrf3.pdf>
6. Project visualization source code - <https://github.com/yusufnb/cs-410-project>