MSc thesis

Master's Programme in Computer Science

A Method for Wavelet-Based Time Series Analysis of Historical Newspapers

Jari Avikainen

December 3, 2019

Faculty of Science

University of Helsinki

Supervisor(s)

Prof. Hannu Toivonen

Examiner(s)

Prof. Hannu Toivonen, Dr. Lidia Pivovarova

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki,Finland

Email address: info@cs.helsinki.fi
URL: http://www.cs.helsinki.fi/

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Koulutusohjelma — Utbildningsprogram — Study programme Master's Programme in Computer Science

Tekijä — Författare — Author

Jari Avikainen

Työn nimi — Arbetets titel — Title

A Method for Wavelet-Based Time Series Analysis of Historical Newspapers

Ohjaajat — Handledare — Supervisors

Prof. Hannu Toivonen

Työn laji — Arbetets art — Level MSc thesis	Aika — Datum — Month and year December 3, 2019	Sivumäärä — Sidoantal — Number of pages 43 pages

Tiivistelmä — Referat — Abstract

This thesis presents a wavelet-based method for detecting moments of fast change in the textual contents of historical newspapers. The method works by generating time series of the relative frequencies of different words in the newspaper contents over time, and calculating their wavelet transforms. Wavelet transform is essentially a group of transformations describing the changes happening in the original time series at different time scales, and can therefore be used to pinpoint moments of fast change in the data. The produced wavelet transforms are then used to detect fast changes in word frequencies by examining products of multiple scales of the transform.

The properties of the wavelet transform and the related *multi-scale product* are evaluated in relation to detecting various kinds of steps and spikes in different noise environments. The suitability of the method for analysing historical newspaper archives is examined using an example corpus consisting of 487 issues of *Uusi Suometar* from 1869–1918 and 250 issues of *Wiipuri* from 1893–1918. Two problematic features in the newspaper data, noise caused by OCR (optical character recognition) errors and uneven temporal distribution of the data, are identified and their effects on the results of the presented method are evaluated using synthetic data. Finally, the method is tested using the example corpus, and the results are examined briefly.

The method is found to be adversely affected especially by the uneven temporal distribution of the newspaper data. Without additional processing, or improving the quality of the examined data, a significant amount of the detected steps are due to the noise in the data. Various ways of alleviating the effect are proposed, among other suggested improvements on the system.

ACM Computing Classification System (CCS)
Information systems → Information retrieval → Retrieval tasks and goals → **Information extraction**
General and reference → Cross-computing tools and techniques → Evaluation

Avainsanat — Nyckelord — Keywords

historical newspapers, wavelet transform, step detection

Säilytyspaikka — Förvaringsställe — Where deposited

Helsinki University Library

Muita tietoja — övriga uppgifter — Additional information

Algorithms study track

Contents

1 Introduction

The aim of this thesis is to examine the applicability of a wavelet transform-based method for change detection in time series generated from historical newspaper data. The change detection method examined in the thesis was developed as a part of NewsEye[†], an EU-funded[‡] project that aims to provide improved tools and methods for performing historical research using newspaper archives as the source material.

Section 1.1 of the thesis provides background for by discussing the use of historical newspapers as a source of information in historical research. Sections 1.2 and 1.3 discuss earlier research related to the topic, and Section 1.4 presents an overview of the remaining sections of the thesis.

1.1 Background

During the last 15–20 years, *digital humanities* has emerged as a new and growing field that combines computational tools and techniques to humanities research [22]. It has even been described as "the next big thing" [8].

One of the goals of digital humanities research is to give humanities researchers the possibility to access and analyse the immense amount of data available in the world from their own computers. One way of progressing towards this goal is to digitise information that currently exists only in analog format, such as microfilm archives of old newspapers. Another equally important task is to provide the tools necessary for accessing the data, sifting through it, and performing various kinds of sophisticated analysis to the data.

Various projects to that end exist, such as the NewsEye project, that aims to introduce "new concepts, methods and tools for digital humanities by providing enhanced access to historical newspapers for a wide range of users.[15]" As can be seem from the quote, the main focus of NewsEye is research using historical newspapers as a source. This is because newspapers, due to their very purpose of distributing information, contain unparalleled amount of information on cultural, political and social events.

[†]newseye.eu

[‡]This work has been supported by the European Unions Horizon 2020 research and innovation programme under grant 770299 (NewsEye)

In order to achieve its goals, the NewsEye project needs to provide improved solutions for a variety of problems, from improving the quality of the digitised versions of the newspapers to providing highly automated tools for exploratory analysis (see e.g. [25]) of the data. The list of tasks that need to be at least partially automated in order to reach the project's goal includes analysis problems such as named entity recognition, topic modelling and event detection.

The focus of this work is event detection. In order to limit the scope of the problem, *events* are here defined not as things that happen in the physical world, but simply as *sudden changes in the newspaper contents within short time intervals*. The assumption naturally is that these changes in the data are caused by things happening in the world, but the aim is simply to pinpoint the events in the data for further analysis by other means.

1.2 Event detection in media

Previous research on automated event detection in historical newspapers is difficult to find. The most relevant discovered paper is one by Van Cahn, Markert and Nejdl [24], where they investigate the evolution of a flu epidemic using German newspapers. In their work they use human annotators to build a corpus consisting of articles relevant to their research question. From the resulting corpus they search for spikes in word frequencies for keywords such as *influenza*, *epidemic*, and *disease*. They gain additional information by examining co-occurrences of various keywords.

Widening the search to event detection in other types of written media we find that a lot of the research uses Twitter[*] posts as source data. Twitter is a *microblogging* service, where users communicate via short (maximum length is 280 characters) posts, that can contain references to other users (marked using the @-character before the user-specific username), and so-called *hashtags*, which are specially marked keywords (e.g. *#MeToo*, related to the Me Too-movement) that can be used to search for other related posts more easily.

In [1], Atefeh and Khreich provide a survey of techniques for event detection in Twitter. Unfortunately, the methods used for event detection in Twitter are not easily adaptable for historical newspaper data due to few key differences in the properties of the two types of media.

[*] www.twitter.com

An obvious one is the lack of structure in the digital versions of old newspapers. Twitter posts or Tweets have a clear structure, with one post consisting of limited amount of text and easily distinguishable keywords in the form of hash tags and usernames. In addition, due to the length limit for the posts, each posts typically pertains to a single topic.

The digitised newspapers, on the other hand, are much more limited in regards to this sort of structure. At the most basic form, the newspaper data consists of machine-readable text produced by a more or less advanced optical character recognition (OCR) or automated text recognition (ATR) method. If article separation hasn't been performed to the data, a single text file contains a single page of the newspaper. This makes methods used with Twitter data, such as topic-based clustering (see e.g. [17], [18]) less useful, since a single page might contain articles concerning any number of topics, and a single article might be divided between two or more pages. The newspaper data is usually also missing easily recognisable keywords similar to the hashtags, unless additional analysis has been performed on the data to produce these kinds of labels.

1.3 Step detection using wavelet analysis

Due to the lack of structure in the newspaper data, the focus is shifted to analysing time series. Time series can be produced from the digitised newspapers e.g. based on word frequencies, similar to the work by Van Cahn et al. [24]. These time series can then be analysed in order to find spikes, steps, and trends in the data.

Mallat and Zhong [14] use *wavelet transforms* for characterising different types of steps. They show that wavelet transform can be used to detect and classify different variation points in signals, and describe a fast algorithm (referred from here on as the *fast wavelet transform* or *FWT*) for computing a discrete wavelet transform [14]. They also demonstrate that the wavelet transform can be used for encoding and reconstructing images.

Rosenfeld [19] suggests a way to improve step detection performed using differences of averages, i.e. by calculating

$$d_k(i) = \left| \frac{f(i+k) + \ldots + f(i+1)}{k} - \frac{f(i) + \ldots + f(i-k+1)}{k} \right|,$$

where a high value for $d_k(i)$ indicates the presence of a step within k samples from i. Rosenfeld suggests that instead of using a single value for k, better results can be achieved by calculating the differences $d_k(i)$ using multiple values for k, and examining the point-wise product of the differences.

Sadler and Swami [20] propose combining wavelet transforms with the idea suggested by Rosenfeld. They use the fast wavelet transform algorithm by Mallat and Zhong [14] to calculate the wavelet transform of a signal in multiple scales (corresponding to the difference signals $d_k(i)$ in Rosenfeld's work), and then calculate point-wise product of the different scales of the transform for performing edge detection. They call this product a *multiscale product* and proceed to analyse its properties and effectiveness in step detection.

1.4 Thesis structure

This thesis describes an experiment at using the approach proposed by Sadler and Swami for detecting sudden changes in the contents of historical newspapers. First, Section 2 gives a short overview of wavelet transform in general and the *fast wavelet transform* algorithm by Mallat and Zhong [14] in more detail.

Section 3 describes the method used for detecting changes in the newspaper contents. First, *time series* are produced from the original data using existing tools, such as the *NLTK* [9] and *UralicNLP* [6] libraries for Python. The produced time series are then processed using the Mallat and Zhong's *fast wavelet transform* [14], and *multiscale products* of the transformed time series are used for step detection similar to the analysis by Sadler and Swami [20]. Finally the detected steps are used to detect points in time with large number of changes in the general contents of the newspapers with the assumption that these changes are caused by some events of interest.

Section 4 describes both the synthetic data and the newspaper corpus used for testing the method, and the tests that were performed in order to evaluate the effectiveness of the method. The tests are devised in order to provide some insight into the following research questions:

1. How well does the step detection used in the method perform in the presence of noise?
2. (a) How do the features of the corpus, such as high amount of noise or missing data for certain months, affect the performance of the method as a whole?
   (b) Can the effects of these features be alleviated?

The results of the tests are presented in Section 5. Section 6 contains discussion of the results, together with answers to the questions posed above. In addition, some additional properties of the method, and their effects on the obtained results are discussed, together with some suggestions on how the method could be improved further.

2 Wavelet analysis

2.1 What is a wavelet transform

The wavelet transform is a tool for time-frequency analysis., in other words figuring out what kind of frequencies a given signal contains at different times. It was first formalized by Grossmann and Morlet [5], and has since been used in various signal processing applications, such as ECG analysis [3, 21], image compression (e.g. [14, 10, 2, 23]) and anomaly detection (e.g. [11, 16, 12, 7]).

Figures 2.1 and 2.2 demonstrate how different features of the original signal appear in the wavelet transform. Sharp edges appear as spikes that have the same height and approximate location over the first four scales of the transform, after which they combine with the neighbouring features. Slower changes in the signal appear only in the higher scales of the transform. The noise added to the latter part of the signal is fairly prominent in the first scale of the transform, after which the number of maxima in the transform caused by the noise rapidly decreases.

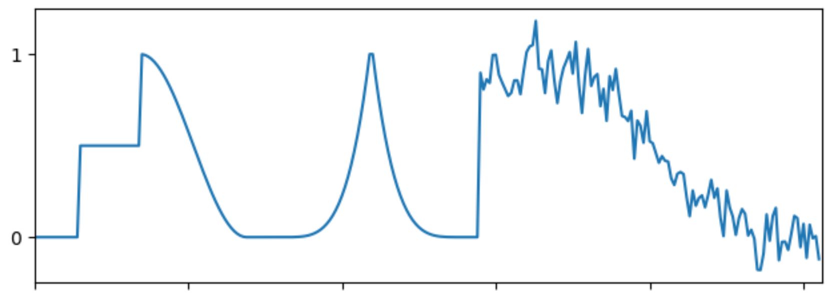

Figure 2.1: An example signal similar to one used by Mallat and Zhong [14] for demonstrating the features of the wavelet transform

2.2 Continuous wavelet transform

Wavelet analysis is a way of performing time-frequency analysis with some resemblance to the better known Fourier transform. The continuous wavelet transform can be described

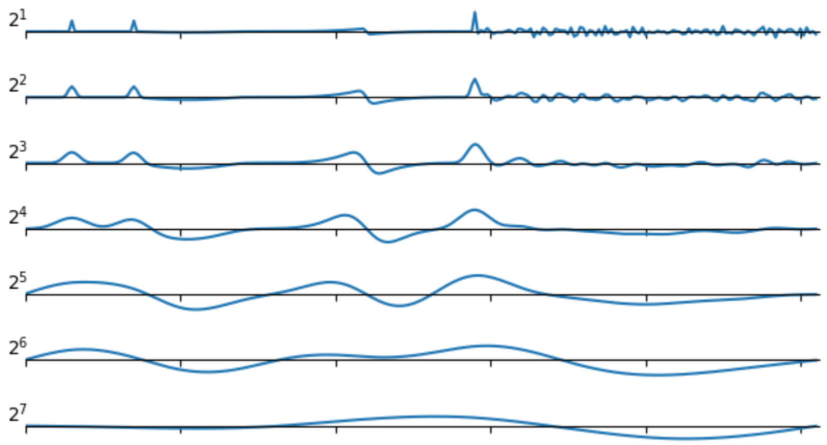

Figure 2.2: The first 7 scales of the wavelet transform of the signal in Figure 2.1

by formula

$$T^{wav}_{a,b}(f) = |a|^{-1/2} \int f(t)\varphi\left(\frac{t-b}{a}\right)\,dt,$$

where φ is called a wavelet function and must satisfy $\int \varphi(t)\,dt = 0$. This formula gives the value of the wavelet transform of function $f(t)$ at time b using scale a.

The a and b in the formula are the scaling and translation parameters. Large values of a correspond to low frequency components of the signal, and smaller values correspond to high frequencies. At the same time, small values for a make the wavelet function more narrow, improving the time localisation for high-frequency components. Translation parameter b simply shifts the center of the wavelet along the time axis [4].

To illustrate the effect of parameters a and b, their effect on an often-used wavelet function called the Mexican hat wavelet ($\varphi(x) = (1-x^2)\exp(\frac{-x^2}{2})$) is shown in Figure 2.3. The figure contains three versions of function $\varphi(\frac{t-b}{a})$ with different values for parameters a and b. The solid line shows the base function with values $a = 1$ and $b = 0$. The dashed line shows how increasing b shifts the function to the right, and at the same time how decreasing a makes the function narrower. The dotted version shows the opposite version: negative values for b shift the function to the left, and values larger than 1 for a make the function wider. Note that these effects on the shape of the function are separate from each other, i.e. changing only b has no effect on the width of the function, and changing only a doesn't affect the location of the function.

The continuous wavelet transform is similar to the Short-time Fourier transform (STFT),

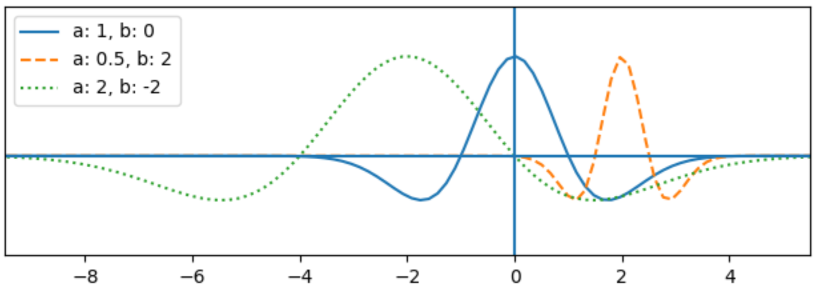

Figure 2.3: The effect of scaling and translation parameters a and b on the shape of the "Mexican hat" wavelet function [4]

which can be formulated as follows:

$$T^{STF}_{\omega,\tau}(f) = \int f(t)g(t-\tau)e^{-i\omega t}\,dt.$$

Here, function $g(t)$ is a window function centered on zero. In other words, the value of $g(t)$ is nonzero only in a limited area around the origin ($t = 0$). This means that for a fixed value of τ, the frequency information obtained from the STFT pertains only to the part of signal where the window function $g(t-\tau) > 0$.

A notable difference between the Fourier and wavelet transforms is in the width of the analysis windows. In Fourier transform, the width of $g(t-\tau)$ is fixed and therefore the time and frequency resolution are the same for all frequencies in the transform. In wavelet transform, decreasing a makes the wavelet function $\varphi(\frac{t-b}{a})$ narrower, which means that for higher frequencies (corresponding to the narrower wavelet function) the time resolution of the analysis is improved at the cost of poorer frequency resolution [4].

2.3 Discrete wavelet transform

In order to be able to work on digital signals, a discrete version of the wavelet transform is needed. To obtain the discrete form of the wavelet transform, both a and b need to be take only discrete values. The scaling parameter a is replaced with a_0^m, where $a_0 > 1$, and larger values for m correspond to wider wavelet functions, and improved frequency resolution [4].

Since the width of the wavelet is proportional to a and therefore also to a_0^m, the discrete version of the translation parameter b needs to depend on a_0^m as well: for narrower wavelet, the shifts along the time axis need to be smaller in order to cover the whole signal.

Accordingly, the discrete version of b becomes $nb_0a_0^m$, where $b_0 > 0$ sets the width of the time increments in relation to the width of the wavelet and n moves the wavelet along the time axis. Combining the discrete versions of the parameters, we get the discrete wavelet transform

$$T_{m,n}^{wav}(f) = a_0^{-m/2} \int f(t)\varphi(a_0^{-m}t - nb_0)\, dt.$$

The effects of changing m and n on the shape and location of $\varphi(a_0^{-m}t - nb_0)$ is demonstrated in Figure 2.4 using the Mexican hat wavelet from Figure 2.3.

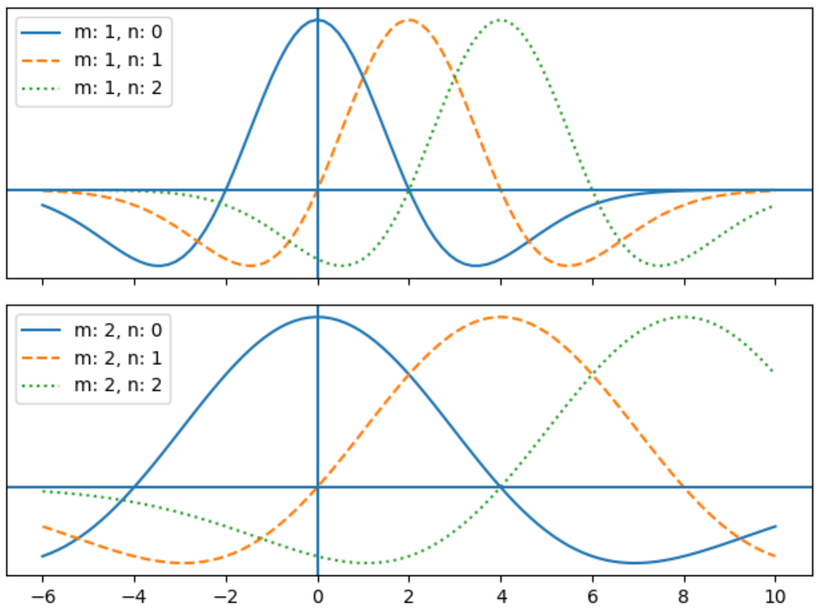

Figure 2.4: The discrete version of the wavelet in Figure 2.3 with $a_0 = 2$ and $b_0 = 1$.

2.4 Fast Wavelet Transform

Mallat and Zhong [14] present the Fast Wavelet Transform (FWT), a fast algorithm for calculating a discrete wavelet transform. They use a quadratic spline wavelet shown in Figure 2.5, with scaling parameters that belong to the dyadic sequence $(2^j)_{j\in\mathbb{Z}}$. This corresponds to setting $a_0 = 2$ in the discrete wavelet transform discussed in Section 2.3.

The following describes the FWT algorithm by Mallat and Zhong [14] using slightly simplified notation. The input of the algorithm is the signal to be transformed, denoted as S_1. For each scale 2^j, the algorithm applies a high-pass filter G_j and a low-pass filter H_j separately to the input signal S_{2^j} (the $*$ symbol denotes convolution). The output W_{2^j} from the high-pass filter is the wavelet transform of the input at scale 2^j and is part of

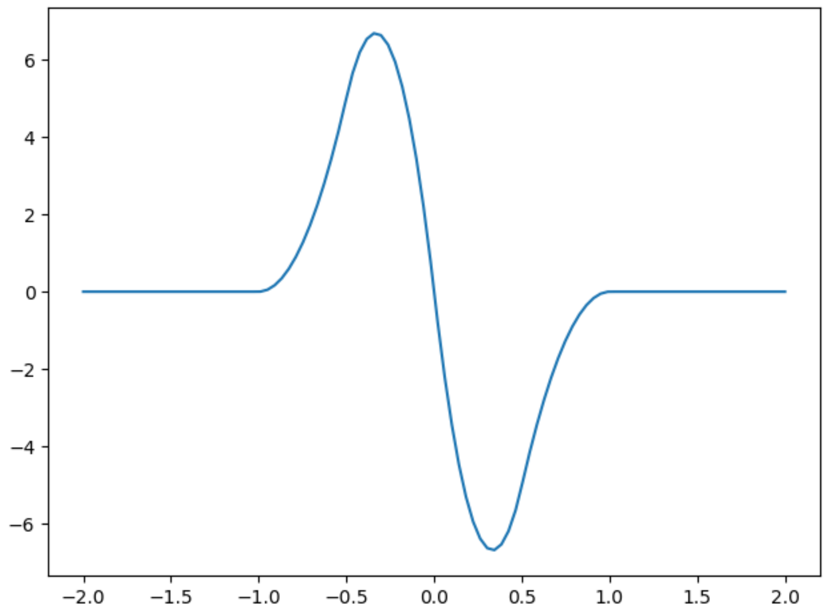

Figure 2.5: Quadratic spline wavelet used by Mallat and Zhong [14]

the resulting wavelet transform. The scaling factors λ_j are used to counteract a distortion introduced by the discretisation of the wavelet transform, and are explained in more detail below. The output $S_{2^{j+1}}$ from the low-pass filter contains the remaining lower frequency components of the signal that are not included in W_{2^j}, and becomes the source signal for the next scale.

The output of the algorithm is formed of signals $W_{2^j}, 0 \leq j < J$, which are the transformed versions of the input signal in different scales, and signal S_{2^J}, which contains the remaining low-frequency components of the original signal.

```
$j = 0$
while $(j < J)$
    $W_{2^j} = \frac{1}{\lambda_j} S_{2^j} * G_j$
    $S_{2^{j+1}} = S_{2^j} * H_j$
    $j = j + 1$
end while
```

The high-pass and low-pass filters G_0 and H_0 corresponding to the wavelet in Figure 2.5 are given in Table 2.1, together with filters G_1, G_2, H_1, and H_2, which correspond to higher scales of the transform. Filters for higher scales are obtained by adding $2^j - 1$ zeroes between each value in filters G_0 and H_0.

The outputs W_2^j are scaled by λ_j to compensate for an error caused by the discretisation.

n	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
H_0						0.125	0.375	0.375	0.125				
H_1				0.125	0	0.375	0	0.375	0	0.125			
H_2	0.125	0	0	0	0.375	0	0	0	0.375	0	0	0	0.125
G_0							-2.0	2.0					
G_1						-2.0	0	2.0					
G_2					-2.0	0	0	0	2.0				

Table 2.1: Filters H and G for the first three scales of the fast wavelet transform algorithm by Mallat and Zhong [14]

If the signal contains an instant step*, the wavelet transform should contain a maxima of equal amplitude in all scales. In their algorithm, Mallat and Zhong use the λ_j values in Table 2.2 to ensure that amplitudes across scales are correct.

j	0	1	2	3	> 3
λ_j	1.50	1.12	1.03	1.01	1

Table 2.2: Values for λ_j as given in Mallat and Zhong [14]

When calculating values near the beginning and end of the signal, some of the values needed for the convolution would be outside the actual signal. This can cause artefacts in the results, and these *border effects* need to be addressed somehow.

If the signal has non-zero values at the ends, simply assuming that the values outside the signal are all zero effectively creates a sudden step to the beginning of the signal. This imaginary step will then show up in the wavelet analysis, and avoiding such artefacts is desirable.

In order to avoid such border effects, Mallet and Zhong suggest using a periodisation technique. The signal with length N is basically assumed to be a part of longer, repeating signal, that has a period of $2N$ samples, and the signal is extended to that length using mirror symmetry: if the actual signal is $d_{1\leq n\leq N}$, then for $N < n \leq 2N : d_n = d_{2N+1-n}$. As a result, there will be no discontinuities at the ends of the signal.

The assumption used for periodisation also limits the number of scales needed for full analysis to $\log_2(N)+1$, since when the scale is equal to the period of the signal ($2^j = 2N$),

*for instance a signal consisting of values `[1, 1, 1, 1, 1, 5, 5, 5, 5, 5, 5]` as opposed to `[1, 1, 1, 1, 2, 3, 4, 5, 5, 5, 5]` the remainder signal S_{2^j} is constant and equal to the mean value of the original signal[14]. This gives us the time complexity of the algorithm. One iteration (calculating one scale of the transform) takes $O(N)$ time, and since we only need to calculate $O(\log(N))$ scales, the total time complexity of the algorithm is $O(N \log(N))$.

3 A method for event detection using wavelet transforms

The method for event detection consists of five distinct phases, each of which is described in the following subsections. Section 3.1 describes the process used to produce time series out of the raw data in the corpus. Next, the time series are transformed using the *fast wavelet transform* algorithm by Mallat and Zhong [14] (Section 3.2). Then, *multiscale product* used by Sadler and Swami [20] is calculated for the transformed time series (Section 3.3). Steps are detected from the multiscale product using *adaptive thresholding*, and the detected steps are given weighted scores (Section 3.5). Finally, steps from different time series are analysed together to find large concentrations of changes in the data (Section 3.6).

3.1 Time-series generation

Processing the data from the raw text into token-specific time series is performed as follows. First, the texts are tokenised, and tokens containing non-alphabetical characters are removed. The remaining tokens are lemmatised using UralicNLP [6] library by Mika Hämäläinen. Tokens that cannot be lemmatised by UralicNLP are discarded, which removes most of the remaining OCR-related problems, such as tokens that are segments of words (e.g. *tehn*, *toia*, *sseja* and so on).

The number of times each lemma appears on any given day in the data is then stored in the form of lemma-specific time series. In addition, for each day, the total number of tokens before lemmatisation (i.e. the number of tokens consisting solely of letters) is stored. This number is used for calculating frequencies for the lemmas, which are stored into separate time series in the form of IPM values, i.e. the number of times the lemma appears per million tokens in the corpus. Next, lemmas appearing in the whole corpus less than ten times are discarded as well, since they are too rare to provide any reliable insights into the changes in the data. In addition, time series for stopwords are retained, but stored separately from other data.

3.2 Wavelet transform

The first phase is to perform a wavelet transform to each of the time series to be examined in order to detect possible spikes and steps in them. The generic description of the fast wavelet transform used in the method is given in Section 2.4. Sadler and Swami [20] present the Matlab code for performing the transform, and this code was used as the basis for the implementation.

In their implementation, Sadler and Swami invert the high pass filter G in Table 2.1, and the same inverted version is also used in this work. The inverted filter produces more naturally interpretable responses (positive peak for a positive step and vice versa), and therefore seems like a sensible choice.

The ends of the signal are treated using mirror-symmetric extension of the signal, as suggested by Mallat & Zhong [14]. This means that at each scale $2^j, 0 \leq j < \log_2(2N)$ of the transform, the remainder signal $S^d_{2^j}$ is extended from both ends with its mirror image, producing a signal of length $3N$. The purpose of the signal extension is to avoid introducing discontinuities into the signal, as these would then appear in the event detection as extra detected steps. The convolutions of the signal with the low- and high-pass filters are calculated using this extended version, and the results are then trimmed back to the original length, by keeping the N samples from the middle.

This method results in a number of effects to the resulting wavelet transform, two of which will be discussed in more detail, since they need to be taken into account in further analysis of the signal. First of all, the method will inevitably produce some artefacts to the wavelet transform close to the beginning and end of the signal. This will be discussed more in Section 3.4. Second, the transformed signal is shifted half a sample to the right (or left, depending on the implementation).

Since the high-pass and low-pass filters H_0 and G_0 used by Mallat and Zhong [14] are of even length, the results from the first convolutions cannot be trimmed exactly from the middle. Instead, the filtered signal is shifted half a sample to left or right, depending on which end of the signal is trimmed more.

Figure 3.1 demonstrates this effect on a signal consisting of four samples with a unit step at 2. The maximum of the transformed signal should coincide with the steepest slope in the original signal (between samples 1 and 2) but is instead shifted half a sample to the left or right. Shifting the signal to the right makes more sense since that way the peaks in the transform coincide with the first sample after a step instead of seeming occur before the actual step.

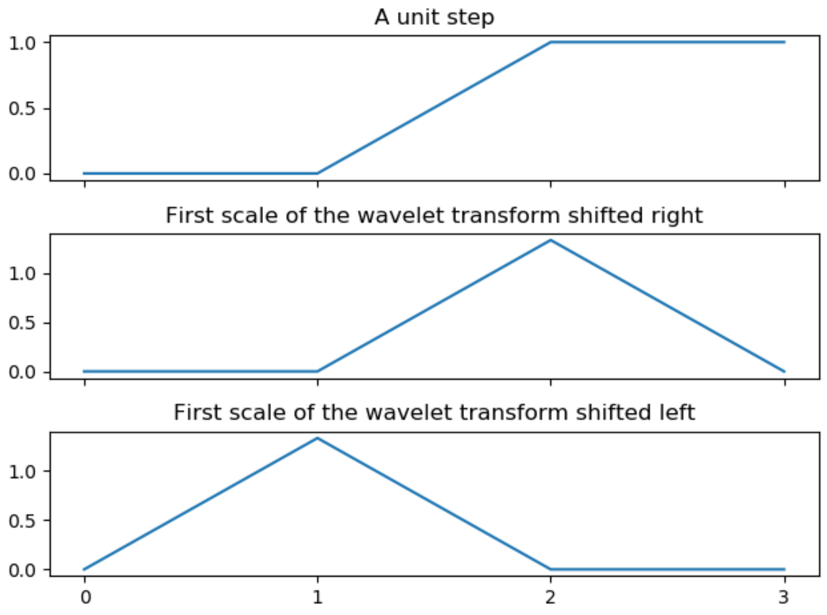

Figure 3.1: The first scale of the wavelet transform for a unit step

As an example of a (partial) wavelet transform using real data, Figure 3.2 shows a time series (3.2.a.) and the first three scales of the wavelet transform for the time series (3.2.b.–d.). The time series describes how well the corpus covers each month in the data. The value is the percentage of days for which there are newspapers issues in the corpus. As we can see, the distribution of the data is quite uneven, which has certain effects on the analysis. This will be discussed more in Section 4.

From Figure 3.2.b.–d. we can see how the wavelet transform behaves in the first three scales. In the smallest scale (Figure 3.2.b.) it produces the most narrow extrema, but the high-frequency noise in the data is also the most prominent. Especially the minima related to the two decreasing steps in 1901 and 1915 are almost indistinguishable from the surrounding noise.

On the other hand, in the third scale in Figure 3.2.c much of the noise has already been filtered out, but at the cost of wider extrema that correspond to the steps in the data. For the spike visible in the data at 1876 this means that the locations of the extrema in the wavelet transform do not accurately correspond to the location of the spike anymore. Also, the step at 1915 is becoming more difficult to differentiate from the surrounding noise.

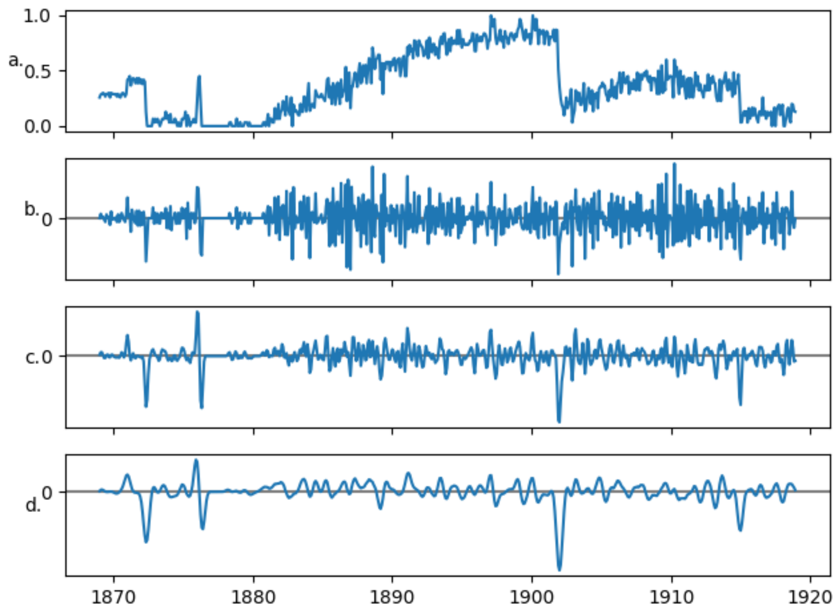

Figure 3.2: A time series of newspaper coverage per month (a.) and the first three scales of the wavelet transform of the time series (b.–d.).

3.3 Wavelet product

There are multiple ways of analysing and processing the transformed signal further. In this work, the chosen approach is to use *multiscale products*, i.e. point-wise products of multiple scales of the wavelet transform. Similar method has been used e.g. Xu et al. [26] in their method for filtering magnetic resonance images, and Sadler and Swami [20] for edge detection.

Using products of different scales is inspired by Rosenfeld [19], who uses a similar method for improving the edge detection accuracy when using differences of averages at different scales. Using the product of multiples scales is based on two observations. For one, sharp edges produce maxima across multiple scales in the wavelet transformation at approximately same location, and therefore the multiscale products will have large values at locations corresponding to the edges.

Second, the number of maxima caused by noise decreases quickly for increasing scales. According to Mallat and Hwang [13], the number of local maxima in the wavelet transform of white Gaussian noise is halved for every scale. In addition, the locations of the maxima caused by the noise are not correlated across scales. Since the value of the multiscale product is small when any of the factors are small, the multiscale product for noise tends to have small values for most of the time. The distribution for the multiscale product of Gaussian noise is examined briefly in Section 5.2.

The multiscale product used in this thesis is calculated using the first three scales of the wavelet transform. The decision to use the first three scales for the product similar to Sadler and Swami [20] is based on the following factors. First of all, smaller scales correspond to the fastest changes in the signal, and the further along the scales we progress, the slower are the changes that are detected by the wavelet transform. Since this work focuses on using steps for detecting events and not e.g. trends over larger timespans, the larger scales are not useful.

Using larger scales also increases the number artefacts appearing close to the start and end of the signal due to the way the fast wavelet transform algorithm described in Section 2.4 works. This distortion is discussed in more detail in Section 3.4.

As an example of the wavelet product, Figure 3.3.b. shows the wavelet product for the newspaper time series from Figure 3.2. It is easy to see that the resulting product indeed has minima at the locations where the original signal has the most prominent decreasing steps (i.e. around 1872, 1901, and 1915) and a maximum followed by a minimum around the location of the spike in original data at 1876.

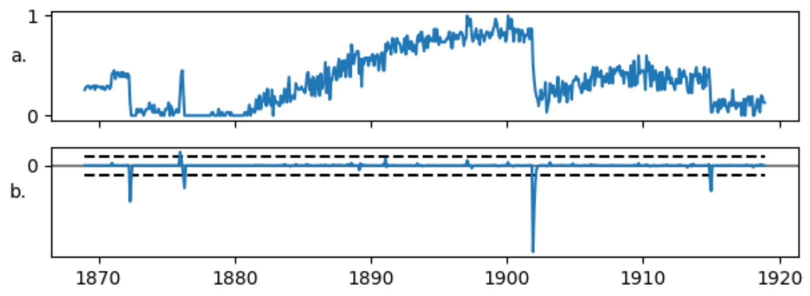

Figure 3.3: The time series from Figure 3.2 and the product of the first three scales of its wavelet transform. The dashed lines indicate the selected step thresholds, described below in Section 3.5.

At the same time, the signal to noise ratio of the product is much better than in any of the three individual scales (Figure 3.2.b., c. and d.) used for calculating the product. The small increasing step at early 1870s is barely visible due to its relatively small size. This is as expected, since the step in question has similar magnitude to the noise elsewhere in the data.

3.4 Problems near the ends of the signal

Figure 3.4 show the proportions by which the samples in the original signal affect the value of the transformed signal. For the first scale of the wavelet transform, the value of the transformed signal at time t is affected only by the values of the original signal at times $t-1$ and t.

On the other hand, for the third scale of the transform, 14 values from the original signal are needed, from $t-7$ up to $t+6$, with values at $t-1$ and t having the largest effect on the result. The solid line in the figure shows the mean of the three scales, corresponding to the effect that each of the samples in the original signal have on the final multiscale product.

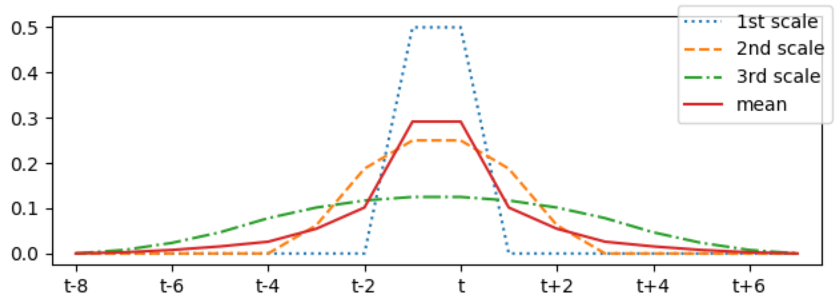

Figure 3.4: The amounts by which the surrounding samples in the original signal participate in the calculation of the first three scales of the wavelet transform. The solid line shows the mean of the values for the three scales.

As can be seen from the figure, when calculating the wavelet transform and the multiscale product closer than eight samples from the start of the signal or seven samples from the end, part of the samples needed for calculating the values for at least one of the scales are located outside the actual signal. Simply assuming those values to be zero (or any other fixed value) effectively assumes that the signal had the selected value before the measurements started. Now, if the signal actually has a (mean) value of something else than the assumed value at the beginning, this imaginary jump from zero to that real mean value will be visible in the resulting wavelet multiscale product.

As explained in Section 2.4, Mallat and Zhong propose using mirror-symmetric extension of the signal in order to minimise artefacts caused by the proximity of the signal borders to the wavelet transform [14]. This approach was also used by Sadler and Swami [20], and adapted to the version of the transform used in this thesis as well.

It's worth noting that even though the signal extension method does work in a way, the values for the wavelet transform and the multiscale product close to the start and end of the signal should still be taken with a grain of salt. Figure 3.5 shows the proportion of assumed data (i.e. values generated by the mirror extension) used for calculating each value in the multiscale product for a signal of 16 samples. For the first value (at index $t = 0$), the proportion of assumed data is actually more than 50 % due to the implementation details of the algorithm.

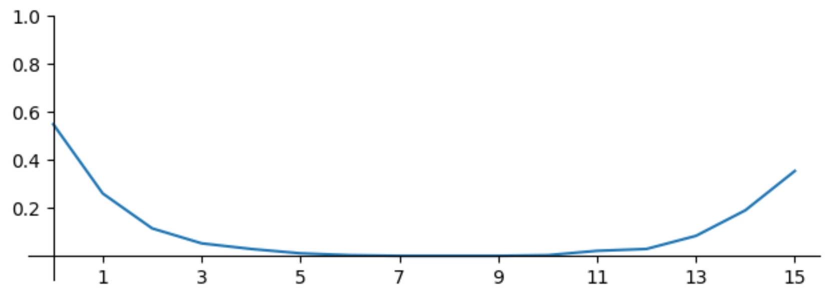

Figure 3.5: The proportion of assumed data used for calculating the wavelet product of a time series of 16 samples

As a demonstration of the artefacts caused by the proximity of the signal borders to the wavelet transform, Figure 3.6 shows the resulting wavelet products for signals that contain a single unit impulse at different locations within the signal. In Figures 3.6.a–c the minimum corresponding to the sample after the unit impulse (i.e. the decreasing step) become increasingly exaggerated when the impulse is located near to the start of the signal, while the maximum disappears completely. Figure 3.6.d shows the expected wavelet product with no interference from border effects. In Figures 3.6.e–h the maximum becomes exaggerated while the minimum disappears.

3.5 Step detection and evaluation

After obtaining the wavelet products as described above, the products are analysed in order to extract the step information. This process is fairly straightforward and consists of finding local extrema in the product and interpreting them as steps. When using an odd number of scales for the wavelet product, as is done here, the maxima in the product indicate increasing steps and the minima indicate decreasing steps. If instead an even number of scales is used, all of the extrema corresponding to steps will be positive, and

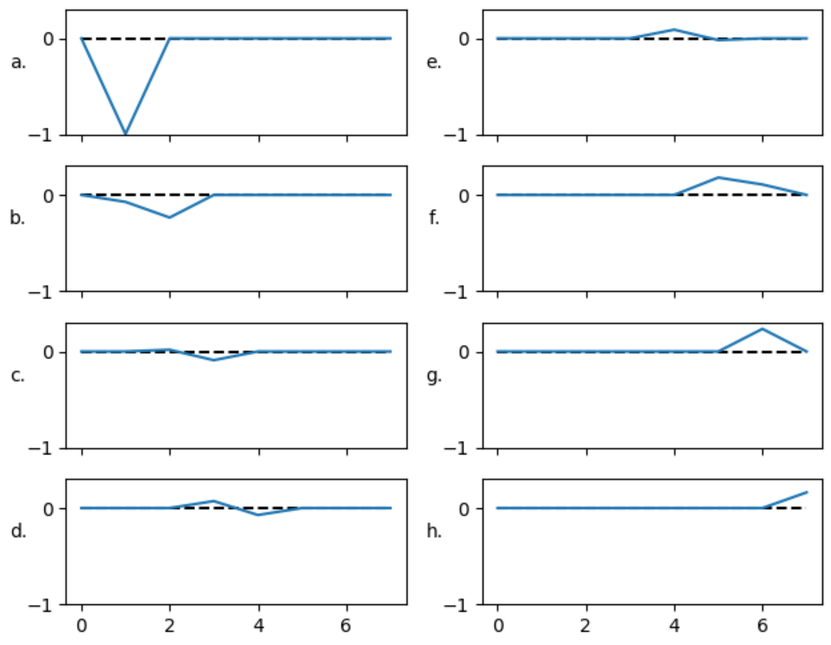

Figure 3.6: Wavelet products corresponding to unit impulses at time points 0–7

the direction of each step needs to be detected e.g. based on the highest scale used in the product.

The main problem is in deciding the threshold that the extrema need to exceed in order to indicate the existence of a step. In this work a relatively simple method is used: the threshold is set to two times the standard deviation of the wavelet product being analysed. This ensures that only the extrema that exceed the signal-specific noise level are reported.

An example of the automatic thresholding can be seen in Figure 3.3.b., where the dashed lines indicate the threshold selected by the step detection. For this example, the three decreasing steps at 1872, 1901, and 1915 exceed the threshold clearly, as does the falling edge for the spike at 1876. The rising edge of the same spike also crosses the threshold, although not so clearly, whereas the small increase at 1871 doesn't.

If the value of the product stays outside the threshold for more than one sample, the location with the highest absolute value for the product is chosen to represent the step. This means that the step is considered to happen at the point of the fastest change.

Figure 3.7 shows a closer look of the decreasing step in Dec 1901 from Figure 3.3. Similar to Figure 3.3, 3.7.a. shows the original time series and 3.7.b. shows the resulting wavelet product, with the dashed lines showing the location of the step threshold. Here the wavelet product is smaller than the negative threshold for three samples from Dec 1901 to Feb

1902. As can be seen, values further away from zero correspond to faster changes in the originating time series. The highest absolute value for the wavelet product occurs in Dec 1901, and accordingly, that month is recorded as the time when the step occurs.

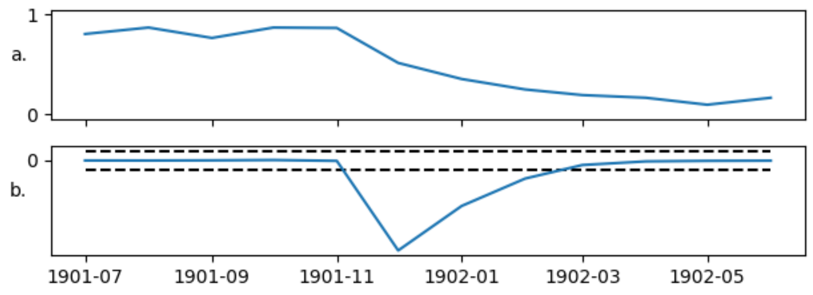

Figure 3.7: A closer look to the area around the beginning of 1902 in Figure 3.3

The location of each detected step is then stored, together with a base score which is set to the absolute value of the wavelet product at the step location divided by the threshold value. This score is always ≥ 1, since the wavelet product has to be larger than the threshold for all steps. The base scores for the steps in Figure 3.3 are listed in Table 3.1.

Step	May 1872	Jan 1876	May 1876	Dec 1901	Jan 1915
Score	3.86	1.44	2.42	9.27	2.73

Table 3.1: Base scores for the steps in Figure 3.3

The purpose of calculating the base score in this way is to make the steps more comparable to steps from other time series. The actual values of the wavelet product vary significantly due to the different scales in the time series, but the score calculated in this way gives a better idea on how distinctive the step is in its own time series.

3.6 Event detection

After obtaining a set of steps, the next step is finding *events* from the step information. The simplest way of doing is is by aggregating all of the steps detected per month, and using a sum of their scores as the total score for the event. The problem with this approach is that it treats all steps equally: ten steps with a score of one will result in the same event score as one step with a score of ten.

Groups of a few large steps are probably more interesting than times with a larger amount of really small steps. In order to account for this, the event detection used in this thesis uses only a set number of steps with the highest scores for calculating the event score. In this way, only the most significant steps for each time point affect its score, hopefully producing useful results.

4 Empirical evaluation

This section presents the data and experiments used for evaluating the usefulness of the method described in Section 3. The results of the experiments are then presented in Section 5.

Section 4.1 discusses the corpus data used in latter parts of evaluating the method, and Section 4.2 describes the experiments used for evaluating the method. Section 4.2.1 deals with the properties of the multiscale product used as part of the step detection and section 4.2.2 examines the effects of noise on step detection in a single time series. Finally, Sections 4.2.3 and 4.2.4 look into the two features found in the data that pose problems for event detection.

4.1 The corpus used for testing the method

The data used for testing the method consists of a collection of Finnish newspaper issues from years 1869–1918. The collection contains 5352 issues of Uusi Suometar (years 1869–1918), and 2064 issues of Wiipuri (years 1893–1918). The data in the collection contains OCR data divided by page, and various metadata including the publication date.

The quality of the OCR is fairly low, and the text contains a high amount of noise, misspellings, split or combined words etc. The newspaper issues are also spread quite unevenly within the time range, with approximately 60 percent of the issues being from the 15-year range 1887–1901 whereas issues from the first 15 years (1869–1883) form only 9 percent of the collection.

Figure 4.1 shows for each month the percentage of days for which at least one newspaper issue exists in the data. From the figure we can see that until about halfway through 1880s the number of newspaper issues per month stays fairly low, with especially small number of newspaper being included in the data between 1872–1882, except for a clear spike in early 1876. After 1882 the number of issues included in the data increases until a sharp drop in 1901, and another in 1915. The differences in the number of issues per month are mostly due to issues missing from the dataset, instead of reflecting the actual number of issues published.

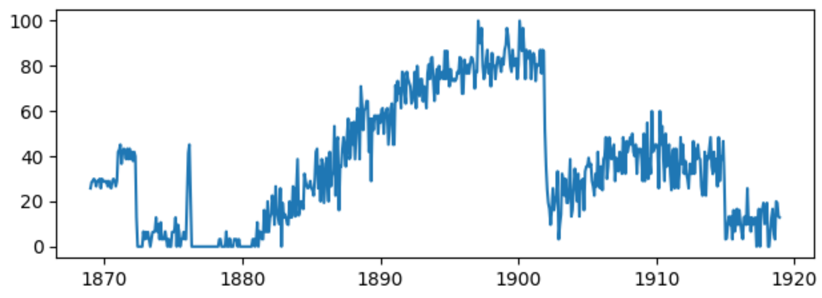

Figure 4.1: Percentage of days with newspaper data in each month

Using the method for generating time series described in Section 3.1, the lemmatisation phase produced 99967 lemmas from the corpus, and discarding lemmas appearing in the corpus less than ten times reduced the number to 48342.

Figure 4.2 is an example of a time series for a single token, in this case token *suomi*. Unsurprisingly, the top graph with total token counts has a similar overall shape as the graph in Figure 4.1 showing the per-month coverage of the corpus. As for the IPM (items per million) values (bottom panel), having less data seems to increase the variance in the IPM values, which again is as expected due to the lower number of available samples (in this case, newspaper issues) decreasing the reliability of the calculated mean values.

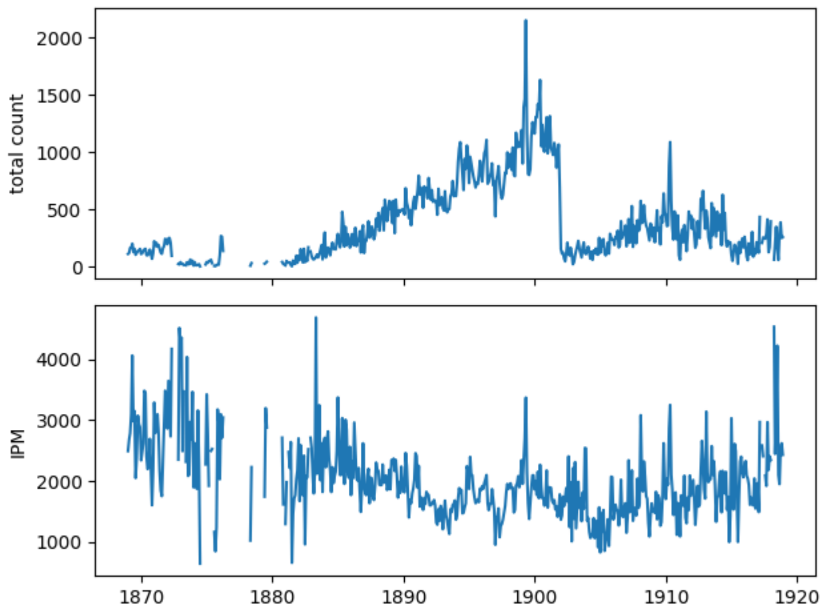

Figure 4.2: Time series of the token counts and IPM-values for the token *suomi*. The gaps indicate months for which there is no data in the corpus.

4.2 Testing the method

In order to evaluate the effectiveness of the method described in Section 3, various experiments on synthetic and real data were performed. The experiments are described below, and their results and some suggestions based on the results are presented in Section 5.

4.2.1 Properties of the multiscale product

The effectiveness of the step detection technique described in Section 3.5 was tested for various combinations of steps and signal to noise ratios. The purpose of these tests was to verify that the step detection itself performs as expected, and to detect possible error sources in this part of the method. Test time series were created with steps of varying height and width, and precision and recall of the step detection was measured while adding increasing amounts of noise to the time series. The results of these tests are described in Section 5.1

Figure 4.3.a. shows the time series used to test the effect of step width to the corresponding wavelet product. Each rising step is followed by a falling step of equal height after the amount of samples defined by the step width. The step width is increased from 1 sample to 16 samples, with the intervals between each pair of rising and falling steps set at 16 samples.

Figure 4.3.b. shows the time series used to demonstrate the effect of increasing step heights to the wavelet transform and the resulting multi-scale product. The spaces between the steps and the step width were set to 10 samples to prevent interference from the neighbouring steps and the step height is increased in intervals of one. The width is based on results from the previous test which agree with the analysis in Section 3.3 that shows that a sample in the wavelet product at time t is affected only by samples in the original time series between times $t-7$ and $t+6$ inclusive (see Figure 3.4).

4.2.2 The effects of noise on the step detection

First, a test signal consisting of 10000 samples of Gaussian noise (i.e. noise sampled from the Gaussian distribution) with variance $\sigma^2 = 1$ was generated. A wavelet transformation was performed to the signal and the wavelet product was calculated to see how the values of the resulting wavelet product are distributed. The first and last 10 samples of the

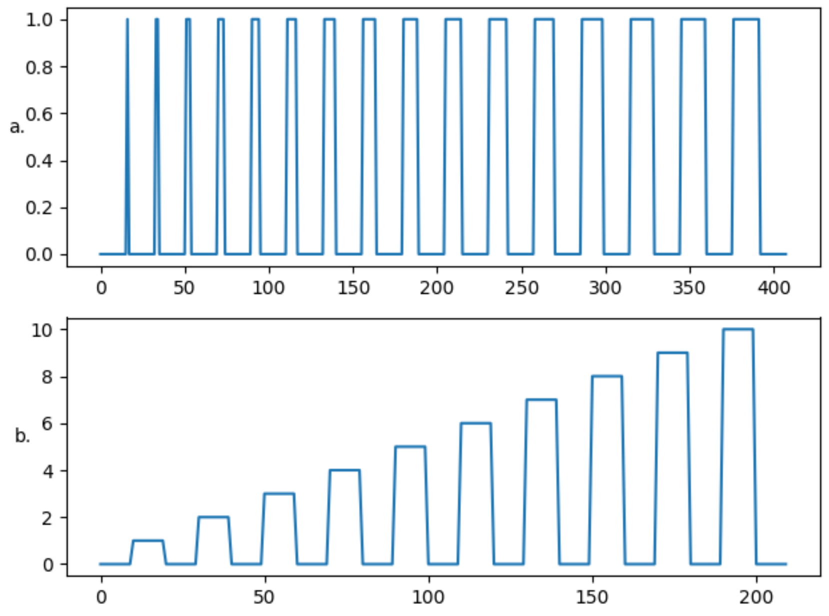

Figure 4.3: The signals used for testing the effects of step width (a.) and height (b.) to the wavelet product.

wavelet product were discarded to remove possible interference caused by the proximity of the start and end of the signal.

Following this, *precision* and *recall* values for different combinations of signal-to-noise ratio (SNR) and step width were estimated. *Precision* is defined as the percentage of real steps in all of the detected steps, and *recall* is the percentage of detected real steps of the total number of real steps.

Precision and recall values were estimated using a test signal with five steps of equal height (see Figure 4.4). SNR was defined as

$$SNR_{dB} = 10 \log_{10} \frac{A^2}{\sigma^2},$$

where A is the step height and σ^2 is the variance of the added noise.

Each trial consisted of the following steps. First the step height in the test signal was adjusted to produce the desired SNR for noise with variance $\sigma^2 = 1$, and the noise was added to the signal. Then step detection as described in Sections 3.2–3.5 was performed, and the resulting steps were compared to the known ground truth to obtain the precision and recall values.

For each tested combination of step width (between 1–10) and SNR (-10–20dB), 5000 trials were performed, which was more than enough to obtain stable results over multiple

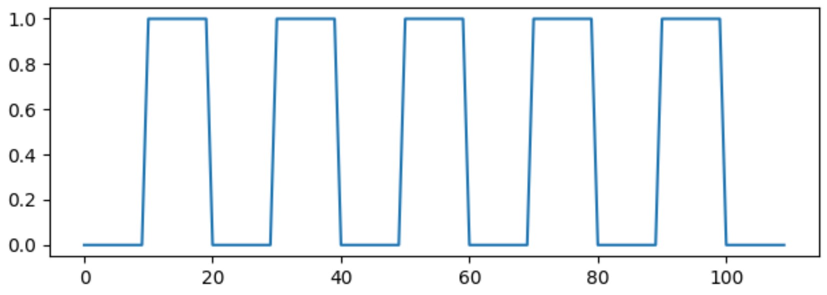

Figure 4.4: The shape of the signal used in estimating the precision and recall of the step detection using various combinations of step width and SNR

repetitions. The results are reported in Section 5.2.

4.2.3 The effect of data coverage on the step counts

In order to get an idea on how well the step detection works using the actual corpus, the total numbers of steps found from the corpus was compared to step counts obtained from synthetic data sets spanning the same time period as the real corpus, i.e. years 1869–1918.

The synthetic data sets were generated as follows. For each month in the included time span, n values of Gaussian noise were sampled, and the value for the month was set to the mean of those n samples. Here values for n represent the number of days in each month for which the imaginary corpus contains data.

The first synthetic time series was generated using the same values for n that appear in the real data. The purpose was to test whether the resulting step counts from Gaussian noise would resemble those obtained from the actual corpus, and in what ways. The average number of detected steps for each month was estimated over 10000 iterations of generating the data and detecting the step locations.

In addition, a second test was made using time series of steadily increasing n in order to visualise to effect of n to the number of detected steps. Again, 10000 iterations were performed, and the average number of detected steps for each month was calculated. The results of these tests are presented in Section 5.3.

4.2.4 The effect of OCR errors to the step counts

When considering the effects of OCR errors to the results obtained with the described method, it is useful to consider how the OCR errors affect the data. A simple way to model OCR errors is to simply replace random characters in the original text with different characters. In most cases this means that using the current system from reading the data from the corpus, most words with replaced characters are discarded from the data. Some changes of character simply change the word to a different valid one, but these are probably in the minority of all changes. If we have the same probability of change for each of the characters in a text, longer words will have increasingly high chance of being discarded due to one of the characters being changed.

For a single time series (corresponding to a single word), this discarding of words will simply decrease the word counts throughout the document, assuming that the OCR error rate is similar throughout the corpus. This decrease will decrease the signal-to-noise ratio for that word, with results similar to those of increasing the noise level, discussed in Section 5.2. If there is a sharp change in the OCR quality, for instance caused by a change in the typeface used in the newspaper, this should result in steps being detected for a large number of words at the same time point, assuming that the change in quality is significant enough.

Since the effects of OCR error should be effectively the same as examined with the earlier tests, no separate tests for analysing these effects were performed.

5 Results

5.1 Properties of the multiscale product

Figure 5.1 shows the effect of step width to the amplitude of spikes in the resulting wavelet product. The signal used for testing is depicted in Figure 5.1.a. and the resulting wavelet product is presented in Figure 5.1.b. The width of each of the steps has been marked below 5.1.b. for improved legibility.

As can be seen from the figure, the spike heights in the wavelet product increase with the step width until steps of width seven, after which the step width doesn't affect the magnitude of the spike. The lower heights for small step widths are cause by destructive interference between the neighbouring rising and falling steps. In the remaining experiments, ten samples is used as a sufficient distance between steps in order to avoid interference from neighbouring steps.

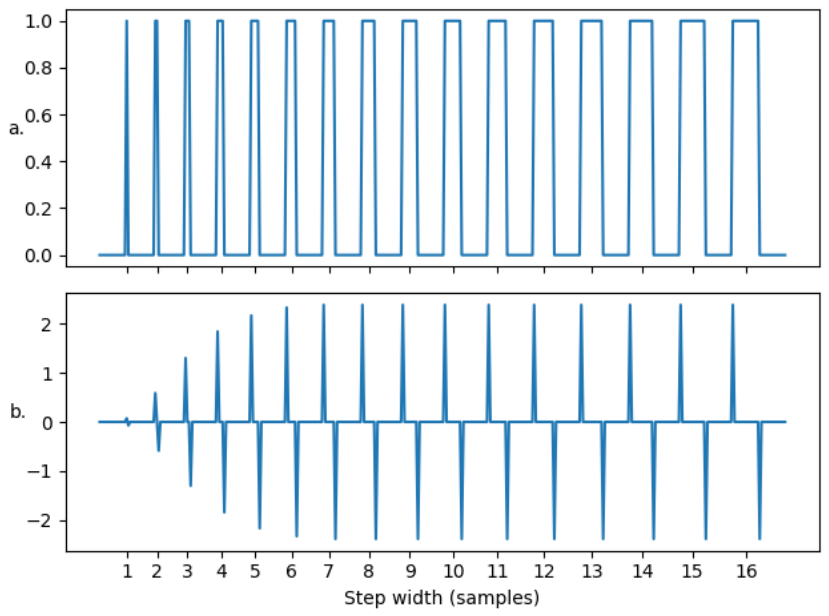

Figure 5.1: Growth of the wavelet product for increasing step width.

Figure 5.2 shows a sequence of steps with increasing heights (a.) and the corresponding wavelet product (b.). The widths of the steps and the intervals between them have been set to 10 samples based on the results of the previous test. This distance between separate steps is enough to prevent interference from neighbouring steps. The values of the wavelet product show cubic growth, which is expected: each scale of the wavelet transform should scale linearly in proportion to the original signal, and the wavelet product is the product of three scales of the transform, resulting in cubic growth in relation to the step height, other parameters being constant.

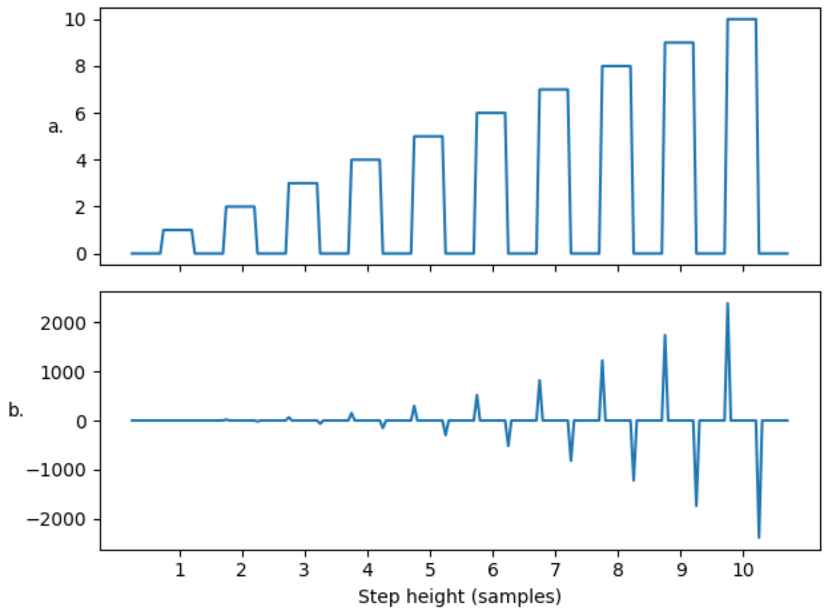

Figure 5.2: Growth of the wavelet product for increasing step height.

5.2 The effects of noise on the step detection

Figure 5.3 shows the distribution of values in the wavelet product for signal consisting of Gaussian noise with zero mean and standard deviation $\sigma = 1$. The figure is essentially a reproduction of a similar figure presented by Sadler and Swami [20]. Standard deviation of the wavelet product is approximately 3.2, and the black lines in the image show the values of 2σ used as the threshold for step detection. Approximately 4.4% of the samples in the wavelet product fall outside of the thresholds resulting in false positive steps. A normal distribution with the same standard deviation of 3.2 is also drawn to the image for comparison.

Figure 5.4 shows the effect of step width on the precision of step detection for different signal-to-noise ratios. For most of the step widths, the precision levels are pretty close regardless of SNR, with higher step widths obtaining slightly better results throughout.

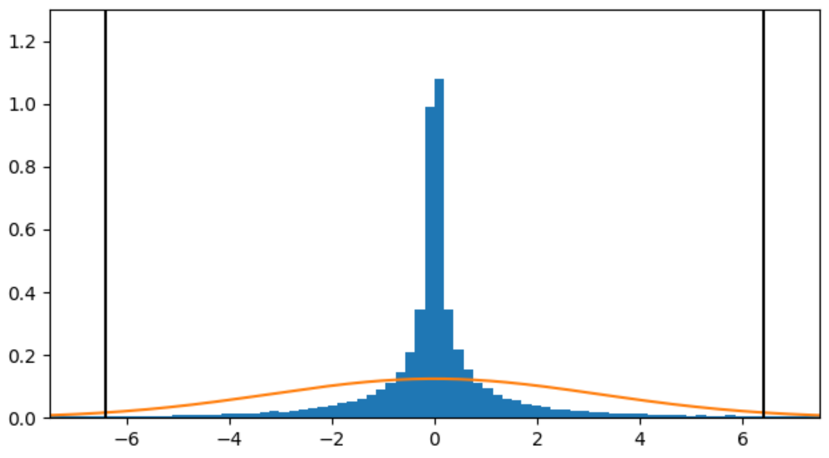

Figure 5.3: A histogram of the wavelet product values for Gaussian noise.

Precision for step widths of one and two is clearly worse from the others. For step width two being approximately 0.1 lower than the main group between SNRs 2–12 dB, and for step width of one (i.e. spikes in the signal) the precision is approximately 0.4 below the main group around SNR of 10dB.

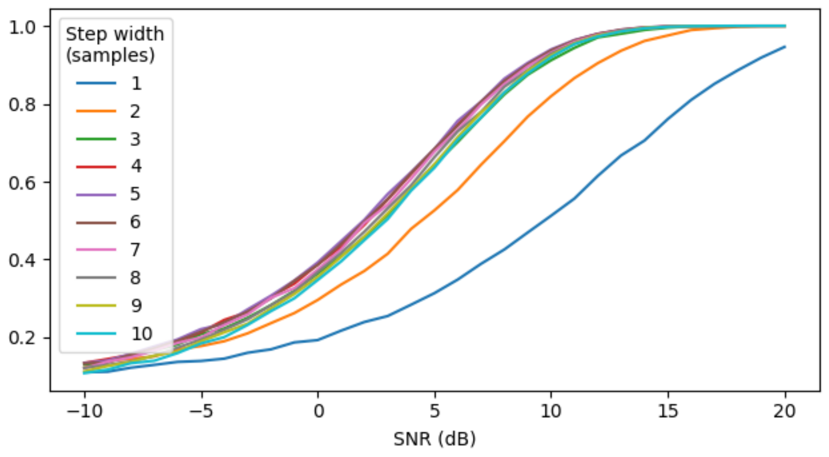

Figure 5.4: Precision for different step widths with varying SNR.

Figure 5.5 shows the estimated recall values for the step detection for various step widths and SNRs. Compared to the precision values, the recall values for different step widths are more spread, with distance between the steps approximately halving for each increase in step width. In terms of recall, the spikes do quite badly, staying under 0.5 even for SNR of 20 dB.

The poor recall results for the spikes and narrow steps are likely due to the fact that the rising and falling edges interfere with each other quite strongly, causing one of the two

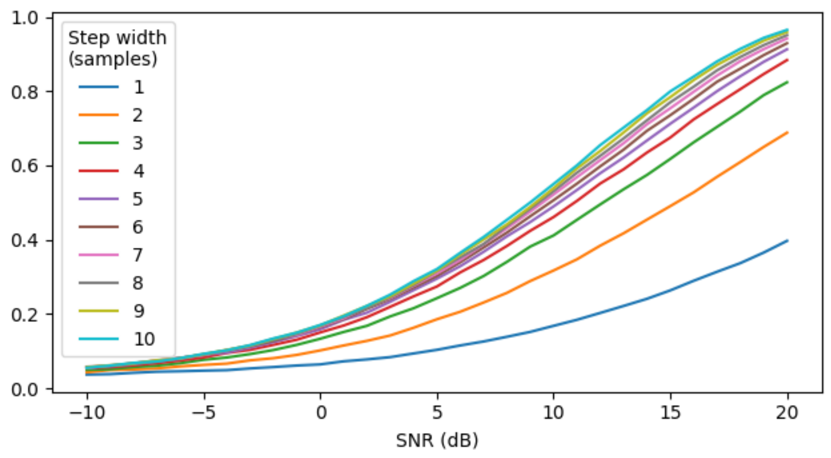

Figure 5.5: Recall for different step widths with varying SNR.

to stay undetected. Figure 5.6 shows recall values when detecting either edge of the step is counted as detecting the whole step. In this case, recall levels similar to the ones in Figure 5.5 for SNR of 20dB are reached already at around 13dB. Especially the narrowest steps are detected significantly better when making this allowance.

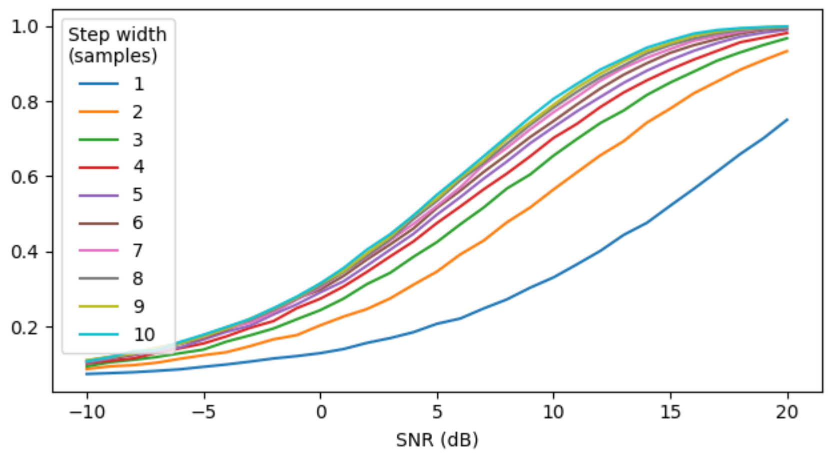

Figure 5.6: Recall for different step widths with varying SNR when detecting either edge of the step is counted as detecting the entire step.

5.3 The effect of data coverage on the step counts

As mentioned earlier in Section 4.1, the coverage of the included time span in the corpus is quite uneven. For some months, only a few days worth of newspapers are included in the corpus while for others, the number goes up to thirty.

In order to investigate the effect of the amount of available data on the step counts, the first test was to perform step detection for each of the words in the actual corpus, and count the the total number of steps detected for each month in the data. Figure 5.7 shows the number of steps detected in the corpus for each month.

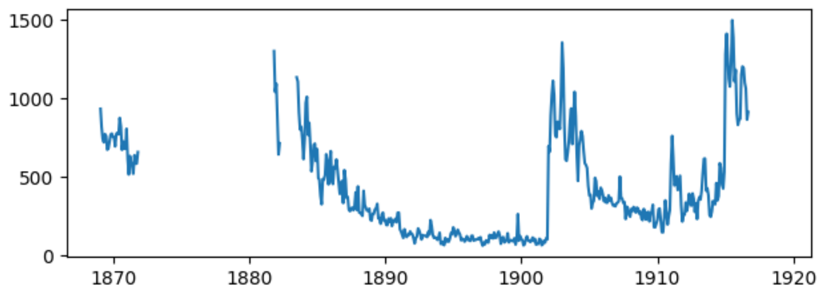

Figure 5.7: The number of steps per month detected by the step detection

As can be seen from comparison to Figure 5.8, repeated here from page 24 for convenience, the number of detected steps seems to have a strong inverse correlation with the amount of data available for the month in question, with less data producing more steps. This makes sense since the monthly IPM values are essentially estimated means for the daily IPM values, and having less data points available would naturally make the estimates less accurate, and increase their variance, which results in large number of false positives.

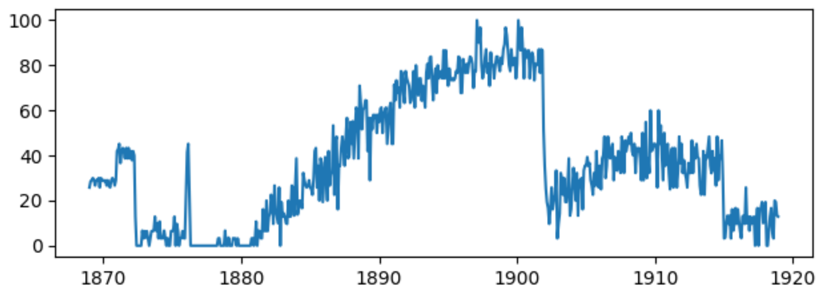

Figure 5.8: Percentage of days with newspaper data in each month, repeated from page 24 for convenience

Figures 5.9 and 5.10 show average step counts per month for two different synthetic data sets. Figure 5.9 shows the result when sampling Gaussian noise instead of values in the corpus, but using the same amount of samples per month as the real corpus contains.

It is easy to see that the overall shapes of Figures 5.7 and 5.9 are quite similar. One of the most clear differences between the graphs seem to be the spike in January 1874, which is missing from the real data, likely due to some difference in the way how months with no data were handled. Of more interest is probably the small (in comparison) spike in 1911 in the actual data (Figure 5.7), which is missing from Figure 5.9.

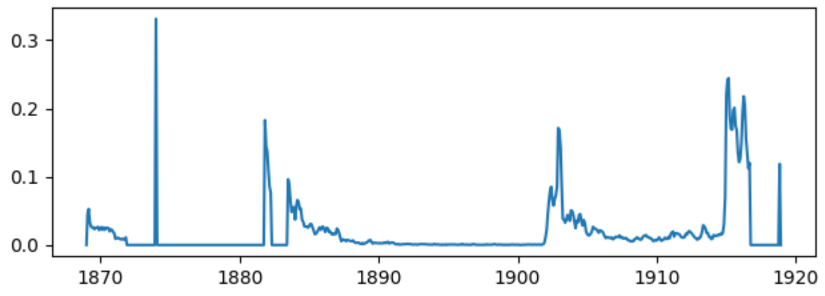

Figure 5.9: The step counts for synthetic data simulating the actual corpus data coverage with normally distributed word IPM values

Figure 5.10 shows how the number of detected steps increases when the number of samples used for estimating the monthly mean value is decreased. For this graph, 20 values were generated using each number of samples, decreasing from 30 to 1. The marks in the x-axis show the beginning of the segment with the given number of samples. It can be seen that the number of detected steps starts to slowly increase at around ten samples per data point, and especially for the last three, the rate of detected steps increases quite rapidly.

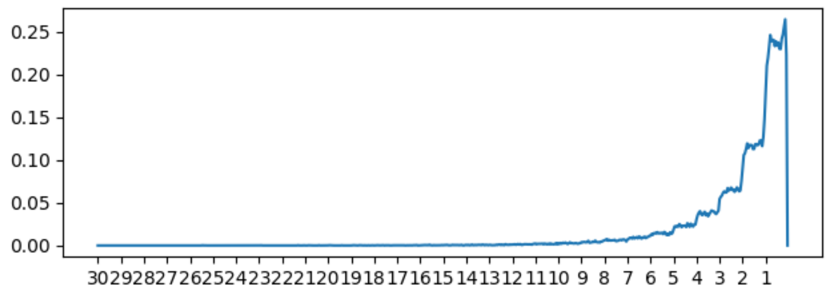

Figure 5.10: The false positive rate for different amounts of issues per month

It should be noted that the false positive rates are not directly tied to any certain number of samples or noise. For instance, if we generate another data set similar to the one used for Figure 5.10, but using only sample amounts from 30 to 11, we get the result shown in Figure 5.11. The shape of the curve is essentially the same, but now the areas with 11–13 samples have high false positive rates compared to the others. This is caused by the fact the the system automatically adjusts the detection threshold lower due to smaller overall amount of noise, which results in smaller noise levels producing the same amounts of false steps.

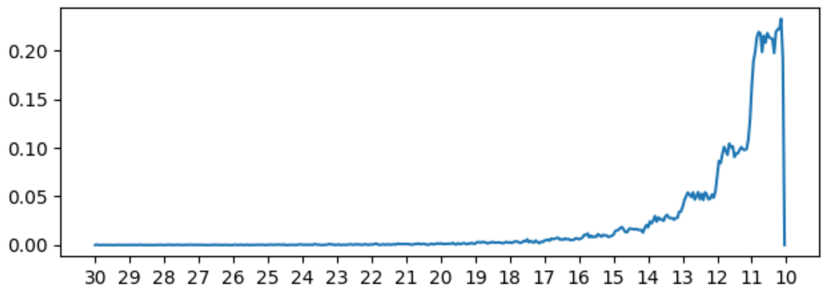

Figure 5.11: The false positive rate for different amounts of issues per month, with higher overall number of available data

5.4 Performance of the event detection

Finally, the event detection was performed on the real corpus, and the resulting events and their scores were examined in more detail. Figure 5.12 show the event scores described in Section 3.6 for all points on the time series. The scores were calculated using four different values for the number of steps to be included in calculating the score.

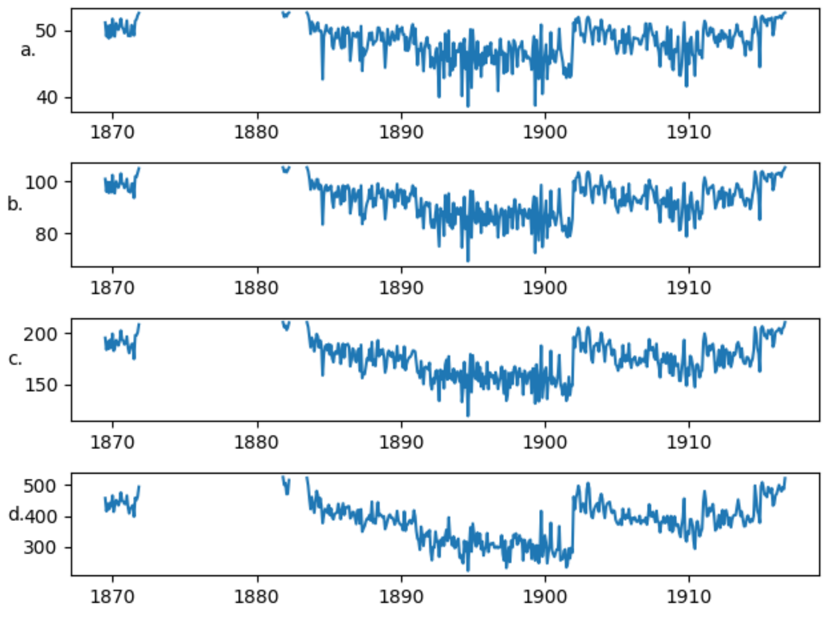

Figure 5.12: The event scores for different time points, using a. 5, b. 10, c. 20, and d. 50 highest scoring steps for calculating the event score

In all of the graphs, the overall shape of the graph is quite similar to the one in Figure 5.9 showing that the event detection is affected significantly by the effects described in Section 5.3.

6 Discussion

6.1 Effects of noise and missing data on the system

Based on the results, wavelet analysis seems to be a reasonable choice for step detection, especially when it can be assumed that the steps are wider than one or two samples. For spikes, detection of both rising and falling edges in quite poor, resulting often in only one of the two being detected. This is visible in Figure 5.5, where the recall for width of one is fairly poor. In order to classify spikes correctly, some additional analysis mechanisms should probably be employed after finding the step locations using the step detection method described in this work.

Another feature of the step detection examined in this work is the automatic scaling employed in deciding the detection threshold for steps. Selecting the threshold value based on the standard deviation in the *multiscale product* works reasonably well when the noise level in the data is fairly constant. When using the threshold value of two times the standard deviation, as was done in this thesis, the rate of false positives for a signal consisting of Gaussian noise is around 4.4 % (see Section 5.2). Any actual steps in the data increase the standard deviation, resulting in reduced false positive rate.

On the other hand, if the noise level is *not* constant, the parts of the signal with less noise cause the detection threshold to become smaller, resulting in increased amount of false positives in the more noisy parts. This is demonstrated in Figures 5.10 and 5.11, where cutting the noisiest parts of the data in Figure 5.10 (the areas with 1–10 samples) simply drops the detection threshold and results in the curve in Figure 5.11 with a really similar overall shape.

OCR errors in the data have similar effects to noise, and as can be seen from Figures 5.4 and 5.5, will cause a decrease in the precision and recall properties of the step detection. Especially the recall for spikes and narrow steps in the data is quite poor in the presence of noise, and OCR noise will therefore reduce the system's ability to detect them.

6.2 Alleviating the effects of uneven temporal distribution of data

Uneven distribution of the data in the corpus causes significant distortion in the detected steps, and needs to be addressed in order to improve the reliability of the system. Two ways of doing this were considered, each with their own downsides.

The first way is to add weights to the scores (see Section 3.5) calculated for each step based on the amount of data available at that time point. Smaller amounts of data mean that the detected steps are less reliable, and therefore should be given less weight when comparing to other steps.

However, selecting the weights in a useful way is not trivial, because the difference in reliability between different amounts of available data depends on the overall levels of available data in the time series, as demonstrated in Figures 5.10 and 5.11. When the smallest amount of monthly issues in the time series is 11, the false positive rate for months with 11 issues is over 20% (figure 5.11, whereas for another set of data where some months have only single issues, the false positive rate for 11 issues per month is around 1%.

The second, easier way is to change the way the time series are formed. Since the number of samples used per data point is the key factor in adjusting the noise level, it should be possible to generate the time series by forming even sized groups of samples that might span different amounts of time. In this way, it should be possible to avoid the varying noise levels caused by different number of samples.

On the other hand, for time periods with less available data, this would result in poorer time resolution, since a single group would contain the data from a longer time period. Also, since the time intervals would be longer, this might mean that the changes happening in those periods would seem faster and therefore more interesting than they should. Still, this seems like a solution that should be experimented on in order to measure its usefulness.

Yet another method that would be an interesting subject of further research would be to look for ways to adjust the step threshold locally instead of having a single threshold for the whole time series. A simple example would be to calculate the step threshold separately for each time point, using only the n closest values of the multiscale product, with some large enough value for n.

6.3 Problematic features of the method, and suggestions for improvements

One of the problems detected during the testing of the system is that the signal extension at the start and beginning of the signal may cause significant artefacts in the analysis (see Section 3.4). Combined with the the way the step threshold is selected for each time series these artefacts can basically ruin the whole time series. As shown in Figure 3.1, even a small spike caused by noise can be significantly exaggerated if it happens to be located in the first sample of the time series. By itself this simply means that there will be a false positive falling edge in the analysis results, and this could be fixed by simply ignoring all steps detected within five or so samples of the start and end of the signal.

However, since the proximity of the signal border causes the spike to seem ten times its real size, it will also cause the step threshold to increase, potentially causing any real steps in the data to go unnoticed. Simply ignoring the step results close to the borders will not correct this problem. A simple fix would be to ignore the same samples in the multiscale product also when deciding the step threshold. In other words, instead of calculating the standard deviation of the whole multiscale product, the standard deviation would be calculated only using the parts of the product not affected by the artefacts, i.e. everything except the first seven and the last six samples.

Another related problem in the implementation is how the mirror extension is implemented. Currently the mirror extension is performed separately at the start of every loop in the algorithm, and the resulting signals are then trimmed back to the original size. This means that for calculating the first three scales of the wavelet transform, the mirror extension if performed not once but three times, each time distorting the original signal.

Instead, it the algorithm should be modified so that it would perform the extension just at the beginning, and then use the extended signal without trimming it back down at every repetition. This should at least slightly reduce the artefacts close to the signal borders, even though removing them completely is not possible. Unfortunately this improvement wasn't added to the system due to time constraints, but it should be an easy modification for further research projects.

6.4 Conclusions

Although the step detection works reasonably well even in a noisy environment for steps wider than a few samples, its applicability on event detection remains uncertain. Additional research is needed to find ways of reducing the impact of features in the data such as the uneven temporal distribution and the high amount of noise. Improving the quality of the used data is one way to improve the situation, and could lead to improved quality of the detected steps and the related events. Some minor improvements unrelated to the issues with the data were also suggested, but by themselves they aren't likely to improve the overall quality of the event detection significantly.

Bibliography

[1] F. Atefeh and W. Khreich. “A survey of techniques for event detection in twitter”. In: *Computational Intelligence* 31.1 (2015), pp. 132–164. DOI: 10.1111/coin.12017.

[2] S. G. Chang, B. Yu, and M. Vetterli. “Adaptive wavelet thresholding for image denoising and compression”. In: *IEEE transactions on image processing* 9.9 (2000), pp. 1532–1546. ISSN: 1941-0042. DOI: 10.1109/83.862633.

[3] S.-W. Chen, H.-C. Chen, and H.-L. Chan. “A real-time QRS detection method based on moving-averaging incorporating with wavelet denoising”. In: *Computer methods and programs in biomedicine* 82.3 (2006), pp. 187–195. ISSN: 0169-2607. DOI: 10.1016/j.cmpb.2005.11.012.

[4] I. Daubechies. *Ten lectures on wavelets.* Vol. 61. CBMS-NSF Regional Conference Series in Applied Mathematics. SIAM, 1992. ISBN: 9781611970104.

[5] A. Grossmann and J. Morlet. “Decomposition of Hardy functions into square integrable wavelets of constant shape”. In: *SIAM journal on mathematical analysis* 15.4 (1984), pp. 723–736. DOI: 10.1137/0515056.

[6] M. Hämäläinen. “UralicNLP: An NLP Library for Uralic Languages”. English. In: *Journal of open source software* 4.37 (2019). ISSN: 2475-9066. DOI: 10.21105/joss.01345.

[7] W. He, Y. Zi, B. Chen, F. Wu, and Z. He. “Automatic fault feature extraction of mechanical anomaly on induction motor bearing using ensemble super-wavelet transform”. In: *Mechanical Systems and Signal Processing* 54-55 (2015), pp. 457–480. ISSN: 0888-3270. DOI: 10.1016/j.ymssp.2014.09.007.

[8] A. Liu. “The state of the digital humanities: A report and a critique”. In: *Arts and Humanities in Higher Education* 11.1-2 (2012), pp. 8–41. DOI: 10.1177/1474022211427364.

[9] E. Loper and S. Bird. “NLTK: the natural language toolkit”. In: *arXiv preprint cs/0205028* (2002).

[10] S. M. LoPresto, K. Ramchandran, and M. T. Orchard. “Image coding based on mixture modeling of wavelet coefficients and a fast estimation-quantization framework”. In: *Proceedings DCC’97. Data Compression Conference*. IEEE. 1997, pp. 221–230. DOI: 10.1109/DCC.1997.582045.

[11] T. Lotze, G. Shmueli, S. Murphy, H. Burkom, et al. "A wavelet-based anomaly detector for early detection of disease outbreaks". In: *Workshop on Machine Learning Algorithms for Surveillance and Event Detection, 23rd Intl Conference on Machine Learning*. International Conference on Machine Learning. 2006.

[12] W. Lu and A. A. Ghorbani. "Network anomaly detection based on wavelet analysis". In: *EURASIP Journal on Advances in Signal Processing* 2009 (2009), p. 4. ISSN: 1110-8657. DOI: 10.1155/2009/837601.

[13] S. Mallat and W. L. Hwang. "Singularity detection and processing with wavelets". In: *IEEE transactions on information theory* 38.2 (1992), pp. 617–643. ISSN: 1557-9654. DOI: 10.1109/18.119727.

[14] S. Mallat and S. Zhong. "Characterization of signals from multiscale edges". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 14.7 (1992), pp. 710–732. ISSN: 1939-3539. DOI: 10.1109/34.142909.

[15] NewsEye. *What is NewsEye about?, accessed 14.10.2019*. URL: https://www.newseye.eu/about/.

[16] A. Noiboar and I. Cohen. "Anomaly detection based on wavelet domain GARCH random field modeling". In: *IEEE transactions on geoscience and remote sensing* 45.5 (2007), pp. 1361–1373. ISSN: 1558-0644. DOI: 10.1109/TGRS.2007.893741.

[17] S. Petrović, M. Osborne, and V. Lavrenko. "Streaming first story detection with application to twitter". In: *Human language technologies: The 2010 annual conference of the north american chapter of the association for computational linguistics*. HLT '10. Association for Computational Linguistics. 2010, pp. 181–189. ISBN: 1-932432-65-5.

[18] S. Phuvipadawat and T. Murata. "Breaking news detection and tracking in Twitter". In: *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. Vol. 3. IEEE. 2010, pp. 120–123. DOI: 10.1109/WI-IAT.2010.205.

[19] A. Rosenfeld. "A nonlinear edge detection technique". In: *Proceedings of the IEEE* 58.5 (1970), pp. 814–816. ISSN: 1558-2256. DOI: 10.1109/PROC.1970.7756.

[20] B. M. Sadler and A. Swami. "Analysis of multiscale products for step detection and estimation". In: *IEEE Transactions on Information Theory* 45.3 (1999), pp. 1043–1051. ISSN: 1557-9654. DOI: 10.1109/18.761341.

[21] O. Sayadi and M. B. Shamsollahi. “Multiadaptive bionic wavelet transform: Application to ECG denoising and baseline wandering reduction”. In: *EURASIP Journal on Advances in Signal Processing* 2007.1 (2007). ISSN: 1687-6180. DOI: 10.1155/2007/41274.

[22] P. Svensson. “Humanities computing as digital humanities”. In: *Defining Digital Humanities*. Routledge, 2016, pp. 175–202. DOI: 10.4324/9781315576251.

[23] D. Taubman and M. Marcellin. *JPEG2000 image compression fundamentals, standards and practice: image compression fundamentals, standards and practice*. Vol. 642. The Springer International Series in Engineering and Computer Science. Springer Science & Business Media, 2012. ISBN: 9781461507994.

[24] T. Van Canh, K. Markert, and W. Nejdl. “A Framework For Historical Russian Flu Epidemic Exploration From German Newspapers.” In: *Digital Humanities 2017 Conference Abstracts* (2017), pp. 630–633.

[25] R. W. White and R. A. Roth. “Exploratory search: Beyond the query-response paradigm”. In: *Synthesis lectures on information concepts, retrieval, and services* 1.1 (2009), pp. 1–98. DOI: 10.2200/S00174ED1V01Y200901ICR003.

[26] Y. Xu, J. B. Weaver, D. M. Healy, and J. Lu. “Wavelet transform domain filters: a spatially selective noise filtration technique”. In: *IEEE transactions on image processing* 3.6 (1994), pp. 747–758. ISSN: 1941-0042. DOI: 10.1109/83.336245.