Master Thesis
Semi-supervised opinion mining: learning sentiment towards vaccination on Dutch tweets

Nicole Walasek[*] N.WALASEK@STUDENT.RU.NL

First supervisor: Antal van den Bosch[**] A.VANDENBOSCH@LET.RU.NL
Second Supervisor: Tom Heskes[***] T.HESKES@SCIENCE.RU.N
Third Supervisor: Florian Kunneman[****] F.KUNNEMAN@LET.RU.NL

[*] *ICIS, Radboud University, Toernooiveld 212, 6525 EC Nijmegen, The Netherlands*

[**] *CLS, Radboud University, Erasmusplein 1, 6525 HT Nijmegen, The Netherlands*

[***] *ICIS, Radboud University, Toernooiveld 212, 6525 EC Nijmegen, The Netherlands*

[****] *CLS, Radboud University, Erasmusplein 1, 6525 HT Nijmegen, The Netherlands*

Abstract

Nowadays, online social media platforms like Twitter or Facebook provide a valuable framework for individuals to share and discuss ideas and opinions regarding various topics, including health related issues. Platforms like Twitter allow for an exhaustive aggregation and analysis of user-generated content which facilitates monitoring of public opinions and sentiment over time towards certain topic of interest. Keeping track of these developments can be crucial for health professionals in order to understand and address the public opinion and behaviour with regard to health related topics. One such topic is vaccination. The goal of this thesis is to perform sentiment analysis towards vaccination on Dutch tweets referring to vaccination. I will present results for using a framework named co-training in order to iteratively label unlabeled instances while adding those labeled with high confidence to the training set. The experiments demonstrate that the co-training framework can be successfully applied to improve sentiment analysis towards vaccination on Dutch Twitter messages. In total a best F_1 score of 0.72 on the negative class and 0.70 on the non-negative class has been reached with the best performing co-training classification system.

1. The importance of online opinion mining for vaccination

Nowadays, online social media platforms like Twitter or Facebook provide a valuable framework for individuals to share and discuss ideas and opinions regarding various topics, including health related issues. Within the past decade these kinds of platforms have spread globally and witnessed a rapid growth in the number of users, reaching people from various demographic groups, ethnicities and occupations (Perrin 2015). Twitter, a popular news and social networking service, where messages ("tweets") are restricted to 140 characters, has reported 317 million monthly and 100 million daily active users in January 2017 with a total of 500 million tweets sent per day (Aslam 2017). These characteristics of platforms like Twitter allow for an exhaustive aggregation and analysis of user-generated content which facilitates monitoring of public opinions and sentiment over time towards certain topics of interest. Keeping track of these developments can be crucial for health professionals in order to understand and address the public opinion and behaviour with regard to health related topics. One such topic is vaccination.

Outbreaks of vaccine preventable diseases are a serious public health issue. Its likelihood is reported to increase either for an overall decline in vaccine uptake (Jansen et al. 2003), or with an increase in the number and size of communities with very low vaccination rates. These low vaccination rates are often considered to be the result of so-called opinion clustering (Salathé and Bonhoeffer 2008, Omer et al. 2009). Opinion clusters are clusters of (unvaccinated) individuals who share the same sentiment concerning vaccination. Recognizing and addressing public concerns regarding vaccination is important for evolving successful immunization strategies in order to control the spread of infectious diseases.

The structure of the Twitter network[1] allows not only to study sentiment towards vaccination but also to make inferences about how these sentiment and behavioral patterns cluster along temporal, geographic and demographic dimensions (Campbell and Salathé 2012, Huang et al. 2017). This way Public Healthcare Organizations try to detect social discussion communities that are identified to influence vaccination decision-making in order to adapt immunization strategies to avoid new outbreaks of infectious diseases (Bello-Orgaz et al. 2017).

Furthermore, Signorini et al. (2011) have demonstrated in the past that Twitter can be successfully used not only to track users' interest in and concern about H1N1 influenza, but also to predict disease outbreaks in real time.

The goal of this thesis is to perform sentiment analysis towards vaccination on Dutch tweets referring to vaccination. The task poses several challenges:

1. Identify whether a tweet is relevant, i.e. addressing vaccination
2. Disentangle general sentiment from sentiment towards vaccination
3. Train a (multiple) classifier(s) on only a small number of labeled instances each of them being at most 140 characters long

The first challenge can be overcome by checking whether a tweet contains keywords related to vaccination, such as "vaccination" or "immunization". However, there are also tweets which contain these keywords and are not related to vaccination, as for instance "A vaccine against hypocrisy would also be nice"[2].

Concerning the second challenge some tweets might be about vaccination but express a sentiment that is not related to vaccination. Take for instance "I hate rain. Later on heading to a doctor's appointment for vaccination.". Although this tweet conveys negative sentiment ("I hate rain") and refers to the topic of vaccination, the expressed sentiment is not related to vaccination.

1. Tweets contain specific markers like # or @ to denote topics or user references. Moreover, tweets hold information about the geographic location of a user, as well as other user information inferred from his Twitter profile. Linking all of these information allows to build graphs visualizing different aspects of the Twitter network.
2. Note, that the English example tweets are artificial and only added to illustrate the challenges.

Lastly, text classification on short documents is a demanding task. Many traditional text mining methods rely on word frequencies and relations between words within a document as features in order to infer similarity between documents and class membership. However, a small number of short training documents might not provide enough information for these features to be meaningful to differentiate between different classes (Man 2014, Chen et al. 2011, Timonen and Kasari 2012).

In particular my work focuses on the third challenge: I will present results for using a framework named co-training in order to iteratively label unlabeled instances while adding those labeled with high confidence to the training set. In the past, co-training has been successfully applied to sentiment analysis (Biyani et al. 2013, Xia et al. 2015). However, to the best of my knowledge co-training has not been applied to sentiment analysis of Twitter messages before.

This work is conducted in the context of a project of the Centre for Language and Speech Technology at Radboud University[3], commissioned by the RIVM[4]. The goal of the project is to develop a dashboard for monitoring the public stance towards vaccination.

The experiments demonstrate that the co-training framework can be successfully applied to improve sentiment analysis towards vaccination on Dutch Twitter messages. In total a best F_1 score of 0.72 on the negative class and 0.70 on the non-negative class has been reached with the best performing co-training classification system.

2. Background and related work

In this section I am going to present some background literature on sentiment analysis of Twitter data, as well as some work concerning machine learning from both labeled and unlabeled data, i.e. semi-supervised learning. The latter part of the literature review focuses on the specific problem that I only have relatively few labeled examples to train on. I will further discuss the characteristics of the given data in the data section.

2.1 Sentiment Analysis of Twitter data

The Oxford English dictionary defines sentiment analysis as follows:

> "The process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral.[5]"

As such it poses an instance of a classical text mining problem. Successfully performing sentiment analysis provides an important component of understanding the semantics of text. It can be applied to different levels of granularity, as for instance document or sentence level. Most of the work on sentiment analyses focuses on product reviews (Medhat et al. 2014). The difference between tweets and typical product reviews is that tweets are restricted in length and contain attributes like hashtags (indicated by the # symbol) and author references (@author) that product reviews not necessarily show.

Here, I will focus on sentiment analysis on tweet level.

Agarwal et al. (2011) present a system for classifying tweets into positive, negative and neutral. They experiment with two versions of the task: a binary classification task of identifying positive and negative tweets as well as the 3-way task of distinguishing between all three categories. Furthermore, they compared three models: a unigram model as baseline, a feature based model using 100 handcrafted features as well as a tree kernel based model (avoiding the need for manual feature

3. `http://www.ru.nl/clst/`
4. `http://www.rivm.nl/`
5. `https://en.oxforddictionaries.com/definition/sentiment_analysis`

engineering). The authors find that the feature based model is not able to outperform the baseline model while the kernel based model outperforms both. Following a feature analysis they find the combination of prior polarity probabilities of words with their part-of-speech tags to be most valuable for the classification task. These classical natural language features outperform twitter specific features like emoticons or hashtags. They train their models on 5,127 manually annotated tweets balanced over the three categories. Throughout prepossessing they make use of an emoticon dictionary and an acronym dictionary which they designed themselves. The tree kernel approach involves representing each tokenized tweet by certain tags, as for instance stop word, positive emoticon, English word or @-reference. In order to calculate the similarity between tweets all possible sub-trees are compared. For the two-way classfication task they report an average test accuracy of 71.35% using the kernel based method and 60.60% for the three-way classification task. The other two main models yielded lower accuracy scores. Using combinations of the models the authors were able to achieve a 74.61% test accuracy in the two-way task and a 60.83% accuracy in the three-way task.

For my particular problem rather than just distinguishing between positive, negative and neutral sentiment I am interested in identifying positive, negative and neutral sentiment *towards* vaccination. Part-of-speech tags proved to be effective in the work by Agarwal et al. (2011). However, identifying POS-tags requires an additional classification system which might introduce additional error to my system. Moreover, the syntactic dependencies modelled by the POS-tags might already be implicitly present in the data. Therefore I will focus on generic features based on term frequencies to represent the dependencies.

Other related work involves the use of emoticons in tweets for distant supervised learning of sentiment classification of Twitter messages (Go et al. 2009). More specifically the emoticons are used as noisy labels to extract training data representing positive and negative sentiment. The authors experiment with different classifiers, namely naive Bayes, Maximum Entropy and SVM's, as well as different feature sets consisting of unigrams, bigrams, the combination of the two and part-of-speech tags. They report an accuracy of 83% trained on both unigrams and bigrams as the best result. The training data consisted of 1.6M tweets containing equal numbers of positive and negative tweets.

The idea of exploiting emoticons in order to inexpensively label tweets originates in Read (2005). This approach was not feasible for my specific problem as an emoticon alone does not suffice to express sentiment towards a particular topic such as vaccination. However I decided against removing emoticons from the tweets during the preprocessing step in order to preserve the sentiment conveyed by them.

Moreover, Barbosa and Feng (2010) demonstrate the usage of "meta-information" (as they refer to it) in addition to classical features like unigrams for performing sentiment analysis on tweets. These meta-information include part-of-speech tags, as well as prior subjectivity and polarity inferred from the subjectivity lexicon[6] provided in (Riloff and Wiebe 2003). In doing so they propose a two-step classification method: first training a classifier to distinguish between subjective and objective tweets and secondly training another classifier to differentiate between positive and negative sentiment.

Other work on sentiment classification of twitter messages approaches a multilingual framework, while also investigating the portability of the proposed systems from one language into another (Boiy and Moens 2009, Cui et al. 2011, Davies and Ghahramani 2011, Narr et al. 2012). The tweets for my particular problem are in Dutch, i.e. non-English, which in general makes the use of existing polarity lexicons more difficult given that their frequency and extent is usually much lower for non-English languages. However, as I do not consider tweets from another language than Dutch I did not look into previous research done in this field.

6. The lexicon can be found at `http://mpqa.cs.pitt.edu/`

Lastly, recent work by Du et al. (2017) describes a system used for sentiment analysis on HPV vaccine related tweets. They annotate a gold standard set of 6,000 tweets and train and optimize an SVM in a hierarchical classification scheme on various feature sets. The hierarchical method is composed of three consecutive steps: (1) train an SVM to classify tweets into "related" and "unrelated" groups, (2) a second SVM for dividing the "related" group into positive, negative and neutral tweets, and (3) another SVM model to categorize the negative tweets into five more fine-grained classes. This way the authors aimed to tackle class imbalance in the data. They achieve a micro average F_1 score of 0.73% and a macro average F_1 score 0.4986% for the optimized hierarchical SVM classifier on 10-fold cross validation.

Many of the papers discussed here utilize Twitter as platform for inexpensive access to large amounts of unlabeled data. In what follows I am going to present and discuss some literature that deals with text classification from both labeled and unlabeled data.

2.2 Text classification from labeled and unlabeled data

Nowadays, ongoing technological advances allow us to produce and store massive amounts of data of various forms. Some of them reside on social media, while others are stored in databases by companies or governmental organizations like the health care sector or political institutions. These databases often contain invaluable information. However, at the same time the size of these databases makes it infeasible to extract knowledge and information from them by hand. Therefore, there is great need for automatic processing of these data in order to extract structured information and relationships. In case of textual data these structured information can be for instance sentiment, named entities, dates, actors or relations between actors.

Natural language processing and text mining are fields that provide possible tools for the automatic extraction of information from text. The literature on automatic classification and processing of text is vast. However, most of the proposed methods share one commonality: they heavily rely on labeled data for training. As indicated earlier it is very easy and inexpensive to get access to large databases for research and engineering purposes. However, labelling a subset of these data is a time-consuming and tedious process that usually has to be done manually. There is a trade-off between the time and effort put into manually labelling an additional amount of data X and the resulting gain in classification accuracy Y. Considering that large quantities of unlabeled data are easily accessible in most domains and the fact that usually the gain in accuracy is low compared to the costs of labelling, the question arises how to use unlabeled data to improve learning algorithms.

Many attempts have been made to address this issue: Platforms like Amazon Mechanical Turk[7] can be used to recruit workers to label large amounts of data, methods for automatic (often noisy) labelling of data have been proposed (for instance Read, (2005)) and unsupervised or semi-supervised algorithms for various tasks have been developed (Ghahramani 2004, Ko and Seo 2000).

As I was given access to a small amount of labeled tweets and a large set of unlabeled data I focused on semi-supervised learning algorithms that make use of the few available labeled training data and attempt to incorporate the unlabeled data as well.

Co-training is a semi-supervised machine learning paradigm that requires only a few labeled examples based on which two classifiers are trained, in order to label unlabeled examples in an iterative process. The rationale behind using two classifiers is that each of them constitutes a different *view* on the data, thereby employing complementary information thereof. Ideally it would be the case that the two views are *conditionally independent* given the class and *sufficient* in the sense that each classifier is able to accurately predict each class. In the end the most confident predictions on the unlabeled data are used to iteratively train both classifiers. The idea of co-

7. https://www.mturk.com/mturk/welcome

training as described here can be originally traced back to Blum and Mitchell (1998). Co-training has been successfully applied to sentiment analysis in the past (Biyani et al. 2013, Xia et al. 2015).

In the context of classification bootstrapping can be viewed as another semi-supervised learning technique which has for example been successfully applied to the task of learning emotion hashtags from Twitter data (Qadir and Riloff 2013). However, this technique is associated with the risk of being limited to instances that are highly similar to the labeled examples with respect to the system's predictive power. Contrary to this, co-training is designed in a way to increase the generalizability of the classification system by using two (or potentially more than two) classifiers, each of which captures a unique set of features in the data. Furthermore, the combination of multiple classifiers and certainty measures over the assigned labels can further increase generalizability by including those instances as positive examples where some of the classifiers are only weakly confident in their decision. Naturally, this method would require careful tuning of the decision thresholds to prevent the classification system from diverging too far away from correct class assignment.

Wang and Zhou (2007) propose a variant of the classical co-training framework with weaker independence assumptions for the classifiers. Moreover, they discuss and analyze successful termination criteria of the training process. The work conducted by Yu (2014) provides an example where this slightly weaker notion of co-training is applied to opinion mining. Judging whether the independence assumption for two views holds is not always straightforward. Therefore, it is important to consider the feasibility of weaker independence notions also for my own work.

Yu et al. (2011) propose an interesting approach combining Bayesian inference and co-training, thereby automatically providing the property of having a certainty measure associated with each classification. A different way to integrate a Bayesian component into the problem of sentiment analysis is an ensemble method based on Bayesian model averaging (Fersini et al. 2014). It considers both the uncertainty and reliability associated with each single classifier in comparison to the whole ensemble. Although I did not employ a Bayesian approach for incorporating a certainty measure into the co-training framework I found the idea of having such a confidence measure useful.

An alternative to the co-training framework, which still preserves its general idea, is a semi-supervised learning algorithm that is based on the combination of Expectation-Maximization and a naive Bayes classifier (Nigam and Mccallum 2002). In case that the co-training framework would have failed this might have been an interesting starting point for an alternative approach.

3. Data

Twitter is a social networking and real-time microblogging service. Messages posted by Twitter users are called tweets and restricted to a length of 140 characters. Usually tweets can be considered informal, to contain colloquial language, emoticons and numerous spelling errors. Another characteristic of Twitter is the use of hashtags (indicated by a "#" in front of a word) in order to refer to certain topics.

Labeled data is needed to train a classifier on sentiment specific to the context of vaccination. In order to obtain labeled data, the Twitter search API is not the best option as it only permits to retrieve tweets from one week in the past, which would not lead to large amounts of data. However, in the case of Dutch tweets there are options to query tweets from years in the past, such as TwiNl[8] (Sang and van den Bosch 2013), a databse of Dutch tweet ID's dating back to December 2010, and Coosto[9], a commercial platform that offers a payed dashboard to monitor Dutch tweets. For this particular project the RIVM's licence has been used to collect tweets from Coosto.

Within the project of the RIVM and Radboud University, tweets have been collected from both TwiNl and Coosto in several stages. The stages are listed in table 2. The table lists the time period

8. `https://TwiNl.surfsara.nl/`
9. `https://www.coosto.com/nl`

Annotator	Number of tweets annotated
1	6,404
2	4,878
3	73
4	2
5	2
6	1
7	1
8	1

Table 1: Subset of TwiNl2 and Coosto data that have been annotated by independent annotators. The table shows how many tweets have been annotated per annotator. In total 6,541 of the 8,260 tweets have been annotated by more than one person.

of retrieval, the corresponding queries and the number of tweets retrieved. The number of tweets in the table refers to the number of tweets after removing retweets and tweets containing URLs[10]. The URL heuristic has been used to distinguish between tweets representing news, i.e. not containing sentiment and opinionated tweets, i.e. tweets potentially containing sentiment.

Parts of all three databases have been labeled. The Relevancer tool (Hürriyetolu et al. 2016) has been used to cluster tweets, i.e. group tweets that are highly similar in their formulation. After clustering, 8,395 tweets of the TwiNl1 dataset have been labeled by one person as "positive", "negative", "neutral" or "don't know". The tweets from the most coherent clusters have been used for labelling. Consequently, tweets belonging to one cluster were assigned the same label. For clusters in which the tweets were not similar in sentiment, the label "not unified" was given. This subset of labeled data has been used as test set.

Although the clustering of tweets can speed up the annotation (as more tweets can be annotated at once), it does lead to a bias of the annotated tweets: only the tweets that have similar neighbours are included. In order to obtain a more diverse set of labeled tweets, for parts of the TwiNl2 and Coosto dataset single tweets were annotated by independent raters. In total, a subset of 8,260 tweets have been annotated by 1 up until 8 annotators. Table 1 shows how many tweets have been annotated per annotator. In total, 6,541 of the 8,260 tweets have been annotated by more than one person. This subset of labeled data has been used as training data.

As negative sentiment towards vaccination is considered to be more relevant than positive or neutral sentiment for the purpose of the study initiated by the RIVM, I also prepared a binary classification task distinguishing between negative and non-negative sentiment by merging the positive and neutral tweets for the labeled tweets from the TwiNl1 dataset. I disregarded tweets that were labeled as "don't know". In order to decide on a final label for a tweet in case that it has been annotated by multiple raters (second subset of labeled data) the majority label has been assigned. In case that a majority could not be established the tweet was labeled as "don't know".

Considering only the tweets that have been annotated by the first two annotators (compare table 1) and disregarding tweets that have been labeled as "don't know" 2,713 tweets have been used to calculate interrater reliability measures. I used Cohen's Kappa κ and Krippendorff's α. Both measures converged on the same values, namely 0.522 for the multiclass problem and 0.618 for the binary classification task. In conclusion, agreement can be considered to be substantial. I only used this subset of the annotated tweets as both measures are defined to be calculated for two raters, so the measures reflect agreement between raters and not directly between assigned labels independent of the annotator.

10. This does not hold for the TwiNl1 dataset.

Another random subset of the collected data has been made available to me as “unsupervised”, i.e. unlabeled data. In total I was given 178,569 of these tweets, while 15,586 of them were explicitly retrieved for keywords relating to vaccination whereas the other tweets were also related to diseases. An argument could be made that the latter one are a better representation of the targeted problem domain and should therefore perform better within the co-training framework. In order to rule out such an effect I tested the effect of this difference in tweets in one of my experiments.

Table 3 summarizes some important characteristics of the three data splits, used for training, testing and unsupervised learning respectively. As indicated earlier retweets and URLs had not been removed from the TwiNl1 data which served as basis for the test set. Therefore I added an extra preprocessing step to remove these in order to account for the difference in training and test data. All numbers reported in the table refer to the datasets after removing retweets, URLs and “don’t know” tweets. It can be seen that the training set displays imbalance in the number of positive and negative tweets. I will discuss experiments addressing this imbalance in the following sections.

Figures 1-4 display the most frequent 50 hashtags and bigrams (after preprocessing and excluding stop words) for the four datasets listed in table 3. It can be observed that the individual datasets cover roughly the same range of topics. The assumption that the training and test data are similar and possibly generated by the same abstract underlying distribution is an important perquisite for successful performance on the test data. Figure 5 shows a wordcloud for the positive tweets and a wordcloud for the negative training tweets. A wordcloud represents the most frequent words in a given text and adjusts the size of the words proportionally to their frequency. To create the wordclouds I have used the preprocessed[11] tweets after removing stop words and @-references.

From this representation it can be inferred that there is large overlap between the words used to describe negative and positive tweets. Terms related to vaccination and diseases appear to be abundant in both categories. As the tweets were specifically retrieved for keywords relating to vaccination and diseases this result was to be expected. However this means that the classifiers will have to be sensitive enough to pick out small differences, i.e. differences within lower frequency words, between the classes. Just by examining the two wordclouds it is difficult to make out a clear distinction between the positve and negative tweets. However, the wordcloud for the negative tweets contains terms like “bijwerkingen” (side effects) or “slecht” (bad) while the positive wordcloud contains terms like “goed” (good). A well trained algorithm should be able to utilize these differences.

4. Methods: semi-supervised sentiment classification

In this section I am going to describe the co-training framework that I used for sentiment classification of Dutch twitter messages. First, I will discuss the preprocessing pipeline that I designed to prepare the tweets for classification. Secondly, I will illustrate the co-training algorithm and its assumptions. All parameter tuning described in this section has been applied to a subset of 10% of the training data solely held out for this purpose.

All parts of the project are implemented in Python 2.7.

4.1 Preprocessing

Prior to classification the tweets have been tokenized based on spaces between words while not splitting words containing “-” or “ ’ ”. All tokens have been lowercased. Addtionally, hashtags (indicated by the # symbol), author mentions (indicated by @), numbers and emoticons have been preserved using regular expressions. The tweets were stripped off all other punctuation marks. Due to the shortness of tweets this rather minimalistic preprocessing design has been chosen to preserve as much expressiveness as possible. The use of hashtags, emoticons and also spelling errors can be a very important feature to distinguish between the different sentiment classes. Following the same line of reasoning I decided against removing stop words. Finally, I settled on an approach that

11. The preprocessing will be explained in the next section

Dataset	Retrieval Period	Number of tweets[1]	Queries[2]
TwiNl1[3]	2011-06 2016	170,092	queries divided into queries relating to vaccination and to diseases: **vaccination:** vaccinatie, inenting, vaccin **diseases:** bmr, dktp, hpv, rotavirus, buikgriep, waterpokken, bmrv-vaccin, gordelroos, hepatitis a, herpes zoster, windpokken, windpokken, meningkokken b, griepprik, griepvaccinatie, influenza
Coosto	2011-08 2016	133,567	bmr, difterie, dktp, gordelroos, hepatitis b, hib-ziekten, hpv, kinkhoest, mazelen, meningokokken, pneumokokken, polio, rodehond, rotavirus, tetanus, waterpokken, (vaccinatie OR vaccin OR rijksvaccinatieprogramma OR intenting) - (dier OR teken OR pluim OR vastgoed OR hond OR gewas OR landbouw OR griepprik)
TwiNl2	01.01.2012 - 08.02.2017	103,381	queries divided into queries relating to vaccination and to diseases: **vaccination:** vaccinatie, vaccin, vaccineren, rijksvaccinatieprogramma, vaccinatieprogramma, intenting, inenten **diseases:** bmr, difterie, dktp, gordelroos, hepatitis, hpv, kinkhoest, mazelen, meningokokken, pneumokokken, polio, rodehond, rotavirus, tetanus, waterpokken, bof, baarmoederhalskanker

[1] Number of tweets after removing retweets and tweets containing URLs.
[2] with and without hashtags, case insensitive
[3] Retweets and URLs have not been removed from these data

Table 2: Retrieval of Dutch tweets based on keywords relating to diseases and vaccination.

involves using generic features extracted from the minimally preprocessed tweets rather than highly specific features like part-of-speech tags.

	Train	Test	Unsupervised[1]	Unsupervised vaccination
Avg. number of hashtags[2]	0.60	0.24	0,52	0,42
Number of positives	2,373	330	-	-
Number of negatives	814	1,072	-	-
Number of neutrals	1,108	1,059	-	-
Size[3]	4,295	2,416	162,983	15,586

[1] Unlabeled data excluding the set specifically relating to vaccination
[2] per tweet
[3] After removing retweets, URLs and tweets labeled as “don’t know”.

Table 3: Overview over the three datasets used for training, testing and unsupervised learning.

Original tweets	Preprocessed tweets
.@KBrinkman_OLVG Kosteneffectiviteitsonderzoek loopt. Tussentijds bepleit Jan v. Bergen @soaaids sowieso hpv-vaccin voor startende homo’s.	@kbrinkman_olvg kosteneffectiviteitsonderzoek loopt tussentijds bepleit jan v bergen @soaaids sowieso hpv vaccin voor startende homo’s
Net eerste vaccinatie hepatitis a+b gehad.. En nu al een stijve arm.. Fijn dit :/	net eerste vaccinatie hepatitis a b gehad en nu al een stijve arm fijn dit :/
@TrampsLikeMe ik dacht ’ ze gaat toch niet de keuze voor vaccinatie vergelijken met keuze over (te gemakkelijk) omgaan met zindelijkheid.	@trampslikeme ik dacht ze gaat toch niet de keuze voor vaccinatie vergelijken met keuze over te gemakkelijk omgaan met zindelijkheid
Vandaag 10.30 Hepatitis B vaccinatie #ROCA12VELP	vandaag 10.30 hepatitis b vaccinatie #roca12velp

Table 4: A few examples of original raw (left column) tweets and preprocessed (right column) tweets.

Table 4 displays a few examples of tweets before and after preprocessing has been applied.

4.2 Co-training framework

My goal was to develop a classification system for analyzing the sentiment of Dutch Twitter messages towards vaccination. I utilized unsupervised, i.e. unlabeled tweets in order to increase the performance of my classification system. To do so I implemented the co-training algorithm (Blum and Mitchell 1998) which uses two distinct views (x_1 and x_2), i.e. different feature sets, of the data in order to iteratively label and retrain one or two classifiers (h_1 and h_2) on unlabeled instances (U) of the training data. Algorithm 1 illustrates the co-training algorithm as described in the original paper by Blum and Mitchell (1998). The co-training algorithm is initialized with a set L of labeled training examples (consisting of data x and labels y), a set U of unlabeled examples, two classifiers

```
Data: $L$, a set of labeled training examples; $U$, a set of unlabeled examples
Result: Co-trained classifiers $h_1$ and $h_2$
initialization;
create a pool $U'$ of examples by choosing $u$ examples at random from $U$;
while still elements in $U \wedge$ not exceeded the maximum number of iterations $k$ do
    $k = k + 1$;
    use $L$ to train a classifier $h_1$ that considers only the $x_1$ view of $x$;
    use $L$ to train a classifier $h_2$ that considers only the $x_2$ view of $x$;
    select from $U'$ $p$ most confidently labeled examples by $h_1$ as positive examples;
    select from $U'$ $n$ most confidently labeled examples by $h_1$ as negative examples;
    select from $U'$ $p$ most confidently labeled examples by $h_2$ as positive examples;
    select from $U'$ $n$ most confidently labeled examples by $h_2$ as negative examples;
    add these self-labeled examples to $L$;
    randomly choose $2p + 2n$ examples from $U$ to replenish $U'$;
end
```

Algorithm 1: Co-training algorithm as presented by Blum and Mitchell (1998).

Co-training parameters	Explanation	Default value in original paper
U'	size of pool of unlabeled data	75
k	number of iterations	30
p	number of most confidently positively labeled examples to add	1
n	number of most confidently negatively labeled examples to add	3
c_1	lower confidence bound of classifier 1	-
c_2	lower confidence bound of classifier 2	-

Table 5: Co-training parameters and default values. c_1 and c_2 have not been used in the original paper.

h_1 and h_2 and two distinct views x_1 and x_2 of the data (including both the labeled data L and the unlabeled data U as x_1 and x_2 just represent transformations of the data into different feature spaces). In each iteration h_1 is trained on the x_1 view of L and h_2 on the x_2 view of the same data. Both classifiers label a subset U' of the unlabeled data U. From U' the p (n) most confidently labeled examples that have been assigned by h_1 to the positive (negative) class are added to the training dataset L. The same is done for the second classifier h_2. In total $2p + 2n$ examples are added to L in each iteration. After each iteration $2p + 2n$ examples are randomly chosen from U to replenish U'. The co-training algorithm terminates if the maximum number of iterations k is exceeded or if U is empty.

After training both co-training classifiers on the different views, the classifiers are used to label the test data. For all cases where the classifiers agree the label is immediately assigned. In case of disagreement the sum over the class probabilities predicted by each classifier is computed and the label corresponding to the maximum probability is assigned.

Table 5 displays the parameters of the co-training algorithm and their default values in the paper by Blum and Mitchell (1998). Blum and Mitchell did not use a minimally required confidence level on the assigned label before adding a newly labeled instance to the training pool. Instead they add the p and n most confidently labeled examples without an additional step of evaluating the confidence of the assigned label.

I propose a variant of the co-training algorithm in which I incorporate such a lower confidence bound for each classifier. Furthermore, I use two sets of labeled training examples L_1 and L_2, one for each view, instead of just one. In the beginning both are initialized with the available labeled training examples. Throughout co-training I add the examples labeled based on view 1 to L_2 and the examples labeled based on view 2 to L_1. This way I want to increase the generalizability of the two classifiers. The idea being that only adding the instances labeled by a different classifier - instead of adding all newly labeled instances - directs more attention to labeled instances that are structurally different from what the classifier has been trained on before. This way it enables the algorithm to learn to recognize and correctly classify those structurally different examples and moreover to discover more of those examples within the set of unlabeled data that add novel aspects for learning to the respective classifier.

In summary this particular adaptation is supposed to increase the informativeness of the labeled examples added to each respective classifier to ensure that learning progresses.

Algorithm 2 displays these adaptations to the original framework. The differences between the adapted and the original version (algorithm 2) can be summarized in three points:

1. Each classifier is trained on its own specific training dataset (L_1 and L_2), respectively
2. The most confidently labeled examples by each classifier are only added if the conditional probability $P(c = C|x_j)$ of a labeled instance (represented by its view x_j) belonging to a particular class C (either positive or negative) exceeds a predefined (classifier specific) threshold
3. The instances labeled by classifier h_1 are added to the training dataset L_2 of h_2 and vice versa

The prediction on the test data by the co-training system, i.e. by the individual co-training classifiers, is realized in the same way as for algorithm 1.

Moreover, I implemented the co-training algorithm to also allow for more than two classes, i.e. non-binary problems.

4.3 Assumptions

As outlined in the introduction the co-training algorithm makes two important assumptions:

Assumption 1 (*Conditional Independence*)**.** The two views, i.e. the feature sets are conditionally independent given the class label.

Assumption 2 (*Sufficiency*)**.** Each view is in itself sufficient for correct classification.

If the conditional independence assumption holds, on average each added example should be as informative as a random example from the underlying "population" of examples. Assumption 2 states that each feature set, given a large amount of training examples, is in itself sufficient to allow for accurate classification of the test data. Consequently incorrect classifications would not be a result of the chosen representation of the data being inappropriate and inadequate but rather a result of the limited amount of training data.

4.4 Selecting views and classifiers

In order to select the two distinct views of the data for co-training I considered a range of different feature sets. As indicated earlier I focused on generic features which are not highly problem specific, i.e. features that can be regarded as a different representation of the data. For example, in contrast to explicitly using part-of-speech tags as features I am relying on the classifiers to learn the syntax

```
Data: $L$, a set of labeled training examples; $U$, a set of unlabeled examples
Result: Co-trained classifiers $h_1$ and $h_2$
initialization;
create a pool $U'$ of examples by choosing $u$ examples at random from $U$;
initialize two sets $L_1$ and $L_2$ with all available labeled training data $L$;
set $p$ and $n$ to represent the proportions of positive and negative instances in the labeled data
  $L$;
while still elements in $U \wedge$ not exceeded the maximum number of iterations $k$ do
    $k = k + 1$;
    use $L_1$ to train a classifier $h_1$ that considers only the $x_1$ view of $x$;
    use $L_2$ to train a classifier $h_2$ that considers only the $x_2$ view of $x$;
    select from $U'$ $p$ most confidently labeled examples by $h_1$ as positive examples, if
      $P(c = p|x_1) > c_1$;
    select from $U'$ $n$ most confidently labeled examples by $h_1$ as negative examples, if
      $P(c = n|x_1) > c_1$;
    select from $U'$ $p$ most confidently labeled examples by $h_2$ as positive example, if
      $P(c = p|x_2) > c_2$;
    select from $U'$ $n$ most confidently labeled examples by $h_2$ as negative examples, if
      $P(c = n|x_2) > c_2$;
    add the self-labeled examples by $h_1$ to $L_2$;
    add the self-labeled examples by $h_2$ to $L_1$;
    randomly choose $2p + 2n$ examples from $U$ to replenish $U'$;
end
```

Algorithm 2: Adapted co-training algorithm.

implicitly from the chosen representation. All features described in the following have been extracted from the preprocessed tweets.

I extracted the following feature sets[12]:

TF-IDF (word) Term frequency - inverse document frequency assigns a vector of numerical values to each document. Each entry in the vector represents one token in the corpus. The n-th entry in the TF-IDF vector for document x represents how important that n-th word in the corpus is in that document. The score is determined by the product of the term frequency (how often does a term occur in the document) and inverse document frequency which reflects how much information that word provides for identifying the particular document (in how many documents does the term occur, i.e. very frequently as for instance "the" vs. rare terms). Cosine similarity between TF-IDF vectors of documents can be used to assess the similarity between two documents.

A TF-IDF vectorizer has been trained on word level on all available training tweets (after preprocessing) in an n-gram range of 1-2. This particular range allows bi-words like "bible belt" as well as single words to be represented as features. This resulted in feature vectors of $1 \times 51{,}978$ dimensionaility per tweet. The machine learning library *scikit-learn* (Pedregosa et al. 2011) has been used to extract both the TF-IDF features on word and character n-gram (see below) level.

12. I also experimented with polarity values (extracted with the python library *pattern*) (Smedt and Daelemans 2012)) and descriptive statistics of the tweets like average number of words, its standard deviation, average number of hashtags and average word length (plus standard deviation). However, initial classification attempts based on these features on the development data lead to much poorer results than the other feature sets. Consequently I decided to discard these attempts and ideas.

TF-IDF (character) A TF-IDF vectorizer has been trained on character n-gram level on all available training data (after preprocessing) in an n-gram range of 2-5. Choosing character n-grams instead of word n-grams for a TF-IDF representation has the advantage of automatically accounting for spelling variants and spelling errors. The dimensionality of the feature vectors is 1×278,412.

Word2vec Word2vec embeddings (Mikolov et al. 2013) are distributed representations of words. Each word is represented by a feature vector with a predefined number of features. The embeddings are produced by training a two-layer neural network on a large corpus of text. During training for each word in the corpus a context of n words around it is considered. Thereby words that share a common context in the corpus are also closely located in the feature space that is spanned by the resulting word embeddings. The goal of Word2vec is to use word co-occurence statsitics in order to learn linguistic context and thereby semantics of words. The embeddings have been trained with *gensim* (Řehůřek and Sojka 2010) on all available tweets, including training, testing and unsupervised data. The test data have been also used for training simply to increase the amount of training data in order to get a better feature representation. The parameters used for training the Word2vec representations were: feature dimensionality = 500, window size = 3, minimum word count = 1, number of cores = 3, number of iterations = 100, training algorithm is CBOW (sg = 0); other parameters were set to default values. Usually Word2vec embeddings are trained on sentence level. However, given the nature of the problem I decided to train on tweet level. To arrive at one feature vector per tweet, Word2vec embeddings for all words in a tweet have been averaged. As addtional features I added the absolute number of positive entries in the resulting feature vector, absolute number of negative entries, the mean of the current feature vector, its standard deviation, as well as the sum over all entries. The resulting feature vector per tweet therefore had a dimensionality of 1×505.

Doc2vec As an alternative to Word2vec *gensim* also offers Doc2vec. While Word2vec is usually used to measure similarity between words, Doc2vec uses word level embeddings with a pooling approach (similar to the averaging described above) to infer similarity between whole documents (Le and Mikolov 2014). The following parameter settings have been used for training: feature dimensionality = 500, window size = 3, minimum word count = 1, number of cores = 3, number of iterations = 100; other parameters were set to default values. The same statistics as in the Word2vec case have been added on top of the Doc2vec vectors, resulting in feature vectors of size 1×505.

It is difficult to make an a priori decision for which combination of the presented feature sets the conditional independence assumption holds. The intuition behind the assumption is to choose the most distinct features to get complementary "views" of the same data.

To evaluate the conditional independence assumption, I plotted the first two principal components of the feature sets. As each of the described sets spans a high dimensional feature space, a reduction to just two principal components could be viewed as a severe misrepresentation. However, there are no empricical metrics available to estimate the conditional independence. A graphical representation in 2D gives at least some indication of the differences in feature sets.

Figures 6 and 7 display the two first principal components of the Word2vec, Doc2vec and the two TF-IDF variants. Red dots indicate negative tweets while blue dots represent non-negative tweets. I only focused on the distinction between negative and non-negative as this was more relevant for my particular problem than a more fine grained distinction. Not surprisingly this visualization emphasizes that within varieties of word embedding features (figure 6) and within varieties of TF-IDF features (figure 7) the conditional independence assumption does not hold. However, between word embedding and TF-IDF features it looks as if the views are independent given the class label. Therefore I decided to choose a combination of word embedding and TF-IDF features as views for co-training. At this point it is however important to note that based on the first two principal components the two classes do not seem to be linearly separable. This can mean that the two classes become separable once all dimensions are included or that a non-linear classifier is required to solve this problem.

In order to make a final decision on the best combination of feature sets I ran experiments that tested multiple classifiers on each set of features. The underlying hypothesis is that to reach good performance with the co-training framework requires a combination of classifiers and features that yield good results in standalone classification. Moreover, the experiments provide support in choosing view-classifier pairs that satisfy the sufficiency assumption. To find the optimal classifiers randomized search (as implemented in the Python library *scikit-learn*) with 10-fold cross validation has been performed on the development data for both the binary and the multiclass problem. The following classifiers have been tested:

- Multinomial naive Bayes
- Random forrest
- K nearest neighbors
- AdaBoost
- ExtraTrees
- Gradient boosting
- Logistic regression
- Support vector machine

All classifiers are preimplemented in the Python library *scikit-learn.* In addition to these classifiers a baseline classification using the *pattern* library in Python has been integrated. The pattern baseline classifies a tweet as positive in case that the overall polarity score returned by the *sentiment()* function is positive, negative for a negative score and neutral otherwise. As the implementation of Doc2vec in *gensim* did not allow the flexibility I desired for my system with regard to randomizing data and calculating Doc2vec representations on different subsets of the test data, I decided to discard this approach as the cost to benefit ratio did not seem to pay off given the time frame of my project.

The RIVM expressed interest in a system that is specifically tailored to identitfy negative sentiment towards vaccination. Furthermore, precision, i.e. the number of true positives among the tweets labeled as negative, is considered to be more important than recall for the purpose of the research conducted by the RIVM. Therefore, instead of using the harmonic mean between precision and recall (F_1 score) as optimization metric, I use the F_β score which is defined as weighted harmonic mean of precision and recall. Setting $\beta = 0.5$ results in an F-score that assigns more weight to precision than recall. This results in the following formula for the $F_{0.5}$ score:

$$F_{0.5} = (1 + 0.5^2) * \frac{precision * recall}{(0.5^2 * precision + recall)} \tag{1}$$

$$F_{0.5} = \frac{recall + 0.5^2 * recall}{0.5^2 + \frac{recall}{precision}} \tag{2}$$

From equation (2) it can be inferred that a β coefficient of 0.5 discounts the effect of recall.

Figure 8 displays the results for all classifiers on the preselected feature sets. Prior to classification the classifiers have been tuned based on randomized search while optimizing for $F_{0.5}$. I held out another 10% of the training data and retrained the optimized classifiers on all remaining training data while testing on the second held out development set. The results for the binary classification can be viewed in figure 8 (a) and the results for the multiclass task in 8 (b). The results for both tasks show that the different classifiers tend to cluster around classes. Furthermore, some classifiers show a large discrepancy between recall and precision. It is important to note, that although the classifiers have been optimized for $F_{0.5}$ the F_1 score is displayed here. Moreover, there is a larger noticeable variance within results for the negative class than for the other classes.

For the remainder of this thesis I will only discuss the results of the binary classification task. The difference in classification performance on the negative class across both tasks is merely noticeable. Therefore, I do not want to overwhelm the reader with numerous experiments for both tasks in the co-training section and rather concentrate on the binary problem.

The results demonstrate that the pattern baseline serves as a surprisingly strong one with an F_1 score slightly below 0.4 for the negative class and a score of roughly 0.7 for the positive class. Based on the performance of the classifiers I selected the **ExtraTrees classifier** in combination with the **TF-IDF (word)** features and the **logistic regression classifier** in combination with the **Word2vec features**. A feasible alternative to the ExtraTrees classifier would have been the SVM. However, training of the SVM is slow in comparison to other classifiers and most importantly the *sklearn* implementation does not allow the SVM to output probabilities instead of class labels which is crucial for co-training.

Table 6 displays the optimized classifiers and their parameters found throughout the randomized search for the binary classification task. The second row illustrates the explored parameter ranges. All parameters (but the n_jobs, max_iter, and n_estimators) that do not occur in the second row have been set to default values. I manually enabled multicore processing by setting n_jobs = -1 and I set the number of maximum iterations for the logistic regression classifier to 300 and the number of estimators for the ExtraTress classifier to 100. For the latter two parameters it holds that more iterations and more estimators lead to better classification results. However, my time for the co-training experiments was limited so I decided to constrain these parameters while still allowing for reasonable accuracy. The logistic regression classifier with the settings listed in table 6 implements regularized logistic regression using the cross-entropy loss. The ExtraTrees classifier is an ensemble classifier (also called meta estimator) which fits numerous randomized decision trees on subsets of the dataset. Averaging over these sub-results is used to improve accuracy and as a means to overcome overfitting.

After determining the view-classifier pairs for the binary classification task, the lower confidence bounds c_1 and c_2 for both classifiers still had to be tuned on the development set. A co-training experiment using the selected view-classifier pairs and the default parameters (compare table 5) from the original co-training paper was run to compute the optimal confidence bounds. p and n however, were set to represent the proportions of negative and non-negative tweets in the data. The experiments used the adapted version of the co-training algorithm (algorithm 2).

As in a binary classification task an instance has to be labeled with a probability of at least 0.5 to be considered to belong to a specific class the following thresholds were explored for both classifiers: [0.5,0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9, 0.95]. More specifically two co-training experiments were run:

Experiment 1 Determines the optimal threshold for the ExtraTrees classifier in combination with the TF-IDF (word) features. The lower confidence bound c_2 of the logistic regression classifier is set to 0.5. p and n are set to represent the proportions of the negative and non-negative tweets in the labeled training data. c_1 is varied and the results are stored for each threshold. Only the set of “relevant” unsupervised data has been used.

	LogReg + Word2vec	ExtraTrees + TF-IDF (word)
Optimized parameters	C = 1, class_weight = "balanced", dual = False, fit_intercept = True, intercept_scaling = 1, max_iter = 300, multi_class = "multinomial", n_jobs = -1, penalty = "l2", random_state = None, solver = "lbfgs", tol = 0.0001, verbose = 0, warm_start = True	bootstrap = False, class_weight = "balanced", criterion = "gini", max_depth = None, max_features[1] = 0.4, max_leaf_nodes = None, min_impurity_split = 1e-07, min_samples_leaf = 1, min_samples_split = 2, min_weight_fraction_leaf = 0.0, n_estimators = 100, n_jobs = -1, oob_score = False, random_state = None, verbose = 0, warm_start = False
Parameter ranges	C: [0.001, 0.005, 0.01, 0.5, 1, 5, 10, 50, 100, 500, 1000], solver: ['newton-cg', 'lbfgs', 'sag'], multi_class: ["ovr", "multinomial"]	max_features[1] = [0.4, 0.44285714, 0.48571429, 0.52857143, 0.57142857, 0.61428571, 0.65714286, 0.7, 0.74285714, 0.78571429, 0.82857143, 0.87142857, 0.91428571, 0.95714286, 1.]
Parameter explanation	C : float, inverse of ruglarization strength; solver : algorithm used for optimization; multi_class : determines whether a binary ('ovr') problem is fit for each label or whether the multinomial loss is minimized	max_features: if float, then max_features is a percentage and int(max_features * n_features) features are considered at each split.

[1] I decided to not consider fewer than 40% of the features. For that reason the explored range only starts at 0.4.

Table 6: Optimized classifiers for the binary classification task (first row), explored parameter ranges for those classifiers (second row) and explanation of the varied parameters (third row).

Experiment 2 Determines the optimal threshold for the logistic regression classifier in combination with the Word2vec features. The lower confidence bound c_2 of the ExtraTrees classifier is set to 0.5. p and n are set to represent the proportions of the negative and non-negative tweets in the labeled training data. c_1 is varied and the results are stored for each threshold. Only the set of "relevant" unsupervised data has been used.

Figure 9 displays the results of the two thresholding experiments. Based on figure (a) an optimal threshold of 0.55 for the ExtraTrees classifier can be inferred. Figure (b) shows an optimal threshold of 0.5 for the logistic regression. Performance is assessed in terms of the F_1 score. Interestingly, the curves for the negative and non-negative class are roughly parallel to each other. In conclusion the chosen thresholds can be considered optimal for both classes. It is important to note that the threshold of 0.5 for the logistic regression essentially means that the same selection criterion for adding unlabeled instances applies for this classifier as for both classifiers in algorithm 1: in every iteration the maximum number of instances (which is constrained by p, n and U') labeled by logistic regression will be added to the training dataset.

The following section will describe the experiments designed to analyze the performance of the co-training framework.

5. Experiments

It is known that the maximum obtainable performance of co-training is limited in various ways (Pierce and Cardie 2001). One such limitation is a reported degradation effect after a certain number of iterations of the learning procedure (Wang and Zhou 2007). In order to monitor the development of the classification system, the classification results in terms of precision, recall and F_1 score are stored after each iteration. The resulting learning curves will be presented and discussed in the results section.

For all of the following experiments the number of iterations k was set to 100 (as the classification results after each iteration were stored I did not test higher numbers of iterations due to time constraints) and n and p were determined based on the relative proportion of negative and non-negative instances in the training data. I decided against performing an exhaustive search for the optimal settings of p and n as I found the initialization to be a reasonable representation of the training data. However, I set up two additional experiments to test the influence of p and n which will be illustrated below. I focused on varying the pool size U': Each of the experiments was run with the following pool sizes $U' \in [10, 25, 50, 75, 100, 150, 200, 500, 1000]$. In conclusion, each experiment ran 9 times for 100 iterations, resulting in $9 * 100 = 900$ classification results per experimental condition in case that algorithm 2 did not terminate earlier due to exhaustion of the unsupervised data U.

The different experimental conditions varied on four levels:

1. **training data**: Either all of the available training data (condition all) or a stricter variant (condition strict) have been used in which only those instances were considered where all annotators (at least two) instead of just the majority agreed. The larger set consists of 3,436 training instances (training instances left after removing the development data) and the strict version of 2,002 instances.

2. **unsupervised data**: As outlined in the data section the total amount of unsupervised, i.e. unlabeled data, consists of 178,569 of which 15,586 specifically relate to vaccination and not to vaccination and diseases. The general setup of the experiments uses the unsupervised data specifically relating to vaccination. However, one of the experiments uses all of the unsupervised data, that is a mix of tweets specifically relating to vaccination and those loosely relating to vaccination. The goal of this experiment is to test whether performance varies depending on the set of unsupervised tweets (either condition all or condition relevant).

3. **downsampling**: As the training data show an imbalance between negative and non-negative data an option for downsampling has been implemented. Downsampling works in two modes: “NoNeutrals” (condition NoNeutrals) and “DropOut” (condition DropOut). In the first mode all neutral tweets are excluded, while in the latter mode tweets from the majority class (the non-negatives) are dropped out with a probability of 0.4.

4. p **and** n: Both parameters are set to represent the relative proportions of non-negative and negative tweets in the training data. In a variant of this baseline setting both parameters have been multiplied by 10 in order to test the influence of a larger number of instances added per iteration. Yet another variant tests the effect of only adding negative instances as a means to overcome the imbalance in the training data.

	training data	unsupervised data	downsampling	p and n
Experiment 1	all	relevant	none	4 & 1
Experiment 2	strict	relevant	none	4 & 1
Experiment 3	all	all	none	4 & 1
Experiment 4	all	relevant	NoNeutrals	3 & 1
Experiment 5	all	relevant	DropOut	3 & 1
Experiment 6	all	relevant	none	40 & 10
Experiment 7	all	relevant	none	0 & 10

Table 7: Co-training experiments for the binary classification task. The first row represents the baseline experiment and all consecutive experiments differ only on one level in order to solely test the effect of the imposed change. The changes per experimental condition are marked in green.

Table 7 summarizes the executed co-training experiments for the binary classification task. Experiment 1 serves as a baseline experiment and all other experiments only differ with regard to one of the four levels from that baseline experiment. This is crucial in order to be able to explain changes in the performance as a result of the experimental condition rather than of an interplay of multiple changes at the same time.

The system has been implemented to flexibly allow to change all of the listed options.

6. Results

I will present and briefly discuss the results for each of the seven experimental conditions. Table 8 displays the best F_1 score across all pool sizes U' and all iterations per experiment for the negative class. For this best F_1 score the corresponding pool size U', the number of iterations k, and precision and recall scores are shown. Furthermore, the average best (the best within the 100 iterations) F_1 score across all tested pool sizes U' and its standard deviation is displayed. Lastly, the last column shows the number of instances added[13] by each classifier up until the best performing number of iterations. Table 9 displays the same statistics for the non-negative class. However, I will not discuss these results in detail as I geared the co-training framework to perform well in the classification of negative tweets. In total it can be observed that the best achievable F_1 score for the non-negative class is below that of the negative class. Moreover, for all but one experimental condition the best score was achieved before adding any unlabeled instances, i.e. after zero iterations. In comparison to the negative class a larger variance associated with the average best F_1 score across pool sizes is noticeable. Lastly, precision is in general higher than recall for the non-negative class while the opposite is true for the negative class.

In what follows I will briefly describe each experiment and compare each experimental condition to the baseline condition (experiment 1). Thereafter I am going to present and discuss the results for the best performing co-training setup in more detail and compare it to the pattern baseline, as well as to the performance of the individual co-training classifiers in standalone classification.

Experiment 1 - baseline The baseline experiment used all training data, only the relevant unsupervised data, no downsampling and values for n and p that represented the actual proportions of the negative and non-negative samples in the training data, namely 4 and 1. The best F_1 score across all pool sizes U' and iterations for that particular setup was 0.60 (with corre-

13. The maximum number of instances that can be added per classifier per iteration is 5 in experiment 1-5, 50 in experiment 6 and 10 in experiment 7. However, the maximum number is always contrained by the pool size U' in case that $U' < (p + n)$.

	best F_1 score[1]	Pr	R	U'	k	avg. F_1[2] ± std	instances added[3]
Experiment 1	0.60	0.50	0.75	25	11	0.58 ± 0.012	55/55
Experiment 2	0.54	0.46	0.64	10	41	0.51 ± 0.014	234/234
Experiment 3	**0.65**	**0.56**	**0.78**	**1000**	**95**	**0.64 ± 0.003**	475/475
Experiment 4	0.62	0.49	0.84	10	5	0.59 ± 0.011	18/20
Experiment 5	0.61	0.49	0.79	75	69	0.61 ± 0.002	276/276
Experiment 6	0.60	0.50	0.75	10	7	0.57 ± 0.013	60/70
Experiment 7	0.61	0.47	0.85	150	17	0.60 ± 0.004	170/170

[1] of the **negative** class
[2] Averaged across the best F_1 scores for each U'.
[3] by both classifiers up until the best iteration in the format ExtraTrees/Logistic Regression.

Table 8: The table shows the best F_1 score for the negative class per experiment across all pool sizes U' and iterations. For the best F_1 score the accompanying pool size U', the number of iterations, and precision and recall scores are reported. Furthermore, the average F_1 score across all best F_1 scores per pool size U' and its standard deviation is displayed, as well as the number of instances added by each classifier up until the best performing number of iterations. The best result is highlighted in bold.

	best F_1 score[1]	Pr	R	U'	k	avg. F_1[2] ± std	instances added[3]
Experiment 1	0.61	0.64	0.58	10	0	0.58 ± 0.024	0/0
Experiment 2	**0.63**	**0.62**	**0.64**	**10**	**0**	**0.60 ± 0.022**	**0/0**
Experiment 3	0.62	0.74	0.53	1000	19	0.60 ± 0.015	95/95
Experiment 4	0.51	0.69	0.41	10	0	0.46 ± 0.022	0/0
Experiment 5	0.57	0.68	0.49	10	0	0.50 ± 0.026	0/0
Experiment 6	0.60	0.64	0.56	10	0	0.49 ± 0.041	0/0
Experiment 7	0.60	0.65	0.56	10	0	0.44 ± 0.079	0/0

[1] of the **non-negative** class
[2] Averaged across the best F_1 scores for each U'.
[3] by both classifiers up until the best iteration in the format ExtraTrees/Logistic Regression.

Table 9: The table shows the best F_1 score for the non-negative class per experiment across all pool sizes U' and iterations. For the best F_1 score the accompanying pool size U' , the number of iterations, and precision and recall scores are reported. Furthermore, the average F_1 score across all best F_1 scores per pool size U' and its standard deviation is displayed, as well as the number of instances added by each classifier up until the best performing number of iterations. The best result is highlighted in bold.

sponding $U' = 25$ and $k = 11$). The average best[14] F_1 score across all pool sizes was slightly lower with 0.58 ± 0.012. Neither in this experimental setup nor in the following ones did the algorithm terminate before the maximum number of iterations ($k = 100$) was reached. Another termination criterion was that the number of unsupervised examples U was exhausted. However, this was never the case.

14. across the 100 iterations per pool size

Experiment 2 - strict training This experiment tested the effect of using stricter training data. The stricter training data only considered labeled instances where at least two annotators agreed and where all annotators assigned the same label. In total 2,001 labeled examples satisfied this condition. In comparison to the baseline experiment a lower best F_1 score of 0.54 was reached (the corresponding U' and k can be found in table 8). The conclusion would be that the co-training framework performs better using all of the training data rather than only a stricter subset. At this point, it is not possible to argue whether this would also hold if as many "strict" as "normal" training data were available. The decrease in performance is probably not a result of the training data being selected for stricter criteria but of the reduced number of training data. However, one can infer that a larger less restricted (and thereby possibly less accurate) labeled training dataset is not detrimental to the overall learning process.

Experiment 3 - all validation This experiment was set up in the same manner as the baseline condition with one difference: instead of the unsupervised data that were specifically related to vaccination, all unsupervised data have been used to compose U and the random subsets U'. This condition emerged to produce the best results in terms of F_1 score of the negative class, namely $F_1 = 0.65$ with $U' = 1000$ and 95 iterations. The best F_1 score averaged across all pool sizes was 0.64 ± 0.003. This indicates that contrary to my expectations the unsupervised data containing both tweets specifically relating to vaccination, as well as those relating to diseases, proved to be a more valuable addition to the training data. I will discuss the results for this condition in more detail further below.

Experiment 4 - Downsampling 1 In order to account for the imbalance between the non-negative and negative training data all neutral tweets were removed in this experiment. This reduced the number of training instances to 2,564 (660 of which were negatives and 1904 positives). Table 8 displays a best F_1 score of 0.62 for this condition. It appears to be the case that the reduced imbalance in the training data did not drastically impact the F_1 score compared to the baseline condition. It is however noteworthy that the recall of the negative class increased in this condition to 0.84 compared to the 0.75 of experiment 1.

Due to removing all neutral instances one might suspect an effect on the non-negative class in this binary classification. Indeed table 9 shows a drop in the best F_1 score due to a drop in recall. This might be the effect of having a lower amount of non-negative training data. It is important to notice that the test data still included neutral tweets.

Experiment 5 - Downsampling 2 Here, another attempt was made to overcome the imbalance in the training data by randomly dropping training instances of the majority class, i.e. the non-negative class, with a probability of 0.4. This variant of downsampling lead to 2,288 training instances. 660 of these tweets were negative, 514 neutral and 1,114 positive. The results are comparable to those of the previous condition with a best F_1 score of 0.61. Table 9 indicates an increase in F_1 score for the non-negative class due to a higher recall value compared to the previous experiment. This can be explained by the fact that the reduced training data still contain neutral tweets and are therefore a better representation of the test data.

Experiment 6 - increased p & n This experimental manipulation tested the effect of an increased number of unlabeled training instances per iteration. The proportion of negative and non-negative examples was still preserved. As the results for the negative class essentially do not differ from the baseline condition (best F_1 score of 0.60 remains unchanged) it can be concluded that a larger proportion of negative and non-negative instances added per iteration do not impact the F_1 score for the negative class. The fact that the best result emerged for a pool size $U' = 10$ indicates that the algorithm itself constrained the number of unlabeled instances added. Given a large enough value of U' theoretically up to 50 instances can be added per iteration per classifier. This would result in 350 added instances per classifier after

	Measure	CoTrain[1]	Pattern	LogReg[2]	ExtraTrees[2]
Non-negative class	F_1	0.62	0.62	0.44	**0.65**
	Pr	**0.76**	0.57	0.64	0.63
	R	0.52	**0.69**	0.33	0.68
Negative class	F_1	**0.65**	0.38	0.58	0.50
	Pr	**0.56**	0.45	0.47	0.53
	R	**0.78**	0.33	0.76	0.47

[1] Choosing the best co-training system geared towards the negative class. Compare table 8.
[2] Both, the logistic regression classifier and the ExtraTrees classifier were initialized with the same parameter settings as for co-training. The initialization is based on the randomized search whose results can be viewed in figure 8.

Table 10: Comparison of best co-training classifier system to the pattern baseline and the standalone classification performance of logistic regression and the ExtraTrees classifier. The same training and test data have been used for each classification system. The best results per row are indicated in bold.

7 iterations. This might be an indication for the co-training system preventing a degradation effect in performance after adding too many unlabeled instances.

Experiment 7 - no p The last experiment tested the effect of only adding unlabeled instances to the training pool that have been labeled as belonging to the negative class. Eventually this intervention could serve as a means to overcome the class imbalance in the training data. Table 8 indicates that the best results from this condition are most similar to the results from experiment 4 in which all neutral tweets had been removed. The comparatively high recall value of the negative class could be attributed to the increase in negative training instances. Interestingly the result for the positive class (compare experiment 7 in table 9) shows the largest variance associated with the average F_1 score across all pool sizes. This indicates that compared to other conditions relatively low F_1 scores are reached which can be ascribed to the changed proportions in negative and non-negative samples.

Table 10 displays a comparison of the best performing co-training system (the one emerging from experiment 3 in table 8) and three baseline classification systems: the pattern baseline (as described in section 4.4), as well the results for the standalone classification using logistic regression and the ExtraTrees classifier. Both, the logistic regression and the ExtraTrees classifier were initialized with the settings that lead to the best results throughout randomized girdsearch (compare table 6 and figure 8). These were also the same settings that have been used for the initialization of the co-training classifiers. All four classification systems used the same training and test data.

For the non-negative class it can be observed that the co-training system, the pattern baseline and the ExtraTrees classifier perform similar in terms of the F_1 measure with the ExtraTrees classifier slightly outperforming the other two ($F_1 = 0.65$). The co-training system leads to the highest precision score ($P = 0.76$) for the non-negative class. This might be a result of optimizing the individual classifiers for the $F_{0.5}$ score which assigns more weight to precision than to recall.

It is however the case that the co-training system outperforms the three classifiers on all measures on the negative class.

Figure 10 displays the classification results and the convergence behaviour of the best performing co-training system. In figure 10 (a) the F_1 score, precision and recall for both classes are displayed as a function of the number of iterations. After each co-training iteration the classifiers have been refitted and used to make predictions on the test data. Based on these predictions F_1, precision and recall are computed. The blue vertical line marks the best F_1 score for the negative class. It can be observed that the F_1 and recall scores for the negative class are higher than for the non-negative class. Furthermore, the figure indicates that recall is higher for the negative class in comparison to the non-negative class while the reverse is true for precision. Overall there is not a lot of variation in the scores noticeable across iterations. One might expect a steady increase in performance over iterations. At the same time it does not appear to be the case that the performance decreases with an increasing number of labeled instances. Therefore, I would not conclude that the labeled instances are inadequate training instances. However, the lack of an increase in performance might be an indicator of the initial training data being sufficient enough to yield good classification results so that the effect of additional labeled instances is rather small. In order to test this hypothesis one might instantiate the same co-training experiment as experiment 3 in table 8 with (a) a larger number of iterations and (b) a smaller initial training dataset. Experiment (a) would test whether a larger quantity of added unsupervised instances increases performance over time. Experiment (b) would test the hypothesis that the beneficial effects of co-training are more pronounced if the initial amount of labeled data is smaller.

Figure 10 (b) illustrates the convergence of the individual classifiers during co-training as well as of the combined system. Convergence is measured by the mean absolute difference between predicted labels on the test data (vectors containing zeros for the non-negative and ones for the negative class) in consecutive iterations. Blue indicates the convergence behaviour of the ExtraTress classifier, green of the logistic regression and red of the combined classifiers. Within the 100 recorded iterations there still seems to be a steady level of small variation. In conclusion, the classification pattern does not seem to change drastically between consecutive iterations. An explanation might be that the initial training dataset was large enough so that the additional instances added through co-training do not have a large impact on the assigned labels. Theoretically it could be the case that more drastic changes can be observed when increasing the number of iterations.

I performed two follow-up experiments based on the results described above (both used the same training and test dataset as the previous experiments):

1. **Follow-up experiment 1** used the same setup as the best performing co-training system. However, the algorithm ran for 1000 iterations instead of 100.
2. **Follow-up experiment 2** is a replication of follow-up experiment 1. However, it uses a different Word2vec model to extract word embedding features from the data. In parallel to running the co-training experiments I re-trained the original Word2vec model on additional 20,052,878 Dutch twitter messages. Those tweets were not specifically pre-filtered to be related to vaccination and/or diseases. The rationale behind this experiment was that theoretically a Word2vec model trained on a larger corpus should outperform a model trained on a smaller corpus and therefore lead to a better classification performance. With this experiment I wanted to test how much better the predictive capacity of the co-training system can get due to (possibly) better features.

The results of the follow-up experiments can be viewed in table 11 and figures 11 and 12, respectively. With regard to follow-up experiment 1 a new best F_1 score of 0.68 for the negative class and 0.65 for the positive class can be observed (compare table 11). What is more, figure 11 shows a steady increase in performance when increasing the number of iterations (figure 11 (a)), as well as convergence of the classifiers (figure 11 (b)). Both of these patterns were missing in the original experiment that ran for 100 iterations (figure 10). The best F_1 score for the negative class has been reached after 765 iterations with 3,788 instances added by the ExtraTrees classifier and 3,825 instances added due to logistic regression. Compared to figure 10 one can notice more variation in

	Measure	Exp 1	Exp 2
Non-negative class	F_1	0.65	0.70
	Pr	0.79	0.85
	R	0.55	0.60
Negative class	F_1	0.68	0.72
	Pr	0.58	0.63
	R	0.81	0.86

Table 11: Results of the follow-up experiments. Experiment 1 tested the effect of running the best co-training system for 1000 instead of 100 iterations. Experiment 2 used the exact same setup as experiment 1 with an updated Word2vec model.

the first 100 iterations in this experiment. The ExtraTrees ensemble classifier implements randomization of the data which might have lead to this different pattern of variation. Other than that I do not have an explanation for this diverging behaviour. Nevertheless, the follow-up experiment demonstrates that the co-training framework works in utilizing unsupervised, i.e. unlabeled data to improve performance.

The results for follow-up experiment 2 (compare table 11 and figure 12) display the same pattern as follow-up experiment 1. The best F_1 score of the negative class was reached after 511 iterations with 2,534 instances added by the ExtraTrees classifier and 2,555 instances added due to logistic regression. However, it is noteworthy that figure 12 (b) shows stronger oscillations in the predictions between consecutive steps. This might be due to the updated Word2vec features having more impact on the predictions in individual steps. Most importantly table 11 shows a best F_1 score of 0.72 for the negative class and and 0.70 for the non-negative class. Which is a large improvement compared to the baseline results in table 10.

Comparing figure 10 to figures 11 and 12 it can be seen that the ordering of recall, precision and F_1 scores within and between classes is preserved. In conclusion based on the results of the follow-up experiments it can be argued that the co-training framework succeeds to add labeled instances to the training pool that are informative in the sense that they increase the overall classification capacity of the system.

7. Conclusion and discussion

The experiments demonstrate that the co-training framework can be successfully applied to improve sentiment analysis towards vaccination on Dutch Twitter messages. In total a best F_1 score of 0.72 on the negative class and 0.70 on the non-negative class has been reached with the best performing co-training classification system. Thereby the system outperformed the three baseline classifiers (compare table 10 and 11).

In the remainder of this section I am first going to discuss a few characteristics and possible shortcomings of my approach. Thereby I will also discuss the general feasibility of an approach like co-training in the field of machine learning. Finally, I will conclude with a few words on future work and the role and importance of unlabeled data for machine learning.

For the interpretation of the results it is important to keep in mind that for selecting the individual co-training classifiers, as well as the best performing co-training system, decisions have been based on the F_1 score of the negative class. Using this particular decision criterion leads to different results than using for instance the average F_1 score of both classes or of the non-negative class instead.

When reading and interpreting the results this bias should be taken into account. I based my decisions on the F_1 score for the negative class as correctly identifying negative sentiment towards vaccination was more important for my particular problem. Furthermore, heightening the system's sensitivity for the negative class can be viewed as a measure to deal with the class imbalance in the training data.

With regard to the different experimental conditions one needs to be cautious about the fact that some of them reduced the total amount of available training instances (e.g. experiment 2). Therefore, an observed decrease in performance might not necessarily be the result of the experimental manipulation but rather of the reduced amount of training data. In order to purely test the effect of the experimental condition more labeled training instances would be required to make sure that the number of training instances is equal across conditions.

Moreover, I would have liked to investigate the effect of different termination criteria on the co-training algorithm. In contrast to pre-defining a fixed amount of iterations k I would have liked to establish criteria that track the rate of change of the predictions made by the co-training system. This could be realized similar to the convergence measure described in the results section. In the end this might work as a preventive measure against degradation after a certain amount of iterations as reported in (Wang and Zhou 2007).

In order to select the optimal classifiers for co-training I performed randomized search on a subset of the data and tested the resulting classifiers on different pre-selected feature sets. Furthermore, I investigated the effect of different thresholds on co-training. All of these tuning steps have been performed in isolation. Each of them constitutes an optimum on its own. However, there is no guarantee that all the optima taken together form a global optimum. Given more computational power and most importantly more time all of these individual parameters and settings could in principle be tuned in one all-inclusive girdsearch to improve on the results. However, it might be the case that such an approach entails a larger risk of overfitting.

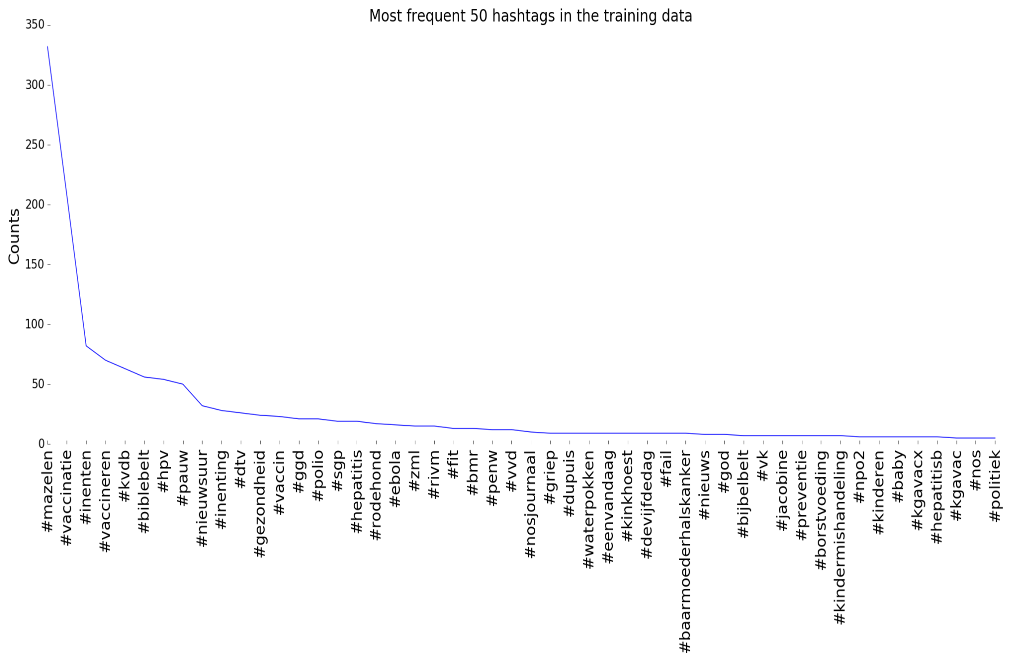
What is more, I am of the opinion that future work (or an extension of this work) should include an additional evaluation step in which the trained co-training system labels large amounts of unlabeled data. These labeled instances are then evaluated by independent raters in order to to measure the accuracy of the assigned labels. Such an evaluation procedure is of course time-consuming and defeats to some extent the purpose of developing a system that integrates unsupervised data in order to handle small amounts of labeled training data. Nevertheless, I think that a project as the one funded by the RIVM enables a framework in which such an extensive evaluation should be realizable. After all, especially when using classification systems that utilize unlabeled data one wants to make sure that the classifier generalizes well and is reliable. Another crucial evaluation step that is missing from the current work would be an extensive error analysis of the miss-classifications of the co-training algorithm. Especially a comparison between errors made by co-training and errors made by standalone classifiers such as logistic regression that have not been trained on additional unlabeled instances could prove very insightful and thereby allow to better target problem cases of the current implementation. I did not incorporate such an extensive analysis as I ran out of time. This however would be the first starting point if I would continue to work on this project.

With regard to co-training as a general tool in machine learning I found that it requires a substantial amount of tuning before being ready for use. Some of the parameters of co-training, such as the number of iterations k and the proportion of negative and non-negative instances n and p have not been optimized. In general it can be concluded that the co-training framework contains many variable parameters that require careful tuning. Most importantly I believe that these parameters are very problem and task specific. This restricts the generelizability of the co-training framework to other problems, domains or even datasets within the same domain. Due to the high number of variable parameters the algorithm is prone to overfitting. Future work on the co-training framework could involve the development of best practices that generalize across datasets and domains so that the algorithm becomes as easy to use as a standard classifier like multinomial naive Bayes.

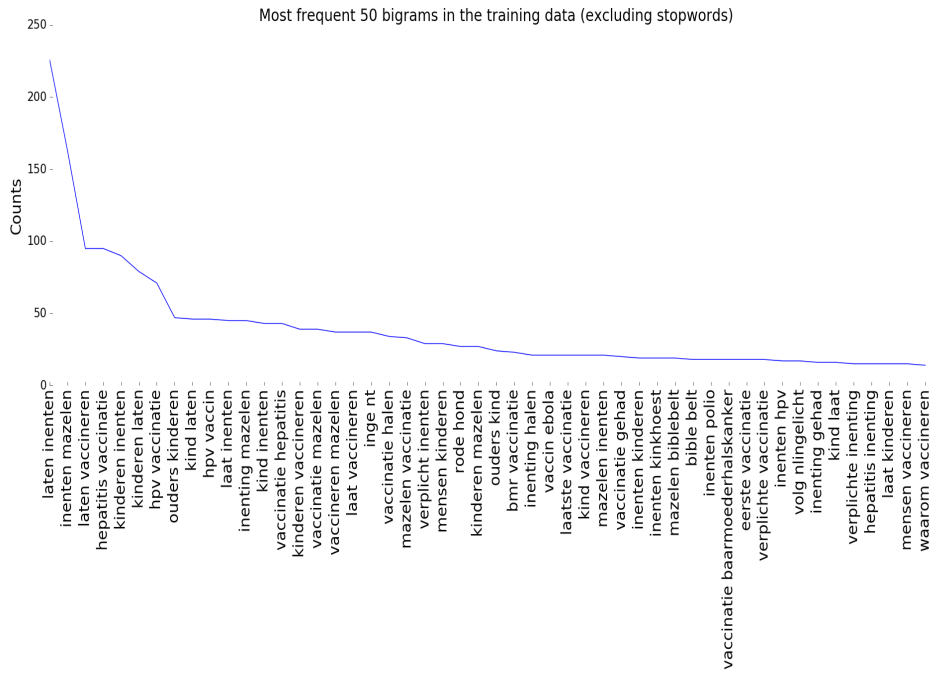
Coming back to the original problem of classifying sentiment towards vaccination on Dutch tweets alternative approaches to co-training or additions to the current system might involve deep learning. One possible realization could be a form of “bootstrapped deep learning”: Once a sufficient amount of data has been confidently labeled by co-training a deep convolutional network could be trained on the labeled data and used in a feedback loop in the same way as the classical co-training classifiers are used. The feasibility of deep learning for text understanding is demonstrated in (Zhang and LeCun 2015). More specifically, they use temporal convolutional networks to solve various text mining related tasks, such as sentiment analysis and text categorization. The input to the network are short batches of text, which are treated as a sequence of characters rather than a sequence of words, allowing the convolutional layers to learn the abstract semantic and syntactic representations on their own.

One of the most constraining perquisites for successfully applying deep learning is a vast amount of labeled data. Currently researchers in the field of deep learning devote attention to this issue and work on developing approaches to overcome it. Reed et al. (2014) demonstrate methods to handle noisy and corrupted labels for the task of image classification. Being able to apply this to textual data like tweets might also serve as an alternative to the co-training approach as a whole.

Referring back to the introduction of this thesis I would like to emphasize that deep learning algorithms are not an exception with regard to suffering from requiring large amounts of labeled data for training. On the contrary most machine learning approaches, and especially those that turn out to be very successful heavily rely on labeled data. Personally I think that access to large volumes of confidently labeled data (a ground truth) is not a realistic scenario. Real data come in many different forms and shapes: most often they are unstructured, unlabeled, correlated and not even for humans easily separable. The unlabeled Dutch Twitter messages constitute an example of realistic data. Labelling these tweets is a tedious and costly matter. At the same time access to these data is very inexpensive. These data and the problem of identifying sentiment towards vaccination within these data showcases why it is so important to develop methods that are capable to make use of large amounts of unlabeled data. Observing the current rise of techniques like deep learning (see for instance Najafabadi et al. (2015)) I do not believe that we are close to moving away from the dependency on labeled data. Nevertheless, I personally am of the opinion that future development of semi-supervised or even unsupervised algorithms will be an important subfield within data science and machine learning as I view the most interesting problems in data science to lie within the class of problems for which access to data is easy but labelling is costly. Medicine is an example of a field which stores massive amounts of data. Advancement with regard to unsupervised learning algorithms could enable the discovery of unknown patterns in the available data which again could have serious implications for the development of treatments and medication. Of course all of this is to some extend speculative. However, I do hope that research in this field of semi-supervised and unsupervised learning algorithms will progress so that eventually the field of machine learning will, next to focusing on engineering problems, e.g. how to get the highest accuracy in the classification of images displaying distracted drivers, also concentrate on more open and research driven problems such as the recognition of patterns in medical data.

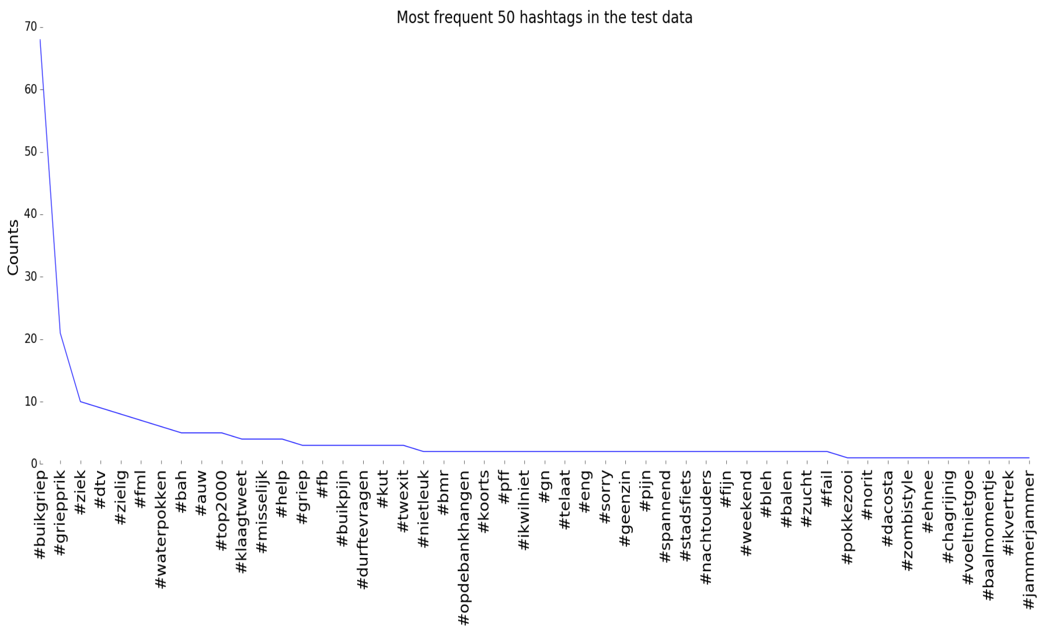


(a) Most frequent 50 hashtags in the training data

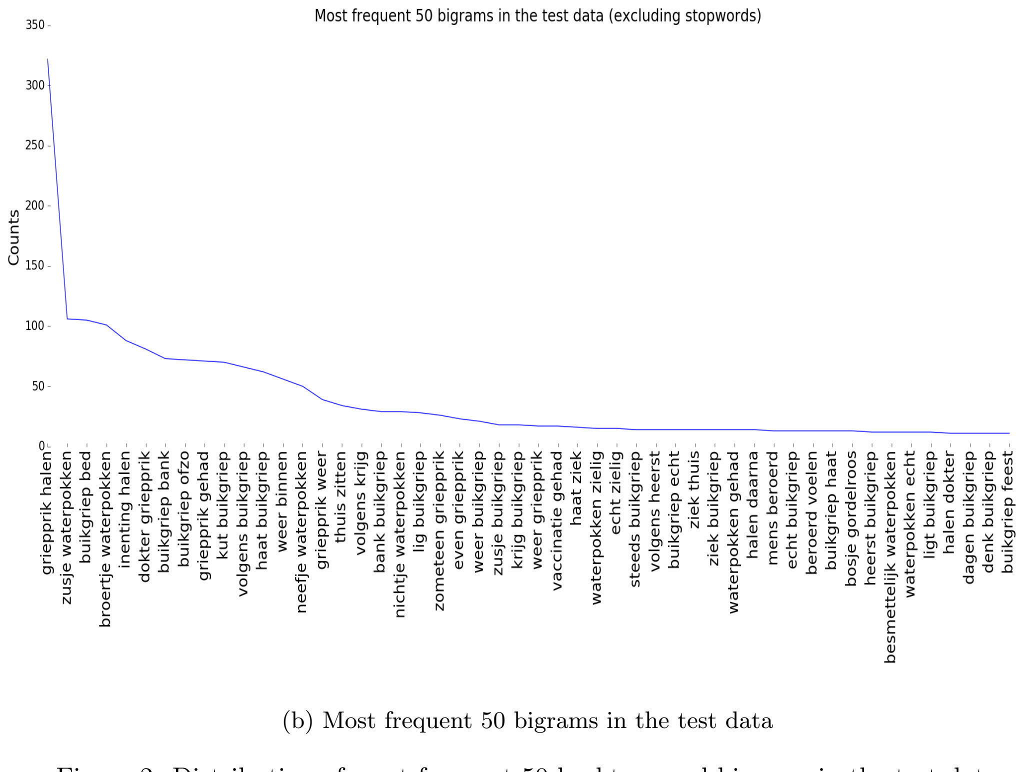


(b) Most frequent 50 bigrams in the training data

Figure 1: Distribution of most frequent 50 hashtags and bigrams in the training data

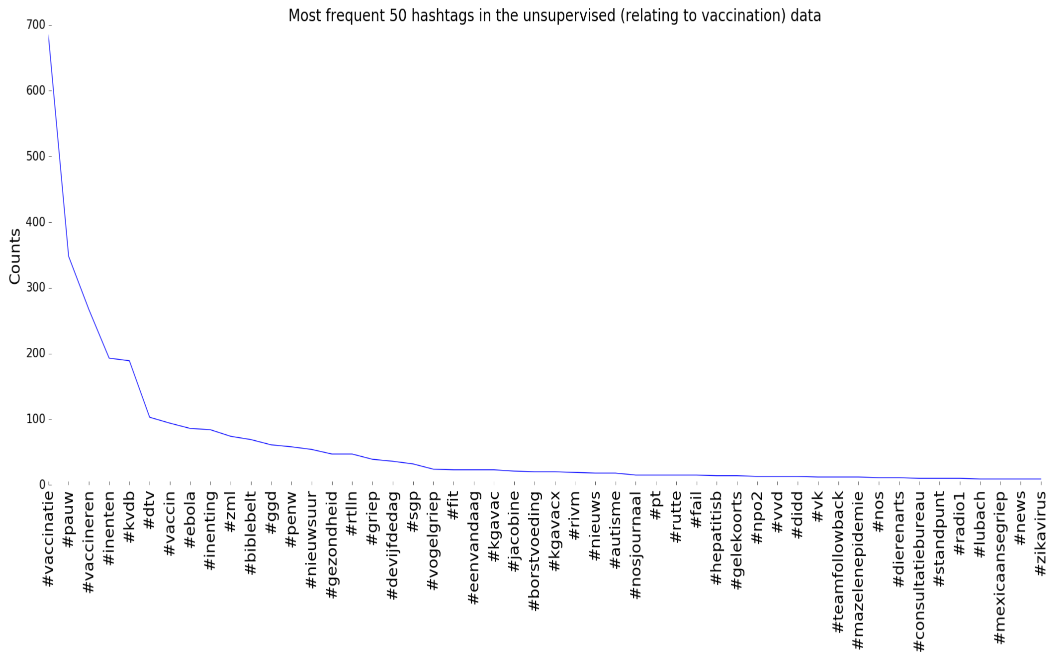


(a) Most frequent 50 hashtags in the test data

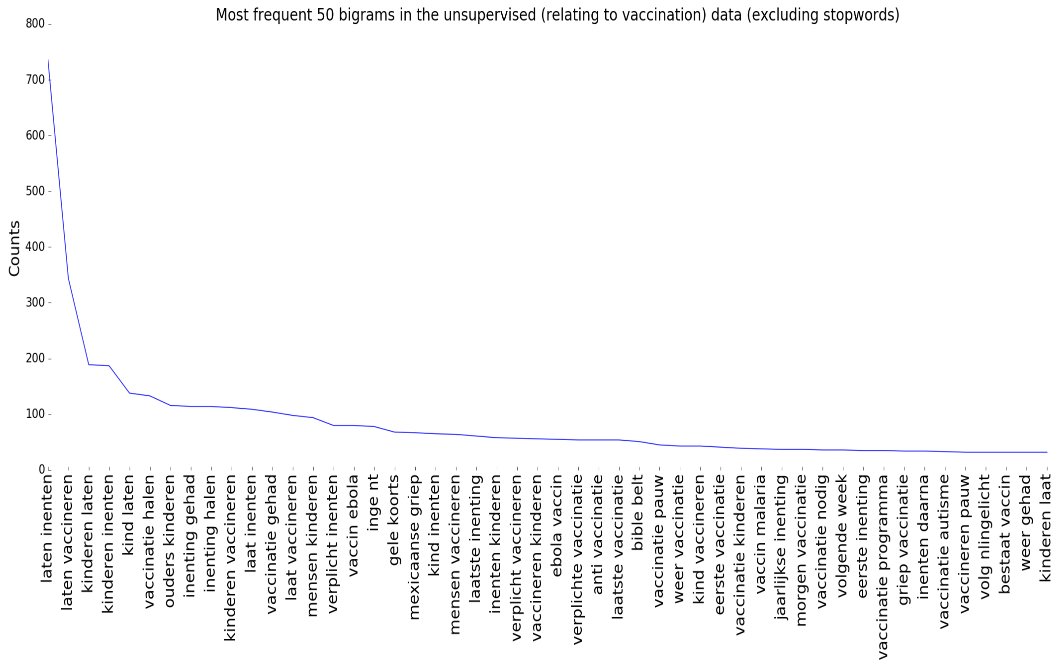


(b) Most frequent 50 bigrams in the test data

Figure 2: Distribution of most frequent 50 hashtags and bigrams in the test data

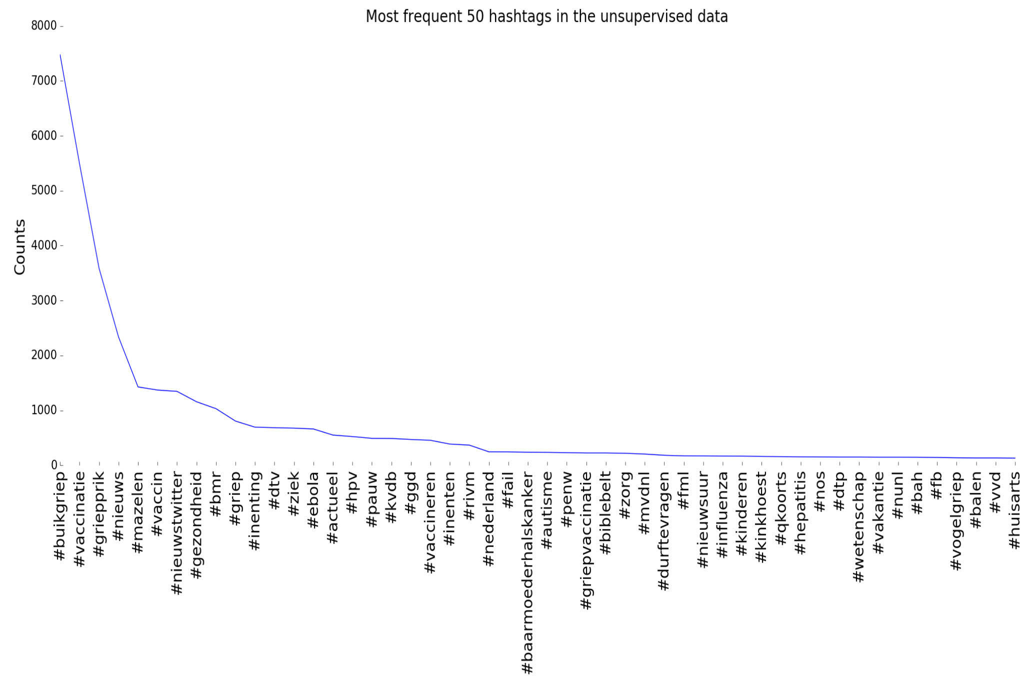


(a) Most frequent 50 hashtags in the unsupervised data relating to vaccination

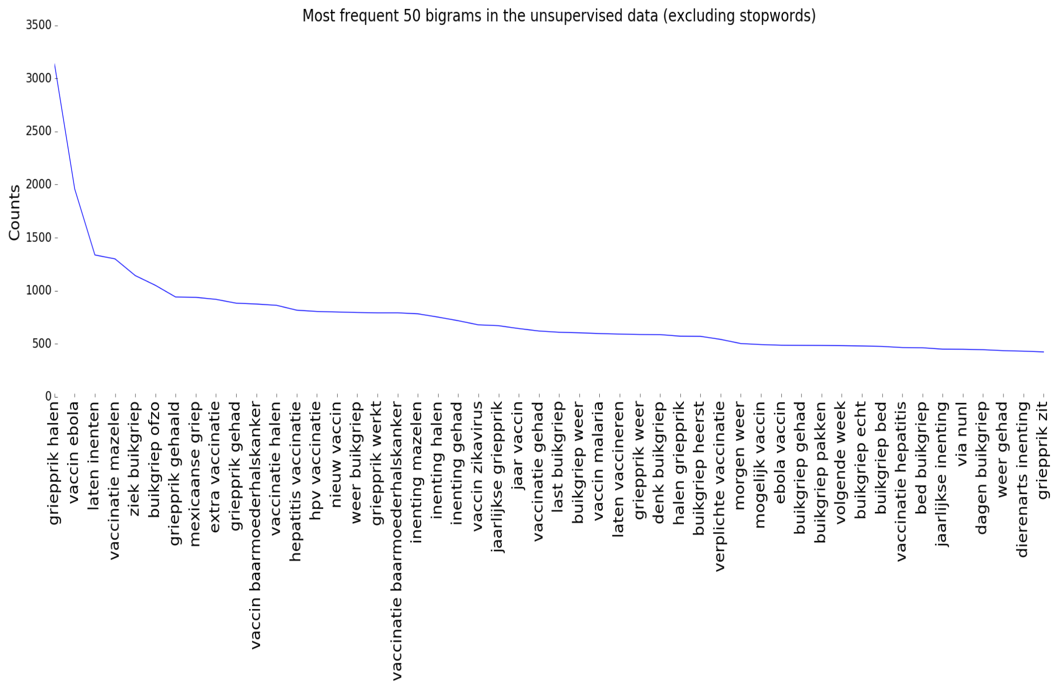


(b) Most frequent 50 bigrams in the unsupervised data relating to vaccination

Figure 3: Most frequent 50 bigrams in the unsupervised data related to vaccination

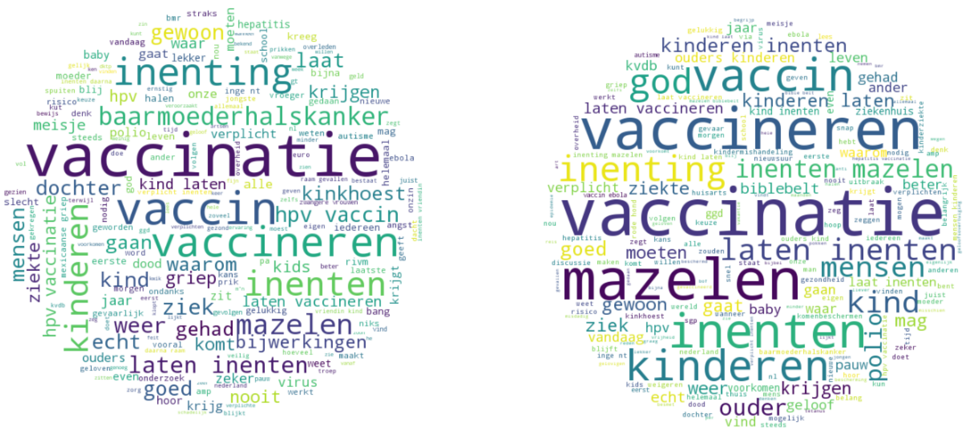


(a) Most frequent 50 hashtags in the unsupervised data



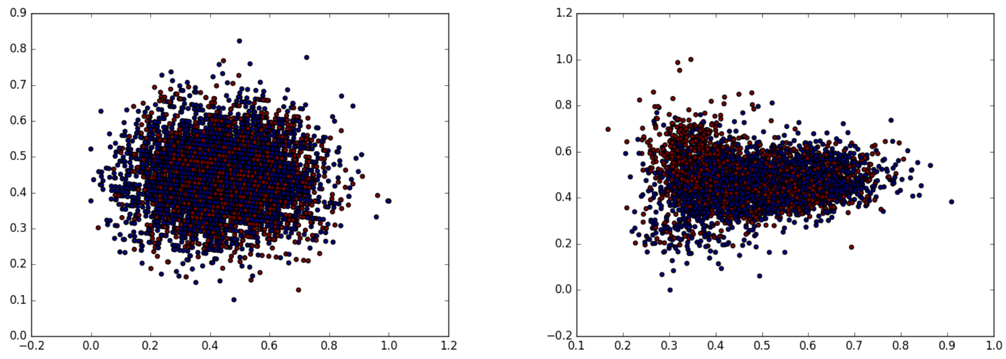
(b) Most frequent 50 bigrams in the unsupervised data

Figure 4: Most frequent 50 bigrams in the unsupervised data



(a) Wordcloud for the negative training tweets (b) Wordcloud for the positive training tweets

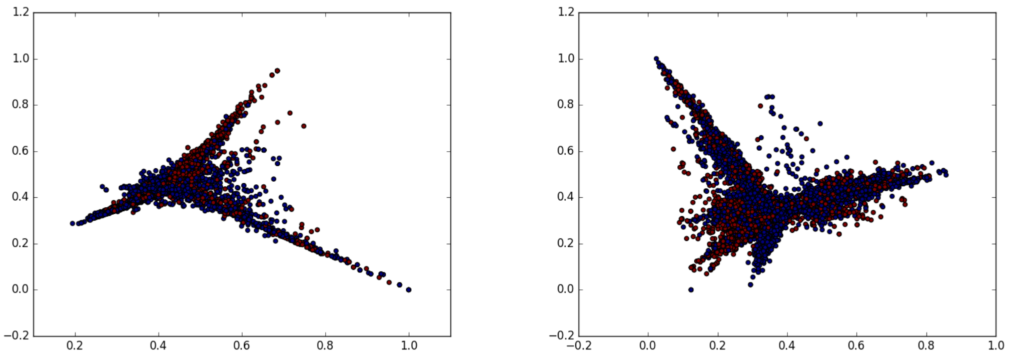
Figure 5: Wordclouds for the positive and negative tweets in the training data



(a) Tweets in Word2vec feature space. Variance explained by the first two principal components is 65% and 33%, respectively.

(b) Tweets in Doc2vec feature space. Variance explained by the first two principal components is 59% and 32%, respectively.

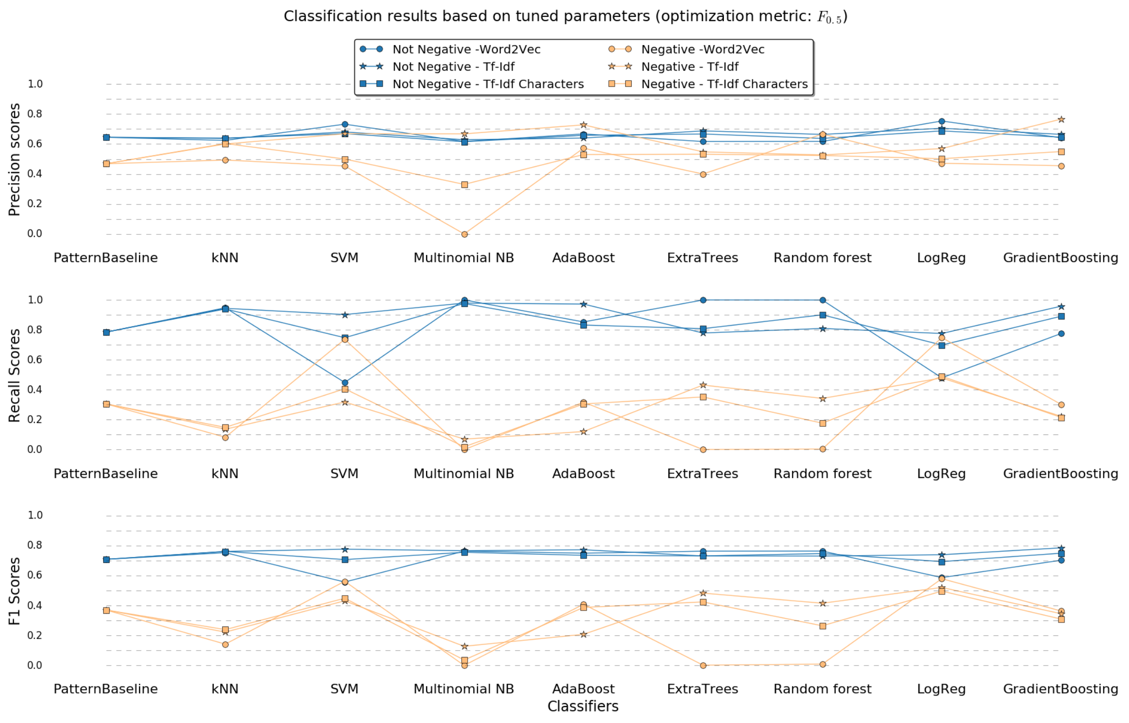
Figure 6: First two principal components of the Word2vec (left) and Doc2vec (right) features. Red dots represent negative tweets and blue dots positive tweets.



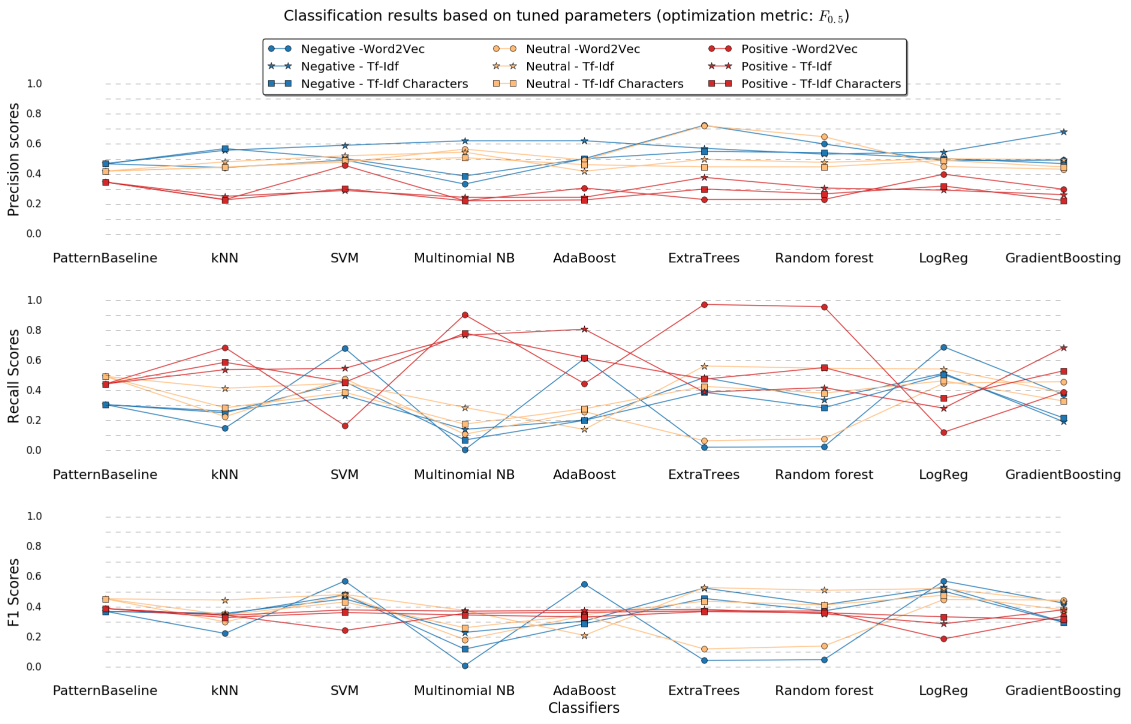
(a) Tweets in TF-IDF word level feature space. Variance explained by the first two principal components is 0.45% and 0.35%, respectively.

(b) Tweets in TF-IDF character level feature space.Variance explained by the first two principal components is 0.27% and 0.14%, respectively.

Figure 7: First two principal components of the TF-IDF word level (left) and TF-IDF character level (right) features. Red dots represent negative tweets and blue dots positive tweets.

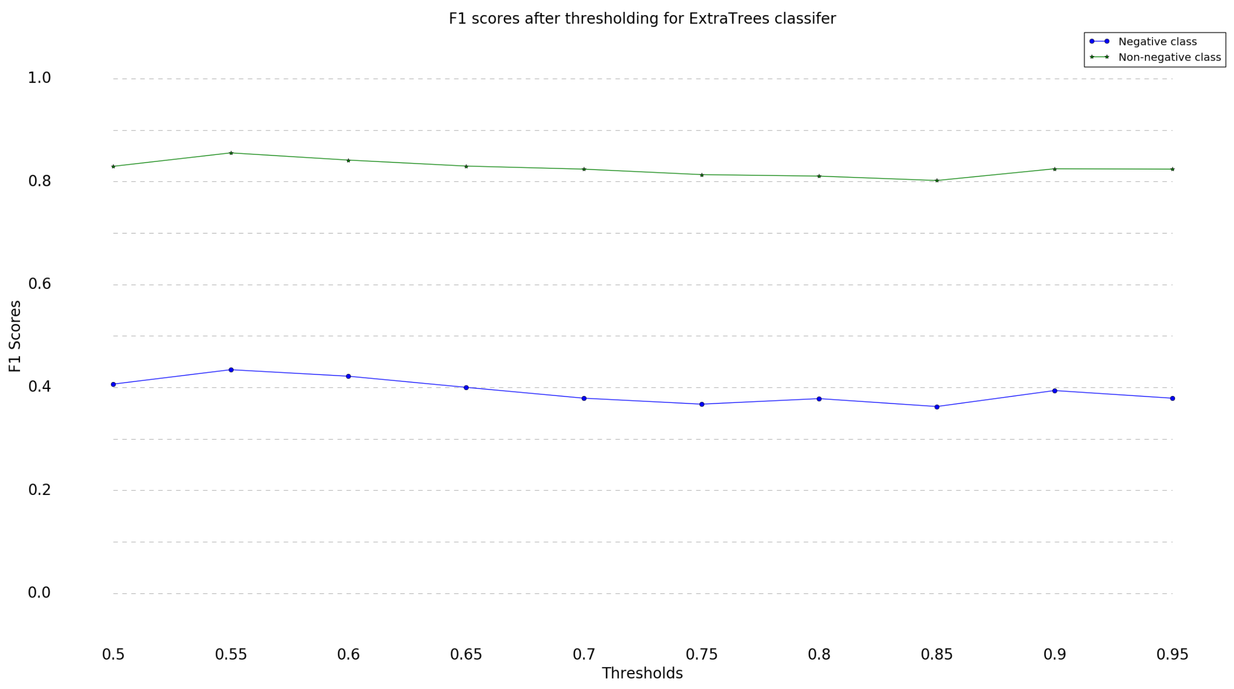


(a) Results for the binary task. Prior to classification the classifiers have been tuned in a randomized search.

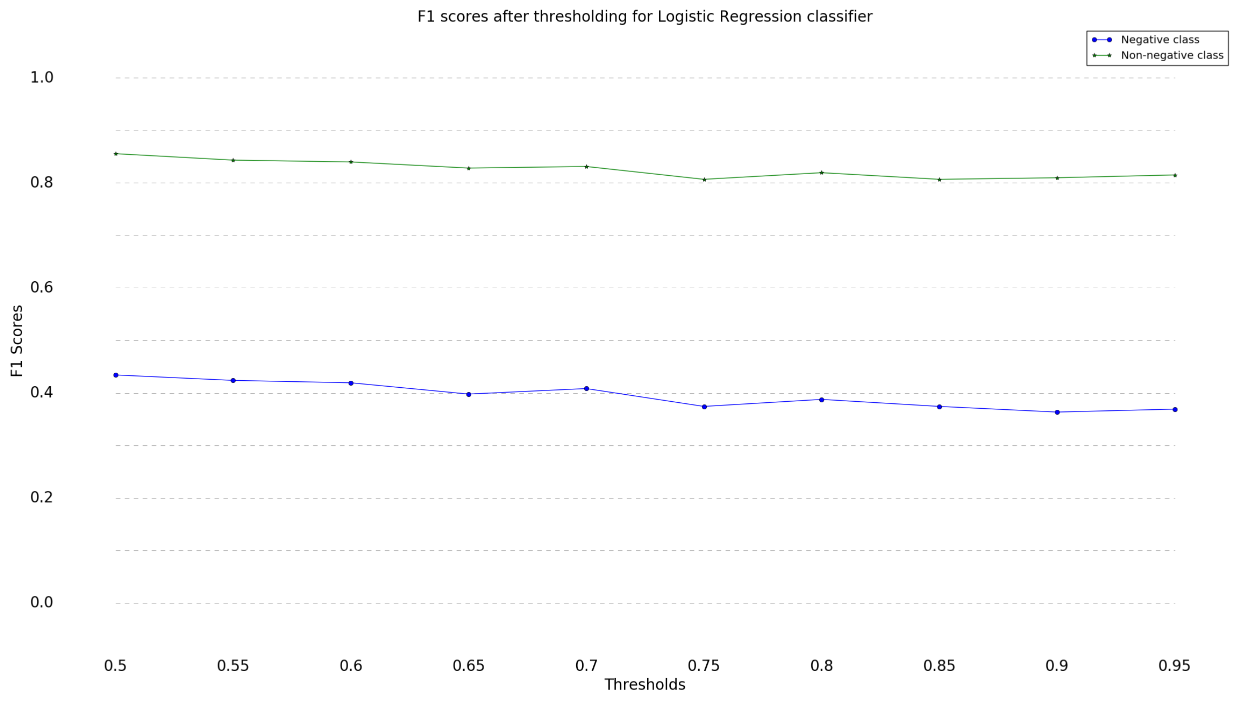


(b) Results for the three-way classification task. Prior to classification the classifiers have been tuned in a randomized search.

Figure 8: Comparison of optimized classifiers across feature sets and classification tasks. Recall, precision and F_1 scores are shown.

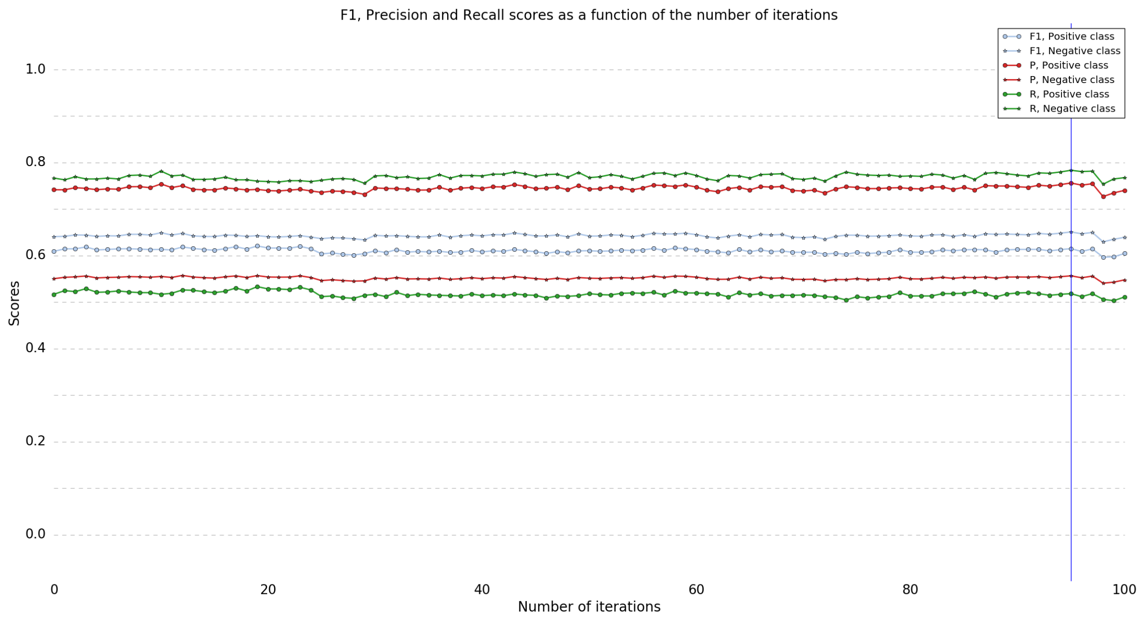


(a) Thresholding results for the ExtraTrees classifier.

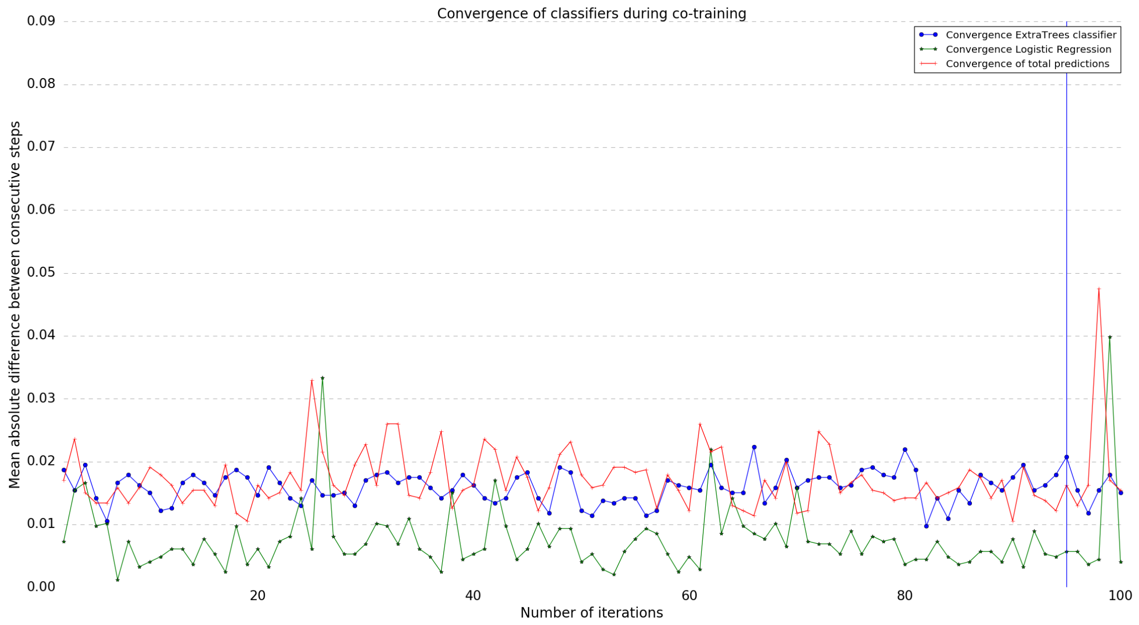


(b) Thresholding results for the logistic regression classifier.

Figure 9: Results of thresholding experiments for the optimized classifiers for $U' = 75$ and $k = 30$.

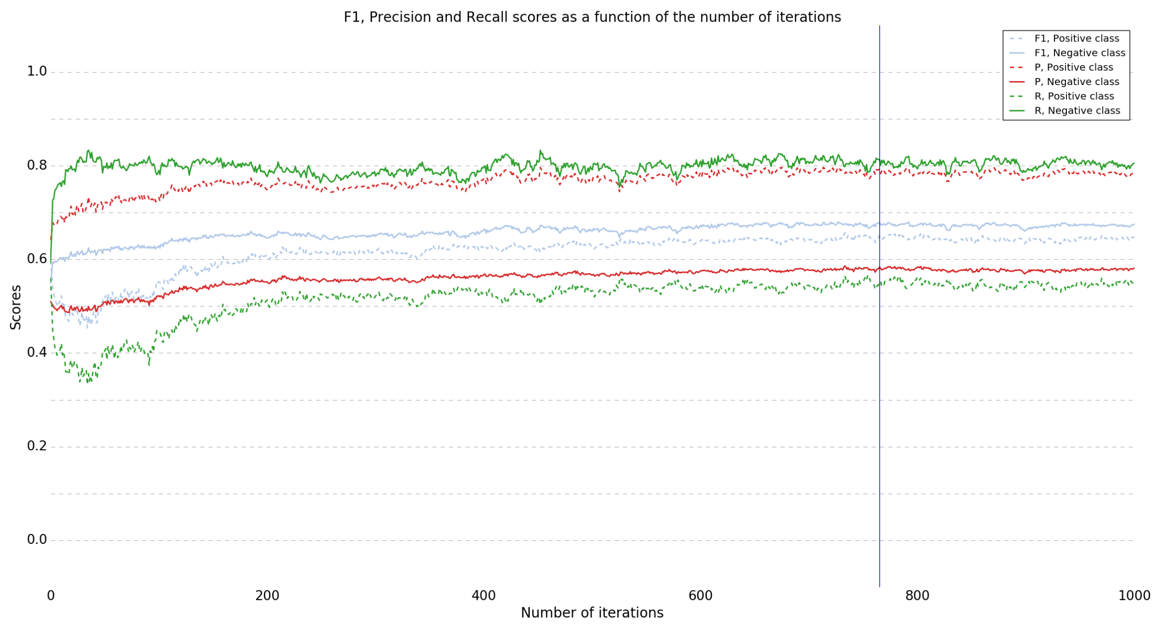


(a) F_1, precision and recall scores for both classes as a function of the number of iterations. The blue vertical line marks the best F1 score for the negative class.

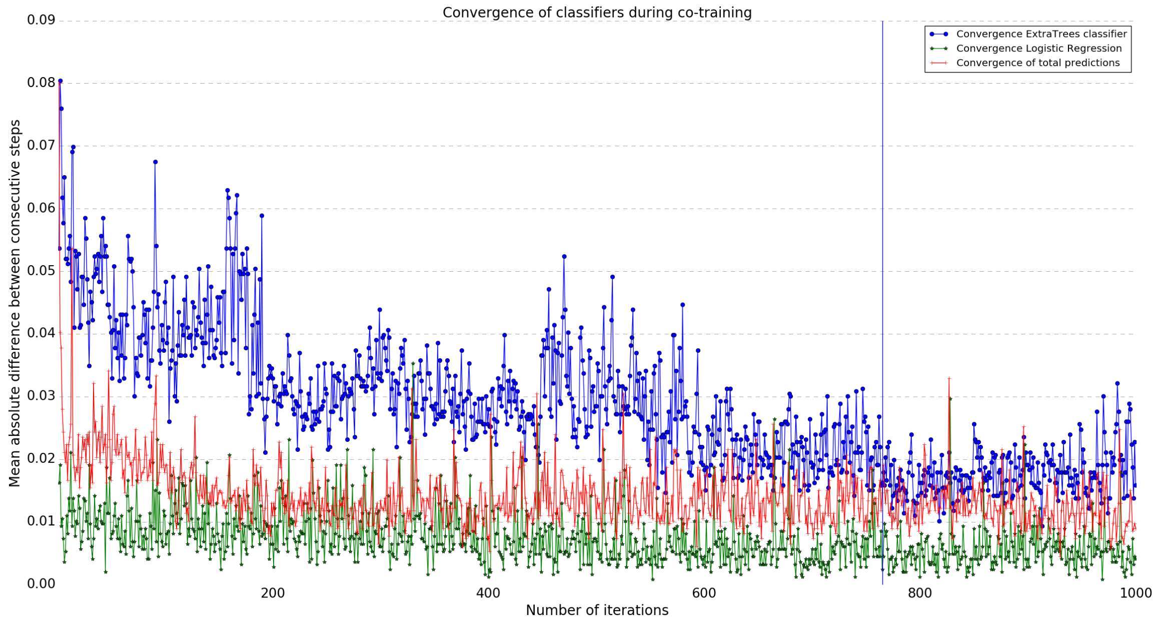


(b) Convergence of the classifiers during co-training. Convergence is displayed for the individual classifiers as well as for the combined system. Convergence is measured in the mean absolute difference in assigned labels between consecutive iterations.

Figure 10: Classification results and convergence behaviour for the best performing co-training system.

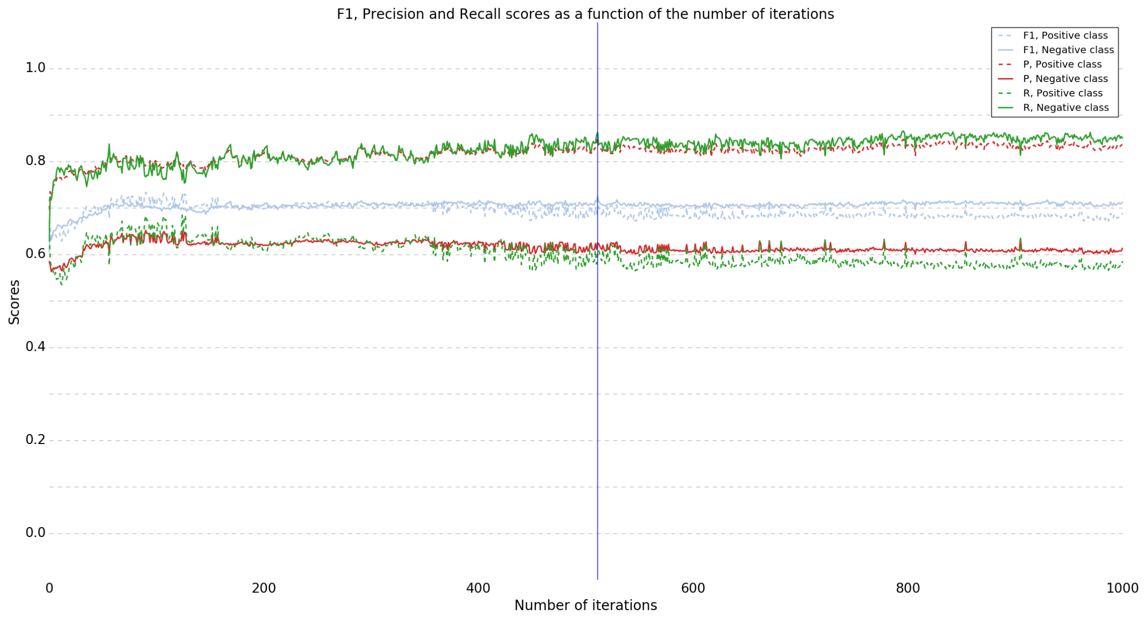


(a) F_1, precision and recall scores for both classes as a function of the number of iterations. The blue vertical line marks the best F1 score for the negative class.

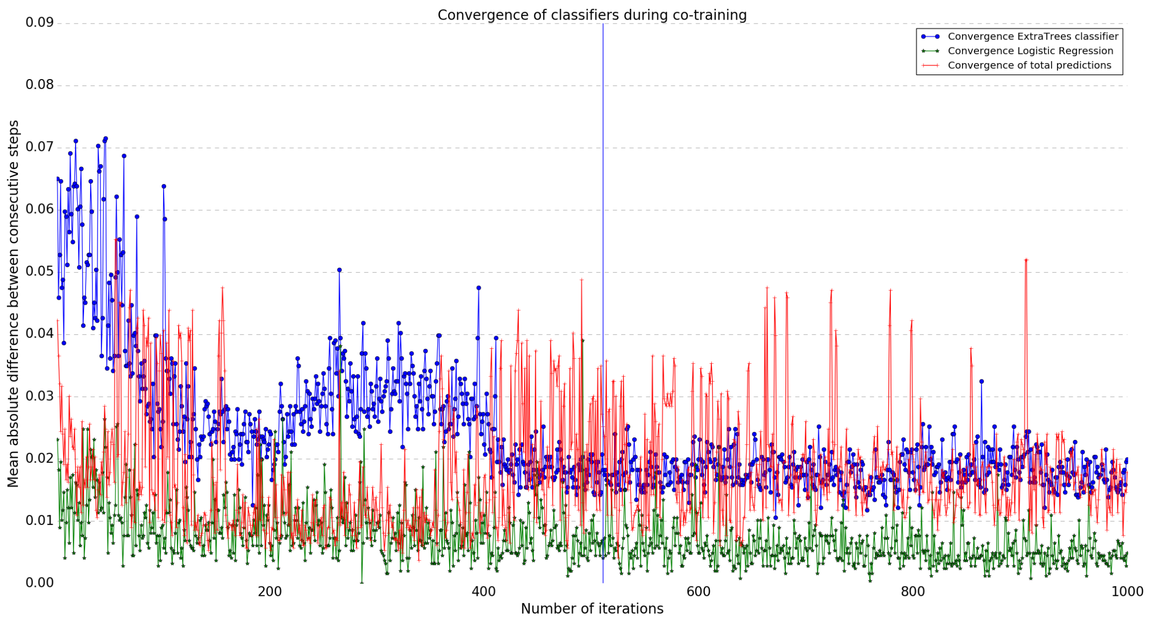


(b) Convergence of the classifiers during co-training. Convergence is displayed for the individual classifiers as well as for the combined system. Convergence is measured in the mean absolute difference in assigned labels between consecutive iterations.

Figure 11: Classification results and convergence behaviour for the best performing co-training system after 1000 (follow-up experiment 1).



(a) F_1, precision and recall scores for both classes as a function of the number of iterations. The blue vertical line marks the best F1 score for the negative class.



(b) Convergence of the classifiers during co-training. Convergence is displayed for the individual classifiers as well as for the combined system. Convergence is measured in the mean absolute difference in assigned labels between consecutive iterations.

Figure 12: Classification results and convergence behaviour for the best performing co-training system after 1000 iterations and using the updated Word2vec model (followup experiment 2).

References

Agarwal, Apoorv, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau (2011), Sentiment analysis of twitter data, *Proceedings of the workshop on languages in social media*, Association for Computational Linguistics, pp. 30–38.

Aslam, Salman (2017), Twitter by the numbers: Stats, demographics & fun facts. https://www.omnicoreagency.com/twitter-statistics/.

Barbosa, Luciano and Junlan Feng (2010), Robust sentiment detection on twitter from biased and noisy data, *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, Association for Computational Linguistics, pp. 36–44.

Bello-Orgaz, Gema, Julio Hernandez-Castro, and David Camacho (2017), Detecting discussion communities on vaccination in twitter, *Future Generation Computer Systems* **66**, pp. 125–136, Elsevier.

Biyani, Prakhar, Cornelia Caragea, Prasenjit Mitra, Chong Zhou, John Yen, Greta E Greer, and Kenneth Portier (2013), Co-training over domain-independent and domain-dependent features for sentiment analysis of an online cancer support community, *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ACM, pp. 413–417.

Blum, Avrim and Tom Mitchell (1998), Combining labeled and unlabeled data with co-training, *Proceedings of the eleventh annual conference on Computational learning theory*, ACM, pp. 92–100.

Boiy, Erik and Marie-Francine Moens (2009), A machine learning approach to sentiment analysis in multilingual web texts, *Information retrieval* **12** (5), pp. 526–558, Springer.

Campbell, Ellsworth and Marcel Salathé (2012), Complex social contagion makes networks more vulnerable to disease outbreaks, *arXiv preprint arXiv:1211.0518*.

Chen, Mengen, Xiaoming Jin, and Dou Shen (2011), Short text classification improved by learning multi-granularity topics, *IJCAI*, pp. 1776–1781.

Cui, Anqi, Min Zhang, Yiqun Liu, and Shaoping Ma (2011), Emotion tokens: Bridging the gap among multilingual twitter sentiment analysis, *Information retrieval technology* pp. 238–249, Springer.

Davies, Alex and Zoubin Ghahramani (2011), Language-independent bayesian sentiment mining of twitter, *The 5th SNA-KDD Workshop11 (SNA-KDD11)*.

Du, Jingcheng, Jun Xu, Hsingyi Song, Xiangyu Liu, and Cui Tao (2017), Optimization on machine learning based approaches for sentiment analysis on hpv vaccines related tweets, *Journal of biomedical semantics* **8** (1), pp. 9, BioMed Central.

Fersini, Elisabetta, Enza Messina, and Federico Alberto Pozzi (2014), Sentiment analysis: Bayesian ensemble learning, *Decision Support Systems* **68**, pp. 26–38, Elsevier.

Ghahramani, Zoubin (2004), Unsupervised learning, *Advanced lectures on machine learning*, Springer, pp. 72–112.

Go, Alec, Richa Bhayani, and Lei Huang (2009), Twitter sentiment classification using distant supervision, *CS224N Project Report, Stanford* **1** (2009), pp. 12.

Huang, Xiaolei, Michael C Smith, Michael J Paul, Dmytro Ryzhkov, Sandra C Quinn, David A Broniatowski, and Mark Dredze (2017), Examining patterns of influenza vaccination in social media.

Hürriyetolu, Ali, Christian Gudehus, Nelleke Oostdijk, and Antal van den Bosch (2016), Relevancer: Finding and labeling relevant information in tweet collections, *International Conference on Social Informatics*, Springer, pp. 210–224.

Jansen, Vincent AA, Nico Stollenwerk, Henrik Jeldtoft Jensen, ME Ramsay, WJ Edmunds, and CJ Rhodes (2003), Measles outbreaks in a population with declining vaccine uptake, *Science* **301** (5634), pp. 804–804, American Association for the Advancement of Science.

Ko, Youngjoong and Jungyun Seo (2000), Automatic text categorization by unsupervised learning, *Proceedings of the 18th conference on Computational linguistics-Volume 1*, Association for Computational Linguistics, pp. 453–459.

Le, Quoc and Tomas Mikolov (2014), Distributed representations of sentences and documents, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pp. 1188–1196.

Man, Yuan (2014), Feature extension for short text categorization using frequent term sets, *Procedia Computer Science* **31**, pp. 663–670, Elsevier.

Medhat, Walaa, Ahmed Hassan, and Hoda Korashy (2014), Sentiment analysis algorithms and applications: A survey, *Ain Shams Engineering Journal* **5** (4), pp. 1093–1113, Elsevier.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean (2013), Distributed representations of words and phrases and their compositionality, *Advances in neural information processing systems*, pp. 3111–3119.

Najafabadi, Maryam M, Flavio Villanustre, Taghi M Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic (2015), Deep learning applications and challenges in big data analytics, *Journal of Big Data* **2** (1), pp. 1, Springer International Publishing.

Narr, Sascha, Michael Hulfenhaus, and Sahin Albayrak (2012), Language-independent twitter sentiment analysis, *Knowledge discovery and machine learning (KDML), LWA* pp. 12–14.

Nigam, Kamal and Andrew Kachites Mccallum (2002), Text classification from labeled and unlabeled data using em, *Machine Learning, Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.*

Omer, Saad B, Daniel A Salmon, Walter A Orenstein, M Patricia Dehart, and Neal Halsey (2009), Vaccine refusal, mandatory immunization, and the risks of vaccine-preventable diseases, *New England Journal of Medicine* **360** (19), pp. 1981–1988, Mass Medical Soc.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011), Scikit-learn: Machine learning in Python, *Journal of Machine Learning Research* **12**, pp. 2825–2830.

Perrin, Pew Research Center, Andrew (2015), Social networking usage: 2005-2015. http://www.pewinternet.org/2015/10/08/social-networking-usage-2005-2015/.

Pierce, David and Claire Cardie (2001), Limitations of co-training for natural language learning from large datasets, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pp. 1–9.

Qadir, Ashequl and Ellen Riloff (2013), Bootstrapped learning of emotion hashtags# hashtags4you, *Proceedings of the 4th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pp. 2–11.

Read, Jonathon (2005), Using emoticons to reduce dependency in machine learning techniques for sentiment classification, *Proceedings of the ACL student research workshop*, Association for Computational Linguistics, pp. 43–48.

Reed, Scott, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich (2014), Training deep neural networks on noisy labels with bootstrapping, *arXiv preprint arXiv:1412.6596.*

Řehůřek, Radim and Petr Sojka (2010), Software Framework for Topic Modelling with Large Corpora, *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, ELRA, Valletta, Malta, pp. 45–50. `http://is.muni.cz/publication/884893/en`.

Riloff, Ellen and Janyce Wiebe (2003), Learning extraction patterns for subjective expressions, *Proceedings of the 2003 conference on Empirical methods in natural language processing*, Association for Computational Linguistics, pp. 105–112.

Salathé, Marcel and Sebastian Bonhoeffer (2008), The effect of opinion clustering on disease outbreaks, *Journal of The Royal Society Interface* **5** (29), pp. 1505–1508, The Royal Society.

Sang, Erik Tjong Kim and Antal van den Bosch (2013), Dealing with big data: The case of twitter, *Computational Linguistics in the Netherlands Journal* **3** (121-134), pp. 2013.

Signorini, Alessio, Alberto Maria Segre, and Philip M Polgreen (2011), The use of twitter to track levels of disease activity and public concern in the us during the influenza a h1n1 pandemic, *PloS one* **6** (5), pp. e19467, Public Library of Science.

Smedt, Tom De and Walter Daelemans (2012), Pattern for python, *Journal of Machine Learning Research* **13** (Jun), pp. 2063–2067.

Timonen, Mika and Melissa Kasari (2012), Statistical approach for term weighting in very short documents for text categorization, *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, Springer, pp. 3–18.

Wang, Wei and Zhi-Hua Zhou (2007), Analyzing co-training style algorithms, *European Conference on Machine Learning*, Springer, pp. 454–465.

Xia, Rui, Cheng Wang, Xin-Yu Dai, and Tao Li (2015), Co-training for semi-supervised sentiment classification based on dual-view bags-of-words representation., *ACL (1)*, pp. 1054–1063.

Yu, Ning (2014), Exploring co-training strategies for opinion detection, *Journal of the Association for Information Science and Technology* **65** (10), pp. 2098–2110, Wiley Online Library.

Yu, Shipeng, Balaji Krishnapuram, Rómer Rosales, and R Bharat Rao (2011), Bayesian co-training, *Journal of Machine Learning Research* **12** (Sep), pp. 2649–2680.

Zhang, Xiang and Yann LeCun (2015), Text understanding from scratch, *arXiv preprint arXiv:1502.01710.*