TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Arvutiteaduse instituut

Raigo Aljand

Assessing Article Quality in Wikipedia Using Machine Learning Algorithms

bakalaureusetöö

Juhendaja: Jaagup Irve
Tarkvarainsener

Tallinn 2014

Abstract

[Here you should briefly describe the aims of your work] [Here you should briefly describe the main problems dealt with] [Here you should briefly describe the main results obtained]

The thesis is in [language] and contains [pages] pages of text, [chapters] chapters, [figures] figures, [tables] tables., etc.

List of Figures

List of Tables

Contents

Lisa 1 **30**

Sissejuhatus

1 Theoretical base

1.1 Programming language

Programming language is a formal language with a set of rules about how the computer should behave. The syntax of the language is usually considered the grammar of the language. For example, if there is a syntax problem in the program, the computer doesn't understand the command and stops. The semantics of the language is considered to be the vocabulary and meaning of the language. If there is a semantical problem, then the program is valid and understandable to the computer, but it is not what the programmer wished to happen. The program will react rather unpredictably. Syntactic sugar is a feature that the programmer can easily implement it in the language and is there for the convenience.

Since a programming language is only a set of rules, it needs an implementation. An implementation is a program that evaluates the syntax into machine actions. There are usually multiple ways to implement this. The most direct approach is to use interpretation, which parses the source code and performs the instructions directly. A second approach is to compile the program, which translates the source code into machine code and optimises it, and then run it. It is also possibe to compile a language into another language and then run from that language's implementation. A compiled program will usually run faster, because the program can be heavily optimised, but compiling the program may take some time. This disadvantage doesn't usually reach the user, because the programmer compiles the program before publishing the machine code to the user. The user can then choose the right file for its computer architecture. Compiling to a lower language has the advantage that the work of adapting for different processor architectures and optimising is already done, but it is possible to compile to a too general source code, which leads to less optimisations.

A third approach is to use just-in-time (JIT) compilation. JIT takes a hybrid approach between running it raw and compiling. While running the program, the JIT compiler will observe what parts of the program are most often run and will compile those parts. Because the analysation is fairly complex and runs parallel to the execution of the program, JIT compilers take a lot more memory.[2] The advantages are that the first operations start immediately and long programs still run almost as efficiently as a precompiled program. The disadvantages are the high memory usage and sudden pauses in the program execution while the necessary part is being compiled. Programming

languages are usually divided into two categories: high level languages and low level languages. The difference is that low level languages are designed more around how the computer works, while high level languages are designed more around the productivity of the programmer. Because high level languages need to do more translation between what the programmer wants and what the computer accepts, they are usually slower than low level languages. Another advantage of lower level languages is that they enable the programmer to have better access to specific operations and more understanding how the program is executed. This is very useful when the main goal is optimisation.

Another way to divide programming languages is static and dynamic programming languages. A programming language is considered static if it uses static typing. Static typing means that the type of a variable is determined prior to the execution of the program. This way the program doesn't have to check the type of a variable at the time of the execution and, therefore, can run faster. Dynamic typing holds the advantage that a variable can hold different types of values at different times and that types can be changed at runtime.

1.1.1 Procedural programming

Imperative programming is a style of programming with the philosophy of changing the machine's state to the required state, which could be a file in a hard drive or a video on the screen. For that the computer executes a sequence of operations in order, which are specified by the programmer. An operation is some change of state in the computer or calculation of some data.

Procedural programming is a subset of imperative programming. Procedural programming tries to divide the program into variables, data structures and procedures. Procedures are meant to group together abstract operations so they could be reused in different situations. In this case an operation can also be a procedure. Data structures are meant to group together conjoined data so they could be moved and manipulated more easily. Variables are pointers to data. They point to the location of the data in the memory which can be then easily retrieved.

1.1.2 Object-oriented programming

Object-oriented programming is another subset of imperative programming. Object-oriented programming tries to divide the program into objects that communicate with each other. Each object has fields and methods. Fields and methods are similar to variables and procedures, but they are tied to the object. Fields are considered to be the objects inner state and methods are considered to be the object's behaviour or object's interface.

The goal of object-oriented programming is encapsulation. Encapsulation means that each object has an inner state, inner behaviour and an interface for other objects to use. An outer object doesn't need to know what is happening within the object. It only needs to know how the object is going to react to an interface procedure. Access protection modifiers are generally employed to better enforce this behaviour. These modifiers are usually tied to a method or field and they describe what other methods are allowed to access these methods or fields. Right to access a field means the right to read or change the field and right to access a method means the right to run the method.[**?**] An object's methods always have access to its objects fields.

An object can inherit another object. The inheriting object is called subobject and the inherited object is called the superobject. The subobject gets the superobject's fields and methods. A copy of the superobject is created and retained in the subobject. When searching for the subobject's methods or fields and they are not found then the superobject is searched for the field or method. A subobject doesn't automatically have access to the superobject's private fields and methods. The subobject can override the superobject's public methods. The type signature of the method cannot change, but the content or the action of the method is changed.

The classic way to implement object-oriented programming is with classes. In class-based programming the object is separated into the class and the instance. The class is an abstraction and the classification of the object while the instance is a actual object with actual data. Usually the class holds the behaviour of the object, which includes the constructor. The constructor is a special method, that is called when a new instance is being created from the object. Inheritance works by remembering the inheritance line and then searching for the methods in the right class.

An interface is a a class that has no fields and all its methods are public. All of the methods are abstract methods, meaning that the methods have no content or implementation. A class can usually implement multiple interfaces

but inherit from only one class. One can't make a instance of an interface, there needs to be an implementing class. An abstract class is a class that has atleast one abstract method. Similar to interfaces, it is impossible to make an instance of them. However they are still classes and and they are inherited, not implemented.

In my humble opinion the rawest object-oriented programming style is protoype-based programming. Prototype-based programming is more used by dynamic languages. It keeps the design of the language minimal by having no wrappers around objects and their attributes. An object is created by creating an empty object or by cloning an existing object. Inheritance works by cloning an existing object and the subtype can replace the old attributes or add new attributes. Abstract objects are objects that will hold only behviour by containing only constant or default attributes.

1.1.3 Functional programming

Functional programming languages are designed around functions. Programming functions are similar to mathematical functions, that it has inputs as parameters and returns a value as output. A function should always return the same output with the same inputs and the inputs should not be changed inside the function. This leads to a particularly stateless form of programming.

Due to the stateless form of this style, functional programming languages usually support immutable datastuctures. Instead of updating a datastructure, it is copied with the new values replaced and the new datastucture is returned. First-class functions and dynamic evaluation of functions are also supported.

1.2 Client-server architecture

Client is process that requires some service. Server offers that service. The client and server communicate with HTTP. HTTP is a protocol in which the client sends the server a request and the server processes the request and sends a response. HTTP is purely plaintext. The HTTP request is divided into headers and content. The request has an URL in the header. URL stands for Uniform Resource Locator. It is a way for the client to request a certain page or other resource on the server. Another thing that the URL can

contain are extra parameters. From those parameters the server can return the resource in a different form. Usually the request content is empty.

Extensible Markup Language (XML) is a protocol to describe data. It tries to nest data between descriptive tags. The tags can be nested and the tags can have attributes. The tags are not preset, so every user can design it's own way of presenting data. The syntax of a XML tag is `<tag attribute1="value1" attribute2="value2">data</tag>`. A tag can't have the same attribute with different values. data can be normal text or more tags. `<tag />` is shorthand for `<tag></tag>` for when the user doesn't have data to insert.

The HTTP response also splits into a header and content. Inside the content is usually the requested data. A browser is an application that sends HTTP requests to servers and parses and visualises the response to the user. To help with the visualisation and interactiveness of the pages, HTML was created. HTML is an Extensible Markup Language (XML) where the tags are focused on giving text some form of context. For example, the a is a tag for a link and the `h1` is a tag for a level 1 header. The browser also parses the HTML into visual cues and interactions with the user.

Often the tags in HTML aren't enough. For that there is JavaScript (JS). JavaScript is a dynamic general-purpose programming language. JavaScript works by registering a JavaScript function with an HTML event, so that when that event is fired, that JavaScript function is run. JavaScript interacts with the user by changing the current HTML the user is seeing. Because JavaScript is a general-purpose programming language, unlike HTML, any arbitrary calculation is possible. JavaScript uses Asynchronous JavaScript and XML (AJAX) to send HTTP requests to outside servers. Despite the name, the returned format doesn't have to be XML.

HTML pages are purely intended for browsers to parse and a human to see. For an another application to get data from a HTML page, it has to web scrape. Web scraping is observing beforehand how a web page is built and later filtering out the necessary data from the HTML. Another way is for the server to offer an Application Programming Interface (API). An API is a way for a program to get data from the server with a HTTP requst in a more formal and machine-friendly form. There are multiple formats for getting the data: JavaSrict Object Notation (JSON), a Domain specific XML and so on.

With JSON the HTTP content is one legal JavaScript Object, which makes it easy to read into JavaScript. Another advantage of it over XML is that there is basic type checking. Javascript allows a value to have a few different

types with different notations: a string, a number, another object, an array; and the three constant values `true`, `false`, `null`.

1.3 Machine learning

Machine learning is used when the programmer doesn't know all about the domain he is writing for. It is then necessary to have the machine learn by itself by some criteria. Machine learning generally divides into two categories: supervised learning and unsupervised learning. Supervised learning is when you have a known data set where for a known input there will be a known output. In unsupervised learning there is no such data set and the programmer is mostly looking for correlation between the inputs. Unsupervised learning is mostly used for data mining.

Logistic regression is a form of supervised learning which is used for classification. Our dataset D will consist of the input of the model, a matrix of the the vectors $\mathbf{x_n}$ and the required outputs y_n for the inputs. A input vector $\mathbf{x_n}$ will consist of the numerical features of the data prefixed with a 1 for the bias. y_n can have only 2 values: 1 or -1. There are more complex forms of logistic regression that can handle more than two values for y but is out of the scope of this paper. N will be the size of our dataset.

$$D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_N}, y_N)\}$$

$$\mathbf{x_n} = [\ 1 \quad x_1 \quad \ldots \quad x_d\ ]^T$$

Similar to another supervised learning algorithm linear regression, logistic regression is a linear model."All linear models make use of a "signal" s which is a linear combination of the input vector $\mathbf{x}$ components weighed by the corresponding components in a weight vector $\mathbf{w}$."[**?**]

$$\mathbf{w} = \left[\ w_0 \quad w_1 \quad \ldots \quad w_d\ \right]^T$$

$$s = w_0 + w_1 x_1 + \cdots + w_d x_d = \sum_{i=0}^{d} w_i x_i = \mathbf{w} \cdot \mathbf{x} = \mathbf{w}^T \mathbf{x}$$

Linear regression will use the signal directly as output, but logistic regression will pass the signal through a sigmoid or logistic function and treat thats output as the probability that $y = 1$.

$$h(\mathbf{x}) = \theta(s)$$

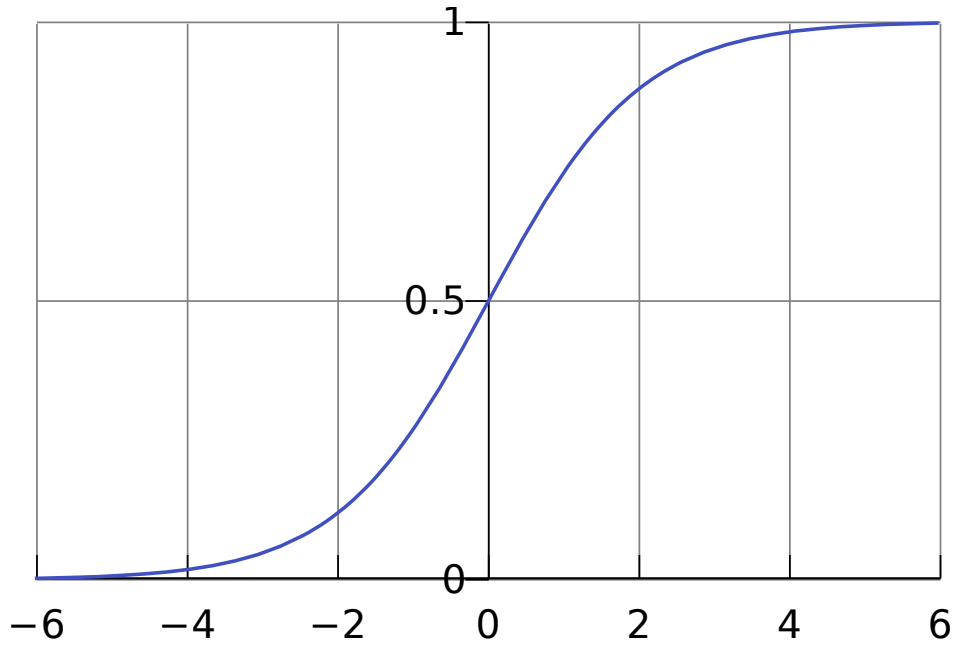

Figure 1: The shape of a logistics curve.

$$\theta(s) = \frac{e^s}{1+e^s} = \frac{1}{1+e^{-s}}$$

As shown in figure 1, the logistic function is good for translating between linear values and probability, because the higher the linear value, the higher the probability of the output value being 1 and the lower the linear value the higher the probability of the output being -1. At input value 0, the probability of it being either value is 0.5.

"We say that the data is generated by a noisy target."[**?**]

$$P(y|\mathbf{x}) = \begin{cases} f(\mathbf{x}) & \text{for } y = +1 \\ 1 - f(\mathbf{x}) & \text{for } y = -1 \end{cases}$$

We want to learn a hypothesis $h(x)$ that best fits the above target according to some error function.

$$h(\mathbf{x}) = \theta\left(\mathbf{w}^T\mathbf{x}\right) \approx f(\mathbf{x})$$

"It's important to note that the data does not tell you the probability of a label but rather what label the sample has after being generated by the

target distribution."[**?**]. The goal of the training will be to calculate the weight vector w so that it minimizes some kind of in-sample error measure.

$$\mathbf{w}_h = \arg\min_w E_{in}(\mathbf{w})$$

Our error measure will be based on likelihood. Likelihood is the probability of generating the data with a model. Likelihood will be high if the hypothesis is similar to the target distribution. *TODO: Controversy? What? Millegipärast pidi likelihood olema controversial, aga ma ei saanud aru, miks.* Let's assume that the data was generated by the hypothesis:

$$P(y|\mathbf{x}) = \begin{cases} h(\mathbf{x}) & \text{for } y = +1 \\ 1 - h(\mathbf{x}) & \text{for } y = -1 \end{cases}$$

$$h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x})$$

Let's try to remove the cases using the property $\theta(-s) = 1 - \theta(s)$.

$$\left.\begin{array}{l} \text{if } y = +1 \text{ then } h(\mathbf{x}) = \theta(\mathbf{w}^T\mathbf{x}) = \theta(y\mathbf{w}^T\mathbf{x}) \\ \text{if } y = -1 \text{ then } 1 - h(\mathbf{x}) = 1 - \theta(\mathbf{w}^T\mathbf{x}) = \theta(-\mathbf{w}^T\mathbf{x}) = \theta(y\mathbf{w}^T\mathbf{x}) \end{array}\right\} P(y|\mathbf{x}) = \theta(y\mathbf{w}^T\mathbf{x})$$

Let's denote an arbitrary hypothesis g, in which case the likelyhood is defined as:

$$L(D|g) = \prod_{n=1}^{N} P(y_n|\mathbf{x}_n) = \prod_{n=1}^{N} \theta(y_n\mathbf{w}_g^T\mathbf{x}_n)$$

To find the best hypothesis, we have to find the best weight vector $\mathbf{w}$.

$$\begin{aligned} \mathbf{w} &= \arg\max_{\mathbf{w}} L(D|h) = \arg\max_{\mathbf{w}} \prod_{n=1}^{N} \theta\left(y_n\mathbf{w}^T\mathbf{x}_n\right) = \arg\max_{\mathbf{w}} \ln\left(\prod_{n=1}^{N} \theta\left(y_n\mathbf{w}^T\mathbf{x}_n\right)\right) \\ &= \arg\max_{\mathbf{w}} \frac{1}{N} \ln\left(\prod_{n=1}^{N} \theta\left(y_n\mathbf{w}^T\mathbf{x}_n\right)\right) = \arg\min_{\mathbf{w}} \left[-\frac{1}{N} \ln\left(\prod_{n=1}^{N} \theta\left(y_n\mathbf{w}^T\mathbf{x}\right)\right)\right] \\ &= \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} \ln\left(\frac{1}{\theta\left(y_n\mathbf{w}^T\mathbf{x}_n\right)}\right) = \arg\min_{\mathbf{w}} \frac{1}{N} \sum_{n=1}^{N} \ln\left(1 + e^{-y_n\mathbf{w}^T\mathbf{x}_n}\right) \end{aligned}$$

We have derived a good form for the error measure, which is the loss function or the average point error.

$$E_{in}(\mathbf{w}) = \frac{1}{N} \sum_{n=1}^{N} \ln\left(1 + e^{-y_n\mathbf{x}_n\mathbf{w}^T}\right) = \frac{1}{N} \sum_{n=1}^{N} e\left(h\left(\mathbf{x}_n\right), y_n\right)$$

$$e\left(h\left(\mathbf{x}_n\right), y_n\right) = \ln\left(1 + e^{-y_n\mathbf{x}_n\mathbf{w}^T}\right)$$

We minimise the error function using gradient descent. Gradient descent works by moving the current value towards the local minimum. With the derivative, one can calculate the necessary direction and the rough distance of the local minimum from the current value. Therefore training works with the formula:

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \nabla E_{in}\left(\mathbf{w}_i\right)$$

Where η is the learning rate. We need the derivative of the point error function and the average point error.

$$\frac{d}{d\mathbf{w}} e\left(h\left(\mathbf{x}_n\right), y_n\right) = \frac{-y_n \mathbf{x}_n e^{-y_n \mathbf{w}^T \mathbf{x}_n}}{1 + e^{-y_n \mathbf{w}^T \mathbf{x}_n}} = -\frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}}$$

$$\begin{aligned} \nabla E_{in}\left(\mathbf{w}\right) &= \frac{d}{d\mathbf{w}} \left[\frac{1}{N} \sum_{n=1}^{N} e\left(h\left(\mathbf{x}_n\right), y_n\right)\right] = \frac{1}{N} \sum_{n=1}^{N} \frac{d}{d\mathbf{w}} e\left(h\left(\mathbf{x}_n\right), y_n\right) \\ &= \frac{1}{N} \sum_{n=1}^{N} \left(-\frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}}\right) = -\frac{1}{N} \sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}^T \mathbf{x}_n}} \end{aligned}$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \left(-\frac{1}{N} \sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}_i^T \mathbf{x}_n}}\right) = \mathbf{w}_i + \eta \left(\frac{1}{N} \sum_{n=1}^{N} \frac{y_n \mathbf{x}_n}{1 + e^{y_n \mathbf{w}_i^T \mathbf{x}_n}}\right)$$

To lower the number of iterations, each feature of $\mathbf{x}$ should be normalised before it is used for predicting or training the model. Normalising means that the program calculates the standard score of each of the features which is then used instead. The standard score subtracts the mean from the features and divides that with the standard deviation of the values. A values standard score floats around the 0 value and roughly has the same absolute value as other standard scores. When normalising $\mathbf{x}$ for predicting, the mean and standard deviation cannot be enhanced with that data, because then the trained model will not be expecting such data. It would mean comparing two fundamentally different sets of data. It is important to note that the bias variable in $\mathbf{x}$ should not be normalised, because it will end in a divide by zero error.

2 Used technologies

2.1 Python

In this project I used Python version 2.7.

Python is a widely-used, general-purpose programming language as shown in figure 2. Python focuses on human readability, simplicity and power of the programmer. It is most likely best expressed by the fact that Python uses indentation to divide its blocks. A language like C uses the curly braces for that and uses the indentation only for clarity of reading. Python, however, forces the programmer to make the program more readable and standardised. Semantically Python is very flexible. It supports functional, object-oriented and procedural programming styles.

To minimise clutter in the language, Python uses dynamic typing, also called duck typing. Duck typing doesn't check if an object implements some certain interface, but simply tries to call or use the method or attribute. The principle is "If it looks like a duck and quacks like a duck, it must be a duck".[**?**, duck-typing] This makes it possible to not use interfaces and generics in the code.

Python has two running modes: the interactive mode and the script mode. The interactive mode is a read-eval-print loop. It reads the user supplied expression and parses it into a data structure, evaluates it and prints the result. An expression can't be multiline and must return only one value. Evaluating a value will return that value and evaluating a variable returns the contents of the variable. The script running mode is similar to the interactive mode, but instead of the user, the expressions are taken from a file and when the end of the file is reached, the interpreter terminates.

2.1.1 Procedural style

Each file in python is a module with the same name as the filename, except without the `.py` extension. Each module can import another module or every object in another modules namespace with the `import` command. When the import command is evaluated, the whole file is evaluated once. A directory can also be a module if it contains a `__init__.py` file. In that case the module is initialised with the `__init__.py` file and it's namespace contains the file modules in the directory.

Jan 2014	Jan 2013	Change	Programming Language	Ratings	Change
1	1		C	17.871%	+0.02%
2	2		Java	16.499%	-0.92%
3	3		Objective-C	11.098%	+0.82%
4	4		C++	7.548%	-1.59%
5	5		C#	5.855%	-0.34%
6	6		PHP	4.627%	-0.92%
7	7		(Visual) Basic	2.989%	-1.76%
8	8		Python	2.400%	-1.77%
9	10	⌃	JavaScript	1.569%	-0.41%
10	22	⌃⌃	Transact-SQL	1.559%	+0.98%
11	12	⌃	Visual Basic .NET	1.558%	+0.52%
12	11	⌄	Ruby	1.082%	-0.69%
13	9	⌄⌄	Perl	0.917%	-1.35%
14	14		Pascal	0.780%	-0.15%
15	17	⌃	MATLAB	0.776%	+0.14%
16	45	⌃⌃	F#	0.720%	+0.53%
17	21	⌃⌃	PL/SQL	0.634%	+0.05%
18	35	⌃⌃	D	0.627%	+0.33%
19	13	⌄⌄	Lisp	0.604%	-0.35%
20	15	⌄⌄	Delphi/Object Pascal	0.595%	-0.32%

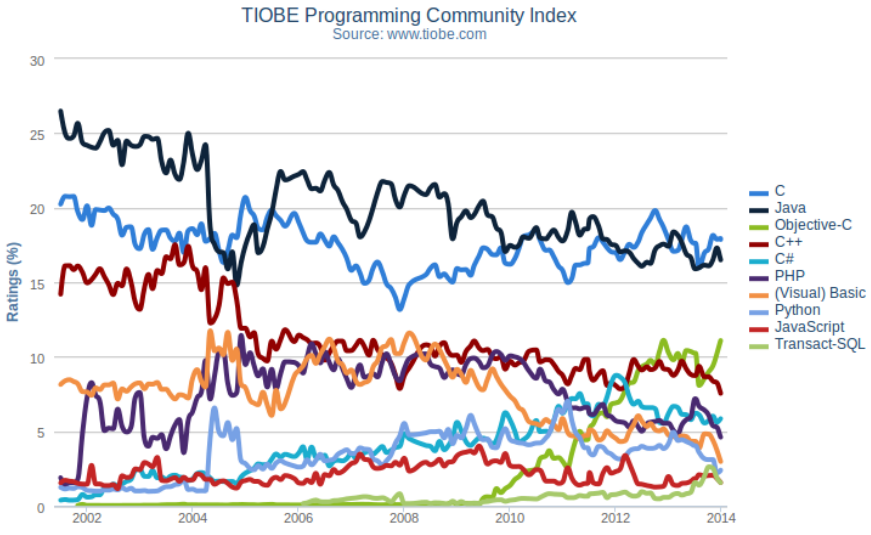

Figure 2: Popularity of python compared to other languages[1]

Procedures in Python are called functions, because they can return one or multiple values, however if a return value isn't specified, it returns the `None` value. Python uses pass by reference so that values inside the object can be changed. However one must still be careful that changing the reference itself won't change the reference in the caller. Also even though numbers are objects and therefore are passed by reference, they are immutable and one can't change their inner value. That makes them in essence pass by value.

The closest feature that gets to a C style stucture is the named tuple. A named tuple is a factory function that will create and return a class. A tuple is an immutable list. The inputs of the function are the class name, the fields of the class and more optional parameters. The returned class inherits from the tuple type and only accepts the previously entered fields. An example:

```
>>> Point = namedtuple('Point', ['x', 'y'], verbose=True)
class Point(tuple):
    'Point(x, y)'

    __slots__ = ()

    _fields = ('x', 'y')

    def __new__(_cls, x, y):
        'Create a new instance of Point(x, y)'
        return _tuple.__new__(_cls, (x, y))

    @classmethod
    def _make(cls, iterable, new=tuple.__new__, len=len):
        'Make a new Point object from a sequence or iterable'
        result = new(cls, iterable)
        if len(result) != 2:
            raise TypeError('Expected 2 arguments, got %d' % len(result))
        return result

    def __repr__(self):
        'Return a nicely formatted representation string'
        return 'Point(x=%r, y=%r)' % self

    def _asdict(self):
        'Return a new OrderedDict which maps field names to their values'
        return OrderedDict(zip(self._fields, self))
```

```
    def _replace(_self, **kwds):
        'Return a new Point object replacing specified fields with new values'
        result = _self._make(map(kwds.pop, ('x', 'y'), _self))
        if kwds:
            raise ValueError('Got unexpected field names: %r' % kwds.keys())
        return result

    def __getnewargs__(self):
        'Return self as a plain tuple.   Used by copy and pickle.'
        return tuple(self)

    __dict__ = _property(_asdict)

    def __getstate__(self):
        'Exclude the OrderedDict from pickling'
        pass

    x = _property(_itemgetter(0), doc='Alias for field number 0')

    y = _property(_itemgetter(1), doc='Alias for field number 1')


>>> p = Point(11, y=22)     # instantiate with positional or keyword arguments
>>> p[0] + p[1]             # indexable like the plain tuple (11, 22)
33
>>> x, y = p                # unpack like a regular tuple
>>> x, y
(11, 22)
>>> p.x + p.y               # fields also accessible by name
33
>>> p                       # readable __repr__ with a name=value style
Point(x=11, y=22)
```

If a variable is nonlocal and the variable in the function is only read, then the interpreter will try to find it from a nonlocal context. However, if the variable is given new value anywhere in the function, the interpreter will assume that the variable is local and the global variable will not be visible. One can make a variable global with the `global` command. The syntax is `global variable`. After that, every reference to the variable will be a reference to the global variable.

2.1.2 Functional style

THIS IS WRONG: FIRST EXPLAIN DEF AND THEN SAY LAMBDA AS THE LIMITED FORM OF DEF Python has full support for first-class functions meaning that a function is an object like any other value. One defines a function with the `lambda` keyword or with the `def` keyword. `lambda` and `def` are very similar, but `def` binds the function value to an identifier and `def` implicitly returns `None`. For `def` to return anything, it needs the `return` keyword. `lambda` is more constrained. It is not allowed to be multiline and it is only allowed to have one expression. The expressions value is then implicitly returned. The syntax of `def` and `lambda` are:

```
def fib(n):
    a, b = 0, 1
    while a < n:
        print a,
        a, b = b, a+b
    return a

f = lambda x: x**2

# Now call the function we just defined:
f(2000)
fib(2000)
```

After the `def` command comes the name of the function, then inside the brackets are the names of the parameters of the function separated by a colon. A function can have optional parameters by giving the parameter a default value like this: `parameter=value`. The expression of the value is only evaluated once, so if the parameters value is changed during the call of the evaluation of the function, the change will be seen in the future evaluations of the function. On the next lines is the body of the function.

The `lambda` command doesn't take a name, therefore brackets are not necessary and the parameters simply follow the `lambda` command again separated by commas. After the colon is the expression to be returned. After the definition and binding to an identity, both function values are of the same type and follow the same rules.

Python also has lexical closures, which gives the function a strong reference to the namespace. This excludes the possibility that the namespace will be garbage collected while the function is still in memory. The function has the

guarantee that the non-local values will exist even after the enclosing context is deleted or garbage collected.

Generator is a data structure to dynamically iterate items. When a function has a `yield` command somewhere inside of the function, when the function is evaluated, the function is actually not run, but a generator data structure is returned. The `yield` command is very similar to `return`, except for one difference. When the generator is iterated, for example in a `for` loop, the function is run until a `yield` command is hit, the value is returned and the function is paused. When the next element is requested, the function is continued again until the next `yield`. This continues until the function reached the end, which signals that the generator has no more elements.

```
def generate_ints(N):
    for i in range(N):
        yield i
```

The advantage of a generator over a list for example, is that lists require that all their elements are in memory at some point, while a generator has only one element at a time in memory. Because when using a generator one can iterate over the items only once, there is also a slight speed increase. The disadvantages are that generators have no such data as the length of the iteration, are immutable and can't be iterated over for a second time.

Since other functional languages have them, Python also has functions `map`, `filter` and `reduce`. `map` applies a function to every item in an iterable and returns a new list with the results of the function. `filter` applies a function to every item in an iterable and returns a new list with the items where the function returned `True`. And `reduce` applies a function for every item in an iterable and returns the value of the last evaluation. In this case the function must have two parameters and the first parameter holds the value from the last evaluation of the function.

A way to avoid using `map` and `filter` is by generator expressions and list comprehensions. List comprehensions are a syntactic sugar to easily create lists. Unlike `map`, list comprehensions don't need to be supplied a function, but can also use arbitrary expressions. A sample list comprehension is: `[2 * item for item in iterable if item % 2 == 0]`. This expression returns a filtered list where each element of iterable is multiplied with 2. The returned list contains only elements that were even before. Therefore the syntax is `[expression for expr1 in sequence1 if condition1 for expr2 in sequence2 if condition2 ...]`. The commands are nested with the right being inside the left and the first expression being the returned inner-

most expression. Generator expressions are syntactically same, except normal brackets instead of square brackets are used and they create a generator instead of a list.

2.1.3 Object-oriented style

Python has a class-based object-oriented style. Python, however, is more dynamic than a normal static class-based language like Java. After an object has been created from a class, one can still change that concrete object's variables and methods. Every property is also public. Properties, that the programmer considers private, are usually prefixed with underscores. Each statement is evaluated top to bottom. If there are multiple properties with the same name, then the last evaluated property is remembered.

```
class MyClass:
    """A simple example class"""
    i = 12345
    def f(self):
        return 'hello world'
```

In Python, methods are functions, that get the object's instance in the first parameter, but are called with the first parameter ignored, like this: `my_object.f()`. If there are brackets after the class name and another class's name inside the brackets, the class inherits from the class in the bracket. A method can access the superclass with the `super` function. The super function takes two arguments: the type of the class of whose the super is being searched for and the second is the object.

Multiple inheritance is supported by putting multiple class names in the brackets supported by comas. If called for a property, that a object doesn't have, the environment will try to find the property from the first named superclass recursively until it hits object class and then the next superclass recursively and so on. The object class is the superclass of every class. If there are no brackets or the brackets are empty, then the object class is an implicit superclass.

2.1.4 Implementation

Python uses the CPython implementation by default. It is the most supported and used. CPython first compiles the python code into C code, which is then run. This implementation also gives the ability to write some parts

of the C code, which would be normally automatically generated, by hand. Since the average generated code is a lot slower than hand written code, you can develop most of the program in Python and the bottlenecks can be written in C.

The second most popular implementation is PyPy. PyPy is a JIT compiler written in Python. It is generally about 6.2 times faster than Cpython. One disadvantage is that it is not possible to integrate C and PyPy. Therefore the program will become generally faster, but it is not possible to incredibly speed up the bottlenecks that might exist in the program. Another disadvantage is that PyPy does not support Python 3 yet.

There are also Jython and IronPython. Jython compiles down to Java bytecode. It allows the programmer to use libraries from Java in his project. IronPython is the same principle, only it compiles to the .NET bytecode. That gives IronPython projects access to .NET libraries.

2.2 MediaWiki

A wiki page is a page that owns a title in that wiki and is supposed to aggregate information on the world wide web about the subject of the title. Most of the content in the wiki pages are created or changed by the users of the wiki. MediaWiki is software for a content-classification wiki. A content classification wiki forbids writing personal knowledge and opinions on the wiki page and only allows writing information that can be referenced. MediaWiki has a separarate web page for discussion on a wiki page.

An interwiki link is a link that refers to another wiki page in the same wiki. This way wikis don't have to duplicate information on different pages and just refer to an another page. Links that link to outside the wiki are called external links. Usually external links are in the page as references. There is a special type of wiki page that are called templates. They are not meant to aggregate information, but that to act as a form template. Whenever a page links to a template, it expands to form a part of the page it is linked on.

MediaWiki has 3 core dependencies. It is completely written in php, therefore it needs php on the system it is running on. It also needs to run on a http server like Apache that is also configured to have php enabled and index.php as a root file. Running MediaWiki is an act of putting the MediaWiki directory in the http server root directory and pointing the browser to the MediaWiki directory. MediaWiki also needs a database service. Officially it supports MySQL 5.0.2+, MariaDB 5.1+, PostgreSQL 8.1+, SQLite

3 and Oracle.

The first time MediaWiki is run, the user is guided through an installation process which will install the database in the database service and create a config file that must be placed in the MediaWiki root directory.

MediaWiki also has an API that exposes the MediaWiki articles to bots without the need to screen scrape. For that in the MediaWiki root directory is the file api.php. Queries made at api.php with the right query parameters will return bot friendly results. For example: `http://en.wikipedia.org/w/api.php?format=json&action=query&titles=Main%20Page&prop=revisions&rvprop=content`

The format parameter tells what format the bot wants the data to be in. Each format also has a version with a fm added to the end. The fm version pretty-prints what the bot would have seen in html so it is easy for the programmer to debug with the browser. Different possible formats include json, xml, serialized php, wddx and yaml, where xmlfm is the default. However json is the recommended format and all the other formats are deprecated.

The action parameter is the second required parameter. It tells what action the bot wants to do in the wiki. The rest of the query parameters are specific to the action. Most important to us is the query action, which is used to query data from wikipedia. The titles parameter is used to specify pages by name to query. Another way to specify pages is using a generator. Generators generate a list of pages. A bot can give a generator additional arguments by prefixing the parameters with the letter g. For example: `http://en.wikipedia.org/w/api.php?action=query&generator=allpages&gaplimit=3&gapfrom=Ba&prop=links|categories`

This example gets the links and categories of the first three pages. The `generator` parameter tells the API which generator use to generate the pages. The `gaplimit` tells the `allpages` generator how many pages should the generator generate at one time. The `gapfrom` tells what page the API should start listing from. The pages are alphabetically ordered. By default the `allpages` generator uses the main namespace to generate pages from. Another significant generator is `random` that generates pages randomly.

To limit a single user from putting a server on too much of a load. The API allows only a certain number of pages to be queried at once. To have bigger queries, the bot has to use continues. To let the server know, that the bot supports continues, the bot has to add the `continue` parameter with an empty value to the query. When the query is big enough, the server returns a dictionary under a `continue` parameter. The bot then takes the dictionary

and queries again with the same arguments except the dictionary added as query parameters and arguments. The original `continue` argument will be overwritten. If it is not possible to continue, the server will not return a continue parameter.

2.3 PyWikiBot

PyWikiBot is a Python framework which allows an application to communicate with a MediaWiki instance through the MediaWiki API. Firstly (IN MY HUMBLE OPINION) the core component of PyWikiBot is the Site object. The site object holds the connection to the MediaWiki instance that PyWikiBot is supposed to query from. It is instantiated without arguments. Instead it takes the instantiation data from the current PyWikiBot configuration.

PyWikiBot is configurable with the `user-config.py` file. On a linux system the `user-config.py` file has to be in the directory `~/.pywikibot`. The `user-config.py` file is created with the script `generate_user_files.py`. It will ask the necessary questions and then generate the `user-config.py` file in the necessary location.

All bots in Wikipedia should have a user account that the bot is using to query and make changes from. All those user accounts should also be flagged as bots, so the admins can make better informed decisions in case of high load. PyWikiBot follows this principle by not being able to run anonymously. When generating user files, PyWikiBot will ask what will be the user name to run with. When running the bot, PyWikiBot will also ask the user for the password. PyWikiBot will remember the password and it is usually not necessary to write the password in again even after a computer restart.

The site that PyWikiBot will query from is dependent on the family and the language. These two are also asked when generating user files. A family is a group of sites that are grouped together according to some theme. A language is the the 2 character code of the language that the site was written in. The url of the site doesn't have to be `ll.family.tld`, where `ll` is the language and `tld` is the top level domain of the site, like how it is with wikipedia. Each site can have it's own url.

2.3.1 Page

The page object holds data about a single wiki page. A page instantation can have 3 arguments. First is the source from which the page will be loaded.

The second is the title of the page. The third is the namespace from which the page is loaded. The first argument is obligatory and the instantiating values depend on it. If the source is an another page, then it will make a copy of the page with the title overridden if it is given. If the source is a site, then it reads the title and namespace argument and creates a link from them. If the source is a link instance then it is remembered and the rest of the arguments are ignored.

A link represents a link to a page. A link has 3 instantiation arguments. The first is the text of the link. The second is the site that the link is on. And the third is the namespace to default to if the link text doesn't contain the namespace where the link points to. The text is the only obligatory argument. If no site is given then a new site is initialised. Unless otherwise specified the default namespace will be the main namespace.

Another way to get pages is to use pagegenerators KAS SEE LAHKU EI KIRJUTATA?. These generate pages using the MediaWiki generator API. There is a PyWikiBot pagegenerator for each MediaWiki generator.

3 The analyser

The purpose of this project is to separate good articles from average articles.

3.1 Prerequisites

The author used Ubuntu 13.04-13.10 to make this program and hasn't tested installing and running it on other systems. Proceed on your own caution.

I switched to arch so I should test and document setting it up for arch.

This project requires the Python 2 interpreter on the system. Python 3 is not possible, because PyWikiBot doesn't support it. One can install it with the terminal command `sudo apt-get install python2`. The Python library numpy is also necessary to be installed. One can use the command `sudo apt-get install python-numpy`.

A user-config.py configuration file must also exist. To generate it, there is a script in the pywikibot folder. It's named `generate_user_files.py` and and it must be run with the Python interpreter. This is a sample installation process:

```
project-directory/pywikibot$ python generate_user_files.py

Your default user directory is "~/.pywikibot"
How to proceed? ([K]eep [c]hange)
Do you want to copy user files from an existing pywikipedia installation? No
Create user-config.py file? Required for running bots ([y]es, [N]o) y
1: anarchopedia
2: battlestarwiki
3: commons
4: fon
5: gentoo
6: i18n
7: incubator
8: lockwiki
9: lyricwiki
10: mediawiki
11: meta
12: mywiki
13: oldwikivoyage
```

```
14: omegawiki
15: osm
16: southernapproach
17: species
18: strategy
19: test
20: testviki
21: vikidia
22: wikia
23: wikibooks
24: wikidata
25: wikimedia
26: wikinews
27: wikipedia
28: wikiquote
29: wikisource
30: wikitech
31: wikiversity
32: wikivoyage
33: wiktionary
34: wowwiki
Select family of sites we are working on, just enter the number not name (defaul
This is the list of known language(s):
ab ace af ak als am an ang ar arc arz as ast av ay az ba bar bat-smg bcl be be-x
The language code of the site we're working on (default: 'en'): et
Username (et wikipedia): Analysebot
Which variant of user_config.py:
[S]mall or [E]xtended (with further information)? S
Do you want to add any other projects? (y/N)
'~/.pywikibot/user-config.py' written.
Create user-fixes.py file? Optional and for advanced users ([y]es, [N]o)
```

Whenever there seems to be a question, but no answer, it is because the default one was correct and pressing enter without an answer selects the default answer. Often the default answer is marked by being upper case. In the real installation process the tilde signs would actually be written out user home directories. It was changed in this paper for generalisation. Also the username of the bot will be different for installations.

Before the user can use the bot in any way, the bot needs to be trained for the first time with the train command. The bot expects a training file to exist, which will be created at the time of training the bot. One can do that

with the command `python main.py train`.

3.2 Interface

This project is run through the Command Line Interface. The user will find the file to run in the pywikibot/analyse folder. The file is called main.py and it needs to be run in the python environment with the command `python main.py`. After the running command the user can add subsequent commands for the program to do different tasks, for example `python main.py train`. The `train` command will either train the program for the first time or will re-train the program if the for example the wiki has been updated. Then it will test the algorithm and give the user the accuracy of the algorithm.

The `find` command will find all the good articles in the current wiki. This command will take a long time because the bot will have to go through the whole wiki to check every page. *Kas ma lisan ka nime järgi filtreerimise, et aega vähendada?* It will then write the list of good pages in a file named `good_pages.pkl` using the Python library `pickle` and print it out on the terminal. *Kas ma teen midagi muud selle listiga?*

3.3 Architecture

Since, PyWikiBot is a framework and not a library, all of the code is under the pywikibot folder. In it is another pywikibot folder and the analyse folder. In the pywikibot folder is all of the code for the PyWikiBot framework and under the analyse folder is the code written for this paper.

In the analyse folder, the main module is designed to be the interface between the user and the bot. The other modules are libraries for the main module. In the config module is the configuration variables for the program. *There is the relative path of the file to keep the results of the training and the relative path to hold to list of found good pages.*

In the model module is the PageModel class. PageModel encapsulates a wiki Page and is the data structure for a page in this project. PageModel has a function to return a page data that is understandable and easily calculable with for the analyser train function. It also has the property, which will return the label to which the Page belongs to. For training the label is preset, but for finding, the label is predicted.

3.3.1 Machine learning

The analyser module is the code specific to the machine learning. It has three functions meant to be used outside the module. The `train` function will take a list of `PageModel`s and save the result in a training result file. The `predict` function will take a `PageModel` and return whether the `PageModel`'s label is `GOOD` or `AVERAGE`. This project implemented logistic regression for prediction and training.

The learning data consists of a set of pages that were hand picked by the Vikipeedia team that were considered good pages. The good pages in our set get the label `GOOD`. Then the program asks for the same amount of random pages which will get the label `AVERAGE`. Most likely the `AVERAGE` set will have both really good articles and really bad articles besides average articles. Over large numbers, however, it will average out and produce good results. Those two sets are added together and shuffled. Then 70% of it will be used to train the program and 30% will be used to test and let the user know the precision of the bot.

Each page has 7 features: the text of the page, the pages that refer this page, the pages that this page links to, the images this page links to, the external links this page has, the templates this page links to and the categories this page is in. To keep the model simple, the algorithm uses only the count of each feature. These 7 features with a prefix of the number 1 make up the vector x.

To keep the number of iterations small, The values of x are normalised before they are trained or predicted with. It helps keep all the values of x around the same size and around the 0 value. The bias prefix 1 is added after the normalisation so it wouldn't be normalised. This all happens in the `prepare_x` function.

The result y is the numerical value of the label of the page. If it was a good page, y = 1. If the page was average, then y = -1. The program calculates the probability that y = 1 with the given x vector.

For the function implementing the gradient descent, I am using the source's example function with slight modifications:

```
def gradient_descent(z, y, w_h=None, eta=1.0, max_iterations=10000, epsilon=0.00
    if w_h == None:
        w_h = np.array([0.0 for i in range(z.shape[1])])
```

```
# save a history of the weight vectors into an array
w_h_i = [np.copy(w_h)]

for i in range(max_iterations):
    subset_indices = range(z.shape[0])
    # subset_indices = np.random.permutation(z.shape[0])[:N/8] # uncomment f

    grad_E_in = np.mean(np.tile(- y[subset_indices] /
                                    ( 1.0 + np.exp(y[subset_indices] * w_h.d
                                (z.shape[1], 1)).T *
                        z[subset_indices], axis=0)

    w_h -= eta * grad_E_in
    w_h_i.append(np.copy(w_h))

    if np.linalg.norm(grad_E_in) <= np.linalg.norm(w_h) * epsilon:
        break

return np.array(w_h_i)
```

3.3.2 Searching

The bot finds the good pages using a brute force mechanism. It requests all the pages from wikipedia and then tries to predict whether the page is good or average. It skips all pages with exceptions, mostly that would be redirects.

Kokkuvõte

[Kirjelda lühidalt töö põhieesmärke] [Kirjelda olulisemaid tulemusi] [Too välja olulisemad järeldused (2(3)] [Esita võimalikke edasiarendusi (vajadusel)]

Summary

[Selgita ülevaatlikult ja põhjalikult oma töö eesmärke] [Selgita ülevaatlikult ja piisava põhjalikkusega oma töös käsitletud probleeme] [Kirjelda ülevaatlikult ja põhjalikult oma töö tulemusi ja järeldusi]

References

[1] TIOBE Software BV. TIOBE software: Tiobe index. `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`, March 2014. [Online; accessed 10-April-2014].

[2] Necrolis. compiler — why does JIT'ed code consume so much more memory than either compiled or interpreted code? — stack overflow. `http://stackoverflow.com/a/8675550`, December 2011. [Online; accessed 10-April-2014].

Lisa 1