NTNU
Norwegian University of
Science and Technology

Predicting the Spread of Pandemic Influenza based on Air Traffic Data and Social Media

Martin Almvik
Mikael Rino Solstad

Master of Science in Computer Science
Submission date: June 2016
Supervisor: Konstantinos Chorianopoulos, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Abstract

Influenza is one of the most common viral infections in the world, and puts millions of people at risk every year. Currently, detecting influenza is done by hospitals reporting data, which is very slow. Modern day technology has given a rise of interest in the field of influenza awareness, especially on the topic of detection. Researchers have found that mining messages from Twitter can accurately find influenza outbreaks that correlates with hospital reported data.

Health officials need to be prepared for influenza outbreaks, so that measures can be made to minimize the impact of a potential epidemic. Rvachev and Longini has created a mathematical model for the spread of influenza by air travel, which we have implemented as a public API.

Our implementation of the model use influenza related tweets as a real-time source of influenza incidents, as well as publicly available flight data, to be able to predict spread in case of an influenza pandemic. The results created by this implementation correlates to a reasonable degree with Rvachev and Longini's original results, and have been tested on more recent data. The final system also has a simple interface, which visualizes the spread using Google Maps with an isopleth overlay.

Sammendrag

Influensa er en av verdens mest vanlige virusinfeksjoner, og setter millioner av folk i fare hvert år. I dag oppdages influensautbrudd ved at sykehus rapporterer data til en offisiell instans, noe som er tregt. Moderne teknologi har gitt økt interesse innen bevisstgjøring av influensa, spesielt på temaet influensadeteksjon. Forskere har funnet ut at å ekstrahere meldinger fra Twitter kan finne influensautbrudd med høy nøyaktighet, som korrelerer med data rapportert fra sykehus.

Helsemyndigheter må være forberedt på influensautbrudd, slik at man kan gjøre tiltak for å minimere konsekvensene av en potensiell pandemi. Rvachev og Longini har laget en matematisk modell for influensaspredning gjennom flytrafikk, som vi har implementert som et offentlig API.

Vår implementasjon av modellen bruker influensarelaterte tweets som en real-time kilde av influensatilfeller, i tillegg til offentlig tilgjengelig flydata, for å kunne forutse spredning ved en eventuell influensapandemi. Resultatene fra denne implementasjonen korrelerer til en rimelig grad med de originale resultatene til Rvachev og Longini, og har blitt tested på nyere data. Sluttsystemet har et enkelt grensesnitt, som visualiserer spredning ved bruk av Google Maps med et isopleth lag over.

Acknowledgements

We would like to thank Konstantinos Chorianopoulos for supervising us through this year. He has given us a lot of good advice, and inspired us to complete this project. We would also like to thank Maria Letizia Jaccheri for setting us up with Konstantinos and letting us use her office for meetings. A special thanks goes out to Hans Jacob Rivertz, who provided mathematical help when needed. Finally, we would like to thank our friends and family for supporting us throughout this semester.

Table of Contents

List of Tables

List of Figures

Abbreviations

ILI	=	Influenza Like Illness
SVM	=	Support Vector Machine
EHEC	=	Enterohemorrhagic Escherichia coli

Chapter 1

Introduction

1.1 Background and motivation

The influenza virus is dangerous for millions of people at risk every year. If people become infected with new strains of influenza, there could be little immunity among the general population. This, together with large air traffic volumes can contribute to rapid spread of the disease, which can potentially create a pandemic if not stopped. Health officials need to constantly monitor influenza activity, which currently is done by local hospitals reporting confirmed influenza incidents. This is usually done at a set time interval, e.g. every one or two weeks. It is important for health officials to know early when an epidemic is imminent, so that measures can be taken to minimize the impact of a potential epidemic or pandemic. Generally, there is a need for faster and automated influenza detection, which potentially can save money, time, and even lives.

Lately, there has been a surge in the interest of data available from social media and finding real-world trends in these. Earlier research have found connections between social media trends and different real-world events, like political elections, earthquakes, or most relevant to us, influenza outbreaks. Flutrack[1] is a service that scans Twitter for influenza-related tweets on a daily basis, and displays the tweets on a map at their respective locations. Others have tried applying machine learning to Twitter mining, and have been able to increase the general relevancy of the collected tweets. Finding influenza trends through Twitter can be a lot quicker than current methods, and analyzing these large amounts of tweets can potentially be very helpful in the case of a pandemic.

An aspect of minimizing the impact of an influenza pandemic is knowing where and when the virus will spread. By knowing this, health officials can try to contain the virus if possible, or vaccinate the people at risk. This thesis is heavily based on a paper written by Rvachev and Longini[1], who created a mathematical model for the spread of influenza

[1] `Flutrack.org`

through air travel. By implementing the model, and using it together with air travel data, we can forecast the spread of an influenza pandemic originating from a specific city.

This thesis' contribution to the field will be an open-source system which is able to detect an influenza epidemic, and then model the spread through air traffic. We have found that a general problem with the health informatics field, is that contributions are not easily available and accessible to the general public. This makes it hard for researchers to build on each other, and to improve the different software out there. We hope that our system can be improved on by others, and that the result will be an available and open system that can raise the influenza awareness of the general public.

1.2 Objective and research questions

1.2.1 Objective

The objective is to design, create, and implement the system that is specified in this thesis. The system should be able to analyze influenza trends in major cities, identify increase in Twitter influenza activity, and forecast global spread of potential epidemics. It should also visualize real time influenza trends and the forecast of future spread in an intuitive and user friendly manner.

1.2.2 Research questions

In order to achieve the objective stated above we have formulated two key research questions, which can be used to see if we have accomplished what we wanted. By answering these questions, we can conclude whether or not the proposed system will work as desired.

RQ1: How can we use real-time Twitter data to monitor influenza activity?
RQ2: Is it possible to predict the spread of an influenza pandemic through air travel?

1.3 Thesis structure

Chapter 2 presents the previous work done on this subject, and lays a foundation on which we will base the research in this thesis on.

Chapter 3 describes the system and its design. We will break down the different parts of the system, and describe how they work together to form the end result. This will give the reader a look into the different decisions we have faced throughout the project.

Chapter 4 shows how the system is validated and tested, and presents the results from these tests. We will show how the results of our system compares to the results of others, and how we have done this.

Chapter 5 will discuss the results provided in chapter 4, and will provide further insight into the system.

Finally, in chapter 6 we will conclude the project. We will look at the outcome of this thesis, and describe further work needed.

Chapter 2

Previous work

This chapter will describe some of the previous work done in the subjects of our thesis, as well as some systems which may have similarities to ours. Note that this whole chapter, excluding section 2.1, 2.6, and 2.7, is repeated from our specialization project in the autumn of 2015 [2]. Some minor additions have been made to section 2.3, and chapter 2.8 has been revised.

2.1 ILI surveillance

A lot of research effort has gone into detecting influenza trends in search queries [3, 4, 5], especially with the Google Flu Trends service. The service provided statistical data concerning search queries of symptoms to Google's search engine. The service has since been shut down, although the historical data is still available. This method of ILI surveillance has proven effective in a lot of research, and is in some ways the predecessor to using Twitter for self reported ILI incidences.

Social media in general is a source with vast amount of data, which previously has been underestimated. Lately, a surge in the interest of detecting trends in this data source has taken place. Research has shown that mining this data can detect general trends in many real-word events [6], and more specifically can be used in detecting influenza trends [7, 8].

Twitter.com[1] is a micro-blogging service on the web that allows users to post small, character limited messages called *tweets*, that becomes publicly available. This has caused an increased interest in research attempting to detect influenza epidemics through the analysis of tweets specifically [9, 10, 11, 12]. This topic is of high importance, as it can help the response times for the prevention of a potential pandemic, as well as increasing the overall awareness of the general population. Recently, Thapen, Simmie, Hankin, and Gillard developed a disease nowcasting system, which aims to aid public health officials in their

[1] `Twitter.com`

work with monitoring disease levels [13]. This system uses data from Twitter, where they classify tweets that are relevant, and places them in a location network.

2.2 Spread by air travel

Billions of people travel by airplane yearly, and international travel by airplane has become a significant source of the spread of influenza. In 2009, a study examined the effect of influenza spread by flights from Mexico in march and April, 2008 [14]. Travelers from Mexico were unknowingly carrying the influenza (H1N1) virus around the world. Out of the 20 countries with the most travelers arriving from Mexico these two months, 16 of them have been confirmed to have imported the Mexican influenza virus. This shows that there is a relevant correlation between international travel and the spread of influenza.

This correlation has also been proved empirically by Brownstein, Wolfe and Mandl. They looked at the flu seasons from 1996 through 2005, and found an inverse correlation between the volume of inter-regional travel in the US and the transnational spread of the influenza virus [15]. Increased travel resulted in an earlier influenza peak. Interestingly, after the flight reduction proceeding the terrorist attack in September 2001, the virus took 68% longer to spread transnationally than the previous two seasons.

2.3 Prediction

The ultimate goal of our research would be to create a system that could predict where in the world outbreaks of influenza would spread based on travel data and known outbreaks. The work of Rvachev and Longini provides a mathematical model for forecasting the spread of influenza based on information from the initial city in the transportation network to experience the disease [1].

In their research, Rvachev and Longini uses the model to “forecast” the 1968-1969 Hong Kong influenza pandemic in 52 cities based on background data. The model successfully forecasts the broad patterns of the geographic spread of the pandemic influenza.

In [16] Grais, Ellis, and Glass uses the model devised by Rvachev and Longini to simulate the spread of an influenza pandemic like the Hong Kong influenza, with travel data from the year 2000. Their simulations show that the influenza spreads from Hong Kong to its neighboring cities first, but also to cities with high rates of travelers arriving from Hong Kong at early stages of the pandemic. This suggests that increased airline travel is a large contributor to the spread of infectious agents, and that forecasting where outbreaks might occur globally could help health professionals prevent large pandemics. Flahault, LEtrait, Hazout, Ménarés, and Vallerion have implemented this model with travel by train as the travel data [17], which proves that the model is general enough to work with travel data other than air as well.

In [13], Thapen, Simmie, Hankin, and Gillard have created a system for forecasting the number of tweets mentioning different influenza like symptoms. They gather tweets related to different illnesses based on mentions of selected symptoms and using historical tweet counts, together with a graph network to predict future tweet counts. Their network is constructed as nodes which correspond to locations, with connecting edges between them which are weighted by transportation data. These weights are derived from information about the tweets, as they are looking at the user location of the tweets. If one user have tweeted from more than one location in a given day, this is used as evidence for traveling between these locations.

2.4 Geolocation

Using Twitter data to report occurrences of influenza is most useful in such a setting if the location of the person reporting an illness is known. Twitter provides four options for users to represent their location from a tweet. 1. Exact GPS location. This method uses the GPS position given by the device used to post the tweet. 2. GPS coordinates of a place. This method gives a bounding box of four coordinates corresponding to an area. Usually a city or municipality. 3. User profile location. Twitter users have the possibility of reporting a location on their user profiles. 4. Time zone associated with the twitter account. Twitter infers a time zone on a user's profile when the account is created. The time zone is gathered from the local device the account was created on, and can be changed by the user at a later time.

Both option 1 and 2 are disabled by default, and users therefore have to manually opt to provide such information with each tweet. These two possibilities combined are present on 2.02% of all tweets. Exact GPS coordinates are present in 0.91% of all tweets, and GPS coordinates of a place are present in 1.92% of all tweets. 2.70% of twitter users provide GPS location with either a place or an exact location [18].

When creating an account, twitter allows for users to input their location. Such locations are more static, and are associated with a user's main location rather than the actual tweet location. Furthermore, such locations are just a text string, which can be anything the user desires. Twitter provides place suggestions when editing this location, but a user may opt to give any text string. Main problems with this are humorous locations such as "Heaven" or "Neverland".

As with user profile location, the time zone associated with a twitter account is also a static location indicator. Furthermore, time zones cannot indicate any specific location, but rather distinguish between major regions of the world. This information could be used to disambiguate city names or region names. If a user has profile location set to Birmingham without a mention of country, a look at the time zone could establish whether the user is located in Birmingham, Great Britain or Birmingham, Alabama.

2.4.1 User profile location accuracy

The work of Graham, Hale, and Gaffney [19] analyzed twitter location data and made some interesting findings regarding geolocation. They gathered a number of tweets using the whole earth as a bounding box, securing that every tweet in their data set included a valid geolocation. Using a sample of 1000 tweets from unique users in a specific area (city) in their data set, the researchers tested the accuracy of self reported user profile locations. Using Google's Geocoder, the researchers could determine a location within the correct bounding box for 45.8% of the tweets. Of the resolved locations which fall outside the correct bounding box, 5.8% of these are locations within the correct bounding box. There was multiple reasons as to why the geocoder wasn't able to place the user locations inside the correct bounding box, including abbreviations of location names, and multiple locations stated.

2.4.2 Carmen

Carmen [20] is a system developed for inferring a tweet's location based on the user profile location. The system uses a list of places constructed from common user locations extracted from millions of tweets. Duplicates are merged (New York City and NYC) and invalid locations are removed through a combination of automated filters and manual review. This list is called the human curated list (HCL). This list is then extended by adding places found in a process called geolocation by association [21]. This method includes places such as "bmore" or "balto" and identifies that these are abbreviations for the actual location "Baltimore" by observing that users with these locations frequently communicate with users that has "Baltimore" as their location [20]. This extended list is called the automatically extended list (AEL).

When evaluating this system, the researchers tested it on a set of 1% of all public tweets from the first 9 days of March 2013 (43,656,388 tweets). The system was able to resolve a location on the city level for 57.9% of the tweets using the HCL, and on 63.4% of the tweets when using the AEL [20].

2.5 Visualization

The thematic map was popularized in the 19th century by Charles Joseph Minard, a french cartographer and engineer [22]. A thematic map, unlike a general reference map, is designed to show a specific theme, and does not care about the usual spatial characteristics of the map. Normally, it is used to visualize statistical data in relation to spatial information. For example, it can be used to show economical aspects of a region, or in our case, show the spread of the influenza virus and the risk of influenza associated with a specific geographic location.

There are several types of thematic maps used for displaying statistics. A widely used one is the choropleth map, which shows statistical data over predefined regions, often represented by shadings or different colors [23]. One challenge with this kind of map is

optimizing the number and the range of the class intervals used as the statistical foundation [23]. An example of a choropleth map is shown in figure 2.1[2].

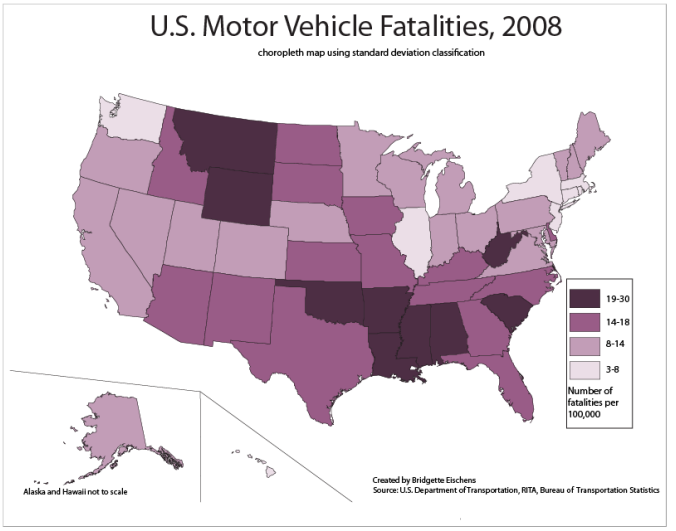


Figure 2.1: A choropleth map, showing motor vehicle fatalities of the different states of USA

On the other side of the map continuum is the isopleth map, which displays data continuously. Schmid and MacCannel defines an isopleth map as a map where isopleths connect equal rates or ratios for specific areas [24]. An example of such a map is a population density map. This type of map can be hard to create dynamically though, partly because one will have to use areal interpolation to create the dividing lines in the area [24]. An example of an isopleth map can be seen in figure 2.2, which is used as an example of heatmap.js' capabilities[3].

[2]Figure taken from `http://www.d.umn.edu/~eisch032/ChoroplethLab5.png`
[3]`http://www.patrick-wied.at/static/heatmapjs/`

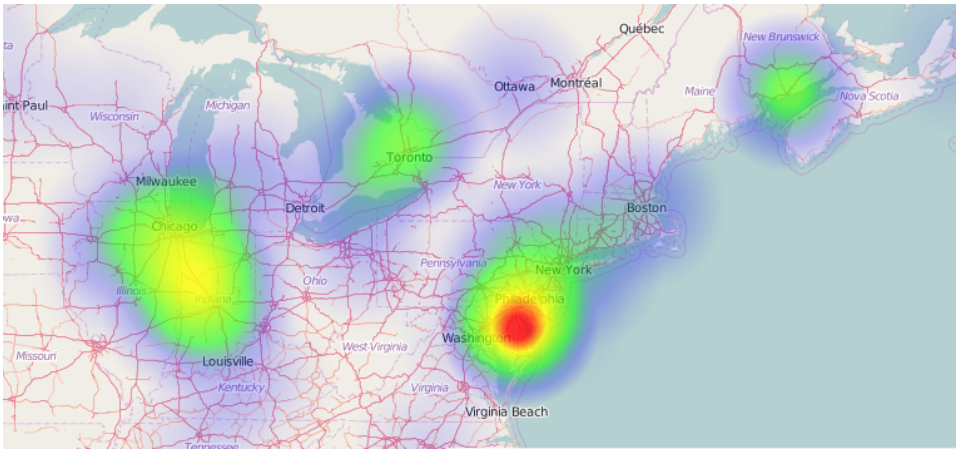


Figure 2.2: An isopleth map, also known as a heatmap

In between these two different thematic maps is the dasymetric map, which is considered the midway between the discrete nature of choropleth maps and the continuous nature of isopleth maps. Eicher and Brewer [25] did a study on the implementation and evaluation of dasymetric maps, and defines it as: "A dasymetric map depicts quantitative areal data using boundaries that divide the mapped area into zones of relative homogeneity with the purpose of best portraying the underlying statistical surface." The popularity of dasymetric maps has been increasing over the years with the introduction of geographic information systems, but still lack standardization [25]. Similar to the isopleth maps, the dasymetric map is usually made with the use of areal interpolation. See figure 2.3 for an example of a dasymetric map with its error. This figure is taken from Eicher and Brewer's research [25].

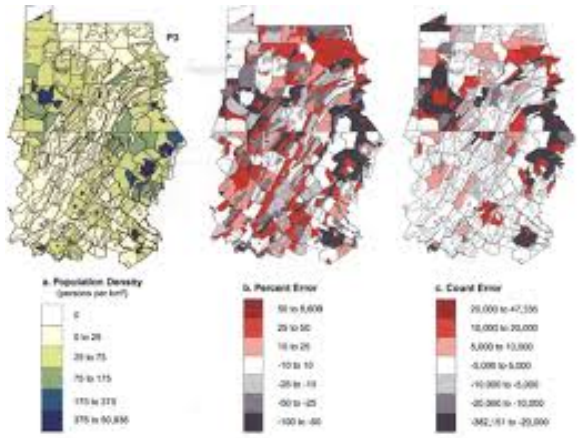


Figure 2.3: An example of a dasymetric map with its associated error

2.6 Machine learning

Using Twitter as a data source means that one will get a lot of data to process. Machine learning has become more and more used with the increased amount of data stored around the world. Companies use machine learning on large datasets to extract trends and information on otherwise unknown topics, and researchers use it to find patterns previously unexplored. In the case of influenza and Twitter, there has been several attempts to create an influenza detection system using machine learning to filter out irrelevant or misleading tweets.

Aramaki, Maskawa and Morita [11] tried to use a support vector machine [26] based sentence classifier to detect cases of influenza with Twitter, and found that their method had higher correlation of 0.89 compared to the gold standard data from IDSC (Institute of Decontamination Sciences) than the state-of-the-art methods at that time. These state-of-the-art methods included Google Flu Trends and simple Twitter keyword search. This research proves that tweets can be used to accurately detect influenza at a reasonable degree of certainty, and that using machine learning and natural language processing (NLP) is an effective way of filtering these tweets. Work done by Culotta [27] confirms that the use of a classifier can be beneficial for filtering out tweets that are not influenza-related, but states that the classifier needs to take use of relatively advanced NLP techniques, e.g. n-grams and synonyms, in order to achieve sufficient correlation. He also states the importance of pre-processing of the tweets, as they often feature informal syntax and spelling mistakes.

Lamb, Paul and Dredze [12] confirms the importance of differentiating types of tweets, in that a tweet regarding influenza not necessarily implies a case of influenza. In their research, they for every tweet first decided if the tweet was influenza related or not, then if the tweet was influenza related they decided of the tweet concerned influenza awareness or an actual case of influenza and if the tweet concerned the author or someone else. Overall, they found that evaluating only influenza related tweets gave better correlation than any other twitter based classifiers at that time.

2.7 Open-source software

Open-source software has become increasingly popular in the later years. The trend of exposing source code has created a new way of developing and maintaining software, also in the research community. However, there are still a big part of the academic world where source code never gets released. This makes it hard for other researchers as ourselves to find a basis for research in creating software. We therefore want to emphasize on making our software open-source, so that it can be reused and studied by others. Recently, a movement trying to bring awareness to this issue has arisen. SoftwareX[4] is a journal that focuses on acknowledging the importance of software and source code as research, and values software as research contribution as much as textual reports. The goal of the journal is to make it easier for researchers and engineers to build upon each other's software,

[4] `http://www.journals.elsevier.com/softwarex/`

rather than rebuilding it from scratch.

Work done by Baldwin and Clark [28] suggests that modularity and option value are two important architectural features when it comes to the success of an open-source system. In their paper, they define modularity as: ”A complex system is said to exhibit modularity if its parts operate independently, but still support the functioning of the whole” [28, p. 6]. A modular system is often based on a stable platform of code on which compatible modules can be built upon. Modularity is important in an open-source system because of it’s ability to support adding features and different parts of the system independently, which in turns increases the capabilities of the system. The second feature the article mentions is option value, which is the possibility to change the design as one experiments and tries different methods. This is supportive of the open-source ideology in that one wants to be able to add new features which previously was not thought of. Modularity and option value are thus two important architectural tactics we can implement in order to make it easier to contribute to an open-source system.

2.8 Similar systems

There have been several attempts to create real-time influenza surveillance systems, especially in the later years. Most health institutions report data to some organ, which oversees the potential of an epidemic, but these methods are slow. Normally, such reports are done at a set time interval, for example every week. This reporting provides accurate influenza detection, but it may detect an influenza too late. Real-time surveillance systems have been created to combat this issue. They may not be as accurate as reported data from hospitals, but it’s possible to detect epidemics much earlier, and the findings often have a high correlation with reported data. This section will feature some of the existing systems that somehow relates to the system we are trying to make.

2.8.1 Flutrack

Flutrack uses data from tweets publicly available in the twitter API. The system gathers tweets, processes them and places markers on a modified Google Maps module. The system uses linguistic filtering to filter out tweets that are not influenza-relevant, like ”Fever cough diarrhoea upset stomach joint aches headache ... I lost my appetite” [29]. When the relevant tweets have been found, the system uses geolocation to find the respective tweet’s coordinates. This system will be the basis of our research, as it is open source and provides a foundation of the end result.

2.8.2 NowTrending.HHS.gov

NowTrending.HHS.gov[5] is a health service provided by the US government which gathers tweets concerning everything from natural disasters to influenza. The system shows relevant statistics about the tweets, as well as visualizing this as a heatmap on their respective

[5] `http://nowtrending.hhs.gov/`

locations. While this service uses similar methods as our research will do, it has a much broader scope, and covers a lot more tweets than necessary in our case. Our system will also implement a prediction module, which this system is missing.

2.8.3 Flutracking.net

Flutracking.net[6] gathers data from a weekly survey distributed to Australian citizens. The system has quickly become a great example of how an online influenza-like illness tracking system can work. It's results corresponds with national reports [30], and accurately pinpoints Influenza outbreaks. Flutracking.net updates it's data weekly along with a map to visualize it. Our project will seek to be even more real-time than that, and this system proves that collecting symptoms from people can provide satisfyingly accurate results.

2.8.4 Bluedot.global

Bluedot.global[7] began as a research program at St. Michael's Hospital in Toronto with the goal of understanding how infectious diseases spread worldwide. The main focus of the program was to study how air travel has connected the world population, and with it the international spread of diseases. The program analyzes air traffic, weather conditions, livestock density, etc at real-time to predict potential pandemics. However, the system is highly closed, is not available to the public population, and is first and foremost meant for health officials.

2.8.5 NLP flu warning

Aramaki, Maskawa and Morita [11] has created a influenza surveillance system with a 0.97 correlation at the outbreak and early spread of an influenza epidemic, which outperforms many other systems. The system can be found at http://mednlp.jp/influ/, but is in Japanese and only covers Japan. The system uses machine learning (SVM based approach) to filter out tweets that are not influenza related, which has improved the resulting correlation greatly.

[6] `http://www.flutracking.net/`
[7] `http://bluedot.global/`

Chapter 3

System Design

This chapter will describe the design of the different components in the system. It will elaborate on what their purpose is, and how they were built. The chapter will also cover some fundamental principles on which the system is built upon.

3.1 Architecture

The designed architecture of a system is essential for it to thrive as open-source, as mentioned in chapter 2. The goal of our system is to create a platform on which others can contribute and expand. The main architectural feature we have chosen to focus on is modularity. The idea is to have a set of modules which are more or less independent from each other that together form the system in its entirety. The system consists of a back-end to process data, as well as a front-end to visualize it. The back-end consists of some python modules for gathering and processing data, as well as some exposed REST API endpoints made with Django rest-framework. The endpoints are accessible to anyone, but are primarily used by the front-end. The front-end is a basic AngularJS application which queries the back-end for data, and then visualizes it.

3.1.1 Modularity as the main architectural trait

The back-end is implemented with modularity in mind, so that it can be extended easily, both by us and potential other contributors. It is divided into modules, each with separate concerns, to minimize coupling and ensure high cohesion. One module is for the API and its respective endpoints, another handles the tweet metadata as well as the gathering of these, and another one processes the tweets and calculate prediction. By structuring the code in this way, it is easy to build upon these modules independently, or even build completely new ones without overlapping the concern of the other existing ones. This also paves way for option value, as the options are endless when it comes to implementing new modules or changing existing ones. A diagram of the implementation of this modularity

can be seen in figure 3.1.

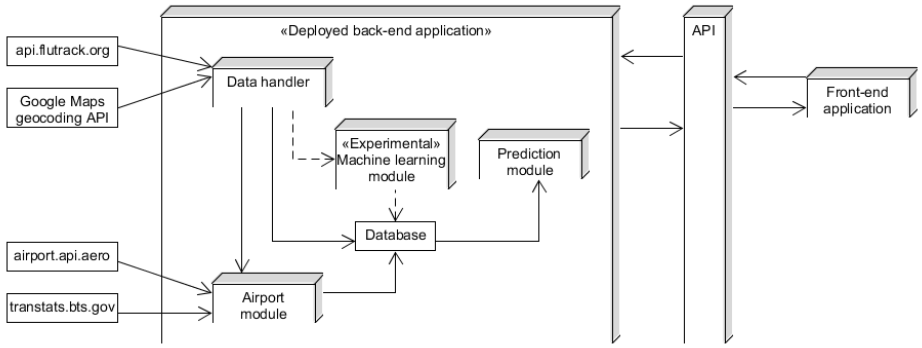


Figure 3.1: Diagram of the modules and their relations in the back-end

3.2 API design

The REST API is designed with the aforementioned traits in mind. As of now, the API consists of two endpoints, and is publicly available. The service is available at `http://flutrack-backend.herokuapp.com/`, and returns JSON responses. An overview of the available endpoints can be seen in table 3.1. Further details about the endpoints can be found below.

Endpoint	Description	Returns
/tweets	An endpoint to get the available twitter data	Returns a JSON response with all available influenza-related tweets
/prediction	An endpoint for predicting the spread in a city	Returns a JSON response with day-to-day infection data about each city on each day

Table 3.1: REST API endpoints

3.2.1 GET /tweets

The /tweets endpoint returns the available influenza-related tweets from the system in the respective city, as well as some metadata. Each entry in the returned list represent a city in the defined city matrix, which contains location as latitude and longitude, the amount of tweets for each of the eight previous weeks, a field stating whether the city is under an influenza epidemic, and a field stating whether the city has an increasing trend. The purpose of the endpoint is for the user to be able to collect the data used by the system.

Resource URL
`http://flutrack-backend.herokuapp.com/tweets`

Resource information

Response format	JSON
Availability	Public

Parameters
None

Example result


```
[
  {
    "city": "Atlanta",
    "location": {
      "lat": 33.7489954,
      "lng": -84.3879824
    },
    "weeks": [
      14,
      1,
      4,
      21,
      5,
      4,
      16,
      20
    ],
    "increasing": false,
    "epidemic": false
  },
  {
    "city": "Bangkok",
    "location": {
      "lat": 13.7563309,
      "lng": 100.5017651
    },
    "weeks": [
      125,
      4,
      4,
      0,
      2,
      0,
      37,
      10
```

```
    ],
    "increasing": false,
    "epidemic": false
  }
]
```

3.2.2 GET /prediction

The /prediction endpoint returns spread data for each city for the whole forecast time period, based on an index city where the influenza outbreak has originated. Each day is represented as an array of the tracked cities with the respective computed morbidity rates on that day.

Resource URL
`http://flutrack-backend.herokuapp.com/prediction`

Resource information

Response format	JSON
Availability	Public

Parameters
None

Example result
Snippet from a request to `https://flutrack-backend.herokuapp.com/prediction`. This returns the morbidity, location, and city name of each of the cities for each day in the forecast. Each day is represented as an array of the registered cities. The example here is limited to only showing the two first days of results from Atlanta for simplicity.

```
[
  [
    {
      "location": {
        "lng": -84.3879824,
        "lat": 33.7489954
      },
      "city": "Atlanta",
      "morbidity": 22,
    }
  ],
  [
    {
      "location": {
        "lng": -84.3879824,
        "lat": 33.7489954
```

```
    },
    "city": "Atlanta",
    "morbidity": 5
   }
  ]
]
```

3.3 Data sources

The system is highly dependent on a high quantity of good quality data in order to perform as expected. We need a lot of tweets in order to establish that an epidemic has originated in a certain city, which needs to be identified with an acceptable degree of accuracy The prediction module is also dependent on flight data, more specifically how many passengers fly from and to each city in the set of 52 cities described in section 2.3. Additionally, we need some metadata to be able to connect airports, cities and tweets in a structure which is usable.

3.3.1 Twitter

Twitter provides two different APIs, the REST API and the streaming API. The REST API lets us search directly with a specified query, and returns a JSON response with a non-exhaustive set of tweets which match this query. The API has limitations, and only allows for 180 queries per 15 minute window. The streaming API provides a connection directly onto the incoming stream of tweets, which gives us access to a majority of the tweets real-time. The streaming API, as the REST API, provides the ability to limit the results with a search query.

With the REST API, we found that it is hard to find a broad enough query which satisfies the accuracy level and quantity needed for this kind of system. What we saw was that the more specified our query was, the fewer results we got. For example, searching only for the keyword 'influenza' returns tweets like "ANIMALS 16 dogs, from 3 Chicago area shelters, naturally infected with canine influenza A H3N2 ... #medsocnews", which clearly does not represent an influenza infected individual. This query generates a lot of tweets, so many in fact, that it quickly exceeds the query limit of 185 queries per 15 minute window. The REST API returns a maximum of 100 tweets per page. When that is returned, another query must be made to request another 100 tweets, which means that the limit is reached after acquiring 18000 tweets. By narrowing the query, we can find more relevant tweets, for example with the query 'influenza+OR+flu+AND+sore+AND+throat+AND+fever'. One example tweet we get with this query is "A torn up stomach, sore throat, fever, body aches and chills - I either have the flu or I really should've stayed...", which is likely to be a case of an infected individual. However, with this query, we find only 107 matching tweets, which is not nearly enough for our purpose. Thus, the main challenge with using the REST API is to find a balance between accuracy and quantity, where we have tweets

that depict a case of influenza, and we have enough of them, preferably in the thousands.

With the streaming API, we need to gather for a longer time, as the stream is real-time and does not provide any historical tweets. Applying only 'influenza', or similarly 'flu', as the only keyword filtering is ineffective. In a ten minute period of streaming, we received zero tweets with either of the keyword filters. For the streaming API to be effective, we need more keywords to search for. We therefore tried some more complex filters with several search terms. With Tweepy[1], the streaming API is set up with a list of keyword searches, where each element in the list represent a disjoint search term. It also allows for joint keyword searches, which are entered as several words within an element in the list. One of the symptoms of influenza is sore throat, and this has to be represented as one search term in order to get accurate results. This is possible with Tweepy, as one would enter ['sore throat'], and it would return tweets which contains both of those words. We tried a search with the most common symptoms associated with the influenza, namely ['fever', 'sore throat', 'cough', 'runny nose', 'headache'], and found that we received a steady stream of tweets. In a ten minute period, we collected 342 tweets of mostly inaccurate results. Terms like 'headache' and 'cough' is often used in other situation, where it has no connection with the influenza virus. For example, the tweet '@User ya *cough* my prom date' has nothing to do with influenza, but is still collected through the stream.

3.3.2 Air travel

The system has a module for calculating daily passenger flow between different cities. The system needs two data sources to make this possible. One source for passenger statistics between airports, and one source for which airports are associated with a particular city.

Passenger statistics

The system uses the database T-100 market from United States Department of Transportation[2]. This offers passenger statistics between airports of the United States and the rest of the world. The database only offers statistics from flights that are connected by at least one U.S airport. This means that no travel statistics between cities outside the U.S is available to our current system.

Airport information

The system's airport module makes a call to SITA's Airport API[3] to look up all associated airport codes for every city in the influenza prediction model. The module stores the information about each city in a database, so that the system does not need to make a call to the API for each calculation. This is only necessary when a new city is added to the influenza model. The module ensures that the system is extensible with minimal amount of modification. By automatically looking up each airport code for a city in the matrix, one only needs to add a city name to the list of cities for the system to be able to work

[1] `http://www.tweepy.org/`
[2] `http://www.rita.dot.gov/bts/home`
[3] `https://www.developer.aero/Airport-API/API-Overview`

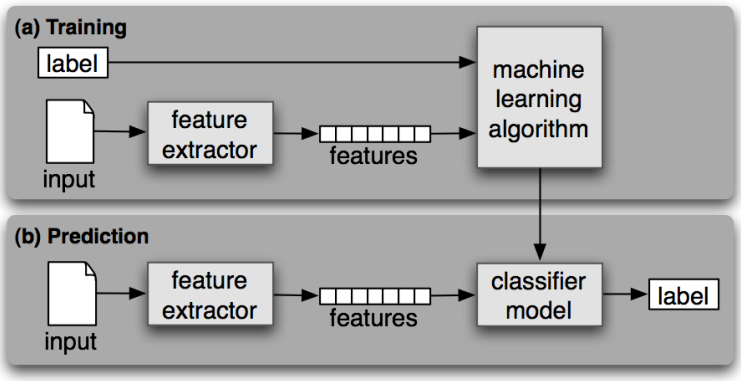


Figure 3.2: The learning and prediction of a classifier. (a) shows how the classifier learns, while (b) shows how it uses the learning to label new data.

properly. This will still work even though the system is extended with data from other sources than the United States Department of Transportation[4].

3.4 Machine learning

As explained in section 3.3.1, we generally found that it was hard to get accurate results in high enough quantities based only on raw twitter search. Machine learning, as mentioned in chapter 2, can be helpful to further filter the received tweets. We attempted to implement a two-class sentiment analyzer, that would recognize if a tweet concerns a case of influenza or not. This process will be explained in this section.

3.4.1 Classifier design

Lamb, Paul and Dredze [12] collected an annotated dataset of tweets for their research, and has kindly published it[5]. We can train our classifier on this dataset, and test it on tweets gathered from the Twitter APIs. An overview of the classifier's structure can be seen in figure 3.2.

The classifier begins by preprocessing all the tweets in the learning dataset, converting all characters to lower case, replacing all URLs with 'URL', replacing all username mentions à la '@flutrack' with 'AT_USER', and trimming extra unnecessary characters, whitespaces, and hashtags. Furthermore, we created a feature vector based on all the words in the dataset, excluding common words that does not contribute to the semantics of the tweets, like 'a', 'and', 'the', etc. To further increase the accuracy of the training, we added the support for bigrams in addition to the unigrams. A tweet containing the unigram 'influenza' seems like it could be a case of influenza. However, if we also look at possible bigrams

[4] `http://www.rita.dot.gov/bts/home`
[5] The dataset can be found at `http://www.cs.jhu.edu/~mpaul/downloads/flu_data.php`

like 'not influenza' or 'influenza news', it is apparent that it doesn't necessarily have to be related to an infected individual. The end result is a feature list with every unique word found in the tweets. The classifier's task is to mark each feature in the feature list with a positive to negative ratio. This way, the classifier should be able to distinguish influenza-related tweets from other non-significant tweets. This is done by creating a profile based on the feature list for each tweet. For example, if the feature list is `['influenza', 'jacket', 'cough']`, then each tweet would produce a profile looking something like the snippet below.

```
{
   contains('influenza'): True,
   contains('jacket'): False,
   contains('cough'): False
}
```

Based on the classifier's opinion on the importance of each word, the classifier will come to a conclusion deciding if the tweet confirms an influenza infected individual or not.

We attempted to implement and test two classifiers, namely Naive Bayes [31] and Maximum Entropy [32]. Each of them has their advantages, but we found that they produced similar results. Lamb, Paul and Dredze's [12] research showed that classifying more than once proved effective. As stated in chapter 2, they classified with regards to whether a tweet is influenza related before they classified based on if the tweet concerns awareness or an actual case, as well as deciding if the tweet talks about someone else or themselves. We attempted the same approach, but without self/others-classification. We implemented a related/not related-classifier and an awareness/infection-classifier. The system first trains the related/not related-classifier, and then the awareness/infection-classifier. As a result, we can first feed the related/not related-classifier with tweets, and then feed the tweets which got classified as related into the awareness/infection-classifier in order to filter out awareness tweets, such as news stories.

3.5 Prediction algorithm

This section will describe how the system plans to predict influenza outbreaks, what data it uses, and how it structures this data to form meaningful output.

3.5.1 The model

The model for spread of influenza is the same model as the one described in Rvachev and Longinis work [1]. For a more detailed explanation, see their work. The model divides a city's population in four different, disjoint states: Susceptible, latent, infectious, and removed. A fraction of the population is assumed to be susceptible to the disease initially. For the city that is the source of the epidemic, the number of latent and infectious individuals are calculated for the days leading up to the outbreak and set to zero for all of the other cities. At the start of the forecast, the number of susceptible, latent, and infectious

individuals are calculated for each city for each day up until the forecast horizon. The model uses a matrix of average daily number of passengers that travel by airplane between each city in the model. This is then used as a probability for any infectious individual traveling to another city and infecting that city's inhabitants.

Key values

The model uses different key values that are calculated in the original paper. This includes a set of infection probability distributions, the length of incubation and infection period, daily infectious contact rate, and fraction of susceptible population. These values are all taken directly from the original paper.

Key values	
Length of incubation period	2
Length of infection period	8
Daily infectious contact rate	1.055
Fraction of susceptible population	0.641
Fraction of ill individuals reported	0.3

Table 3.2: Key values concering characteristics about influenza [1].

Seasonal swing

Because influenza is a respiratory disease, it exhibits a characteristic seasonal pattern of higher infection rates in colder months, and lower infection rates in warmer months. The model therefore implements a scaling factor for seasonality, in which the daily infectious contact rate is lowered by a certain factor depending on which month it is, and in which part of the world the city is in. The world is divided into three zones: Northern hemisphere, tropical hemisphere, and southern hemisphere. Cities located in the tropical hemisphere do not exhibit seasonal swinging. This seasonality scaling factor is identical to the seasonality scaling factor used by Grais, Ellis, and Glass [16]. Table 3.3 show how this scaling factor is applied to the cities in the different zones of the world at different times of the year.

Month	Southern hemisphere	Northern hemisphere
January	0.10	1.0
February	0.25	0.85
March	0.55	0.70
April	0.70	0.55
May	0.85	0.25
June	1.0	0.10
July	1.0	0.10
August	0.85	0.25
September	0.70	0.55
October	0.55	0.70
November	0.25	0.85
December	0.10	1.0

Table 3.3: Different seasonal scaling factors applies for northern and southern hemispheres. Cities in the tropical hemisphere do not exhibit seasonal swinging [16].

Flight data

To make the system easily extensible, it needs to be able to add cities to the model on demand. Our only source of passenger traffic is on a per-airport basis, which means that various airports are associated with each city. Statistics for each of these airports needs to be gathered and categorized together with other airports belonging to the same city. Therefore, the system uses a module for handling passenger travel data between cities.

3.6 Geolocation

Geolocation is a very important aspect to our system. The geolocation must be accurate to city level, coordinates beyond that is not needed. The tweets we gather from Flutrack has exact coordinates, which we need to reverse geocode to the name of the city or town. We keep a MongoDB database of all the required cities, which for now is the 52 cities listed in Rvachev and Longinis research [1]. In each document in the database, there is data on population count, point coordinates, the name of the city, as well as city bounds as coordinates. The point coordinates are there to have a point to represent in the front-end map, the population count is used in the prediction, and the city bounds are used to determine which city a tweet is coming from. The system looks up these bounds for each city through Google Maps Geocode API[6]. When an incoming tweet is processed, the system checks if it is located within one of the bounds in the database, and adds the city name to the tweet document in the database if it is. If it's not, then the system reverse geocodes the location through the Google Geocode API, and adds the result to the database document. The Geocode API has a limit of 2500 requests per day, which is not enough for the amount of tweets returned by the Flutrack API. This is solved by introducing this saved bounding box for each city, which makes it simple to lookup the city from where tweets originate.

3.7 Front-end and visualization

Being able to clearly and efficiently visualize the data our system produces is imperative. We have chosen to use the Google Maps API together with Heatmap.js[7] as the visualization tools in our application. A heatmap is an isopleth map, and is efficient at portraying data in relation to each other. We decided in our specialization project [2], that using an isopleth map for visualization would be the best choice. With this setup, it's easy to see the general area and intensity of the influenza spread.

The front-end is built using AngularJS[8], and features a viewable Google map with a heatmap layer on top. The page also features a mechanism to change the amount of days in the future to predict an influenza epidemic. A screenshot of this can be seen in figure 3.3. The front-end collects tweets with their associated metadata from the API, namely

[6] `https://developers.google.com/maps/documentation/geocoding/intro`
[7] `http://www.patrick-wied.at/static/heatmapjs/`
[8] `https://angularjs.org/`

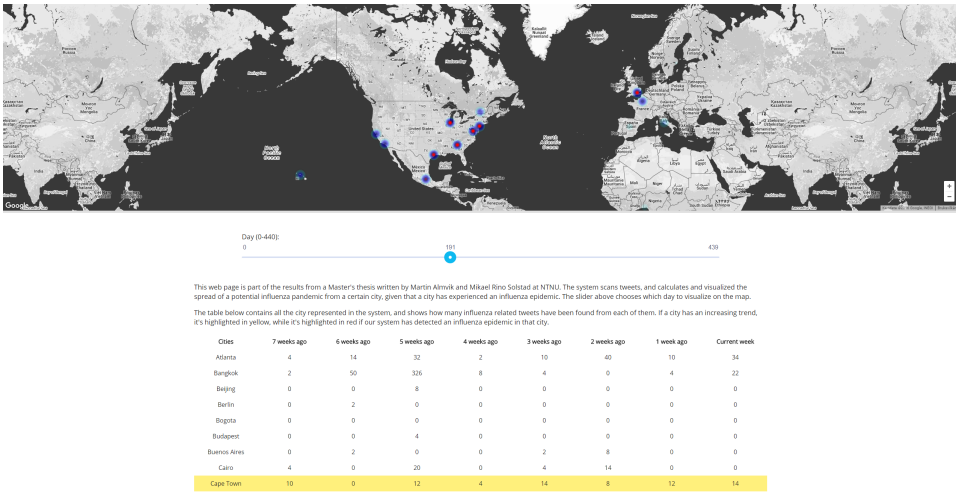


Figure 3.3: Screenshot of the visualized data. This visualization shows the state of the spread 191 days in the future when Honolulu is the pandemic origin. Cities marked yellow means that they have an increasing trend, and should be watched.

from the /prediction endpoint. Ultimately, this feature can be merged together with the original Flutrack system, and work as one total influenza awareness service. The front-end application can be found at `http://flutrack.almyy.xyz`

3.8 Deployment

Both the back-end and front-end are hosted on Heroku, a cloud hosting service with flexible scaling and pricing. The service is based on scalable *dynos*, which are small linux containers that run a single user command. Each dyno operates within the app's *slug*, a compressed copy of the application and its dependencies. The dyno is scalable horizontally, meaning one can have several dynos working together in the same slug, or vertically, which means one can upscale a single dyno with more available memory and general computing power. The price depends on the scaling of the application, but Heroku offers a free version, where one can get a single dyno per application with some limitations. These free dynos can only be active 18 hours in a 24 hour time frame. If the dyno exceeds this limit, it will be forced into sleep for six hours. Heroku offers a simple way to host small applications, as we can just push a specific git branch to a specific remote, which then automatically gathers and compresses the application's slug and deploys the application on a desired dyno.

By using Heroku, one has access to many different add-ons, which provides extra functionality or customized content. Our back-end uses a MongoDB add-on, namely mLab

MongoDB[9], which provides 496 MB of storage for free. The service is used to store tweets with their respective metadata, as well as city information.

[9] https://elements.heroku.com/addons/mongolab

Chapter 4

Results and testing

This chapter presents our results. It will elaborate what types of data was needed for testing the different modules of the system, how we acquired these, and how we used them to validate that our system was implemented correctly.

4.1 Prediction model

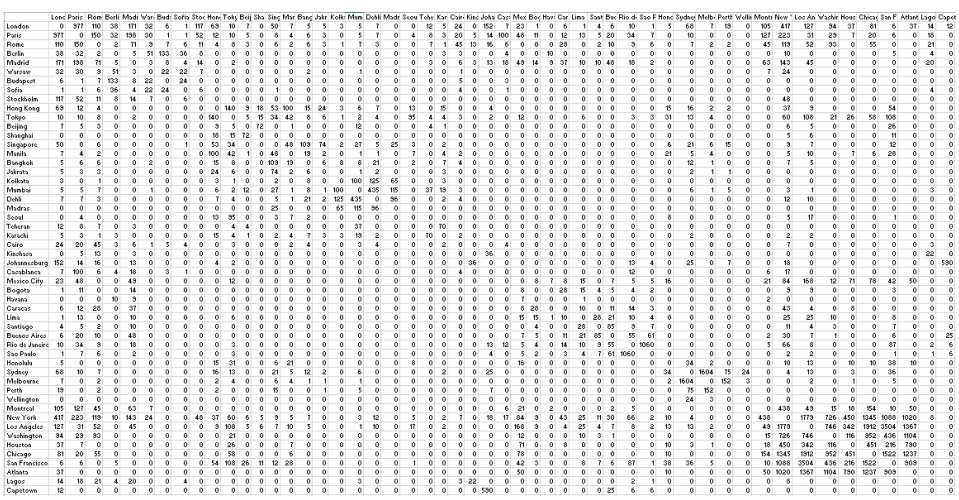
This section will describe what parts of the model for global spread of influenza needs to be validated, how we intend to test the different parts, and will present the results attained by these tests.

4.1.1 Data

To validate the model for global spread of influenza, we decided to try and replicate the results of Longini and Rvachev [1]. If we could reproduce the results for the global spread of influenza, using the 1968 air traffic volumes as our source, this would give a strong indication that our implementation of the model would be correct.

To reproduce the results Rvachev and Longini produced, we would have to use the same data as was originally used in their article. We therefore copied the transportation matrix from their work, and implemented the model with this matrix.

The model is implemented with the same 52 cities as used in the original article, and with the same population count as used in the original paper. The transportation matrix contains the same air traffic volumes as used in their work, to give the same prerequisites when implementing the model.



	Lond	Paris	Rom	Berli	Madr	Wars	Budp	Sofia	Stoc	Hels	Tokyo	Beij	Sha	Sing	Mal	Bang	Jakr	Kuala	Mum	Delhi	Mad	Seou	Toky	Kar	Cairo	Kisa	Joha	Cape	Mex	Bog	Hon	Cur	Lima	Sant	Bue	Rio d	Sao P	Hous	Sydn	Melb	Perth	Wellin	Mont	New Y	Los An	Washi	Hous	Chica	San F	Atlant	Lago	Capet
London	0	377	110	38	171	32	6	1	117	63	10	7	0	50	7	5	5	3	5	7	0	0	12	5	24	0	152	7	23	1	0	6	1	4	6	10	1	5	68	7	19	0	105	417	127	34	37	61	6	37	14	12
Paris	377	0	150	32	139	30	1	1	52	12	10	5	0	6	4	6	3	0	5	7	0	4	8	3	20	5	14	100	46	11	0	12	15	5	20	34	7	0	10	0	0	0	127	223	31	23	7	20	6	0	16	0
Rome	110	150	0	2	71	3	7	6	11	4	0	3	0	6	2	6	3	1	7	3	0	0	7	1	45	13	16	6	0	0	0	28	0	2	10	9	6	0	7	2	2	0	45	119	52	13	0	15	0	0	21	0
Berlin	38	32	2	0	5	51	133	36	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	0	0	4	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	5	0	4	0
Madrid	171	139	71	5	0	3	8	4	14	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	6	3	13	18	43	14	3	37	10	10	46	10	2	0	0	0	0	63	143	45	0	0	0	0	20	0
Warsaw	32	30	9	51	3	0	22	22	7	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	24	0	0	0	0	0	0	0
Budapest	6	1	7	133	8	22	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sofia	1	1	6	36	4	22	24	0	5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	4	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0
Stockholm	117	52	11	8	14	7	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0
Hong Kong	69	12	4	0	0	0	0	0	0	140	9	16	53	100	15	24	3	6	7	0	13	0	15	0	4	0	0	0	0	0	0	0	0	0	0	0	0	15	16	2	2	0	0	37	9	0	0	0	54	0	0	0
Tokyo	10	10	6	0	2	0	0	0	0	140	0	5	15	34	42	8	6	1	2	4	0	35	4	4	3	0	2	0	12	0	0	0	6	0	0	3	3	31	13	4	0	0	0	60	108	21	26	58	108	0	0	0
Beijing	7	5	3	0	0	0	0	0	0	3	5	0	72	0	1	0	0	0	12	0	0	0	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	5	0	0	0	26	0	0	0
Shanghai	0	0	0	0	0	0	0	0	0	18	15	72	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	6	0	0	0	11	0	0	0
Singapore	50	8	6	0	0	0	1	0	53	34	0	0	0	46	109	74	2	27	5	25	3	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	21	6	15	0	0	3	7	0	0	0	12	0	0	0
Manila	7	4	2	0	0	0	0	0	100	42	1	0	48	0	13	2	0	1	1	0	7	0	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	21	5	4	0	0	0	5	10	0	7	6	28	0	0	0
Bangkok	5	6	6	0	0	2	0	0	15	8	0	0	109	19	0	6	8	8	21	0	2	0	7	4	0	0	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	7	5	0	0	0	0	0	0	0
Jakrata	5	3	3	0	0	0	0	0	24	6	0	0	74	2	6	0	0	1	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1	1	0	0	0	0	0	0	0	0	0	0	0	0
Kolkata	3	0	1	0	0	0	0	0	3	1	0	0	2	0	8	0	0	100	125	65	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mumbai	5	5	7	0	0	1	0	0	0	6	2	12	0	27	1	8	1	100	0	435	115	0	37	19	3	0	0	0	0	0	0	0	0	0	0	0	0	6	1	5	0	0	3	1	0	0	0	0	0	0	3	0
Dubli	7	7	3	0	0	0	0	0	7	4	0	5	1	21	2	125	435	0	35	0	1	0	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	10	0	0	0	0	0	0	0
Madras	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0	0	65	115	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Seoul	0	4	0	0	0	0	0	0	0	13	35	0	0	3	7	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	5	17	0	0	0	1	0	0	0
Tokyo	12	8	7	0	3	0	0	0	0	4	4	0	0	0	0	0	0	0	37	0	0	0	0	0	70	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Karachi	5	3	1	3	0	0	0	0	15	4	1	0	2	4	7	3	3	19	2	0	0	70	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	2	2	0	0	0	0	0	0	0
Cairo	24	20	45	3	6	1	5	4	0	0	3	0	0	0	2	4	0	0	3	4	0	0	0	2	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	3	0
Kinshasa	0	0	5	13	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	0
Johannesburg	152	14	16	0	13	0	0	0	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	13	4	0	25	0	7	0	0	18	0	0	0	0	0	0	0	590
Casablanca	7	100	6	4	18	0	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	6	17	0	0	0	0	0	0	0	0
Mexico City	23	48	0	0	43	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	5	16	0	0	0	0	21	84	168	12	71	78	42	50	0	0
Bogota	1	11	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	28	15	4	5	4	2	0	0	0	0	0	0	0	0	0	0	9	3	0	0	0	3	0	0	0
Havana	0	0	0	10	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0
Caracas	6	12	28	0	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0	0	10	0	11	14	3	0	0	0	0	0	43	4	0	0	8	0	0	0	0	0	0	0	0	0
Lima	1	13	0	0	10	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	15	1	10	0	28	21	10	4	0	0	0	0	0	25	25	10	0	0	8	0	0	0	0	0	0	0	0
Santiago	4	5	2	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	28	0	65	9	7	0	0	0	0	0	0	0	11	4	3	0	0	7	0	0	0	0
Buenos Aires	6	20	10	0	48	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	5	0	11	21	85	0	55	61	0	0	0	0	0	2	30	7	1	0	0	6	0	0	6	0	0	25	0
Rio de Janeiro	10	34	9	0	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	12	5	4	0	14	10	9	55	0	1060	0	0	0	0	0	5	66	8	0	0	0	87	0	2	6	0	0
Sao Paulo	1	7	6	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	5	2	0	3	4	7	61	1060	0	0	0	0	0	2	2	0	0	0	1	0	1	6
Honolulu	5	0	0	0	0	0	0	0	15	31	0	0	0	6	21	0	0	0	0	0	0	0	8	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0	54	2	0	0	0	10	13	0	10	10	38	10	0	0
Sydney	68	10	7	0	0	0	0	0	16	13	0	0	21	5	12	2	0	6	0	0	0	0	2	0	0	25	0	0	0	0	0	0	0	0	0	0	0	34	0	1604	15	24	0	4	13	0	3	0	36	0	0	0
Melbourne	7	0	2	0	0	0	0	0	2	1	0	0	5	4	3	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1604	0	152	3	0	0	2	0	1	0	5	0	0
Perth	13	0	2	0	0	0	0	0	2	0	0	0	15	0	0	1	0	5	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	15	152	0	0	0	0	0	0	0	0	0	0	0	0	0
Wellington	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	3	0	0	0	0	0	0	0	0	0	0	0	0
Montreal	105	127	45	0	63	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	21	0	2	0	0	0	2	5	0	0	0	0	0	0	0	0	0	0	439	43	15	18	154	90	50	0	0
New York	417	223	113	90	143	24	0	0	48	37	60	6	5	3	5	7	0	0	3	12	0	5	0	2	7	0	18	17	84	3	0	43	25	11	30	66	2	10	4	0	0	0	438	0	1773	726	450	1345	1086	1020	8	0
Los Angeles	127	71	52	0	45	0	0	0	0	9	105	5	6	7	10	5	0	0	1	10	0	17	0	2	0	0	0	0	168	9	0	4	25	4	7	8	2	13	13	2	0	0	49	1779	0	746	342	1912	3504	1967	0	0
Washington	94	29	53	0	0	0	0	0	0	0	21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	10	3	1	0	0	0	0	0	0	0	15	726	746	0	116	952	436	1104	0	0
Houston	37	7	0	0	0	0	0	0	0	26	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	71	0	0	8	0	0	0	0	0	10	3	1	0	0	18	450	342	116	0	451	216	790	0	0
Chicago	81	20	55	0	0	0	0	0	0	58	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	78	0	0	0	0	0	0	0	0	10	0	0	0	0	154	1345	1912	952	451	0	1522	1237	0	0
San Francisco	6	6	0	5	0	0	0	0	0	14	108	24	11	12	28	0	0	0	0	0	0	1	0	0	0	0	0	0	42	3	0	0	8	7	6	87	1	38	36	5	0	0	10	1093	3504	436	216	1522	0	1613	0	0
Atlanta	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	10	0	0	0	0	50	1020	1367	1104	730	1237	1613	0	0	0
Lagos	14	16	21	4	20	0	0	4	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	3	22	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0
Capetown	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	590	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.1: The original transportation matrix reproduced from Rvachev and Longini's work [1].

Grais, Ellis, and Glass.

In addition to using the data and results from Rvachev and Longinis original work, we decided to use one other source as validation of a correctly implemented model. The work of Grais, Ellis, and Glass [16] utilizes the same influenza model with air traffic volumes and city population from the year 2000. By substituting the transportation matrix from Rvachev and Longini's work with the transportation matrix from the work of Grais, Ellis, and Glass, we have two different points of reference to compare against. Confirming the model against two different sources gives an even stronger indication that the model is correctly implemented.

Because the work contains air traffic volumes from the year 2000, we can also use their results as a point of reference when examining if the system is able to predict the spread of influenza using only the currently implemented data sources for air traffic volumes. To validate this, the system will run a forecast with the transportation matrix generated by the system itself using the T-100 market database[1] from the year 2000 as the only data source. The resulting forecast will be compared with the forecast of Grais, Ellis, and Glass, which used many different sources for air traffic volumes, including the T-100 market database [16].

Key values

In addition to data about the different cities, the model uses a set of infection distribution probabilities which are taken from the original paper. These key values, such as maximum length of incubation and infection period, daily infectious contact rate, fraction of suscep-

[1] `http://www.transtats.bts.gov/databaseinfo.asp?DB_ID=111`

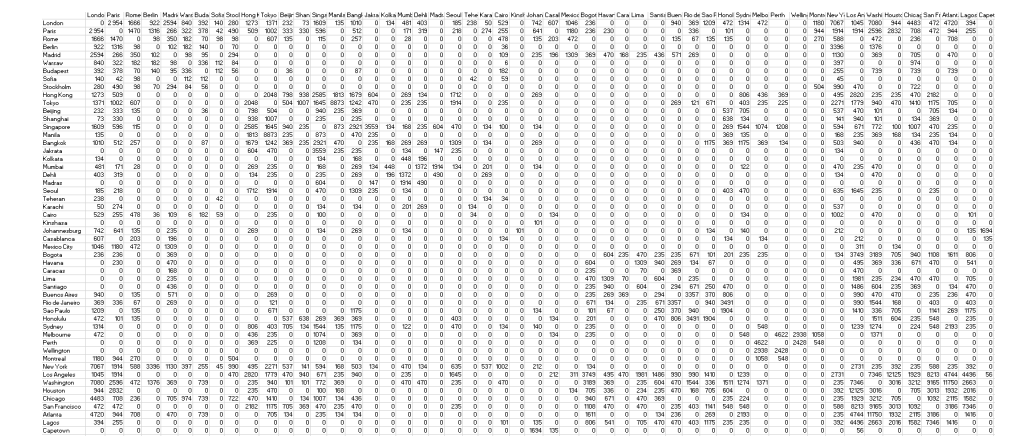


Figure 4.2: The transportation matrix used in the work of Grais, Ellis, and Glass, to model the spread of influenza in 2000. [16]

tible population and fraction of newly ill individuals reported are all taken directly from the work of Rvachev and Longini. This reduces the risk of miscalculating any values, and gives the best basis for validating our implementation of the model. We are ensuring that the system has implemented the correct values for these key values by unit testing.

4.1.2 Validation

Rvachev and Longini [1] presents their results as a forecast of predicted daily reported morbidity compared with observed daily reported morbidity. In order to validate that our system implements the model proposed by Rvachev and Longini correctly, we compare the results of our predicted daily reported morbidity, with the predicted daily morbidity of the original article. If we are able to create the same forecast with the same input data, that's a strong indication that we've implemented the model correctly.

Gathering results

We decided that the best way to compare the results with our own system's results, would be to produce a forecast on a similar format. We did this programmatically, and ran the forecast over 440 days, as done in the original article. The computer then divided daily morbidity into four day intervals, and plotted the results. We then took a picture of these results, and fitted them to the same size and matrix as in the original paper. From there, we manually plotted the results with photo editing software, in color coding. This was done so we could easily determine whether or not our results are correlating with Rvachev and Longini's results.

When comparing our system with the results from Grais, Ellis, and Glass [16], we decided to calculate the temporal progression of the forecast in the same way as done in their paper. This means calculating the number of days since the epidemic outbreak each city expe-

riences its peak influenza activity. The results can then be used to calculate correlation between the two resulting forecasts.

After comparing our results with that of Grais, Ellis, and Glass for the temporal progression of the forecast using 1968 air traffic volumes, we found that the results weren't correlating as much as we had expected. We therefore decided to manually create a temporal progression of the results given in Rvachev and Longini's work. This was done to ensure that the results from Grais, Ellis, and Glass were in fact reliable, as there are many sources of errors when trying to recreate the results. This would also give us another point of comparison to further validate our implementation, giving us a numerical value of the statistical correlation between the forecast generated from our system, and the original forecast done by Rvachev and Longini [1].

4.1.3 Results

The original forecast from Rvachev and Longini's work can be seen in the appendix in figures A.1 and A.2, while our systems forecast can be seen in figures A.3 and A.4. Figures A.5 and A.6 show our results superimposed on to the original results, to highlight the correlation of the two forecasts.

Rvachev and Longini's forecast show that the pandemic spreads from Hong Kong to the neighboring cities first, then progresses to the northern hemisphere and eventually reaches the southern hemisphere. From the results it is easy to see that our forecast is very similar to the forecast produced in the original article. It seems our forecast is able to predict the epidemic peak correctly, with an error of around 2 weeks, for most of the 52 cities on the list. The forecast displays a notable lack of correlation for the four cities: Sydney, Melbourne, Perth, and Wellington, as it predicts that Wellington would not experience influenza activity at all, and the three cities in Australia would reach their epidemic peak 272 days before the forecast of Rvachev and Longini.

Grais, Ellis, and Glass also runs their implementation of Rvachev and Longini's model with air traffic volumes from 1968. The results are presented as a temporal progression of the forecast of peak epidemic activity. Comparing the temporal progression forecast in the work of Grais, Ellis, and Glass [16], with our forecast, and a manually created temporal progression from the results of Rvachev and Longini, the results can be seen in table 4.1.

City	Rvachev and Longini	Grais, Ellis, and Glass	Solstad and Almvik
Hong Kong	0	0	0
Manila	37	4	35
Singapore	44	15	41
Jakarta	57	35	51
Sydney	348	300	74
Bangkok	80	62	75
Honolulu	87	65	78
Madras	94	81	88
Mumbai	97	83	91
Melbourne	373	340	105
Perth	381	360	105
Dehli	125	112	118
Kolkata	132	135	123
Karachi	155	137	142
San Francisco	132	115	142
London	182	121	143
Tokyo	136	121	145
Los Angeles	155	125	148
Teheran	144	130	153
Mexico City	177	175	154
New York	163	145	156
Atlanta	157	130	161
Chicago	159	140	166
Washington	159	138	167
Houston	159	140	168
Paris	191	150	171
Montreal	181	160	174
Stockholm	202	145	175
Madrid	195	160	176
Rome	188	160	178
Rio de Janeiro	204	190	180
Seoul	178	163	185
Warsaw	214	172	186
Berlin	226	167	187
Lima	200	195	189
Beijing	188	175	191
Cairo	230	172	191
Caracas	200	197	191
Shanghai	188	180	194
Bogota	238	210	199
Casablanca	224	190	201
Sao Paulo	234	218	206
Lagos	224	180	210
Budapest	245	200	222
Sofia	250	210	222
Kinshasa	245	213	224
Johannesburg	303	218	320
Buenos Aires	377	285	328
Capetown	320	287	341
Santiago	416	340	367
Wellington	385	360	0
Havana	0	202	0

Table 4.1: Comparing the dates for epidemic peak activity with air traffic volumes from 1968 [16, p. 1068].

The left column in table 4.1 shows the temporal progression produced from Rvachev and Longini's work. The middle column is the temporal progression produced in the work of Grais, Ellis, and Glass [16], and the right column is the temporal progression from our own system. Table 4.2 show the respective statistical correlations between the forecasts. The table includes a calculation done with the most deviating results from both our own forecast and from Grais, Ellis, and Glass' forecast.

Comparison	All cities	Without Oceania and Havana
Grais, Ellis, and Glass - Rvachev and Longini	0.919	0.971
Solstad and Almvik - Rvachev and Longini	0.587	0.978
Solstad and Almvik - Grais, Ellis, and Glass	0.447	0.966

Table 4.2: Calculating the correlation between the results of Rvachev and Longini, Grais, Ellis, and Glass, and ourselves. The rightmost column is calculated correlation between the data when removing Sydney, Melbourne, Perth, Wellington, and Havana from the data sets.

The average difference in peak day activity for each city between Rvachev and Longini's forecast and Grais, Ellis, and Glass' forecast is 32.327 and drops to 29.0 when excluding Havana from the forecast. The average difference in peak day activity for each city between Rvachev and Longinis forecast and our own forecast is 37.731 days and drops to 15.813 days when excluding Wellington, Sydney, Perth, and Melbourne. The average difference in peak day activity between the forecast of Grais, Ellis, and Glass and our own forecast is 43.1, and drops to 20.49 when excluding Havana, Wellington, Melbourne, Perth, and Sydney from the forecast. The differences are summarized in table 4.3

Difference	All cities	Without Oceania and Havana
Grais, Ellis, and Glass - Rvachev and Longini	32.327	29.0
Solstad and Almvik - Rvachev and Longini	37.731	15.813
Solstad and Almvik - Grais, Ellis, and Glass	43.1	20.49

Table 4.3: The average difference in peak day activity for each city. The rightmost column is average difference when removing Sydney, Melbourne, Perth, Wellington, and Havana from the data sets.

City	Grais, Ellis, and Glass	Solstad and Almvik
Hong Kong	0	0
Singapore	36	21
Sydney	36	30
Johannesburg	36	39
Melbourne	36	33
Wellington	36	41
Perth	36	29
Bangkok	40	26
Manila	40	25
Jakarta	48	33
Capetown	48	73
Mumbai	56	38
Honolulu	60	45
Madras	68	50
Sao-Paulo	76	58
Kolkata	84	61
Mexico City	84	153
Santiago	92	100
Lagos	92	62
Bogota	112	123
Caracas	120	127
Lima	120	91
Rio De Janeiro	120	69
Havana	120	88
Atlanta	124	124
Washington	132	122
New York	132	131
Chicago	132	127
Houston	132	121
San Francisco	132	122
Los Angeles	132	124
Delhi	132	40
London	134	126
Buenos Aires	140	105
Rome	140	129
Montreal	140	129
Madrid	140	130
Karachi	144	58
Seoul	144	137
Paris	144	128
Beijing	148	132
Tokyo	148	137
Shanghai	148	132
Cairo	152	141
Budapest	152	141
Berlin	156	145
Stockholm	156	145
Warsaw	160	148
Casablanca	165	130
Sofia	168	158
Teheran	180	140
Kinshasa	0	77

Table 4.4: Comparing the dates for epidemic peak activity with air traffic volumes from the year 2000 [16].

Table 4.4 shows the same temporal progression but from a forecast using air traffic volumes from the year 2000. The calculated correlation between the two datasets is 0.848 and an average difference in peak day activity at 19.6 days.

City	Grais, Ellis, and Glass	Solstad & Almvik	City
Hong Kong	0	0	Hong Kong
Singapore	0	0	Singapore
Sydney	-1	0	Manila
Johannesburg	-1	0	Bangkok
Melbourne	-1	-1	Perth
Wellington	-1	-1	Sydney
Perth	-1	-1	Melbourne
Bangkok	0	0	Jakarta
Manila	0	0	Mumbai
Jakarta	0	-1	Johannesburg
Capetown	-1	0	Delhi
Mumbai	0	-1	Wellington
Honolulu	0	0	Honolulu
Madras	0	0	Madras
Sao-Paulo	0	0	Karachi
Kolkata	0	0	Sao Paulo
Mexico City	1	0	Kolkata
Santiago	-1	0	Lagos
Lagos	0	0	Rio De Janeiro
Bogota	1	-1	Cape Town
Caracas	0	0	Kinshasa
Lima	0	0	Havana
Rio De Janeiro	0	0	Lima
Havana	0	-1	Santiago
Atlanta	1	-1	Buenos Aires
Washington	1	1	Houston
New York	1	1	San Francisco
Chicago	1	1	Washington
Houston	1	1	Bogota
San Francisco	1	1	Atlanta
Los Angeles	1	1	Los Angeles
Delhi	0	1	London
London	1	0	Caracas
Buenos Aires	-1	1	Chicago
Rome	1	1	Paris
Montreal	1	1	Montreal
Madrid	1	1	Rome
Karachi	0	1	Casablanca
Seoul	1	1	Madrid
Paris	1	1	New York
Beijing	1	1	Beijing
Tokyo	1	1	Shanghai
Shanghai	1	1	Seoul
Cairo	1	1	Tokyo
Budapest	1	1	Teheran
Berlin	1	1	Budapest
Stockholm	1	1	Cairo
Warsaw	1	1	Berlin
Casablanca	1	1	Stockholm
Sofia	1	1	Warsaw
Teheran	1	1	Mexico City
Kinshasa	0	1	Sofia

Table 4.5: Comparing the temporal progressions of both Grais, Ellis, and Glass' forecast and our own forecast using air traffic volumes from 2000. The table shows which zone each city is located in, and both are ordered from earliest to latest epidemic peak day, to highlight the geographical spread and seasonal characteristics of both forecasts.

Forecast using only the systems source for air traffic volumes

When running our system with the T-100 market database as the only source of air traffic volumes, the model is not able to spread influenza when Hong Kong is the epidemic index city. The system predicts a 60 day period of influenza activity in Hong Kong before the epidemic dies out. It's notable that Los Angeles, New York, Chicago, and San Francisco experience influenza activity, but the daily predicted morbidity never rises above 1.

4.2 Machine learning

This section will look at the tests and results done during the procedure described in section 3.4. We attempted to implement a machine learning module for tweet collection, and the results can be found here.

4.2.1 Data

As mentioned in chapter 3, Lamb, Paul and Dredze [12] has provided two datasets from 2009 and 2012 of, respectively, 2366 and 4760 tweets, where each tweet is annotated with a 1 if it is influenza related, and 0 if it is not. Each tweet in the dataset is represented by a tweet ID, and not the actual text posted in the tweet, because of Twitter's privacy policy in regards to publishing personal data. Because of this, we manually looked up each tweet ID in Twitter's API, and the resulting dataset was approximately 72% of the original, amounting to approximately 5130 total available tweets.

4.2.2 Validation

To be able to properly test the accuracy of the classifiers, one should use approximately half of the annotated dataset. This way, we can train the classifier with one half, and use the other half for testing whether or not the classifier is functioning as expected. However, because of the relatively small dataset we had access to, we found that there was no point to test the classifier on a dataset. We could easily see, just by looking briefly at the results, that the classifier had a hard time distinguishing an actual influenza incidence and casual influenza mentions. In order to get a general idea of how the classifiers performed, we found the most informative features of each of the classifiers. This way, we could at least see which words the classifiers emphasized more, and see if the classifiers at least were on the right track.

If we had access to a larger dataset, we could perform tests on a part of it, so that we could find the specific accuracy of each classifier type, and compare them against each other.

4.2.3 Results

Below follows a list of the ten most informative features for each of the classifier implementations. This should give an indication of how effective the classifiers have been in finding the most important words.

Naive Bayes

Related (1)/Not related (0)

```
contains(throat) = True           1 : 0    =    22.3 : 1.0
contains(feeling) = True          1 : 0    =    19.0 : 1.0
contains(im getting) = True       1 : 0    =    18.0 : 1.0
contains(flu news) = True         0 : 1    =    17.4 : 1.0
contains(guide) = True            0 : 1    =    16.6 : 1.0
contains(pregnant women) = True   0 : 1    =    16.2 : 1.0
contains(zombie) = True           0 : 1    =    15.8 : 1.0
contains(flu spreading) = True    0 : 1    =    14.3 : 1.0
contains(coughing) = True         1 : 0    =    14.0 : 1.0
contains(women) = True            0 : 1    =    12.6 : 1.0
```

Awareness (1)/Infection (0)

```
contains(vote) = True             1 : 0    =    26.4 : 1.0
contains(flu vaccine) = True      1 : 0    =    25.1 : 1.0
contains(seasonal flu) = True     1 : 0    =    17.7 : 1.0
contains(vaccine) = True          1 : 0    =    16.3 : 1.0
contains(cow) = True              1 : 0    =    15.7 : 1.0
contains(finally getting) = True  0 : 1    =    15.0 : 1.0
contains(media) = True            1 : 0    =    12.4 : 1.0
contains(seasonal) = True         1 : 0    =    12.2 : 1.0
contains(winter) = True           1 : 0    =    11.0 : 1.0
contains(shots) = True            1 : 0    =    10.4 : 1.0
```

Here, the rightmost numbers are ratios of how often a feature is influenza related or not, as signified by `1 : 0` or `0 : 1` to the left of it. For example, `contains(throat)1 : 0 = 22.3 : 1.0` means that the word 'throat' appears 22.3 times as often than not in tweets that are influenza related.

We see that the related/not related-classifier certainly is on the right track. When actually testing it against tweets gathered real-time from Twitter, however, we generally see that the classifier has problems distinguishing influenza related tweets.

Maximum entropy

Related (1)/Not related (0)

```
-0.003 contains(throat)==True and label is '0'
-0.003 contains(im getting)==True and label is '0'
-0.003 contains(feeling)==True and label is '0'
-0.003 contains(flu news)==True and label is '1'
-0.003 contains(coughing)==True and label is '0'
-0.003 contains(guide)==True and label is '1'
-0.003 contains(zombie)==True and label is '1'
```

```
-0.003 contains(stupid)==True and label is '0'
-0.003 contains(pregnant women)==True and label is '1'
-0.002 contains(hoping)==True and label is '0'
```

Awareness (1)/Infection (0)

```
-0.003 contains(vote)==True and label is '0'
-0.003 contains(flu vaccine)==True and label is '0'
-0.003 contains(seasonal flu)==True and label is '0'
-0.002 contains(cow)==True and label is '0'
-0.002 contains(finally getting)==True and label is '1'
-0.002 contains(media)==True and label is '0'
-0.002 contains(vaccine)==True and label is '0'
-0.002 contains(winter)==True and label is '0'
-0.002 contains(seasonal)==True and label is '0'
-0.002 contains(risk)==True and label is '0'
```

As one can see here, the classifiers have some problems distinguishing important features. The ten most informative features in both cases are within 0.001 significance of each other, and are generally low. This means that the classifier will place a lot of the tweets together, and will have problems separating the positive and the negative tweets.

Generally, tweets often contains informal language, and are prone to grammatical errors. Words like 'sick' and 'cough' are often used outside of describing illness, for example 'Sick of crying. tired of trying. yes, i'm smiling. but inside i'm dying' contains the word 'sick', but is completely irrelevant to us. This is often hard for the classifiers to comprehend, seeing as an occurrence of the word 'sick' usually is a good indicator of a sick individual.

4.3 Tweets

To gather tweets, we first attempted a raw twitter search as explained in section 3.3.1, then tried utilizing machine learning techniques on the raw data as explained in section 3.4 and 4.2.

Due to mediocre at best results from the raw twitter search and machine learning module, as well as a large update to Flutrack, which increased the amount of influenza related tweets by tenfolds, we decided to use the tweet data we get from Flutrack as the basis for the rest of our research. The available API usually provides between 3000 and 4000 tweets for each week back in time, up to 60 days.

4.3.1 Results

Although the tweets returned from Flutrack are mostly good and of relevancy, there are some irrelevant noise as well. At the time of writing this, one particular tweet is circulating

in Thailand, which seems to be an English translation of common symptoms like 'Cough' and 'Sneeze'. A screenshot of Flutrack with this tweet can be seen in figure 4.3. Almost all the markers in the displayed area are repetitions of this one tweet.

Figure 4.3: Screenshot snippet from `Flutrack.org`, where Thailand can be seen to have a lot of retweets of the same English translation of common symptoms.

The system will use the tweets gathered to detect a potential epidemic, which is done by looking at increasing trends of influenza related tweets in each city. Flutrack provides an API which allows collecting the tweets gathered up to 60 days in the past. This gives us data from approximately the latest 8 weeks, which lets us compare the trends of each city.

We saw that at a point where we collected a total of 23767 tweets from 60 days back in time from the Flutrack API, 3981 of them originated in one of the 52 cities, which amounts to approximately 17%. We find that this is sufficient for looking at increases or decreases in the amount of tweets from week to week. A snippet from the collected tweets can be seen in figure 4.4. In this case, Atlanta has an increasing trend the last 3 weeks, which is why it has been marked with yellow.

Cities	7 weeks ago	6 weeks ago	5 weeks ago	4 weeks ago	3 weeks ago	2 weeks ago	1 week ago	Current week
Atlanta	24	12	4	40	12	4	34	44
Bangkok	256	14	8	0	4	0	22	72
Beijing	6	0	0	0	0	0	0	0

Figure 4.4: Snippet of screenshot from the system. Each city has twitter data from 60 days in the past, and the system looks at influenza trends.

4.4 Airports

The module uses SITA's Airport API[2] to look up all airports for each city, and categorizes these airports together. This means that the system is able to automatically identify which airports are associated with any new city added to the model, and is then able to acquire all relevant passenger statistics.

To test if the system is actually able to identify and gather all airport codes associated with a city, we generated a random test dataset consisting of some cities and their associated airports. We then created unit tests to check whether or not the system is in fact able to identify, and extract the correct airports for all cities.

4.4.1 Data

We are interested in looking at travel data between cities, not between airports. The airport module therefore searches through the passenger travel statistics database and adds up passenger travel between all airports between two cities. This allows the system to query how many passengers have flown from Los Angeles to New York City. The result will then be all passengers from LAX (Los Angeles International Airport) to JFK (John F. Kennedy International Airport), and all passengers from LAX to LGA (LaGuardia Airport).

To test if the system is gathering the correct statistics, we have designed a test dataset and are running unit tests to check that the system is acquiring correct values. The test dataset is created by manually searching through the database, and calculating how many passengers should travel between two cities, and then comparing these results with what the system generates.

4.4.2 Validation

To validate that the system is able to find the correct airports from the cities in the list, it is sufficient to manually check that each airport for each city is added. This is done with unit testing, where we generate a test dataset for a selection of cities, and confirming that the module is finding all belonging airports.

4.4.3 Results

The end result is a list of all the airports in the world stored in the database. This is handled by the system, which converts it into a list of all the airports within each city in the city matrix. Figure 4.5 represents the resulting dictionary after our system collects all airports associated with all cities

[2] `https://www.developer.aero/Airport-API/API-Overview`

```
Sao Paulo      ['CGH', 'GRU']
Delhi          ['DEL']
Havana         ['HAV']
Madras         ['MAA']
Perth          ['PER']
Montreal       ['YHU', 'YMX', 'YUL']
Buenos Aires   ['EZE', 'AEP']
Sofia          ['SOF']
Sydney         ['SYD', 'YQY']
Johannesburg   ['JNB']
Singapore      ['SIN', 'XSP']
Los Angeles    ['LAX']
Karachi        ['KHI']
Tokyo          ['NRT', 'HND']
Atlanta        ['ATL']
Honolulu       ['HNL']
Kinshasa       ['FIH']
Rio De Janeiro ['SDU', 'GIG']
Seoul          ['GMP', 'ICN']
Mumbai         ['BOM']
Santiago       ['SCQ', 'STI', 'SCL']
Melbourne      ['MEL', 'MLB']
Chicago        ['MDW', 'ORD']
Stockholm      ['BMA', 'NYO', 'ARN']
Bogota         ['BOG']
London         ['STN', 'LCY', 'LGW', 'LHR', 'LTN', 'YXU']
Casablanca     ['CMN']
Beijing        ['PEK']
Lagos          ['LOS']
Cape Town      ['CPT']
Cairo          ['CAI']
Caracas        ['CCS']
Teheran        ['THR']
Hong Kong      ['HKG']
Washington     ['DCA', 'IAD']
Lima           ['LIM']
Manila         ['MNL']
Wellington     ['WLG']
Houston        ['HOU', 'IAH']
Mexico City    ['MEX']
Budapest       ['BUD']
San Francisco  ['SFO']
Berlin         ['SXF', 'TXL']
Jakarta        ['CGK']
Shanghai       ['SHA', 'PVG']
Warsaw         ['WAW']
Rome           ['CIA', 'FCO']
New York       ['TSS', 'LGA', 'JFK', 'JRB']
Bangkok        ['BKK']
Madrid         ['MAD']
Paris          ['CDG', 'LBG', 'ORY']
Kolkata        ['CCU']
```

Figure 4.5: A snippet of the formatted city list with associated airports used by our system. Data is collected from SITA's Airport API.

4.5 Current system

The final result of this thesis is an open-source Django REST API, as well as a simple Angular front-end to visualize the data. This section will quickly describe the system and it's capabilities.

The REST API is dependent on a lot of external sources of data. It uses flight data from the T-100 market database, tweets from api.flutrack.org, and updated airport codes from SITA's airport API. It has to query these sources at some frequency, so its data is up to date. As of now, this has to be done manually, because of Heroku's restrictions on the free dynos. Optimally, this task should be automated through a scheduled worker dyno, which can run the collection tasks one time per day. The API will detect when an epidemic breaks

out in a single city. This is done by comparing week to week data, and seeing if there's an abnormally large increase in the amount of influenza related tweets. When this happens, the system will calculate the forecast, and update the database. This is then returned to the users on request, together with the week to week tweet data.

The front-end is a simple one-page interface to the API. It features a map of the calculated forecast, a temporal slider which updates the map accordingly, as well as a table of the week to week tweet data. All the underlying data is gathered from the API on load.

Chapter 5

Discussion

This chapter will discuss the findings and results from the previous chapters. It will present explanations for why the results appear as they do, and reasons why the results might differ from what was initially expected.

5.1 Prediction model

The results presented in the previous section clearly show that our implementation of the model produces a very similar forecast of the global spread of influenza as that of Rvachev and Longini. Figure A.5 and A.6 show how our results correlate with the expected results. The influenza activity occurs at roughly the same time period for each of the 52 cities in the model, with the exception of the four cities located in Oceania: Sydney, Melbourne, Perth, and Wellington.

Table 4.1 shows that there is a big lack of correlation between the temporal progressions of the forecast produced from our results, and the forecast produced by Grais, Ellis, and Glass using air traffic volumes from 1968. This led to us creating a temporal progression of the forecast of Rvachev and Longini, to examine whether the results of Grais, Ellis, and Glass correlated with the original report. This also gave us another way of confirming that our forecast correlated with that of Rvachev and Longini in a more precise way.

When comparing our results with the results of Grais, Ellis, and Glass using air traffic volumes from the year 2000, we clearly see that there is a much stronger correlation between the two forecasts. Table 4.4 shows the results.

The correlation between the temporal progression datasets indicate to which extent the influenza spreads to the same areas in the same order. A strong correlation factor indicates that influenza progresses to the correct cities in roughly the correct order, at roughly the correct time, and a weak correlation factor indicates the opposite. This correlation might be strong, even if there are big differences in peak influenza activity.

5.1.1 1968 Hong Kong influenza

From looking at the comparison of our results on top of the results of Rvachev and Longini in figure A.5 and A.6, it is clear that the forecasts correlate to some extent. Peak influenza activity is reached at roughly the same time for both forecasts for all cities with the exception of the four cities in Oceania. The average difference in peak influenza activity for each city is 37.731 days, and the correlation between the temporal progression of the forecasts is 0.587.

Differences

The most obvious differences are of Sydney, Melbourne, Wellington, and Perth. It seems that our forecast predicts Sydney, Melbourne, and Perth to experience peak influenza activity 272.67 days prematurely, and does not produce any influenza activity in Wellington. From figure A.2 we see that Sydney, Melbourne, and Wellington experience actual reported, as well as simulated influenza activity in July and August 1969, while Perth experiences only simulated influenza activity in the same time period. From figure A.4 we can see that our forecast simulates influenza activity for Sydney, Melbourne, and Perth in October and November 1968, and no activity in July and August 1969.

The correlation between the temporal progression of our forecast and that of Grais, Ellis, and Glass is 0.447, which we would not consider a promising result. This means that there is a weak positive linear relationship between the two forecasts. These results surprised us, after preliminary comparisons with Rvachev and Longini looked promising. One big reason for the lack of correlation is due to our simulation of influenza activity in Sydney, Perth, Melbourne, and Wellington. This does not however explain it all, as we would expect similar correlation between us and Grais, Ellis, & Glass and us and Rvachev & Longini if those four cities were the only major difference.

Examining the data from the forecast of Grais, Ellis, and Glass closer, we found some inconsistencies which led to us create a temporal progression of the original forecast in order to compare the two. Rvachev and Longini explicitly states in their paper that ”Only Havana was predicted not to experience influenza activity over the time period of the simulation” [1, p.17], while the temporal progression of the forecast of Grais, Ellis, and Glass shows that Havana reaches peak influenza activity 202 days after the epidemic outbreak in Hong Kong [16, p.1068]. This result suggest that something has been implemented incorrectly, or that the model implemented is perhaps updated in a way which does not immediately become apparent in their article. After comparing the two forecasts, we find that they have a correlation factor of 0.919, which is a good result, but not as good as one would expect when trying to replicate an experiment.

Looking at figure A.9 it seems as Wellington and Perth experience peak influenza activity after 360 days, which is quite similar to the forecast of Rvachev and Longini. However, when looking at figure A.11, it seems as though both Perth and Wellington are not experiencing influenza activity during the simulation. Changing peak influenza activity to 0 for both cities produces a correlation factor between Grais, Ellis, & Glass and Rvachev &

Longini of 0.603, which is much more in line with our results. Comparing the temporal progression of our forecast and that of Grais, Ellis, and Glass after changing peak influenza for Perth and Wellington to 0, the correlation increases to 0.70.

Similarities

Examining the results after excluding Sydney, Wellington, Perth, and Melbourne from the forecast, we get a different picture. The average difference in peak influenza activity decreases from 37.731 to 15.813 days, and the correlation between the temporal progression of our forecast and that of Rvachev and Longini rises from 0.587 to 0.978. This correlation is more in line with what we expected, as even though we are trying to replicate the results, there are many reasons for us not to expect 100% equal results.

When excluding these cities from the results, the correlation between our results and that of Grais, Ellis, and Glass also rises from 0.447 to 0.877. Taking it one step further, and removing the obviously erroneous simulation of Havana, the correlation rises to 0.966. The forecast of Grais, Ellis, and Glass also correlates with that of Rvachev and Longini more when removing Havana from the comparison. The correlation factor rises from 0.919 to 0.971.

When looking at our forecast, after removing the the most diverging results, the simulation of Sydney, Perth, Wellington, and Melbourne, we are able to predict roughly the same forecast as both Rvachev & Longini and Grais, Ellis, and Glass The rightmost column in table 4.2 show how each of the three forecasts correlate with each other in terms of temporal progression. This means that all three forecasts predict mostly the same geographic spread of global influenza with some exceptions.

5.1.2 2000 Hong Kong influenza

We have already shown that in large, the forecasts correlate when using air traffic volumes from 1968, and using the forecast from the work of Grais, Ellis, and Glass with updated air traffic volumes from the year 2000, we can confirm that the models correlate.

Differences

When confirming the forecast using air traffic volumes from 2000, the only available data for comparison is the temporal progression of the forecast. Looking at table 4.4, we can see that there are some obvious differences. Kinshasa is not experiencing any influenza activity in the forecast of Grais, Ellis, and Glass, while our system produces simulated peak influenza activity at 77 days after epidemic outbreak. The overall correlation between the forecasts are 0.848 and the average difference in peak influenza activity is 19.60 days. Removing Kinshasa from the forecast increases the correlation to 0.876 and the average difference decreases to 18.47 days. This means that the diverging result in Kinshasa doesn't impact the forecast too much.

Similarities

Table 4.5 show how both forecasts spread influenza based on geography and seasonality. Even though the forecasts does not spread influenza to the same cities at the same order, both forecasts show similarities in terms of geographic spread. The pandemic progresses from Hong Kong to the neighboring cities before moving on to Australia. Then the pandemic continues to spread to cities in the tropical zone, before eventually progressing to the northern hemisphere.

Overall, the model predicts a similar forecast, but not as similar as expected. There are many potential reasons for this. The model might have been implemented differently by Grais, Ellis, and Glass, since when comparing their forecast to that of Rvachev & Longini, we find some obvious unexpected differences. Taking this into account, one would expect there to be differences between our forecast and the forecast of Grais, Ellis, and Glass when using air traffic volumes from 2000 as well. There might also have been an error in recreating the data from either forecast. Grais, Ellis, and Glass might have an error in their copy of the 1968 transportation matrix, or we might have an error in our copy of the 1968 transportation matrix, or we might have an error in our copy of Grais, Ellis, and Glass' 2000 transportation matrix.

Using the system's generated air traffic volume

Running our system using only its own source of air traffic volumes from the year 2000, results in no mentionable spread of influenza activity outside Hong Kong. The reasons for this is apparent, as the T-100 market database only contains passenger statistics from flights with at least one airport located in the U.S. The transportation matrix for Hong Kong contains average daily passenger statistics for Atlanta, Chicago, Los Angeles, New York, San Francisco, and Washington. No other cities in the matrix has any daily passenger statistics to Hong Kong.

This is perhaps the biggest limitation to our system, as filling the transportation matrix with more data produces a somewhat reliable forecast. The extensibility of the system however, makes it easy for people in the open-source community, us included, to extend the data gathering capabilities of the system. Being able to fill the transportation matrix with different transportation statistics than air traffic, such as train travel statistics has proven feasible in the past. Flahault, Letrait, Hazout, Menares, and Valleron modelled the 1985 influenza epidemic in France using the same model as we have implemented, with train travel statistics between districts in France instead of air traffic volumes [17].

5.1.3 Error sources

Since the forecast of Rvachev and Longini is presented in four-day intervals, composing a temporal progression of peak epidemic activity becomes an estimation process. In figures A.1 and A.2 peak epidemic activity is represented by a cross. The way this data is presented, makes it difficult to pinpoint the exact number of days since epidemic peak activity

in Hong Kong, which is explicitly stated to be July 27th 1968 [1, p. 15].

Seasonality

As the four cities that experience substantially diverging results are all located on the southern hemisphere, and even more specifically in the Oceanic continent, there is reason to believe that there is something incorrect about the seasonal swinging factor. However, other cities in the southern hemisphere are not diverging from the original forecast. If there was an issue with the seasonal factor, we would see more cities in the southern hemisphere experience diverging results. Moreover, our results simulates influenza activity during October and November 1968, which would be during the spring time, early summer in the southern hemisphere, which should decrease influenza activity by scaling the daily infectious contact rate by 0.55 and 0.25 respectively.

5.2 Feasibility of predicting spread

The previous section show that we in broad terms have been able to replicate the results of Rvachev & Longini, and implement the model in such a way that allows for automated updating of necessary information. The model itself can predict the broad patterns of the geographical spread of the Hong Kong influenza [1, p. 20]. Key parameters such as length of incubation and infection period, daily infectious contact rate, and infection distribution could vary among different populations and different strains of influenza. Our model is implemented on the underlying assumption that a new influenza epidemic have the same epidemiological features as the Hong Kong virus (Influenza A subtype H3N1). Results may vary if the next influenza pandemic is of a new strain of influenza.

The model will in broad patterns predict the geographic spread of influenza if an epidemic is detected. As of now, the simulation of the forecast of our system must be run manually once per day. Ideally we would want to use the twitter data to recognize the outbreak of an epidemic in one of the cities that we are tracking, and then automatically run a new forecast with that city as the index city. We are using a method for looking at trends in the Twitter data to identify an epidemic as of right now. For the system to have a reliable algorithm of determining whether an epidemic has occurred or not, more research, and more data is required. Others have identified the outbreak of different types of epidemics with the use of social media, so this is not only feasible, but has already been implemented. In [33] Diaz-Aviles, Stewart, Velasco, Denecke, and Nejdl confirm that their method for detecting epidemics using social media, is faster than all other methods of reporting epidemics for the EHEC outbreak in Germany in May 2011.

The system created by Thapen, Simmie, Hankin, and Gillard [13] uses a different approach for forecasting influenza. Their goal is to predict how many tweets with different symptoms can be expected in the future, whereas ours have been predicting the geographic spread of influenza by computing daily reported incidence numbers. They are also using a different transportation network in that they are gathering all travel information from the tweets they are monitoring, using a user's different posting locations as evidence for travel

between the different locations. This means that they are using actual observed travel, while our system uses statistical travel data. Their results opens up the question if there are more ways of predicting disease outbreaks, and strengthens the hypothesis that it is feasible to predict spread of influenza based on social media.

5.3 Twitter and geolocation

In the previous chapter, we presented some results from the collection of tweets in the system. This section will discuss these results.

5.3.1 Tweets

Replies and retweets

Twitter is dependent on being a vivid community, with users sharing content and reacting to other users' content. This is also represented in influenza related tweets, as many tweets are either replies to other tweets or so-called *retweets*, a direct re-post of the original tweet. Because of these features, tweets with mentions of influenza does not necessarily need to concern the original poster. The example shown in section 4.3.1 is a good indication of this. In this case, a single tweet that features English translations of common symptoms is shared so many times that it seems like there is a big influenza outbreak in Bangkok and Thailand in general, although it does not indicate a single case of influenza infection. Usually, Twitter users with a lot of *followers*, other people who receives the user's tweets, have a tendency to have their tweets retweeted and discussed, which will create noise in the overall data. The example shown earlier clearly shows this type of behaviour where a single tweet is retweeted by many users. This can be hard to avoid if it is not dealt with during the original gathering of the tweets. Even though noise can be detrimental to the system, only approximately 6% of tweets are retweeted[1] and retweets should therefore not be a big problem for our system.

Demographics

Twitter is generally used all around the world, but is a lot more common in the western community, especially in USA. A report from 2009 and revised in 2014, which analyzed over 11 million tweets, reports that 62.14% of tweets are from USA and 16.36% are from the other English speaking countries UK, Canada and Australia[34]. This means that a total of 78.5% of tweets are from these four English speaking countries. This means that the constraint of only including English tweets is not a problem, since they cover the majority of the Twitter activity. However, it also means that the system might not be able to pick up on trends or influenza epidemics outside of the western civilization. We knew that this might become an issue early in the process, which is why we have decided to put most of our focus on Twitter activity in the USA.

[1] Stats from `https://sysomos.com/inside-twitter/twitter-retweet-stats` (2010)

5.3.2 Geolocation

Because the tweets we use are collected from Flutrack, we do not have any say in the geocoding of the tweets. The tweets from Flutrack are geolocated through the Google Geocoding API [29], which is overall quite accurate and provides high resolution results. However, the results can be improved, as discussed in chapter 2, for example with the use of the geolocation service Carmen [20], which can increase the overall number of geolocated tweets. Carmen exists as a python library, and can thus be easily implemented in the API. However, it requires full tweet objects from the Twitter API for optimal results, which the Flutrack API does not provide. Therefore, our API will simply have to trust the Flutrack API in their collection of tweets and their respective geolocation.

5.4 Machine learning

Chapter 4 provided results from testing the implementation of the experimental machine learning module described in chapter 3, which will be evaluated and discussed in this section.

5.4.1 Data

During the ID lookup of the tweets in the dataset, we discovered that approximately 28% of the tweets were missing. This is most likely because of tweets or user profiles being deleted since the original collection of these tweets in 2009 and 2012. Although, this is the largest relevant dataset we could find, it is often generally advised to have an even larger dataset for training. Additionally, to test the accuracy of the trained classifier, one needs a similarly sized annotated dataset, which would require at least double the total amount of annotated tweets.

The dataset is also somewhat biased. In 2009 and 2012, there was an outbreak of H1N1 (more commonly known as swine flu), which may have increased the amount of tweets mentioning this issue. For example, the bigram feature 'swine flu' will be heavily represented in the training dataset, but can barely be found in a collection of tweets from today. An outbreak such as the H1N1 pandemic will also greatly increase the amount of tweets concerning influenza awareness, rather than actual infections. This means that a lot of the tweets will be mentions of news articles or similar, and will add to the noise that the classifier will have to remove, which is hard, seeing as those tweets may have many of the significant n-grams in them.

At the time of writing this, the dataset is between four and seven years old. While Twitter is still used extensively, the trends or norms in the community may have changed since then. A classifier trained on data from 2009 and 2012 may not give the same results in 2016.

5.4.2 Implementation

As shown in section 4.2, the machine learning module did not perform as we wanted. Both the Naive Bayes implementation and the Maximum Entropy implementation seemed to be able to pick up on the most informative features in the dataset, but was not able to classify tweets correctly. This might be because of a relatively small and outdated dataset, as explained in the previous subsection. The trained classifiers places a tweet in one of two categories, influenza related or not influenza related. To be able to accurately classify tweets, the two categories have to be distinct and distinguishable. When the training dataset is too small and the features of the data overlaps, the classifier is not able to clearly distinguish the two classes. When it comes to tweets that concerns an influenza infection, the general language of the tweet is likely to be too similar to a tweet that concerns something else.

Section 2.6 presented several successful implementations of classifiers that are able to accurately detect influenza related tweets, which proves that it is feasible to implement. However, this would require a much larger dataset, which also would require manually annotating it, which would take too long. Additionally, the implementation itself can be improved. Lamb, Paul and Dredze states that n-gram feature sets are insufficient, because of common vocabularies in the classes [12, p. 790]. They solved this by expanding the feature sets to include different grammatical properties, which was proven effective by their results. By doing this, one can increase the gap between the classes, which should increase the overall accuracy of the classifier greatly. Our implementation included unigrams and bigrams as features, but in order to increase the efficiency of the classifier, we could also include 3-gram features.

We implemented a Naive Bayes and a Maximum Entropy classifier which are simple to implement and are easily available as Python libraries. Aramaki, Maskawa, and Morita showed that Support Vector Machine is effective at classifying tweets, both in terms of accuracy and training time [11]. The software developed by Thapen, Simmie, Hankin, and Gillard [13] also proves that machine learning is a helpful tool when determining whether or not a tweet is influenza related. Thus, implementing an SVM-based classifier might produce more accurate and reliable results.

Implementing a machine learning module in the system was not part of the original scope, and were therefore not prioritized. After the initial attempts, we therefore decided to focus on other aspects of the system and use the Twitter data obtained from Flutrack's API instead.

Chapter 6

Conclusion and future work

This chapter will conclude the thesis, and describe further work needed to improve the system.

6.1 Conclusion

The goal of this research has been to create an open-source system which can monitor influenza activity in a specific city, model the pandemic spread by air travel throughout the world, and visualize this in a simple manner. To do this, we have implemented and hosted a public Django REST-API based on a mathematical model created by Rvachev and Longini[1], which uses Twitter data and air travel data to detect a possible epidemic origin and model the spread of the epidemic. The system also consists of a visualized representation of the potential spread, which is deployed and live. The research questions with their respective conclusion can be found below.

RQ1: How can we use real-time Twitter data to monitor influenza activity?

Conclusion: The REST-API is able to gather tweets 60 days from the past from api.flutrack.org, which provides the opportunity to look at changes in the amount of geolocated tweets related to influenza. This allows us to be able to detect a possible epidemic whenever there is an unusual change in the amount of influenza related tweets in a specific city. However, this approach is not completely accurate, and there is no way for the system to know for sure if there is an influenza epidemic anywhere. It does provide a decent estimate, though, as deciding when there is an epidemic is officially done by monitoring influenza incidents in a certain area. Thus, our attempt at detecting an influenza epidemic is comparable to current detection methods, although one would have to view the detection with some healthy scepticism. The implementation therefore answers RQ1 with the fact that monitoring influenza can be done through the analysis of Twitter data, which is also proven by several others [9, 10, 11, 12, 13] as presented in chapter 2.

RQ2: Is it possible to predict the spread of an influenza pandemic through air travel?

Conclusion: Our implementation of the model have been able to reproduce the broad patterns of the results of both Rvachev and Longini [1], and Grais, Ellis, and Glass [16] with high correlation. The implementation assumes an ongoing epidemic in a certain city as detected by the system, and calculates the global spread through available air travel data.

The model is able to predict the broad patterns of the geographical spread of influenza. It does, however, base this prediction on the underlying assumption that any influenza outbreak has the same epidemiological properties as the 1968 Hong Kong influenza virus. Having a reliable source for an epidemic outbreak, together with accurate air traffic data, the system will be able to predict the broad patterns of geographical spread.

We have also found that the air travel data can be replaced with other types of travel data, and achieve similar results, as stated by [17]. We conclude that predicting spread is possible under certain conditions and assumptions, and that further work is required for a more reliable prediction.

Our system should be viewed as a starting point for developing autonomous influenza forecasting systems. It can be considered a basis for building reliable such systems in the future. The open-source paradigm allows for our work to be studied, contributed to, and improved by others. We present the future work that should be prioritized by the open-source community in the following section.

6.2 Future work

The architecture of the system, and the nature of the open-source ideology, makes contributions almost necessary in order for it to function as it should. The system built during this thesis is not perfect, and we encourage researchers, developers or other students to build upon this system and improve it little by little. Appendix B contains a reproducibility document, which is made for others to be able to download and install the system locally. This document is also available as the ReadMe on our GitHub.

The system needs reliable data sources, and currently there are some shortcomings with the data. Especially the travel data is lacking, as the source we have used only contains passengers from and to at least one American airport. Additionally, the data should be updated automatically daily through a scheduled task, instead of having to be done manually.

The machine learning module was not prioritized in this thesis, but could be a great addition to the system. Implementing a working classifier with high accuracy could increase the amount of influenza related tweets, and help us to monitor influenza epidemics more efficiently. The machine learning module in the system can be viewed as a starting point for this implementation.

Finding a reliable way of autonomously detecting an influenza epidemic based on incidence reports is crucial for the prediction to work properly. In [33] Diaz-Aviles, Stewart, Velasco, Denecke, and Nejdl show that their system is able to use Twitter messages to identify an epidemic outbreak of EHEC in Germany in May of 2011 earlier than any then current methods of detection. Their work proves that epidemic detection from Twitter is not only possible, but can also be an effective method of detection.

The modularity and extensibility of the system allows for other types of travel to be included in the prediction module. Others have had success with this exact model and train travel data for example [17], and it is likely that the model can work with other types of travel as well. This means that the system could be expanded with different kinds of travel data, which could generally increase the overall accuracy of the system.

Taking advantage of the extensibility of the system, other cities should be added to the prediction model. ”In order to make the forecast more realistic, it is necessary to expand the number of cities modeled from 52 to at least 152.” [1, p. 20]. Rvachev and Longini explains that the model needs to be implemented with more cities for it to be more realistic, and more research needs to be done to determine which cities might be optimal additions to the model.

Bibliography

[1] Rvachev LA, Longini IM. A mathematical model for the global spread of influenza. Mathematical biosciences. 1985;75(1):3–22.

[2] Almvik M, Solstad M. Calculating and visualizing the risk of influenza infection based on travel data and Twitter data; 2015.

[3] Polgreen PM, Chen Y, Pennock DM, Nelson FD, Weinstein RA. Using Internet Searches for Influenza Surveillance. Clinical Infectious Diseases. 2008;47(11):1443–1448. Available from: http://cid.oxfordjournals.org/content/47/11/1443.abstract.

[4] Ginsberg J, Mohebbi MH, Patel RS, Brammer L, Smolinski MS, Brilliant L. Detecting influenza epidemics using search engine query data. Nature. 2009;457(7232):1012–1014.

[5] Cook S, Conrad C, Fowlkes AL, Mohebbi MH. Assessing Google Flu Trends Performance in the United States during the 2009 Influenza Virus A (H1N1) Pandemic. PLoS ONE. 2011 08;6(8):1–8. Available from: http://dx.doi.org/10.1371%2Fjournal.pone.0023610.

[6] Asur S, Huberman BA. Predicting the future with social media. In: Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on. vol. 1. IEEE; 2010. p. 492–499.

[7] Krieck M, Dreesman J, Otrusina L, Denecke K. A new age of public health: Identifying disease outbreaks by analyzing tweets. In: Proceedings of Health Web-Science Workshop, ACM Web Science Conference; 2011. .

[8] Corley CD, Cook DJ, Mikler AR, Singh KP. Text and structural data mining of influenza mentions in web and social media. International journal of environmental research and public health. 2010;7(2):596–615.

[9] Lampos V, De Bie T, Cristianini N. Flu detector-tracking epidemics on Twitter. In: Machine Learning and Knowledge Discovery in Databases. Springer; 2010. p. 599–602.

[10] Sadilek A, Kautz HA, Silenzio V. Predicting Disease Transmission from Geo-Tagged Micro-Blog Data. In: AAAI; 2012. .

[11] Aramaki E, Maskawa S, Morita M. Twitter catches the flu: detecting influenza epidemics using Twitter. In: Proceedings of the conference on empirical methods in natural language processing. Association for Computational Linguistics; 2011. p. 1568–1576.

[12] Lamb A, Paul MJ, Dredze M. Separating Fact from Fear: Tracking Flu Infections on Twitter. In: HLT-NAACL; 2013. p. 789–795.

[13] Simmie D, Thapen N, Hankin C. DEFENDER: Detecting and Forecasting Epidemics using Novel Data-analytics for Enhanced Response. arXiv preprint arXiv:150404357. 2015;.

[14] Khan K, Arino J, Hu W, Raposo P, Sears J, Calderon F, et al. Spread of a novel influenza A (H1N1) virus via global airline transportation. New England journal of medicine. 2009;361(2):212–214.

[15] Brownstein JS, Wolfe CJ, Mandl KD, et al. Empirical evidence for the effect of airline travel on inter-regional influenza spread in the United States. PLoS Med. 2006;3(10):e401.

[16] Grais RF, Ellis JH, Glass GE. Assessing the impact of airline travel on the geographic spread of pandemic influenza. European journal of epidemiology. 2003;18(11):1065–1072.

[17] Flahault A, Letrait S, Blin P, Hazout S, Menares J, Valleron AJ. Modelling the 1985 influenza epidemic in France. Statistics in medicine. 1988;7(11):1147–1155.

[18] Burton SH, Tanner KW, Giraud-Carrier CG, West JH, Barnes MD. " Right time, right place" health communication on Twitter: value and accuracy of location information. Journal of medical Internet research. 2012;14(6).

[19] Graham M, Hale SA, Gaffney D. Where in the world are you? Geolocation and language identification in Twitter. The Professional Geographer. 2014;66(4):568–578.

[20] Dredze M, Paul MJ, Bergsma S, Tran H. Carmen: A twitter geolocation system with applications to public health. In: AAAI Workshop on Expanding the Boundaries of Health Informatics Using AI (HIAI). Citeseer; 2013. p. 20–24.

[21] Bergsma S, Dredze M, Van Durme B, Wilson T, Yarowsky D. Broadly Improving User Classification via Communication-Based Name and Location Clustering on Twitter. In: HLT-NAACL; 2013. p. 1010–1019.

[22] Robinson AH. The Thematic Maps of Charles Joseph Minard. Imago Mundi. 1967;21:95–108. Available from: `http://www.jstor.org/stable/1150482`.

[23] Gale N, Halperin WC. A case for better graphics: The unclassed choropleth map. The American Statistician. 1982;36(4):330–336.

[24] Schmid CF, Maccannell EH. Basic Problems, Techniques, and Theory of Isopleth Mapping*. Journal of the American Statistical Association. 1955;50(269):220–239.

[25] Eicher CL, Brewer CA. Dasymetric mapping and areal interpolation: Implementation and evaluation. Cartography and Geographic Information Science. 2001;28(2):125–138.

[26] Cortes C, Vapnik V. Support-vector networks. Machine learning. 1995;20(3):273–297.

[27] Culotta A. Towards detecting influenza epidemics by analyzing Twitter messages. In: Proceedings of the first workshop on social media analytics. ACM; 2010. p. 115–122.

[28] Baldwin CY, Clark KB. Does Code Architecture Mitigate Free Riding in the Open Source Development Model. Working paper, Harvard Business School; 2003.

[29] Chorianopoulos K, Talvis K. Flutrack.org: Open-source and linked data for epidemiology. Health informatics journal. 2015;.

[30] Carlson SJ, Dalton CB, Durrheim DN, Fejsa J. Online Flutracking survey of influenza-like illness during pandemic (H1N1) 2009, Australia. Emerging infectious diseases. 2010;16(12):1960.

[31] Rish I. An empirical study of the naive Bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence. vol. 3. IBM New York; 2001. p. 41–46.

[32] Phillips SJ, Anderson RP, Schapire RE. Maximum entropy modeling of species geographic distributions. Ecological modelling. 2006;190(3):231–259.

[33] Diaz-Aviles E, Stewart A, Velasco E, Denecke K, Nejdl W. Epidemic Intelligence for the Crowd, by the Crowd. In: ICWSM; 2012. .

[34] Sysomos. Inside Twitter: An In-Depth Look Inside the Twitter World; 2009, revised 2014. [Accessed 26.05.2016]. [Online] `http://sysomos.com/sites/default/files/Inside-Twitter-BySysomos.pdf`.

Appendix

A Experimental results

A.1 1968-1969 Hong Kong influenza

The results from Rvachev & Longini are presented below. Figure 6.1 and Figure 6.2 Presents the whole inlfuenza forecast as a set of computed daily morbidity counts.

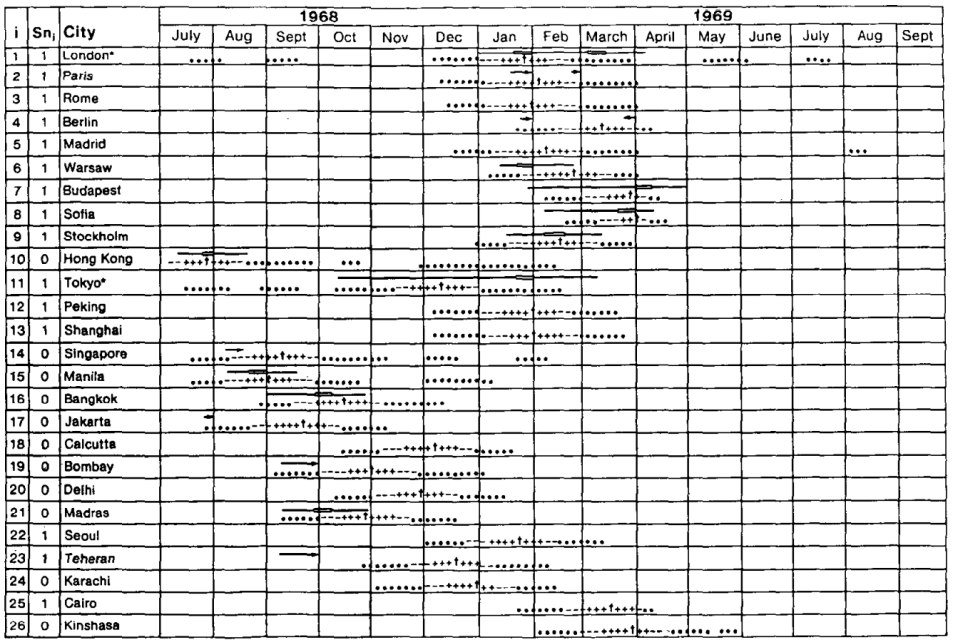


Figure A.1: The original results attained by Rvachev & Longini [1]. The table represent a 4-day forecasted incidence $b(t)$ per 100 000 inhabitant. • indicate $b(t) < 10$, - indicate $10 \leq b(t) < 100$ and + indicate $b(t) \geq 100$. The lines represent actual reported influenza activity.

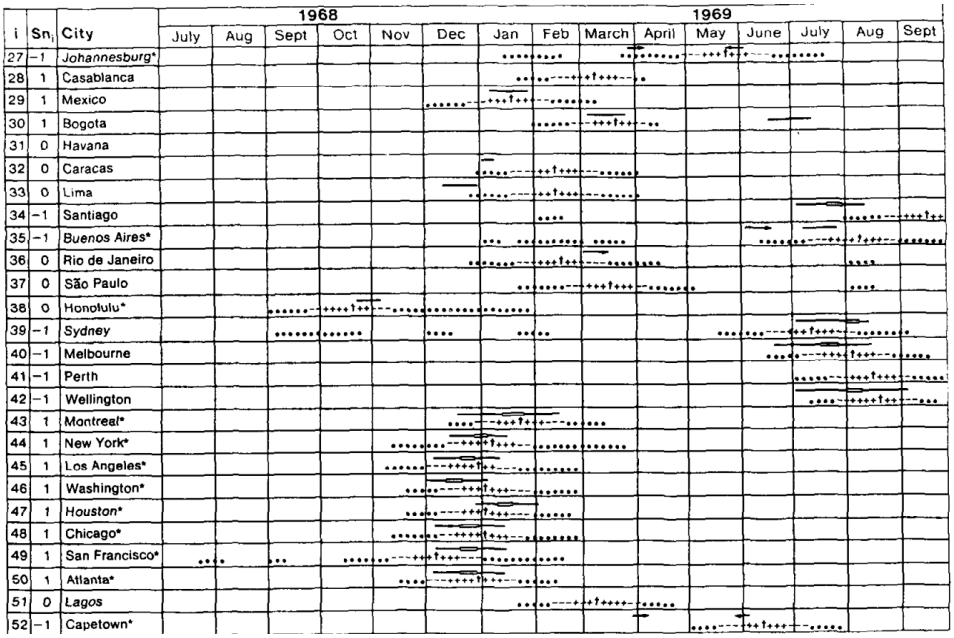


Figure A.2: The second half of the original results from Rvachev & Longini. [1].

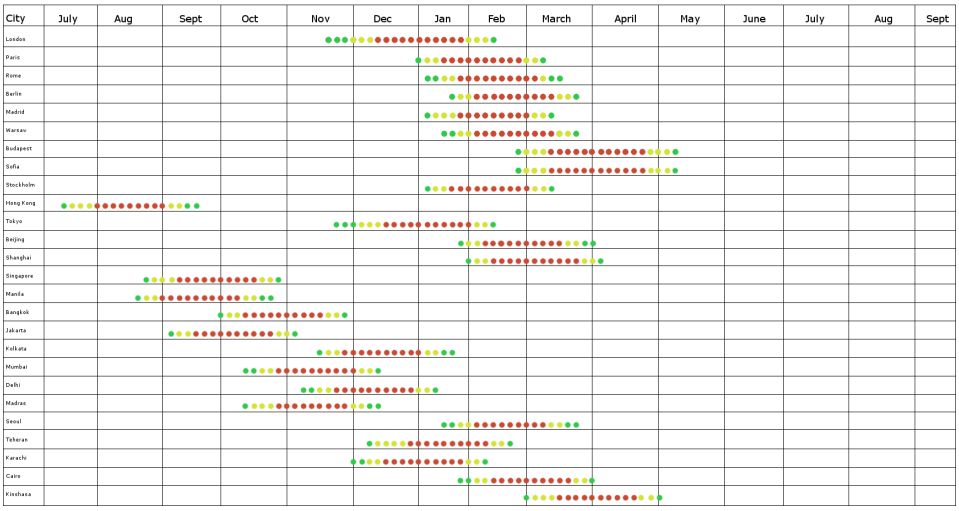


Figure A.3: Results from our system. Green dots represents an 4-day incidence of $b(t) < 10$, yellow dots represents $10 \leq b(t) < 100$, and red dots represents $b(t) \geq 100$.

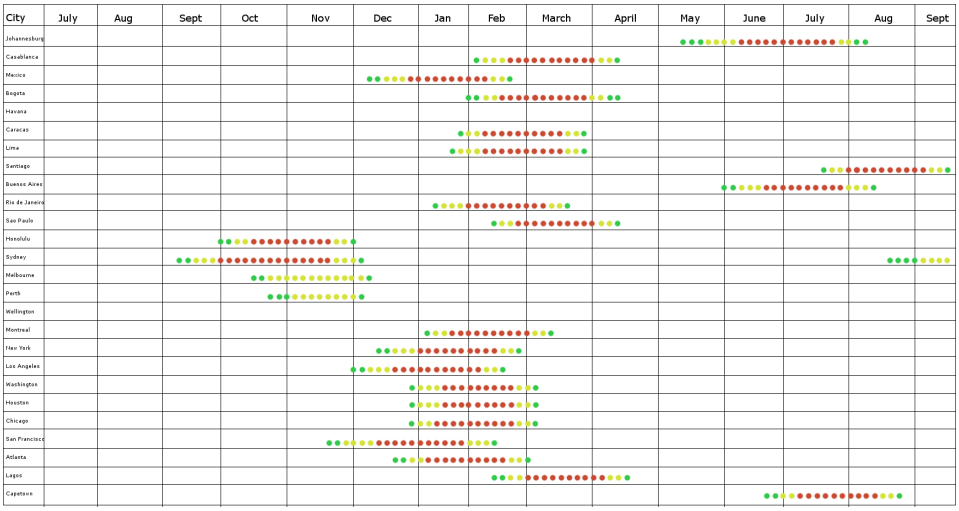


Figure A.4: The second half of our system's forecast.

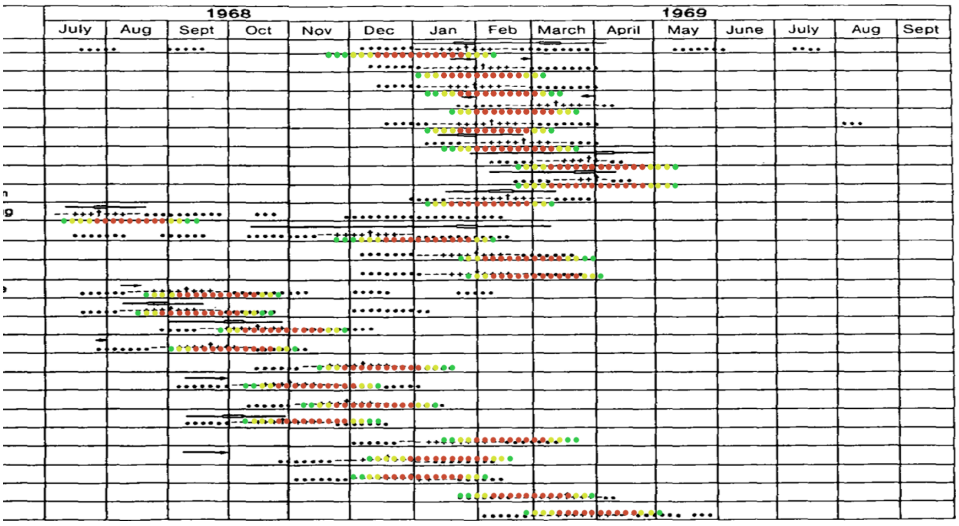


Figure A.5: Our results superimposed on the results of Rvachev & Longini to highlight the correlation.

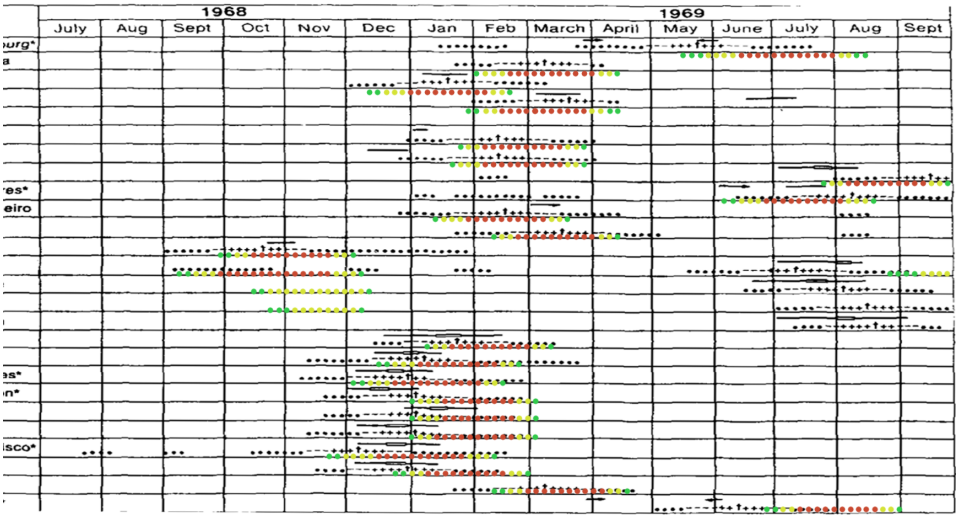


Figure A.6: Notable lack of correlation between our results and Rvachev & Longini's results for the four cities located in Oceania.

Air traffic volumes

	Lonc	Paris	Rom	Berli	Madr	War	Bud	Sofia	Stoc	Hon	Toky	Beij	Sha	Sing	Mar	Bang
London	0	977	110	38	171	32	6	1	117	69	10	7	0	50	7	5
Paris	977	0	150	32	198	30	1	1	52	12	10	5	0	8	4	6
Rome	110	150	0	2	71	9	7	6	11	4	8	3	0	6	2	6
Berlin	38	32	2	0	5	51	133	36	8	0	0	0	0	0	0	0
Madrid	171	198	71	5	0	3	8	4	14	0	2	0	0	0	0	0
Warsaw	32	30	9	51	3	0	22	22	7	0	0	0	0	0	0	2
Budapest	6	1	7	133	8	22	0	24	0	0	0	0	0	0	0	0
Sofia	1	1	6	36	4	22	24	0	6	0	0	0	0	1	0	0
Stockholm	117	52	11	8	14	7	0	6	0	0	0	0	0	0	0	0
Hong Kong	69	12	4	0	0	0	0	0	0	0	140	9	18	53	100	15
Tokyo	10	10	8	0	2	0	0	0	0	140	0	5	15	34	42	8
Beijing	7	5	3	0	0	0	0	0	0	9	5	0	72	0	1	0
Shanghai	0	0	0	0	0	0	0	0	0	18	15	72	0	0	0	0
Singapore	50	8	6	0	0	0	0	1	0	53	34	0	0	0	48	109
Manila	7	4	2	0	0	0	0	0	0	100	42	1	0	48	0	19
Bangkok	5	6	6	0	0	2	0	0	0	15	8	0	0	109	19	0
Jakrata	5	3	3	0	0	0	0	0	0	24	6	0	0	74	2	6
Kolkata	3	0	1	0	0	0	0	0	0	3	1	0	0	2	0	8
Mumbai	5	5	7	0	0	1	0	0	0	6	2	12	0	27	1	8
Dehli	7	7	3	0	0	0	0	0	0	7	4	0	0	5	1	21
Madras	0	0	0	0	0	0	0	0	0	0	0	0	0	25	0	0
Seoul	0	4	0	0	0	0	0	0	0	13	95	0	0	3	7	2
Teheran	12	8	7	0	3	0	0	0	0	0	4	4	0	0	0	0
Karachi	5	3	1	3	0	0	0	0	0	15	4	1	0	2	4	7
Cairo	24	20	45	3	6	1	5	4	0	0	3	0	0	0	2	4
Kinshasa	0	5	13	0	3	0	0	0	0	0	0	0	0	0	0	0
Johannesburg	152	14	16	0	13	0	0	0	0	4	2	0	0	0	0	0
Casablanca	7	100	6	4	18	0	3	1	0	0	0	0	0	0	0	0
Mexico City	23	48	0	0	49	0	0	0	0	0	12	0	0	0	0	0
Bogota	1	11	0	0	14	0	0	0	0	0	0	0	0	0	0	0
Havana	0	0	0	10	9	0	0	0	0	0	0	0	0	0	0	0
Caracas	6	12	28	0	37	0	0	0	0	0	0	0	0	0	0	0
Lima	1	13	0	0	10	0	0	0	0	0	6	0	0	0	0	0
Santiago	4	5	2	0	10	0	0	0	0	0	0	0	0	0	0	0
Buenos Aires	6	20	10	0	48	0	0	0	0	0	0	0	0	0	0	0
Rio de Janeirc	10	34	9	0	18	0	0	0	0	0	3	0	0	0	0	0
Sao Paulo	1	7	6	0	2	0	0	0	0	0	3	0	0	0	0	0
Honolulu	5	0	0	0	0	0	0	0	0	15	31	0	0	6	21	0
Sydney	68	10	7	0	0	0	0	0	0	16	13	0	0	21	5	12
Melbourne	7	0	2	0	0	0	0	0	0	2	4	0	0	6	4	1
Perth	19	0	2	0	0	0	0	0	0	2	0	0	0	15	0	0
Wellington	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Montreal	105	127	45	0	63	7	0	0	0	0	0	0	0	0	0	0
New York	417	223	119	10	143	24	0	0	48	37	60	6	5	9	5	7
Los Angeles	127	31	52	0	45	0	0	0	0	9	108	5	6	7	10	5
Washington	94	29	93	0	0	0	0	0	0	0	21	0	0	0	0	0
Houston	37	7	0	0	0	0	0	0	0	0	26	0	0	0	7	0
Chicago	81	20	55	0	0	0	0	0	0	0	58	0	0	0	6	0
San Francisco	6	6	0	5	0	0	0	0	0	54	108	26	11	12	28	0
Atlanta	37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lagos	14	18	21	4	20	0	0	4	0	0	0	0	0	0	0	0
Capetown	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	Jakr	Kolk	Mum	Dehli	Madr	Seou	Tehe	Kar	Cair	Kins	Joha	Cas	Mex	Bog	Hav	Car	Lima	Sant	Bue	Rio d	Sao F	Honc
London	5	3	5	7	0	0	12	5	24	0	152	7	23	1	0	6	1	4	6	10	1	5
Paris	3	0	5	7	0	4	8	3	20	5	14	100	48	11	0	12	13	5	20	34	7	0
Rome	3	1	7	3	0	0	7	1	45	13	16	6	0	0	0	28	0	2	10	9	6	0
Berlin	0	0	0	0	0	0	0	3	3	0	0	4	0	0	10	0	0	0	0	0	0	0
Madrid	0	0	0	0	0	0	3	0	6	3	13	18	49	14	9	37	10	10	48	18	2	0
Warsaw	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Budapest	0	0	0	0	0	0	0	0	5	0	0	3	0	0	0	0	0	0	0	0	0	0
Sofia	0	0	0	0	0	0	0	0	4	0	0	1	0	0	0	0	0	0	0	0	0	0
Stockholm	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hong Kong	24	3	6	7	0	13	0	15	0	0	4	0	0	0	0	0	0	0	0	0	0	15
Tokyo	6	1	2	4	0	95	4	4	3	0	2	0	12	0	0	0	6	0	0	3	3	31
Beijing	0	0	12	0	0	0	4	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shanghai	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Singapore	74	2	27	5	25	3	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	6
Manila	2	0	1	1	0	7	0	4	2	0	0	0	0	0	0	0	0	0	0	0	0	21
Bangkok	6	8	8	21	0	2	0	7	4	0	0	0	0	0	0	0	0	0	0	0	0	0
Jakrata	0	0	1	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kolkata	0	0	100	125	65	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mumbai	1	100	0	435	115	0	37	19	3	0	0	0	0	0	0	0	0	0	0	0	0	0
Dehli	2	125	435	0	96	0	0	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0
Madras	0	65	115	96	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Seoul	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8
Teheran	0	0	37	0	0	0	0	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Karachi	3	3	19	2	0	0	70	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
Cairo	0	0	3	4	0	0	0	2	0	0	0	4	0	0	0	0	0	0	0	0	0	0
Kinshasa	0	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	0	0
Johannesburg	0	0	0	0	0	0	0	0	0	36	0	0	0	0	0	0	0	0	0	13	4	0
Casablanca	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	12	0	0
Mexico City	0	0	0	0	0	0	0	0	0	0	0	0	0	8	7	8	15	0	7	5	5	16
Bogota	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	28	15	4	5	4	2	0
Havana	0	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	1	0	0	0	0	0
Caracas	0	0	0	0	0	0	0	0	0	0	0	0	8	28	0	0	10	0	11	14	3	0
Lima	0	0	0	0	0	0	0	0	0	0	0	0	15	15	1	10	0	28	21	10	4	0
Santiago	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	28	0	85	9	7	0
Buenos Aires	0	0	0	0	0	0	0	0	0	0	0	0	7	5	0	11	21	85	0	55	61	0
Rio de Janeirc	0	0	0	0	0	0	0	0	0	0	13	12	5	4	0	14	10	9	55	0	1060	0
Sao Paulo	0	0	0	0	0	0	0	0	0	0	4	0	5	2	0	3	4	7	61	1060	0	0
Honolulu	0	0	0	0	0	8	0	0	0	0	0	0	16	0	0	0	0	0	0	0	0	0
Sydney	2	0	6	0	0	0	0	2	0	0	25	0	0	0	0	0	0	0	0	0	0	34
Melbourne	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Perth	1	0	5	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0
Wellington	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Montreal	0	0	0	0	0	0	0	0	0	0	0	6	21	0	2	0	0	0	2	5	0	0
New York	0	0	3	12	0	5	0	2	7	0	18	17	84	9	0	43	25	11	30	66	2	10
Los Angeles	0	0	1	10	0	17	0	2	0	0	0	0	168	9	0	4	25	4	7	8	2	13
Washington	0	0	0	0	0	0	0	0	0	0	0	0	12	0	0	0	10	3	1	0	0	0
Houston	0	0	0	0	0	0	0	0	0	0	0	0	71	0	0	8	0	0	0	0	0	10
Chicago	0	0	0	0	0	0	0	0	0	0	0	0	78	0	0	0	0	0	0	0	0	10
San Francisco	0	0	0	0	0	1	0	0	0	0	0	0	42	3	0	0	8	7	6	87	1	38
Atlanta	0	0	0	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	10
Lagos	0	0	3	0	0	0	0	0	3	22	0	0	0	0	0	0	0	0	0	2	1	0
Capetown	0	0	0	0	0	0	0	0	0	0	590	0	0	0	0	0	0	0	25	6	6	0

	Sydne	Melb	Perth	Welli	Montr	New Y	Los An	Washir	Hous	Chicag	San F	Atlant	Lago	Capet
London	68	7	19	0	105	417	127	94	37	81	6	37	14	12
Paris	10	0	0	0	127	223	31	29	7	20	6	0	18	0
Rome	7	2	2	0	45	119	52	93	0	55	0	0	21	0
Berlin	0	0	0	0	0	10	0	0	0	0	5	0	4	0
Madrid	0	0	0	0	63	143	45	0	0	0	0	0	20	0
Warsaw	0	0	0	0	7	24	0	0	0	0	0	0	0	0
Budapest	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sofia	0	0	0	0	0	0	0	0	0	0	0	0	4	0
Stockholm	0	0	0	0	0	48	0	0	0	0	0	0	0	0
Hong Kong	16	2	2	0	0	37	9	0	0	0	54	0	0	0
Tokyo	13	4	0	0	0	60	108	21	26	58	108	0	0	0
Beijing	0	0	0	0	0	6	5	0	0	0	26	0	0	0
Shanghai	0	0	0	0	0	5	6	0	0	0	11	0	0	0
Singapore	21	6	15	0	0	9	7	0	0	0	12	0	0	0
Manila	5	4	0	0	0	5	10	0	7	6	28	0	0	0
Bangkok	12	1	0	0	0	7	5	0	0	0	0	0	0	0
Jakrata	2	1	1	0	0	0	0	0	0	0	0	0	0	0
Kolkata	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mumbai	6	1	5	0	0	3	1	0	0	0	0	0	3	0
Dehli	0	0	0	0	0	12	10	0	0	0	0	0	0	0
Madras	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Seoul	0	0	0	0	0	5	17	0	0	0	1	0	0	0
Teheran	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Karachi	2	0	0	0	0	2	2	0	0	0	0	0	0	0
Cairo	0	0	0	0	0	7	0	0	0	0	0	0	3	0
Kinshasa	0	0	0	0	0	0	0	0	0	0	0	0	22	0
Johannesburg	25	0	7	0	0	18	0	0	0	0	0	0	0	590
Casablanca	0	0	0	0	6	17	0	0	0	0	0	0	0	0
Mexico City	0	0	0	0	21	84	168	12	71	78	42	50	0	0
Bogota	0	0	0	0	0	9	9	0	0	0	3	0	0	0
Havana	0	0	0	0	2	0	0	0	0	0	0	0	0	0
Caracas	0	0	0	0	0	43	4	0	8	0	0	0	0	0
Lima	0	0	0	0	0	25	25	10	0	0	8	0	0	0
Santiago	0	0	0	0	0	11	4	3	0	0	7	0	0	0
Buenos Aires	0	0	0	0	2	30	7	1	0	0	6	0	0	25
Rio de Janeirc	0	0	0	0	5	66	8	0	0	0	87	0	2	6
Sao Paulo	0	0	0	0	0	2	2	0	0	0	1	0	1	6
Honolulu	34	2	0	0	0	10	13	0	10	10	38	10	0	0
Sydney	0	1604	75	24	0	4	13	0	3	0	36	0	0	0
Melbourne	1604	0	152	3	0	0	2	0	1	0	5	0	0	0
Perth	75	152	0	0	0	0	0	0	0	0	0	0	0	0
Wellington	24	3	0	0	0	0	0	0	0	0	0	0	0	0
Montreal	0	0	0	0	0	438	49	15	18	154	10	50	0	0
New York	4	0	0	0	438	0	1779	726	450	1345	1088	1020	8	0
Los Angeles	13	2	0	0	49	1779	0	746	342	1912	3504	1367	0	0
Washington	0	0	0	0	15	726	746	0	116	952	436	1104	0	0
Houston	3	1	0	0	18	450	342	116	0	451	216	790	0	0
Chicago	0	0	0	0	154	1345	1912	952	451	0	1522	1237	0	0
San Francisco	36	5	0	0	10	1088	3504	436	216	1522	0	909	0	0
Atlanta	0	0	0	0	50	1020	1367	1104	790	1237	909	0	0	0
Lagos	0	0	0	0	0	8	0	0	0	0	0	0	0	0
Capetown	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure A.7: The original transportation matrix reproduced from Rvachev & Longini's work [1].

	London	Paris	Rome	Berlin	Madrid	Vienna	Budapest	Sofia	Stockholm	Hong Kong	Tokyo	Beijing	Shanghai	Singapore	Mexico City	Bangkok	Jakarta	Kolkata	Mumbai	Delhi	Riyadh	Seoul	Tehran	Kyiv	Cairo	Karachi	Lima	Santiago	Buenos Aires	Rio de Janeiro	São Paulo	Honolulu	Sydney	Melbourne	Perth	Wellington	Auckland	New York	Los Angeles	Washington	Houston	Chicago	San Francisco	Atlanta	Lagos	Capetown
London	0	2454	2489	1936	2594	2484	2392	2603	2332	9597	9559	8132	8178	10843	8755	9215	8040	7491	7191	7197	6431	8615	6744	5151	4785	6465	5237	9118	11469	9469	9469	11859	10517	10517	17020	17020	17020	5585	5585	5585	8609	8609	8609	8609	8609
Paris	2454	0	1470	1316	1066	1322	1376	1436	1509	9603	9572	8133	8179	10844	8756	9216	8041	7492	7192	7198	6432	8616	6745	5152	4786	6466	5238	9119	11470	9470	9470	11860	10518	10518	17021	17021	17021	5586	5586	5586	8610	8610	8610	8610	8610
Rome	1470	1470	0	1316	1066	1322	1376	1436	1509	9603	9572	8133	8179	10844	8756	9216	8041	7492	7192	7198	6432	8616	6745	5152	4786	6466	5238	9119	11470	9470	9470	11860	10518	10518	17021	17021	17021	5586	5586	5586	8610	8610	8610	8610	8610
Berlin	1066	1470	98	0	350	182	70	98	0	607	135	0	257	0	0	0	0	0	0	28	0	0	0	0	0	135	203	472	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Berlin	322	1316	98	0	102	182	140	0	70	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Madrid	2594	266	350	102	0	96	95	0	294	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Warsaw	640	322	182	98	0	336	112	64	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Budapest	392	378	70	140	95	336	0	112	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sofia	742	1066	98	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Stockholm	280	490	98	70	294	84	56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Hong Kong	1273	509	0	0	0	0	0	0	0	0	2048	798	338	2595	1613	1679	604	0	269	134	0	1772	0	0	0	269	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tokyo	1371	1002	607	0	0	0	0	0	0	2048	0	504	1097	1645	8873	1242	470	0	235	235	0	1914	0	0	0	235	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Beijing	232	333	135	0	0	0	36	0	0	0	796	504	0	940	236	369	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Shanghai	73	330	0	0	0	0	0	0	0	0	938	1007	0	235	0	235	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Singapore	1603	536	115	0	0	0	0	0	0	0	2585	1645	940	0	1673	1242	363	235	235	0	873	470	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mexico City	1758	547	0	0	0	0	0	0	0	0	2585	1673	1242	363	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Bangkok	1010	512	257	0	0	0	0	0	0	0	2673	1242	363	235	2321	470	235	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Jakarta	0	0	0	0	0	0	0	0	0	0	604	470	0	3559	235	235	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kolkata	134	0	0	0	0	0	0	0	0	0	269	235	0	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Mumbai	461	171	28	0	0	0	0	0	0	0	269	235	0	168	0	269	134	448	0	1372	1914	134	0	201	0	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Delhi	403	315	0	0	0	0	0	0	0	0	134	235	0	235	0	269	0	196	0	1372	0	430	0	0	0	269	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Riyadh	0	0	0	0	0	0	0	0	0	0	0	0	0	694	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Seoul	165	47	0	0	0	0	0	0	0	0	1712	1914	0	0	470	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Tehran	238	0	0	0	0	0	42	0	0	0	0	0	0	100	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Kyiv	50	274	0	0	0	0	0	0	0	0	0	0	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cairo	523	255	478	36	109	6	182	53	0	0	235	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Karachi	50	274	0	0	0	0	0	0	0	0	0	0	0	134	0	0	0	0	0	201	269	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Lima	742	641	195	0	295	0	0	0	0	0	269	0	0	269	0	0	0	0	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Santiago	940	0	0	0	436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Buenos Aires	940	0	185	0	571	0	0	0	0	0	269	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Rio de Janeiro	1046	344	67	0	269	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
São Paulo	1209	0	135	0	0	0	0	0	0	0	671	0	0	0	1175	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Honolulu	472	101	135	0	0	0	0	0	0	0	0	537	638	269	369	369	0	0	0	0	0	403	0	0	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Sydney	1394	0	0	0	0	0	0	0	0	0	806	403	705	134	1544	135	1175	0	0	122	0	0	0	470	0	0	140	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Melbourne	472	0	0	0	0	0	0	0	0	0	436	235	0	0	1074	0	369	0	0	0	0	0	0	0	0	134	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Perth	0	0	0	0	0	0	0	0	0	0	369	225	0	0	1208	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Wellington	190	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Auckland	190	944	270	0	0	0	0	0	0	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
New York	7067	1914	598	3396	1130	397	295	45	990	495	2271	537	141	594	168	503	134	0	0	470	134	0	635	0	537	1002	0	212	0	0	134	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Los Angeles	1045	1914	0	0	0	0	0	0	0	0	2820	1779	470	940	671	235	940	0	0	235	0	0	1645	0	0	0	0	0	212	0	311	3749	495	470	1981	1486	990	390	1440	0	1233	0	0	0	2731	0
Washington	7060	2596	472	1376	383	0	739	0	0	0	235	940	101	101	772	383	0	0	0	470	470	0	235	0	0	470	0	0	0	0	0	3193	383	0	235	604	470	1544	336	1511	1274	1371	0	0	0	235	7346
Houston	944	2832	0	0	0	0	0	0	0	0	235	470	0	0	100	168	0	0	0	0	0	0	0	0	0	0	0	0	194	705	336	0	940	671	0	470	368	0	0	0	235	224	0	0	0	382	2125
Chicago	4483	708	236	0	705	374	739	0	0	722	470	1410	0	194	1007	134	436	0	0	0	0	0	0	0	0	0	0	0	0	0	0	940	671	0	470	0	235	224	0	0	235	1623	3212	705	0	235
San Francisco	4720	472	0	0	0	0	0	0	0	0	2820	1762	705	368	470	235	470	0	0	0	0	0	0	0	0	0	0	0	0	0	0	168	470	0	235	0	470	0	0	0	0	0	0	0	0	0
Atlanta	4720	472	0	0	0	0	0	0	0	0	2820	1762	705	368	470	235	470	0	0	0	0	0	0	0	0	0	0	0	0	0	0	168	470	0	235	0	470	0	0	0	0	0	0	0	0	0
Lagos	394	255	0	0	0	0	0	0	0	0	705	0	0	0	235	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Capetown	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure A.8: The transportation matrix used in the work of Grais et al. [16]

Results from Grais et al.

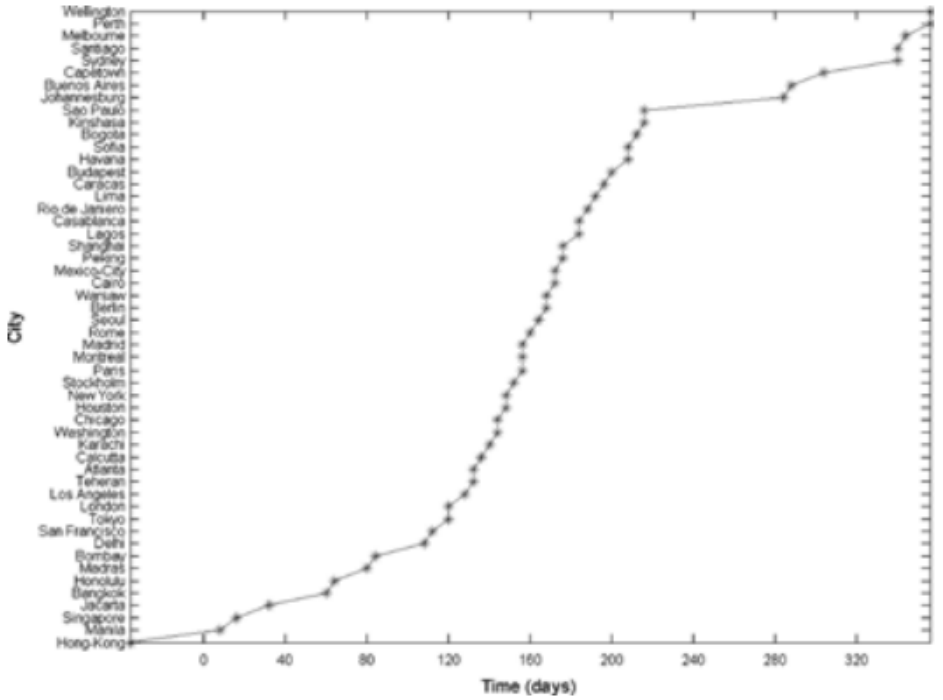


Figure A.9: The temporal progression of the forecast of Grais et al. using air traffic volumes from 1968 [16].

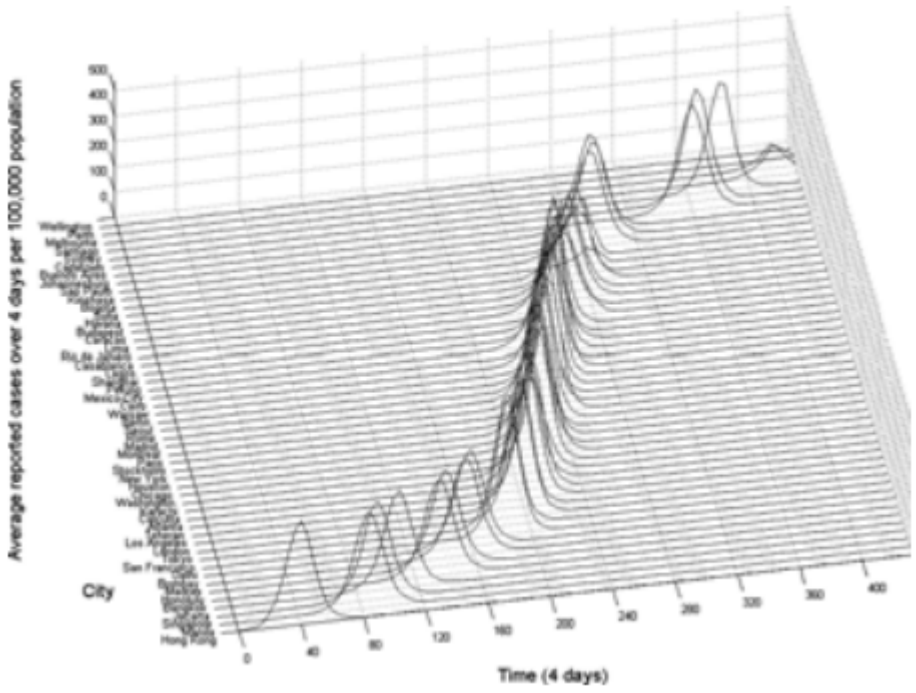


Figure A.10: The average 4-day incidence of Grais et al's forecast using air traffic volumes from 1968 [16].

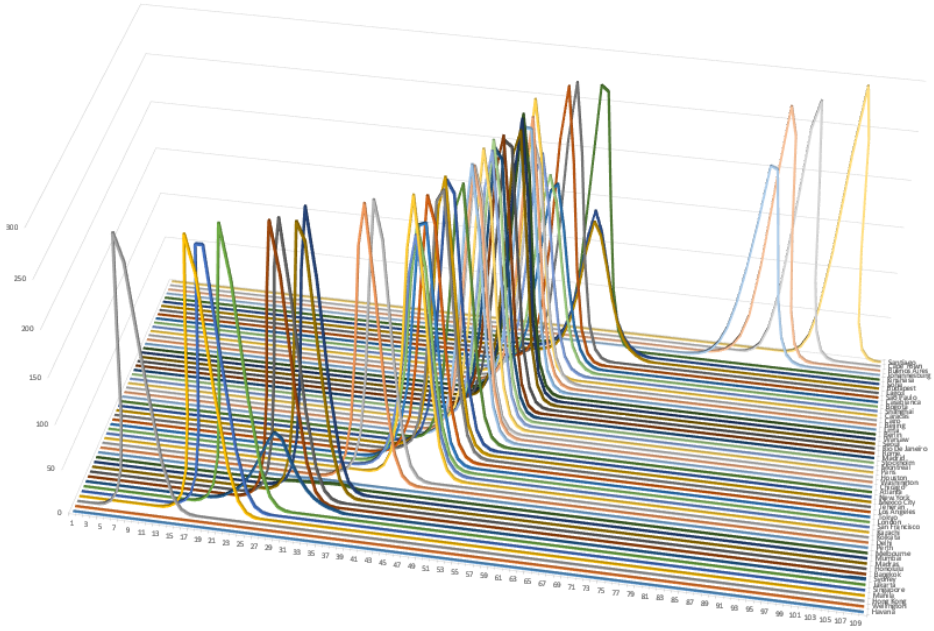


Figure A.11: The average 4-day incidence of our forecast using air traffic volumes from 1968.

B ReadMe (reproducibility doc)

This appendix can also be found on our GitHub at `https://github.com/almyy/flutrack_backend`.

B.1 Flutrack backend

This system is a part of a master's thesis at NTNU, done by Martin Almvik and Mikael Rino Solstad. The thesis was delivered 6th of June, 2016.

B.2 Purpose

The goal is to create an open-source web application (angularJS frontend and Django backend), available for everyone, that will predict where and when the flu will spread through air traffic, and visualize this over a time period. It uses tweets gathered from Flutrack.org for detecting influenza incidences, which the prediction is built upon. Visit their project on GitHub.

Furthermore, the system will use flight traffic data from BTS, and an algorithm to calculate the risk of infection in cities connected via air traffic. The algorithm is created by Rvachev and Longini in 1985, and our results correlates with the results they have found.

B.3 The API

The API is hosted at `flutrack-backend.herokuapp.com`, and has two publicly available endpoints:

Endpoint	Description
GET /tweets	Returns the weekly influenza related tweet count used by our system.
GET /prediction	Returns a list of an entire pandemic forecast, where each element represents one day in the forecast. Each day is represented as a list of all the cities included in the forecast with their respective morbidity and location.

B.4 Front-end

As a visualized interface to this API, we have created a simple single-page front-end. It's hosted at flutrack.almyy.xyz, and the open-source project can be found at GitHub.

B.5 Installation

The system is based on Python and Django, as well as some third party libraries. The steps required to install and reproduce our results is listed below.

Cloning the project

The main source of the system is this Git repository, and everyone is free to use this project as sees fit. The project can be cloned or forked by the use of Git, using this commands:

```
git clone https://github.com/almyy/flutrack_backend.git
```

MongoDB

The system uses MongoDB as datastore, and in order to test the API locally, you need MongoDB, which can be downloaded from here.

Python and pip

We have used Python 3.5.1, but should work on all installations over version 3.0. Install Python from here. This download also includes pip, which we will use to install third party libraries.

After you have downloaded Python and the project, you can navigate into the project directory, and install the following required libraries using pip:

```
pip install django djangorestframework tweepy requests pymongo
    xlrd
```

Django and Django Rest Framework are used to setup the API, Tweepy is used for utilizing the Twitter API, Requests makes it easier to make HTTP requests, Pymongo is used to communicate with our MongoDB instance, and xlrd is used to read .xls-files.

B.6 API keys

The system is dependent on external geocoding, and it is therefore necessary for you to get an API key for the Google Geocoding API. This can be obtained here. The key must be added as an environmental variable named 'GEOLOCATION_KEY'.

B.7 Running the API

Now that you have everything installed, you should be able to run a local instance of the API. First you need to populate the database, so that it has some data to base the prediction on. We first run the setup.py script, which sets some necessary environment variables. If you ever want to deploy the API to a production server, the SECRET_KEY variable should be changed and hidden. In the project root folder, run the following command:

```
python setup.py
```

Then we can run populateDB.py:

```
python flutrack_backend/populateDB.py
```

This step may take a while. If you get import errors, you might have to add the project folder to the PYTHONPATH environment variable manually. This will populate the database with the required cities, airports, as well as some updated tweets from the Flutrack API. When the database is populated and the secret key is set, the API is ready to be run. You can run the following command to start the API locally:

```
python manage.py runserver
```

This hosts the API locally on localhost:8000, and can be browsed using e.g. Postman. When running locally, the API also serves a browsable version through browsers, so you can visit the API in any browser.

Example request: GET localhost:8000/prediction