Implementing a Growing Neural Gas Network to Predict Categories for News Articles

Sara Eriksson

Stockholm University

Bachelor's Thesis in Computer Science
at Stockholm University, Sweden, 2020

Implementing a Growing Neural Gas Network to Predict Categories for News Articles

Sara Eriksson

Bachelor's Thesis in Computer Science (15 ECTS credits)
Bachelor's Programme in Computer Science
Stockholm University year 2020
Supervisor at the Department of Mathematics was Lars Arvestad
Examiner was Chun-Biu Li

Department of Mathematics
Stockholm University
SE-106 91 Stockholm, Sweden

Abstract

Automatically assigning documents of text with labels representing the category of which the document belongs, is a problem within the text mining field. SVT is interested in finding a solution to this problem and use it as a tool for proposing categories for news articles to editors. During this project the Growing neural gas method was evaluated in the purpose of automatically assigning category labels to news articles. Growing Neural Gas is a neural inspired method based on the Neural Gas method, and it is proposed to be used for clustering and classification tasks [2]. The results of this project show that the GNG method did not perform well. Most of the predictions made by the GNG graph were incorrect. The label "Agenda" was the predicted label for more than 90 % of the news articles, but less than 10 % of the articles belonged to this category. However, several improvements can be done, including changing the method for transforming text into numerical vectors, optimizing constants in the GNG algorithm, and considering ambiguous category labels used for training. The performance of GNG was poor, but the improvements could make the method more suitable for automatically predicting categories for news articles.

Implementation av ett Growing Neural Gas-nätverk för att förutspå kategorier för nyhetsartiklar

Sammanfattning

Att automatiskt sätta etiketter som representerar kategori för textdokument är ett problem som tillhör området text mining. SVT är intresserade av att hitta en lösning till det föregående nämnda problemet och använda det som ett verktyg som föreslår kategorier för nyhetsartiklar till redaktörer. Under detta projekt har det utvärderats hurvidare metoden Growing Neural Gas kan användas för att automatiskt sätta etiketter på nyhetsartiklar, som representerar kategorin för varje artikel. Growing Neural Gas är inspirerad av neurala nätverk och är baserad på Neural Gas metoden. Förslagsvis används GNG för att lösa klustings- och klassificerings problem [2]. Resultaten från detta projekt visar på att GNG inte presterade bra, då flesta förutsägelserna gjorda av GNG var inkorrekta. Mer än 90 % av alla artiklar förutspåddes att tillhöra kategorin "Agenda". Dock tillhörde endast 10 % av artiklarna denna kategori. Det finns däremot flera möjliga förbättringar att genomföra för att förbättra resultatet. Dessa förbättringar inkluderar: ändra metod för att omvandla text till numerisk data, optimera konstanter i GNG algoritmen och att granska tvetydiga och oklara etiketter använda i träningsfasen. GNG presterade inte bra i försöken gjorda under projektet, men med de tidigare nämna förbättringarna är det möjligt att GNG skulle kunna vara lämplig för att automatiskt sätta etiketter på nyhetsartiklar som representerar dess kategori.

Contents

Chapter 1

Introduction

1.1 Background

Sveriges Television (SVT) is Sweden's national public television broadcaster, providing news, entertainment and educational video content to everyone in Sweden. Other than video content, the news department at SVT provides news in articles accompanied with videos through their website and mobile app. Some of the articles are labeled with categories, which enables a feature where consumers can select a specific category, to display articles on topics of their choosing. To label documents with categories is a common method to group content. It has several benefits, such as the user experience improvements for external users of the SVT news mobile app, where the user can select a category to browse. Other benefits includes optimizing search features, storage, accessing stored data internally, etc.

The labeling process of news articles at SVT is completely manual today. An editor has to come up with appropriate categories to use and then use the CMS (Content Management System) to set one or more categories for an article. Because this process requires an editor to do extra work and come with categories to set, the possibility of the category not being set at all is quite high. But if there was a tool that automatically proposed categories to editors in the CMS, it would be very easy for editors to set categories

fast. Therefore, there is a desire to have a tool at SVT to automatically set categories for news articles. The task of automatically setting categories on news articles can be viewed and solved in several different ways, but a specific approach will be explored in this project.

1.1.1 Text Mining and Natural Language Processing

The practice of collecting information from text is called text mining. It includes various different methods and data structures, but all of them aim towards the same goal; to collect high quality information from text. Natural language processing (NLP) is one type of text mining, focusing on information extraction from natural language using computers [1]. Previous work within the field of NLP have resulted in several tools that are used on a daily basis by thousands of people, some examples being: sorting "spam" emails, spell checking and automatic completion when writing. During this project some NLP methods will be used, specifically the Growing neural gas method and other methods, to explore their performance when applying them to automatically set categories for news articles.

1.1.2 Clustering and Classification

Cluster analysis or clustering refers to the process of arranging data into groups, or clusters, where data in the same cluster is more similar to each other than data in other clusters. To group news articles into subject based categories could be seen as a clustering problem. But it could also be viewed as a classification problem. Classification is closely related to clustering as its purpose is to assign data into predefined classes [1]. During this project the automation of labeling news articles with categories will be viewed as both a clustering problem and a classification problem. The clustering problem will consist of mapping out clusters of news articles. But solving the clustering problem alone does not provide a way to predict a category name for a unknown article. It is however, possible to use category name labels defined by editors together with the cluster mapping and some classification strategy to determine to which category an unknown article belongs. The benefit of using both clustering and classification methods together, instead of using

only a classification method, is that the clustering could also be used for other purposes other than the one of this project in the future.

There is a number of different approaches to solve clustering problems, such as the k-means algoritm and DBSCAN. These methods are both common but are not the best option for our problem. The k-means algorithm needs to know the number of clusters beforehand and DBSCAN clusters all of the data points at once, making it hard to train and use over time. But the Growing neural gas method does not have these problems [3], making it more suitable for this project. In order to use the Growing neural gas method for classification, labeling strategies can be applied, see more in section 1.1.6.

1.1.3 Artificial Neural Networks

The selected method, the Growing neural gas method, creates an artificial neural network (ANN) [3]. ANNs are computing systems which process information in a way the resembles some learning patterns of the human brain. The difference between traditional computing and ANN's is that traditional computing only can solve problems for which the rules and circumstances are precisely defined, while ANN's have the ability to calculate solutions without having exact rules defining the problem [8].

There are numerous applications of ANNs, including prediction, identification, clustering etc. The ANN maps input to a target output. In other words, it takes given an input and outputs new meaningful information about the input.

In general an ANN consist of a number of nodes, usually referred to as neurons, connected by edges; creating a network. Information is transferred between neurons via the edges, which can be seen as a greatly simplified model of signals transported between human brain cells via synapses. Input is passed to the neurons, which have descriptions of what to do with that information: mostly in form of mathematical calculations.

There are three stages when creating and using ANNs: training, validation and usage. Training is the first step, and during this stage input data is passed to the neurons and the network learns from the input data. Learning

in this context is minimizing the error of the output. After training, validation is done by taking a new set of input data and validate that the network outputs the expected results. Finally, the network can be used for processing data that has not been seen by the network previously and give a output with minimized error.

1.1.4 Self-organizing Maps

Self-organizing maps (SOM) are a specific type of ANN, which the ANN created by the Growing neural gas algorithm partly resemblances [3]. It was introduced by Teuvo Kohonen and he explains that SOMs projects data onto a grid. According to Kohonen it can be viewed as a "similarity graph" where some subset of data is associated with each grid point, and grid points with most similar data should be located closest to each other in the grid [4].

When working with SOMs the usual steps are not the exactly as the one of most ANNs. When creating and using SOMs, two stages are neccesary: training and mapping. Training works much like for most ANNs, but the mapping phase is special for the SOM. In simple terms, this stage assigns input to grid points, declaring that this is where in the similarity map the input belongs [5]. The Growing Neural Gas algorithm creates an ANN with features resembling a SOM, in the sense that they both map clusters with an ANN [2].

1.1.5 The Growing Neural Gas Graph

The Growing neural gas (GNG) algorithm is based on the Neural gas (NG) method introduced by Thomas Martinetz and Klaus Schulten in 1991 [7] and Competitive Hebbian Learning (CHL) introduced by Thomas Martinetz in 1993 [6]. CHL is used for mapping a predefined number of topological centers by adding an edge between the two neurons closest to the input. The term topological center is in this context a neuron in an ANN with the task of mapping clusters of a data set. However, to make use of all neurons in the network CHL needs another supporting method. Without such a method, CHL develops so called "dead units", which are not of any use to

the learning process of CHL. A method that can be used for this purpose is the NG method, which principle is to adapt the k nearest neurons for every input, where k is a large number that decreases for as the algorithm proceeds [3].

The Growing Neural gas method was introduced by Bernd Fritzke in 1995, that uses CHL and is a proposed alternative to using the combination of CHL and NG. This alternative also solves some of the disadvantages a CHL and NG combination brings, such as the need for predefined number of topological centers. As the GNG algorithm will be explained in detail below, it will be shown that the differences between GNG and NG are that GNG can grow over time and GNG has only constant parameters [3].

The Growing Neural Gas Algorithm

The Growing neural gas algorithm creates a graph where:

- each neuron has a vector position (of the same dimension as all other neurons in the graph) in $\mathbb{R}^n$ and a local error variable set to 0,
- each edge is an undirected connection between two neurons, initiated with an age property set to 0.

In short, the algorithm starts of with a graph with two neurons and for each input, the graph adapts the closest node and its neighbors to the current input. After the adaptation, neurons are removed if they do not have any edges connected to them. A neuron is then added if the algorithm has been iterated a certain number of times. The algorithm will continue iterating until a predefined stop criteria has been met, such as set number of passes through the algorithm or a maximum size of the graph [2]. The output of the algorithm is the graph it creates, which then can be used for its intended purpose.

To run the GNG algorithm a input set and a number of constants needs to be defined. The input set is a number of vectors of the same dimension in $\mathbb{R}^n$. In our case these vectors will consist of one input per article, where each input is a vector representing data from that news article, see how that representation is created in section 1.1.7. The constants to be set are:

- ϵ_b and ϵ_n which are used in step 6 of the algorithm described below, to move neurons in the graph.
- a_{max}, the maximum age for an edge in the graph.
- λ which decides the number of iterations of the algorithm between each insertion of a new neuron.
- α, the factor to multiply local error variables with when inserting a new neuron in step 9 of the algorithm.
- d, the factor to decrease all edge ages with each iteration of the algorithm.

The GNG algorithm in detail as defined by Bernd Fritzke [3] consist of the following steps:

1. Start with two neurons a and b with random positions w_a and w_b in $\mathbb{R}^n$.
2. Take an random input ξ in $\mathbb{R}^n$ from the input set.
3. Find the nearest neuron s_1 and the second nearest neuron s_2 to the input ξ.
4. Increase the age for all edges connected to s_1.
5. Update the local error variable for s_1 by adding:

$$\Delta error_{s_1} = |w_{s_1} - \xi|^2$$

6. Update the position of s_1 and all its neighbors (all neurons directly connected with an edge to s_1) according to:

$$w_{s_1} = \epsilon_b(\xi - w_{s_1})$$

$$w_n = \epsilon_n(\xi - w_n)$$

for all neighbors n of s_1.

7. Connect s_1 and s_2 with an edge, if no such edge already exist. If such an edge already exists, set the age of the edge to zero.

8. Remove edges with age exceeding a_{max}. Neurons having no connecting edges after this are to be removed as well.

9. If the number of passes through the algorithm is equal to an integer multiple of an integer λ, add a new node to the graph according to the following steps:

    - Find the node q with the largest error variable in the graph.
    - Find the neighbor f to q with the largest error variable.
    - Add a new node halfway between f and q and assign the new node the error variable $w_r = 0.5(w_q + w_f)$.
    - If there is an edge between f and q, remove it.
    - Add two new edges connecting r with f and q.
    - Multiply error variables of f and q with a constant α. Set the error variable of r to be the same as the error variable of q.

10. Multiply all ages of edges by a constant d to decrease all ages.

11. If the algorithm has not reached some criterion to stop, go to step 2.

12. Return the graph

1.1.6 Labeling Strategies

In order use GNG for prediction (classification) of categories for news articles there is a need for labeling strategies. One can use labeling strategies where labels for categories are collected from the text of the article in some way. But for this project it has been decided to use existing categories, defined by editors at SVT, for the labeling strategy. To include a labeling strategy in GNG, two functions needs to be defined:

- a function to set labels for categories on neurons,
- a function to set labels on input when using the graph for predictions.

Beyer and Cimiano [2] propose several functions for labeling with GNG. They also propose online labeling in GNG, which is when labeling of neurons is performed in batches. When using online labeling together with GNG, the algorithm is extended with a additional step (after step 3 in the GNG algorithm) where labeling of neurons is performed with a labeling function. This means that a neuron can have different labels during different times of the training process, and that a prediction could be performed at any iteration of the algorithm. When using offline labeling with GNG, the labeling of all neurons is done when the training phase is complete, i.e. when the execution of the GNG algorithm is finished.

The functions used for labeling in this project are ones proposed by Beyer and Cimiano [2].

Labeling Neurons

For labeling neurons in this project the Frequency-based method [2] was chosen, a strategy in which neurons stores the number of times it has been assigned a certain label. Let the neuron s_1 in step 2 of the GNG algorithm (the neuron closest to the current input) be called it the winner neuron. Each time a neuron is assigned as winner neuron, that neuron saves the label of the current input and increases the frequency count for that label by 1. Also, let $c \in C$, where C is the set of all labels available. Then, we can define the label of neuron s_1 after time t as

$$l_t(s_1) = \arg\max_c freq_t(c, s_1).$$

Labeling Prediction Input

The strategy chosen for labeling input for prediction was the Single-linkage method [2]. Let $N(c)$ denote the set of all neurons labeled with label c, and d_{new} denote an input for which we want to predict the label. With the Single-linkage method, d_{new} is labeled with label c of neuron n that minimizes the distance to d_{new}:

$$l(d_{new}) = \arg\min_c(\arg\min_{n \in N(c)} |w_n - w_{d_{new}}|^2).$$

1.1.7 Pre-Processing of Text

In most forms of NLP, pre-processing of data is necessary. In this project it was needed since the text of each news articles needed to be transformed into numerical vector representations in order to use them with the GNG method. The pre-processing of text data depends on the intended usage of the data. Before this project started, SVT had some tools implemented and ready to use for pre-processing data from their news articles. These tools contained implementations for removal of stop words, tokenization, lemmatization, and TF-IDF.

Stop words, Tokenization, and Lemmatization

A first step of pre-processing text, is removing stop words. Stop words are the words that are the most common in a text, e.g. in Swedish it could be words such as "och" or "till". These words are removed because they do not provide any significance to the topic of a text.

After removal of stop words, tokenization is a common next step. Tokenization is the process of dividing text into tokens, where a token usually is represented by one word. Tokens with more than one word can occur in cases such as names, e.g. "Sara Eriksson" could be consider one token. During tokenization signs such as period, exclamation mark, question mark, white space, etc. are removed from the data as well.

Finally, some sort of normalization can be performed. Lemmatization is one form normalization where different methods are used to return the base form, called a lemma, of each token. For example the base form of the Swedish words "spelade" and "ungdomar" are "spela" and "ungdom". Methods for lemmatization vary between languages, but SVT uses a tool called the Stagger [10]. This tool is also used for tokenization at SVT.

TF-IDF

After the steps described in section 1.1.7 has been completed, a method called TF-IDF can be used. TF-IDF stands for "term frequency - inverse document frequency" and it is a method used to calculate numerical values

for each token/lemma in a text. The TF-IDF value for a token is determined by an inverse proportion of the number of times a token/lemma occurs in a document to the percentage of documents the term appears in [9].

With TF-IDF, each token/lemma will have a TF-IDF value for each document it occurs in. But in order to use GNG, vector representations of each document/article are needed.

TF-IDF Vectors

It is possible to use TF-IDF values to create vector representations for text in articles, using something called a TF-IDF matrix. A TF-IDF matrix is a matrix where all TF-IDF values for all tokens/lemmas appearing in any article are stored. Let n denote the number of articles used in the calculation of TF-IDF values, m denote the number of tokens/lemmas found in all n articles, $t_{n,m}$ denote the TF-IDF value calculated for token/lemma number m in article number n. Then the TF-IDF matrix M is defined by

$$M = \begin{pmatrix} tfidf_{1,1} & tfidf_{1,2} & \cdots & tfidf_{1,m} \\ tfidf_{2,1} & tfidf_{2,2} & \cdots & tfidf_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ tfidf_{n,1} & tfidf_{n,2} & \cdots & tfidf_{n,m} \end{pmatrix}.$$

Each row in matrix M contains all TF-IDF values for an article, therefore a row can be considered a vector representation of an article. This means that the vector $[token_1, token_2, ..., token_m]$ with all tokens/lemmas, creates a vector space where each token add a new dimension to the vector space. The benefit of creating vector representations this way is that all articles are represented by vectors in the same vector space, which is equivalent to the vectors having the same dimension. This is necessary in order to use vectors as input into the GNG algorithm. However, the disadvantages of using a TF-IFD matrix is that the dimension of the vectors can grow fast, and we need to know all tokens of all articles in advance of creating any vectors.

1.2 Problem Definition

The problem to be examined during this project is the implementation of a Growing neural gas network and testing it's ability for predicting categories of news articles. It will also be evaluated if the method can be used to create a tool for editors where categories to label news articles with are automatically proposed.

Chapter 2

Method

In general, the method consisted of building the GNG graph, pre-processing of data, training the GNG graph and predicting categories for articles. The following sections will explain all of those steps in detail.

2.1 Implementing GNG

Implementation of GNG was done in the programming language Kotlin. The main reason for this was the there was an existing implementation at SVT for collecting data of news articles and some pre-processing of that data, written in Kotlin. More details about the pre-processing tools at SVT in section 1.1.7.

To implement the GNG algorithm, the steps from the defined algorithm in section 1.1.5 was followed. When the finished implementation of GNG was done it was tested with 2-dimensional data before moving on to data of news articles. During these tests the following constants were used for the GNG algorithm: $\lambda = 100$, $\epsilon_b = 0.2$, $\epsilon_n = 0.006$, $\alpha = 0.5$, $a_{max} = 50$, and $d = 0.995$. The number of iterations for the algorithm to run was set to 2000000. These constants where sampled from Fritzke's [3] example.

2.2 Pre-processing of Data

To be able to use text from news articles as input in the GNG algorithm, the data from the articles needed to go trough pre-processing. The data collected for each article was the title concatenated with the body text and categories (if an editor had defined any categories for that article). The categories was left as is and only used for labeling clusters, see section 2.4 for usage of the predefined categories.

The body text and the title of the article needed to be processed, and it was for this data the pre-processing tools at SVT where used. These tools were capable of tokenization, removal of stop words, and lemmatization. An earlier solution to automatically set categories for news articles also used these tools, therefore it was suitable to use the same tools for this project.

After the text of the articles had been trough the three stages of pre-processing above, the resulting data was transformed into vector representations. Using a TF-IDF matrix for the articles. The implementation for TF-IDF was also part of the tools that existed at SVT, leaving only the implementation for creating the TF-IDF matrix for this project. To guarantee that the vector space created contained all lemmas from articles in both the training set and the prediction set, both sets of articles were included when creating the TF-IDF matrix. When the matrix had been generated and saved, all pre-processing of data was completed and the resulting data was ready for use.

2.3 Training the GNG Graph

Once the data was ready to use, the training of the GNG graph could start. The training consisted of creating the GNG graph with the GNG algorithm, with the additional step of labeling of neurons added to the algorithm (see section 1.1.6). During the training, three options were possible when it came to which articles to use the articles as input:

- only use articles with categories set by editors,

- only use articles without categories set by editors,
- use articles with and without categories set by editors.

It is possible to use any of the three approaches, but using articles without predefined categories would probably increase the number of neurons without labels in the graph. Neurons without category labels can in turn lead to getting prediction results without any category labels. One can ignore the neurons without labels when using the graph for prediction, but to save computational time and power, it was selected to use only articles with predefined categories in the training stage.

After filtering out articles without tags, the GNG algorithm was executed with an input set of 1000 articles. It was tested to start the algorithm with more than 1000 articles, but it took too much time to complete the algorithm with a larger amount of articles. A discussion on why the training took a long time can be found in section 4.1.

2.4 Labeling Strategies

In order to use GNG for predictions of categories (classification), labeling strategies were needed. These strategies needed to consist of two types of functions; one function for labeling neurons with a category name label and one function for labeling a prediction input with a category name. There are different approaches as to when the labeling should take place. For this project, online labeling was used. To implement online labeling with GNG, the labeling of neurons was done during training, i.e. during the execution of the GNG algorithm. Because online labeling is not separate to training, it is not dependent on the training being completely finished which makes it more flexible than offline labeling. For this reason the online labeling strategy was chosen for this project. See section 1.1.6 for a more detailed description of online labeling.

The Frequency-based method was chosen for labeling neurons and the Single-linkage method for labeling prediction inputs, see section 1.1.6 for descriptions of these methods. The Single-linkage method was used because it was

the most simple to implement. The frequency-based method was chosen because it has the potential to save more than one label, which made the predictions capable of predict more than one category label per article. However, only one category label was used during this project.

2.5 Prediction with the GNG Graph

After training, the GNG graph was used for predicting categories for 100 random articles not seen by the graph before. The articles used for testing had predefined categories, but these were ignored until the prediction had been completed. When prediction was finished, the labels were used for comparing the results of the graph predictions to the categories set by editors.

The process of training and predicting was done twice with different constants used for the GNG algorithm. The first time the constants from Fritzke's [3] example were used for the GNG algorithm: $\lambda = 100$, $\epsilon_b = 0.2$, $\epsilon_n = 0.006$, $\alpha = 0.5$, $a_{max} = 50$, and $d = 0.995$. The number of iterations for the algorithm to run was set to 10000. The second time, it was tested to see what some changes of the constants would do, therefore the constants were set to: $\lambda = 50$, $\epsilon_b = 0.2$, $\epsilon_n = 0.006$, $\alpha = 0.5$, $a_{max} = 50$, and $d = 0.995$. The number of iterations for the algorithm to run was set to 20000. The constant λ that determined how many iterations of the algorithm were executed between each insertion of a new neuron was decreased, to see if more neurons in the graph increased the performance. The number of iterations was in increased to allow more exposure of the training data to the graph. No further testing of different constants were done, as it was not in the scope for this project to optimize these constants.

Chapter 3

Results

For evaluating of the results of the usage of news articles in GNG, it was only possible to do evaluation of the prediction (classification) performance. Clustering was also done, but the vector representations were of very high dimension ($\sim$ 17000 dimensions), making hard to do any type of evaluation of the clustering alone. Another complication of high dimensional vectors were that it was not possible to use more than 1000 articles during training, by cause of any operation performed with these vectors took a long time. Training the GNG graph took about $8 - 12$ hours in total.

3.1 First Test

The first test of using GNG to cluster news articles and predict categories gave the following results. Out of the 100 new articles sent to the graph for predicting their category, only one was accurate. The graph predicted that 97 of the articles belonged to the category "Agenda". One of the articles did belong to this category, but 96 out of the 97 articles did not. The other three articles, were the prediction was not "Agenda", the predictions were not the same as the editor has set before, see table 3.1 for exact results of these three articles.

Table 3.1: Results from articles using GNG for prediction.

Categories set by editor	Predicted category
Boxholm, Polismorden i Malexander	Musikhjälpen Västerås
Klimattoppmöte i Paris 2015	Nordkorea-krisen
USA:S NYE PRESIDENT	Nordkorea-krisen

3.2 Second Test

The second test gave similar results to the first test, but it was slightly more accurate. Most articles were predicted to belong to the category "Agenda" again. 9 out of the 100 articles were predicted to belong to other categories than "Agenda". See table 3.2 for the results of those 9 articles.

Table 3.2: Results for articles using GNG for prediction.

Categories set by editor	Predicted category
Nelson Mandela död	Boxning
Polismorden i Malexander	Polismorden i Malexander
Prins Harry och Meghan Markle gifter sig	Festvåningar i Södertälje
EU-valet 2019	EU-valet 2019
Almedalen 2019	Regeringsbildningen
Regeringsbildningen	Regeringsbildningen
Val 2018 - Politikerna svarar dig	Val 2018 - Politikerna svarar dig
Val 2018 - Politikerna svarar dig	Val 2018 - Politikerna svarar dig
Ekonomipriset 2017	Nobelpriset 2019

Table 3.2 shows that 5 of the 9 articles not predicted to belong to the category "Agenda" got correct predictions.

Out of all 91 articles predicted to belong to the category "Agenda", 6 of them also had "Agenda" set as category by an editor, making them correct predictions. See table 3.3 for the results for these 6 articles.

Table 3.3: Results from articles using GNG for prediction.

Categories set by editor	Predicted category
SVT granskar friskolor, Agenda	Agenda
Fallet Thomas Quick, agenda, Agenda	Agenda
Hästköttskandalen, Agenda	Agenda
Kris för svenska mjölkbönder, Agenda	Agenda
Rut-avdrag för läxhjälp, agenda, Agenda	Agenda
Läget efter valet, Agenda	Agenda

In total, the second test had 11 correct predictions out of 100 articles, giving it an accuracy of 11 %.

Chapter 4

Discussion

The results of the first test were mostly incorrect, with only 1 out of 100 articles ending up with a correct category label. The reason for the poor performance is hard to pinpoint as there could be several reasons, and they could also be related to each other. Not exposing the graph to enough training data and insufficient constants for the GNG algorithm during training are presumably the two main reasons for the poor performance. Because the labeling strategies that were used have been proven to be successfully used with GNG [2], the training process with news article data can be considered the most uncertain area of the process. The labeling strategies can not be completely ruled out of being a reason for poor performance, however it will not be discussed as more investigation regarding these strategies would be needed. See section 4.1 for details on how the training process and other areas to work on to improve the performance.

The second test was slight more successful than the first test with an accuracy of 11 %, compared to 1 % for the first test. This shows that changing the constants in the GNG algorithm can give the results an impact of at least 10 %. Section 4.1 contains discussions of possible improvements for the future.

4.1 Improvements for the Future

There are several ways to improve usage of GNG for clustering and classification to automatically set categories for new articles. One of the major improvements would be to replace the methods for transforming the text to numerical vectors. First of all, the creation of the TF-IDF matrix caused very high dimensions for all the vectors. This lead to any operation with vectors terribly slow and only 1000 articles could be used for training as higher amounts took too much time. Another disadvantage of using the TF-IDF matrix and TF-IDF values, is the fact that one must know all words/tokens that occurs in any article used for either training or for prediction. This causes some of the advantages of the GNG method, such as not having to do all clustering at once, not being valid anymore.

Other than TF-IDF and the TF-IDF matrix, improvements include optimization of constants used in the GNG algorithm. There are a number of constants in the GNG algorithm that effects the training of the GNG graph. The constants used for this project were ones used by Fritzke [3] and some altered versions of those numbers, but there's no guarantee that the same constants would work well for this project. To find optimized constants could improve performance in a specific use-case of the GNG algorithm.

Lastly, an improvement to the classification part could be to have a look at what types of categories editors set for articles. The first test done of using news article data in GNG resulted with predicting most articles to belong to the category "Agenda". Agenda is a TV-program created by SVT, where politicians discuss and debate different political topics. The problem is, a political topic could be many different things, everything from healthcare to terrorism. Let's say that we have two articles with the following topics:

- politician A said X about the new healthcare budget on Agenda yesterday,
- hospital gets more money in new healthcare budget

If the first article is labeled with category "Agenda", which seems very sensible, there is a chance that the second article also gets labeled with category

"Agenda" since it is also about the new healthcare budget. But in reality the second article is not related to the TV-program Agenda. The question to be discussed is: if a TV-program or similar concepts are suitable categories. If so it would imply that all articles on topics that has been, or going to be, or could be on Agenda, should be labeled Agenda. This is something SVT would have to decide on in the future if automation of setting categories would be based on editors work.

Chapter 5

Conclusions

It is not possible to conclude if the Growing neural gas method is a suitable tool for automatically setting categories on news articles. The results of this project was an unsuccessful use of GNG for automatically setting categories on news articles. However, there is a lot of room for improvements regarding transforming data into text to numerical vectors and optimizing constants in the GNG algorithm. What categories are suitable to use with the GNG method should be examined if development of a tool for automatically labeling news articles with categories continues.

Bibliography

[1] M. Allahyari, S. A. Pouriyeh, M. Assefi, S. Safaei, E. D. Trippe, J. B. Gutierrez, and K. Kochut. A brief survey of text mining: Classification, clustering and extraction techniques. *arXiv preprint arXiv:1707.02919*, 2017.

[2] O. Beyer and P. Cimiano. Online labelling strategies for growing neural gas. In *International Conference on Intelligent Data Engineering and Automated Learning*, pages 76–83. Springer, 2011.

[3] B. Fritzke. A growing neural gas network learns topologies. In *Advances in neural information processing systems*, pages 625–632, 1995.

[4] T. Kohonen. Self-organization of very large document collections: State of the art. In *International Conference on Artificial Neural Networks*, pages 65–74. Springer, 1998.

[5] Y.-C. Liu, M. Liu, and X.-L. Wang. *Application of self-organizing maps in text clustering: a review.* INTECH Open Access Publisher., 2012.

[6] T. Martinetz. Competitive hebbian learning rule forms perfectly topology preserving maps. In *International conference on artificial neural networks*, pages 427–434. Springer, 1993.

[7] T. Martinetz and K. Schulten. A "neural-gas" network learns topologies. *Artificial neural networks*, 1:397–402, 01 1991.

[8] M. Rafiq, G. Bugmann, and D. Easterbrook. Neural network design for engineering applications. *Computers and Structures*, 79(17):1541 – 1552, 2001.

[9] J. Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, volume 242, pages 133–142. Piscataway, NJ, 2003.

[10] R. Östling. Stagger: an open-source part of speech tagger for swedish. *Northern European Journal of Language Technology*, 3:1–18, 2013.