An Infrastructure to Store and Analyse Seismic Data as Suffix Trees

MSc Data Analytics - Project Proposal

Tom Taylor

April 2016

This report is substantially the result of my own work except where explicitly indicated in the text. I give my permission for it to be submitted to the JISC Plagiarism Detection Service. I have read and understood the sections on plagiarism in the Programme Handbook and the College website.

Contents

1 Abstract

The purpose of this project is to develop an infrastructure and tool set for converting raw seismic time series data into a searchable string using SAX (**S**ymbolic **A**ggregate appro**X**imation) and then to store this data as a suffix tree for fast searching and analysis. An interface will then be developed to enable the searching of these suffix trees and provide visualisation of the data. This enable the searching for similar patterns over time or between stations after an event.

The code should ideally be re-usable where possible in a project to receive live streaming data from many stations.

2 Background

2.1 Seismic Waves

Seismic waves take on two main forms, body waves and surface waves. Body waves are those that travel through the interior of the earth and are the fastest travelling. The body waves are comprised of **P** (primary) waves which are compressional waves and arrive first. **S** (secondary) waves are shear waves and travel more slowly, sometimes minutes behind. The surface waves travel only along the earth's crust and will normally arrive much later than the body waves.

A seismic station records movement over three axis: vertical (**z**) alongside horizontal in terms of north-south (**n**) and east-west (**e**). P waves are primarily observed on the z axis and so a P wave can be measured as a simple metric of displacement along the Z axis. S waves are primarily observed as movement across both the n and e axis. The geometry of the fault tends to have a bearing on the orientation of the displacement so the two axis of movement cannot be easily combined in to a single metric for time series analysis.

2.2 Symbolic Aggregate Approximation (SAX)

SAX (**S**ymbolic **A**ggregate appro**X**imation) (Lin et al., 2003) is a technique where by a single dimension of a time series is reduced to a string of symbols for pattern matching. The technique involves first transforming the normalised time-series in to a Piecewise Aggregate Approximation (**PAA**) which is then represented by a fixed number of symbols.

For the PAA, the data is first divided into equal sized time frames, then the

mean deviation from zero of each frame is calculated. An appropriate number of breakpoints symmetrical along the x-axis are created so that they follow a Gaussian distribution and a symbol assigned to each range between the breakpoints. Then for each frame, a symbol is assigned based on which range the mean falls in to. The symbols assigned to each frame are then concatenated in to a string and it is this string that gives the SAX representation of that data. The width of the time frame and the number of discrete regions would be two parameters passed to this process alongside the data.

2.3 Suffix Trees and the related infrastructure at Birkbeck

Suffix trees are a representation of a string designed for performing search related algorithms. They are based on a suffix trie of a given string the trie is constructed using all of the suffixes of the given text **T** with a terminator character not used in the alphabet (**A**). The terminator is deemed to be lexicographically of lower priority than all of the other characters to ensure that a prefix of a string would appear before the string. In the following example, the suffixes of *T: abaaba* would produce the following suffixes with **$** as a terminator.

```
T$:abaaba$: abaaba$
             baaba$      [ Diagram of constructed trie ]
              aaba$
               aba$
                ba$
                 a$
                  $
```

This allows for efficient searching and counting of substrings of T by following the trie from the root node. Each substring (**S**) of **T** is a prefix of a path starting at the root. It also allows for counting of occurrences of the substring by counting the number of leaf nodes that can be reached from the end of the substring. This format is not particularly space efficient as it has an upper bound storage in the order of $O(n^2)$.

A far more efficient way to store and query the trie is to store it as a Suffix Tree. In order to produce a suffix tree from a trie, the non-branching nodes are firstly reduced (or coalesced) in to a single edge. The original string **T** is then stored along side the tree and the labels further reduced to a starting position and offset within T. The leaf nodes become labels to the offset of the suffix in the string. This reduces the upper bound storage to $O(n)$ and results in a tree for our example **T** that looks as follows:

```
[ Include diagram of Suffix Tree for T ]
```

This technique of building the tree is considered naive though as it is very

resource hungry. A far more efficient way to build the tree is using the Ukkonens Algorithm (Ukkone, 1995). This technique allows for a time-linear *O(n)* construction of the tree.

`(Some words about Martyns tooling and the BBK infra)`

3 Objectives

1. Interpretation of raw data from seismic stations (in either the SAC or miniseed format) and storage as time-series data in a time-series database for easy access during conversion to SAX and for later rendering during interactions with the data.
2. Conversion of the data to SAX and storage as a suffix tree.
3. An interface for searching and viewing the raw data.

4 Approach

As laid out in the objectives, the development will produce three separate components.

4.1 Interpretation and Storage as time-series data

The raw data received from the stations is accessed as files in the SAC (Seismic Analysis Code) binary format. These will be read using the ObsPy python library which can read both the headers and return the raw data as a Python object.

This data will be batch processed and imported in to a time series database. Initially OpenTSDB is being considered for performance reasons but there may be scope for this to change depending on how difficult it is to implement this on the Universities hardware. Other options could be InfluxDB or Graphite which are considerably easier to implement but might suffer from scalability issues.

4.2 Conversion of data to SAX and building of the Suffix Tree

Once the raw data has been loaded to the time series database, it will be re-read in batches and passed though an algorithm to produce a SAX string as described in section 2.2. There are a few challenges in making this string usable. Firstly the seismic data

will need to have noise removed and to be normalised across multiple stations. Secondly the parameters of alphabet length and the size of the time window most suitable will need to be empirically found.

Once a suitable SAX string can be created across the dataset, it will be loaded in to a suffix tree using the toolset and infrastructure available at Birkbeck.

4.3 Development of an interface

The interface to query and view the data will be web based as to ensure maximum compatibility with clients. Many open source graph renderers are in existence such as Grafana and Cubism.js and components of both are likely to be used for viewing the raw seismic data alongside the SAX data.

5 Plan

As both ObsPy and the existing infrastructure for Suffix Tree storage are written in Python, this seems the logical choice. Specifically Python 3.5 will be used for all components.

Development will take the approach of Test Driven Development (TDD) where unit tests will be written at a class level before the classes are coded.

References

Lin, J., E. Keogh, S. Lonardi, and B. Chiu
2003. A symbolic representation of time series, with implications for streaming algorithms. *Data Mining and Knowledge Discovery.*

Ukkone, E.
1995. On-line construction of suffix trees. *Algorithmica.*