Whitepaper
TECHNICAL DETAILS

§

What does it do?

The application is designed to read XML transcriptions of ancient historical documents, and allow highly specific searches of content. In fact, as well as being able to search through documents simply by some given text, depending on the mark-up of the text, an almost limitless amount of textual features - for example, it is possible to search for unclear words, abbreviations, additions and deletions, foreign words, headings, notes, spelling errors and corrections, and unclear terms. It is also possible to search for a document by an identifier. Of course, this is not necessary; the application is designed to be easy to use, so that it is usable for anyone from a schoolchild to a researcher.

When a search is performed, the results are brought up, along with suggestions of similar words or possible different spellings of words. The results are shown with a heading and an extract, and metadata about the document, such as date.

When a result is brought up, it shows the original document in all its glory, along with the transcription of the text, and annotations on the text.

The software is not limited to searching for documents, however - it is also possible simply to browse through the library of documents, in chronological order, and through sub-parts of documents.

If a user wishes, they are also able to download the marked-up document, and a scan of the original.

§

Architecture

The application is designed to be run from the browser, allowing it to be accessed anywhere. Anyone can search or browse for documents, then view and download them. The search is handed to *Elasticsearch*, and

then a page is composed from the document database and presented to the user to display the search results. There is also a page to allow remote administration of the documents for certain authorized users. From this part of the application, documents may be uploaded and deleted.

§

Technologies

The application is based on *Ruby/Rails*, allowing it to be deployed and quickly and easily. It also used *Elasticsearch* for the search back-end, making the search abilities very flexible.

§

Machine

The application is designed to be fast and responsive – therefore, given the large amount of documents to be processed and served, it needs large amounts of memory – 64GiB is recommended[1], although less can be used. It's also good to have a multicore machine, although it does not need to be particularly fast.

§

Storage

Since the disk is generally by far the slowest part of a system, it makes a big difference how *fast* your disks are. If at all possible, SSDs should be used, as they are much faster than traditional media. Alternatively, if this is unfeasable, try to use the fastest spinning drives you can - 15,000 RPM drives are good.

§

[1] It's also a bad idea to have *more* than this

Limitations

Because of the way natural language works, it is only possible to give coherent suggestions for languages known by the software - this, unfortunately, means you won't be able to get amazing results for documents written in Linear A. For these languages, regular expressions are provided as a way to specialise search results.