Dept of Computer Science and Information Systems

BSc Final Year Project Form (2015/2016)

1. Proposal

The student should complete parts 1(a), 1(b) and 1(c) below, and then agree the maximum pocket values with the supervisor and put these in part 2(a) below. An electronic version of this form should be uploaded to the Final Year Project page on Moodle no later than **Monday 2nd November 2015**.

(a) Student details

Name:	
Email:	**Project Information Systems and Management Type 4 (BUCI027S6)**

(b) Project details

Title:
Design and development of an Intelligent System for classifying documents

Objectives:

To build an Intelligent System able to analyse and classify documents based on their content.
To demonstrate whether an Intelligent System as such would be able to provide quick but reliable output. In case of a positive result, it will be incorporated within a web API.

Steps (more details in the work plan section):

1. Identify user and technical requirements
2. Design of architecture, components and services
3. Design and implementation of preprocessing, machine learning and natural language processing algorithms
4. Implement and test software
5. Validate and evaluate machine learning algorithms
6. Iterate and improve

If the project results are suitable then create a Web Application:

7. Design of additional architecture, components and services

Description:

Information seeking on the Web can be a daunting and time-consuming experience, especially as most of the time people don't really know they are looking for. Another factor that can deteriorate their experience is the possible lack of familiarity with the search tool they are provided.

It is therefore responsibility of the service providers to aid users in their research process. They ought to provide appropriate tools to speed up, improve search queries and therefore the user experience with the system.

It is human nature to categorise anything that surrounds us. Categorization is the process by which ideas and objects are recognized, differentiated, and understood. [1]
It is therefore a sensible design choice to allow users to narrow the scope of their search by category (topic) in order to display the pertinent results.

The main objective of this project, is to build an Intelligent System to categorise documents by their subject.These categorised documents will be used by GOV.UK website's search engine allowing its users to run scoped queries on its database.

GOV.UK is the current website for the UK government and it's the recommended place to find information on government services.[2] However, it's also a fairly new website and therefore some advanced functionalities, like scoped search by topic, are missing. At present, the search engine on the site allows for scoped searches only by organisations tagged to a document (ie https://www.gov.uk/search?q=prime+minister).

A scoped search by topic, in order to be exhaustive, should require most (ideally all) of the documents to have a topic assigned to them. But this is far from the reality; out of ~170.000 documents, only ~8.000 are tagged to a topic.

There are a few reasons that cause the disproportion between the number of documents with and without tag. The possibility to tag documents was implemented later in the life of the site's publishing tools and, even when implemented, it has not been made a compulsory activity for valid reasons as tagging is a time consuming activity. It requires experts to read and understand documents and this can clash with the fact that often documents need to be published urgently. Enforced tagging might therefore become a counterproductive activity if done in a hurry.

What can be done instead, is making tagging an easier, faster and more pleasing procedure. This the goal of the AI that will be built in this project; suggesting topics for documents to users.
Topic suggestion can be separated in two parts:

- Bulk tagging untagged documents in background
- Real time tag suggesting documents when a publisher is working on it

Resources available:
All the necessary elements required for building the wanted Intelligent System appears to be easily accessible.
Labels:
These are the topics and they are provided. The AI will therefore not be required to infer or create them. But their existence does not exclude the possibility or necessity to create new ones. There might be untagged documents similar enough to each other to be tagged as the same topic, a topic that does not exist yet. The system could therefore take note of such documents, generate a placeholder label and report them to the user.

Topics have a 2 level hierarchy with parent and children topics. Documents can only be tagged to children topics.

Number of parent topics: 50
Number of children topics: 341

Data:

The documents are publicly available, accessible both from the site and from its API.

Training Data:

The tagged documents. Same as the regular data, publicly available and free to use.

Machine learning and natural language processing tools and software:

Python is the programming language of choice, multi-paradigm and available under open source license. There are several and well maintained machine learning and natural language libraries, such as:

- NumPy: Base n-dimensional array package
- SciPy: Fundamental library for scientific computing
- Matplotlib: Comprehensive 2D/3D plotting
- Natural Language Toolkit (NLTK): Natural language processing

Method:

In summary, this is available to build the system:

- Labels
- Labeled training data
- Data

It is evident that this is all that is required for supervised machine learning tasks.
“Supervised learning is the machine learning task of inferring a function from labeled training data.” [3]

Or more specifically, we have all the components required to build a supervised classifier:

A classifier is called supervised if it is built based on training corpora containing the correct label for each input. [...] Classification is the task of choosing the correct class label for a given input. In basic classification tasks, each input is considered in isolation from all other inputs, and the set of labels is defined in advance. Classification tasks have several variants. The one that most reflects our case is multi-class classification, where each instance may be assigned multiple labels.[4]; documents can be tagged to more or one topics.

With its characteristics, our classifier falls within the pattern recognition branch of machine learning. There are many algorithms that could be used to classify documents, these are ones that seem to fit best to our case:

- k-nearest neighbors algorithm (KNN):

“KNN is an non parametric lazy learning algorithm. [...] It does not make any assumptions on the underlying data distribution. [...] It does not use the training data points to do any generalization. [...] there is no explicit training phase or it is very minimal. This means the training phase is pretty fast . Lack of generalization means that KNN keeps all the training data. [...] KNN makes decision based on the entire training data set (in the best case a subset of them). [...]There is a non existent or minimal training phase but a costly testing phase. The cost is in terms of both time and memory. More time might be needed as in the worst case, all data points might take point in decision. More memory is needed as we need to store all training data.” [5]

- Naive Bayes classifier:

A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions. A more descriptive term for the underlying probability model would be "independent feature model". In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature.[6]

Due to the topics' hierarchical structure, it seems reasonable to have a system with the following architecture:
- a Root Classifier sitting at the top of the system connected to every subsequent child classifier. It will analyse whether a document given should be sent to any of its 50 sub-level classifiers
- a child classifier for each Parent topic. This, where appropriate, will assign one or more second-level topic for the document.

Work plan:

I will embrace the principle of agile software development especially focusing on changing requirements, even late in the development of the system.
Each of the following steps is time-blocked to 2/3 weeks which may overlap. They may also be started in different order and be repeated where appropriate.

1. Identify user and technical requirements
    - Interview end-users and technical architects familiar with pre-existing tools
    - Analyse data availability and format
2. Design of architecture components and services
    - Design scripts to collect data and store it in the most suitable form for the following processes
3. Design and implement pre-processing, machine learning and natural language processing algorithms
    - Feature selection
        - Title, corpora, publishing department, ministers, notable and departments mentioned
    - Feature extraction
    - Dimension reduction with:
        - Tokenizing, Stemming, Punctuation removal, Stop words, Speech tagging
    - Choice of most appropriate machine learning algorithm
4. Implement and test software
    - Test Driven Development to help keep the project on track and meet deadlines
    - Unit, feature and integration tests to support to make sure the system is solid and reliable
5. Validate and evaluate machine learning algorithms
    - Use graphs, user research and benchmarks to evaluate the quality of the system's output
6. Iterate and improve
    - Tweak and/or replace machine learning algorithms and document features until a reasonable compromise has been reached between speed and reliability

If the project results are suitable to live within a Web Application:

7. Design of additional architecture components and services
    - Implement API endpoints
    - Write exhaustive documentation for public use

College equipment required: none

None

References:

[1] Cohen, H., & Lefebvre, C. (Eds.). (2005).Handbook of Categorization in Cognitive Science. Elsevier.

[2] GOV.UK - About. https://www.gov.uk/help/about-govuk

[3] Mehryar Mohri, Afshin Rostamizadeh, Ameet Talwalkar (2012) Foundations of Machine Learning, The MIT Press ISBN 9780262018258.

[4] Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit. Steven Bird, Ewan Klein, and Edward Loper http://www.nltk.org/book/ch06.html

[5] Saravanan Thirumuruganathan. https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/

[6] Naive Bayes classifier. Source: http://www.ic.unicamp.br/~rocha/teaching/2011s2/mc906/aulas/naive-bayes-classifier.pdf Contributors: Akella, AlanUS, AllenDowney, Anders gorm, Anna Lincoln, Arauzo, Arichnad, Bewildebeast, Bkell, BlueNovember, Bovineone, Btyner, Caesura, Cagri, Calimo, Can't sleep, clown will eat me, Cancan101, Classifier1234, Coffee2theorems, CommodiCast, Cyp, David Eppstein, Dchwalisz, Ddxc, Den fjättrade ankan, Dianegarey, Dkemper, Doobliebop, Dstanfor, EverGreg, Fcady2007, Gene s, Geoffrey I Webb, Giftlite, Headlessplatter, HebrewHammerTime, Herlocker, Hike395, Icestorm815, Intgr, Jason Davies, JimD, Jklin, John Vandenberg, Johnnyw, Joseagonzalez, Jrennie, Justin W Smith, KKramer, Karada, Karipuf, Kotsiantis, Macrakis, Maghnus, MarkSweep, Mat1971, Mbusux, Mebden, Melcombe, Memming, Michael Hardy, Mitar, MrOllie, Mwojnars, Neilc, Neshatian, Newtman, NilsHaldenwang, Oleg Alexandrov, Olivier, OrangeDog, Orphan Wiki, PerVognsen, Peterjoel, Proffviktor, Prolog, RJASE1, RPHv, RedWolf, Rich Farmbrough,