Tweegion - mapping Tweet to Region

Johannes Gontrum, Matthias Wegel, Steve Wendler

September 4, 2013

Abstract

This project originated from the seminar *Computerlinguistische Analyse von Twitterdaten* by Tatjana Scheffler at Universität Potsdam in summer 2013. During this seminar we were confronted with many problems of computational analysis of social media data, esp. very short texts of media like Twitter.

This project's aim was to find a way to map unseen Tweets to the region a Tweet or at least the Tweet-writer comes from. We were aware that it would only be possible to map a Tweet to a specific region if this Tweet contains at least little hints towards a region. Tweets written in pure standard language are impossible to map. Nonetheless we thought it being worthwhile to touch this region-specific marked Tweets.

We tried out two different approaches whose starting points were different but which were very similar afterwards.The *Geo-location-attempt* trained a model based on Tweets which were sent together with geo data information. The *Regional-words-attempt* trained a model according to a list of words known of being regionally salient words. We found out that the regional words attempt was hardly usable whilst the geo location attempt could reach much better results.

But not only finding and implementing a good algorithm for fulfilling this aim was important. For the three co-workers of this project it was the first project of this manner and it was exciting to learn to develop source code in a team. So we had to learn to work together, think together, try to find a way to communicate and of course to develop code.

Even if the results of our project were not that satisfying as we wished they were, we nevertheless learnt much about collaborating in a project like this. Additionally we are convinced that bad results are improvable by finding more and better region-specific words and of course by accumulating more knowledge in computational linguistics during our studies. So the journey still goes on.

Contents

1 Introduction

1.1 Twitter

Twitter is a web service for sending out short messages—so called microblogging—originated in 2006 and used worldwide nowadays. Only a small fraction of the messages, the so-called Tweets, is written in German. Predominantly used languages are English, Spanish, Portuguese, Japanese, and Malay.[Twl] Also when compared to the population, Germans, Austrians and Swiss do not Tweet very much, whereas the network is extremely popular for instance in the Netherlands, the US, the UK, and Japan.[Twc]

The spatial distribution of sended Tweets within Germany is investigable based on geo annotation, too. It mainly reflects the distribution of population; focal points are above all Berlin and the Ruhr district. But—beside Tweeting in general—especially sending of geo data (so-called geotagging) is rather unpopular in the German-speaking area.

There are three fundamentally different location-related pieces of information that can be sended along with a Tweet. None of these specifications is obligatory for the user, so local dara are not available for all Tweets.

First of all, the user may enter a location to their profile. In a JSON object, it appears as field *location* in the sub-object *user.* Users often enter fantasy places like *Wolkenkuckucksheim (cloud cuckoo land)* here, though. What's more, the information may be ambigous and inaccurate, which is why they are practically ruled out for computational processing.

When sending a Tweet, there is also the option *add location.* There a distinct, named location can be chosen. In a JSON object, it appears as sub-object *place,* including name, type (e. g. *city*), country and a *bounding box,* a rectangle enclosing the place defined by four geo coordinates. As the Twitter developers documentation puts it, "Tweets associated with places are not necessarily issued from that location but could also potentially be *about* that location".

Finally it is possible to send a Tweet's actual point of origin from GPS-enabled devices. The coordinates are saved as sub-object *geo* in the JSON object.

1.2 General Idea

Our aim is a rough placement of Tweets within the German-speaking area in spite of the above-mentioned rare geo data and barely usable location information. As computational linguists we came up with the idea to use

language as an indicator of origin—the pure text of a Tweet already contains information about its origin.

There are two different items here. One the one hand there is the place or the places where a Twitter user has grown up and which influenced his or her language. Mainly those places is what we measure when we search for regional salient expressions of dialectal and colloquial speech. One the other hand there is the current, sometimes very short-timed whereabout which the user is Tweeting from. On a textual level it will rather be reflected in location-related terms (place names, traffic hubs, local events and personages etc.). Furthermore it is what we get to know from Tweets' geo data. Yet geo data are the only data we can use for evaluation of our results. While our geo data driven *Geo-location-attempt* may be evaluated quite suitably, our *Regional-words-attempt* does specifically aim for the item 'language-imprinting origin of a Twitter user' and will therefore inevitably pass the evaluation with a certain handicap.

Our attempt is to divide the German-speaking area into certain regions and to use machine learning to collect regionally salient data from the particular regions. We use the simple bag-of-words model, i. e. we consider unigrams.

1.3 Geo-location-attempt

As allready mentioned above, German Twitter-users only seldom allow Twitter to store geo data. But nonetheless there is a small number of Tweet containing those geo data. The *Geo-location-attempt* uses those data by assigning a specific region to each Tweet. Based on that information we calculate affiliation of all words of the Tweet with particular regions. This will be done for all Tweets in the training corpus and so eventually we create a vector-space model based on the exact knowledge of geographical origins of all Tweet writers.

1.4 Regional-words-attempt

The *Regional-words-attempt* was inspired by the fact that even for such ordinary things like small bakery products or other food exists a confusing mess of words. There is the funny effect that some of these foods got names of towns different from the speakers-town. So a small sausage in Berlin is called *Wiener* whilst in Vienna it is called *Frankfurter*. This shows that the use of language and especially the use of lexical items differs throughout regions. The aim of our regional word attempt was to exploit this fact, try to

map different words to specific regions, and to create a vector-space model of words for each region. In doing so we should be able to compare new Tweets against these models and assign it to one region.

1.5 Expectations

We expected that it was possible to map Tweets to a specific region if this Tweet contains material which will be relatively prominent in this region. But we were aware about the fact that a list serving as the base of our *Regional-words-attempt* was very vulnerable to criticism because we ourselves lay the ground for further calculations. This means that not the real use of language of Twitter-users were the starting point of this attempt but academical investigations. The opposite is the case for the Geo-location-attempt: Here all calculations were done based on real Twitter data.

We therefore expected our *Geo-location-attempt* to show better results than the *Regional-words-attempt.*

1.6 Sources, used corpora

We created two main-corpora: *Regional word corpus* and *Geo data corpus.*

For our regional word corpus we created a list of words which we thought would be good candidates to represent one or at maximum four regions. This list of words was inspired by the work of the *Atlas der deutschen Alltagssprache* – a project of Universität Salzburg and Université de Liège which aim is to collect information about everday use of different words of native German speakers depending on their origin. This list was used to collect 12,878 Tweets from August 1, 2013 to August 8, 2013 using the Twitter-API (Regioword-corpus). Because we thought that some regio-words will be unknown or very rare we decided to switch off language detection by *LangID* and to check only if a Tweet containing one of our words got one of the country IDs *de, at* or *ch.* We are aware of the fact that by chance some of our regio-words could also be words in other languages but we thought that combining this word with the country IDs was sufficient to minimize that chance. For reasons of comparison additionally we created two corpora whose content probably don't contain words from out list - at least we don't care about it: a corpus out of 12,878 Tweets (Scheffler-Regioword-corpus) and a corpus containing all 1,494.994 Tweets (Scheffler-oneday-corpus) from August 29, 2012 - both from the Scheffler-corpus (see below).
Our Geo data corpus has been extracted from the Scheffler-corpus. We created balanced sets, where the amount of Tweets are the same for all regions.

Because the fewest number of Tweets (8787) came from region 6, *Österreich,* we ended up with only 61509 Tweets for all regions.

The source for parts of the regional word corpus and geo data corpus was the *Scheffler-corpus*. This is a collection containing Tweets from April 1, 2012 to April 30, 2012 collected by Tatjana Scheffler using the Twitter-API which are recognized as German Tweets by *LangID*.

1.7 Regions

The considerations of our specific division of the German-speaking area into regions is based on were mainly predetermined by the *Regional word attempt*. The data situation in the *Atlas der deutschen Alltagssprache* showed us the scale at which we could divide the language area based on regional everyday language. We also did the actual defining of the particular regions examining the data there. But Twitter as our field of application did contribute a crucial criterion here, too: We wanted regions to be formed that would provide a sufficient amount of Tweets sent from there. So for instance the data of the *Atlas der deutschen Alltagssprache* also showed a small region around Saarland and Luxembourg to have characteristic idiosyncrasies, but we did not carry over it because of too few expectable Tweets.

2 Algorithms

2.1 Data structure and overview

Our basic datastructure is a vector whose dimensions represent regions, thus we use seven-dimensional vectors for working with our seven defined regions. On one hand we apply vectors to words (types) and on the other hand to Tweets (documents). They show the strength of the word's or the Tweet's affiliation to the individual regions.

So for instance the Word *Porree* could be equally significant for the regions *Ostdeutschland, Norddeutschland* and *Westdeutschland*. Consequently we would guess the probability that a usage of this word indicates origin from one of these regions to be one third for each of them, and to be zero for each of the other regions.

$$\vec{V}_{Porree} = \begin{pmatrix} 0,33 \\ 0,33 \\ 0,33 \\ 0,0 \\ 0,0 \\ 0,0 \\ 0,0 \end{pmatrix} \begin{matrix} \text{Ostdeutschland} \\ \text{Norddeutschland} \\ \text{Westdeutschland} \\ \text{Bayern} \\ \text{Südwestdeutschland} \\ \text{Schweiz} \\ \text{Österreich} \end{matrix}$$

As a seed we use initial word-vectors *(generation 0)*. What comes next is an accumulation stage, during which more regionally salient words are supposed to be found using training data from Twitter. This is done by our main algorithm, which calculates a following generation of word-vectors from a previous one. Thus it can be executed in a loop running any number of cycles you like. Eventually we get a final generation of word-vectors, from which after that we calculate the average vector (normalized to length 1). At this stage our model has passed training and is ready to be applied.

In application phase an unseen Tweet comes in. We calculate a tweet-vector for it and subtract it from the average vector. The difference vector shows the region we eventually assign the Tweet to.

2.2 Main algorithm

The main algorithm calculates a new generation of word-vectors based on an existing generation and a training corpus of Tweets.

It iterates over all Tweets in the corpus. For each Tweet a tweet-vector is calculated. Therefore the existing generation's word-vector for each token

in the Tweet simply is added, provided it exists. Subsequently, the resulting tweet-vector conversely is added to the new generation's word-vector. In case any of the Tweet's tokens are not represented in the old generation of word-vectors they now join for the new generation; thus the number of word-vectors increases. Values for already existing words do also change, though.

In a nutshell, first a tweet-vector is calculated for each Tweet in the training corpus based on the existing generation's word-vectors for the Tweet's words, and then a word-vector is calculated for each word in the corpus based on the tweet-vectors for all the Tweets the word occurs in.

Data:
WV_0: existing set of word-vectors $WV_0(word) = \vec{V}_{word}$
$Tweets$: set of Tweets

Result:
WV_1: enriched set of word-vectors $WV_1(word) = \vec{V}_{word}$

$WV_1 \leftarrow \emptyset$
foreach *Tweet in Tweets* **do**
  $\vec{V}_{Tweet} \leftarrow (0, 0, 0, 0, 0, 0, 0)$
  foreach *Token in Tweet* **do**
    $\vec{V}_{Tweet} \leftarrow \vec{V}_{Tweet} + WV_0(Token)$
  end
  foreach *Token in Tweet* **do**
    $WV_1(Token) \leftarrow WV_1(Token) + \vec{V}_{Tweet}$
  end
end
foreach $\vec{V}_{Word}$ *in* WV_1 **do**
  $\vec{V}_{Word} \leftarrow \text{normalize}(\vec{V}_{Word})$
end
return WV_1

Algorithm 1: Tweegion main algorithm

2.3 Difference between both attempts

The difference between the *Regional-word-attempt* and the *Geo-location-attempt* lies in the seed for the main algorithm. For the *Regional-word-attempt* we manually defined about 200 initial word-vectors based on a source for regional expressions.

With the *Geo-location-attempt*, by contrast, we have the initial word-vectors learned by machine, too. For this purpose we use a special collection of geo annotated Tweets. Based on its coordinates each Tweet is assigned to one of our seven regions. Similarly as we do it in the main algorithm, we now calculate word-vectors based on the Tweets the words occur in.

Our results are ten thousands of initial word-vectors for the *Geo-location-attempt* depending on the size of the special training corpus. So this attempt tendentially needs less cycles of the main algorithm for the same amount of final word-vectors.

2.4 Normalizing versions

Over the course of our work we developed multiple versions of our main algorithm which differ in treatment of the word-vectors at the end of each cycle, i. e. subsequent to the calculation of Tweet and word-vectors. This treatment directly affects the following calculation of tweet-vectors, be it in a possible further cycle of the main algorithm or in the application stage, i. e. for classifying an unseen Tweet.

The original version, *Normalized,* includes a normalization of all word-vectors to sum 1. This means every type is equivalent regardless of the number of tokens. During test runs we found this approach to have an extreme bias towards rare words, since they only occur in one or a few Tweets and are therefore graded strongly salient for a certain region.

$$\text{normalize}(\vec{v}) = \frac{\vec{v}}{\sum_i v_i}$$

Our second version, *Linear,* then came into being by simply omitting the normalization. With this approach, every type now has a weight directly linearly dependent on its frequency as a token. This alternative however, as it turned out, lended high frequent words such a huge weight that regional salience, being more common with words of medium frequency, suffered a heavy loss and only tiny deviations from the average vector remained.

$$\text{normalize}(\vec{v}) = \vec{v}$$

Driven by this experiences we decided to find a trade-off between constant and linear dependency. So we developed the version *Log,* where the vector's length is changed in such a way, that its sum increased by 1 thereby is logarithmized to base 2.

$$sim(\vec{q}, \vec{d_j}) = \frac{\sum_{i=1}^{N} w_{i,q} \times w_{i,j}}{\sqrt{\sum_{i=1}^{N} w_{i,q}^2} \times \sqrt{\sum_{i=1}^{N} w_{i,j}^2}}$$

Figure 1: Cosine similarity

$$\text{normalize}(\vec{v}) = \frac{\vec{v}}{\sum_i v_i} * \log_2(\sum_i v_i + 1)$$

In the case of *Root,* the square root is used rather than the logarithm to base 2.

$$\text{normalize}(\vec{v}) = \frac{\vec{v}}{\sum_i v_i} * \sqrt{\sum_i v_i}$$

2.5 Classifying a tweet

During application an unseen Tweet comes in and is to be assigned to a region. For this purpose a tweet-vector is calculated for it. This is done by searching its text for words we have got word-vectors for. All word-vectors we find are then added up.

Afterwards the *cosine similarity* between the tweet-vector and the average vector is calculated. In case of too large similarity no classification is carried out, see the chapter *Cosine similarity.*

In the event of the tweet-vector being salient enough it is now normalized to length 1. For assigning the Tweet to one of the regions the difference between the average vector and the tweet-vector is calculated, this giving a difference vector pointing to the direction searched for. More precisely, the vector's maximum value reveales the region the vector points to the most. This region is the result of our classification.

2.6 Cosine similarity

Although Tweets may differ from one region to another in some way, a huge percentage of the German Twitter users write their messages exclusively in standard German with no signs of any regional influence whatsoever.
For example the following Tweet just appeared in one of our timelines:

'Ich glaube @Drahflow tippt noch schneller als er redet. ;) #om13'

Keeping in mind that usernames, hashtags, smileys and punctuation are being removed in the pre-processing, this Tweet contains way too ordinary words to be assigned to a specific region. It is written in pure standard language and therefore could be sent from a village in Bavaria as well as from Berlin, and depending on the data we use to train our algorithm with, the result of the program could be *Österreich* or *Norddeutschland* as well.

To face this problem we decided to filter out this kind of indistinguishable Tweets in order to get more reliable results, thus increasing the accuracy of the algorithm.
Our idea was to determine the average Tweet-vector $\vec{\vec{d}}$ of **all** the Tweets in the training-corpus and to compare the Tweet-vector $\vec{q}$ of an **inputted** Tweet with it, using the cosine metric (see Figure 1) as recommended in [JM09, p. 805]. If the similarity between both vectors is smaller than a specified threshold, we continue to map the Tweet to one of the region, otherwise we stop and return a message that the Tweet is written in standard German hence cannot be classified. In the example in figure 2, the vector $\vec{q}$ for the Tweet mentioned above is compared to the average Tweet-vector $\vec{\vec{d}}$, returning a very high similarity of 0.9986. Nevertheless the raw algorithm would state the Tweet was most likely sent from Switzerland.

The most difficult part was to find the right threshold that separates the too ordinary Tweets from the significantly regional ones. To make a guess which percentage of all Tweets are written in standard German, we calculated the cosine similarity of all Tweets in the training-corpus to the average vector, sorted them and had a look at the distribution (figure 3).
The result was not surprising: More than 80% of all Tweets had a similarity of 0.9 or higher. Or, seen from another point of view, only 20% of all Tweets differ enough from the average vector to calculate reliable results.

$$
\begin{aligned}
q &= \text{'Ich glaube @Drahflow tippt noch schneller als er redet. ;) \#om13'} \\
\vec{q} &= (0.427, 0.38, 0.4, 0.39, 0.391, 0.602, 0.41) \\
\vec{\vec{d}} &= \frac{\sum_{i=0}^{N} d_i}{N} = (0.376, 0.336, 0.349, 0.346, 0.361, 0.481, 0.379) \\
sim(\vec{q}, \vec{\vec{d}}) &= 0.9986
\end{aligned}
$$

Figure 2: Example for the similarity calculation

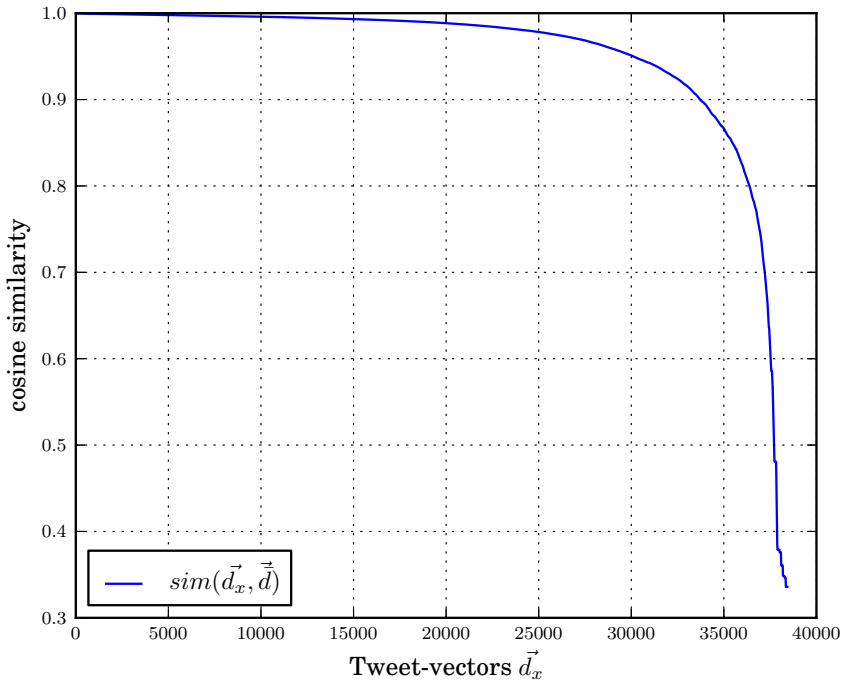


Figure 3: Cosine similarity between all Tweet-vectors and the average Tweet-vector

The result of the algorithm we created (see algorithm 2), is depending on the Tweets to calculate the average vector, a guess of the amount of regional Tweets and the set of word-vectors, which is created in the main-algorithm. By default, the Tweets are the same dataset as used in the main-algorithm. We decided to use a guess on which the similarity threshold is computed instead of directly entering a value for the threshold. This way, we receive more reliable results and are able to compare the use of different datasets.

In the experiments of the *Geo-location-attempt* we tested different thresholds to find a good balance between a reliable classification with a high accuracy and the coverage of as many Tweets as possible and applied the results to further experiments in the *Regional-words-attempt*.

Data:
$tweets$: Set of geo-annotated documents
$guess$: $0 \leq guess \leq 1$, guessed amount of regional Tweets
WV: Set of word-vectors
Result:
$threshold$: $0 \leq threshold \leq 1$, cosine similarity threshold

$tweetvectors \leftarrow \emptyset$;
foreach *tweet in tweets* **do**
  $\vec{tweet} \leftarrow (0,0,0,0,0,0,0)$;
  forall the *token in tweet* **do**
    if $token \in WV$ **then**
      $\vec{tweet} \leftarrow \vec{tweet} + WV(token)$;
    end
    $tweetvectors \leftarrow tweetvectors \cup \{\vec{tweet}\}$;
  end
end
$\vec{average} \leftarrow (0,0,0,0,0,0,0)$;
foreach $\vec{tweet}$ *in* $tweetvectors$ **do**
  $\vec{average} \leftarrow \vec{average} + \vec{tweet}$;
end
$\vec{average} \leftarrow \dfrac{\vec{average}}{l(tweetvectors)}$;
$vectorlist \leftarrow \emptyset$;
foreach $\vec{tweet}$ *in* $tweetvectors$ **do**
  $similarity \leftarrow sim(\vec{tweet}, \vec{average})$;
  $vectorlist \leftarrow append(similarity)$;
end
$vectorlist.sort()$;
$threshold = vectorlist[int(guess \times l(vectorlist))]$;
return *threshold*;

Algorithm 2: Cosine similarity calculation algorithm

3 Geo-location-attempt

3.1 Introduction

While the foundation of all calculations in a regional word attempt is a manually created list of only a few hundred words along with their probability distribiutions, we first wanted to try using machine learning techniques to generate such list automatically.

Although the manually created list and its values are based on scientific research, it could mark the weak spot of the regional word attempt for number of reasons. For example people in a specific region could use a typical word in their everyday language while speaking to their friends and family, but it is not quite sure that they also use this words in their written language, even if it is only their private Twitter account. In addition, the list is way too short to cover only fraction of the words people use on Twitter, so the end results mostly rely on the data that is generated during the main algorithm's loops.

It seemed we had no other choice but to trust this generated values, so we came up with the idea of first skipping the manually created word list and use an automatically created one – based on a corpus of geo annotated Tweets instead.

Before entering the main loop, we had to write an algorithm that learns the distribution in all seven regions for all the words in the corpus. This way we are generating a list that covers a very huge percentage of all words that could occur in a Tweet.

We struggled to find a good solution for representing the regions in a way that would allow us to easily check from which of them a geo annotated Tweet was sent. After a few unsuccessful approaches we decided to define polygons for the regions that neither are overlapping each other nor leave gaps between them. For the value of the points we simply used their longitude and latitude coordinates, which can be represented as floating point numbers.

We did not implement a point-in-polygon algorithm ourself, but used the version of Joel Lawhead published on geospatialpython.com [Law11] instead. To find out where a Tweet was sent from, we iterate over all regions and return the first one, where the point-in-polygon function returns true. In this first step, the tweet-vector starts with the value 0 for all features, except the one standing for the region of origin, which we assign the value 1 to.

Let us assume that the coordinates of some Tweet reveal that its region is *Westdeutschland,* represented by the feature with the index 3. Therefore the tweet-vector is $(0, 0, 1, 0, 0, 0, 0)$. The next step consists in iterating over all

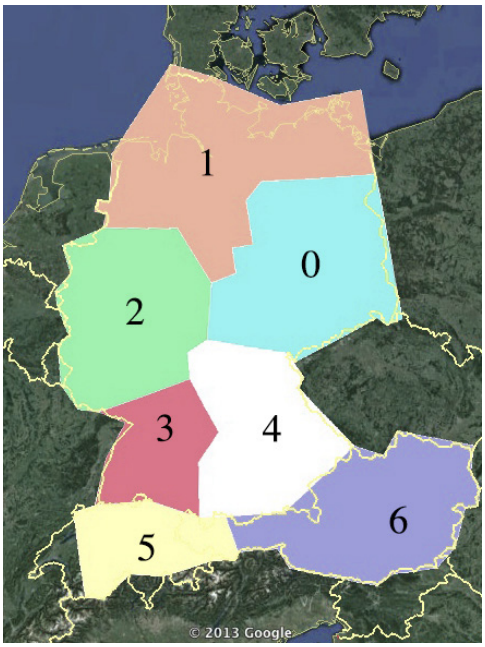


Figure 4: Map of the regions and the index of their feature used in the vectors represented as polygons

Data:
tweets: Corpus of geo-annotated documents
stopwords: List of stopwords
Result:
WV: normalized word-vectors, representing
the probability distribiution for each word

```
1  WV ← ∅;
2  foreach tweet in tweets do
3  |   region ← Classify(tweet);
4  |   t⃗weet ← CreateVector(region);
5  |   forall the token in tweet do
6  |   |   if token ∉ stopwords then
7  |   |   |   WV(token) ← WV(token) + t⃗weet;
8  |   |   end
9  |   end
10 end
11 foreach w⃗ord in WV do
12 |   w⃗ord ← normalize(w⃗ord);
13 end
14 return WV;
```

Algorithm 3: Geo-algorithm

tokens in the Tweet and add the tweet-vector to their according word-vectors. To put it simply, we increase the counter for the specific feature of all occurring tokens by one. In the final step we iterate over all word-vectors and use our `normalize()`-function to attain the probability distribution showing how likely it is that the token is used in a specific region. By this normalization we make sure the format of the outcome is comparable to the initial list of the regional-word attempt and we can use it in the main algorithm without any adjustments.

3.2 Source of data

The datasets used for the generation of the first word-vectors have been extracted from the Scheffler-corpus using the same classification function as described before. Although the corpus mostly consists of German Tweets (filtered with *LangID*), about 20% of the Tweets with geo-location have not been sent from one of the defined regions and therefore have been ignored. In addition, we filtered Retweets, Hashtags for the location-based social network

Example Tweet: $d =$ 'Hello Twitter!' from the region 'Westdeutschland'.

$$
\begin{aligned}
\vec{t}_{hello} &= (1, 3, 2, 5, 2, 1, 0) \\
\vec{t}_{twitter} &= (0, 0, 0, 0, 0, 0, 0) \\
\vec{d} &= (0, 0, 1, 0, 0, 0, 0) \\
\vec{t}'_{hello} &= \vec{t}_{hello} + \vec{d} = (1, 3, 3, 5, 2, 1, 0) \\
\vec{t}'_{twitter} &= \vec{t}_{twitter} + \vec{d} = (0, 0, 1, 0, 0, 0, 0) \\
\texttt{normalize}(\vec{t}'_{hello}) &= (0.04, 0.13, 0.13, 0.22, 0009, 0.04, 0) \\
\texttt{normalize}(\vec{t}'_{twitter}) &= (0, 0, 1, 0, 0, 0, 0)
\end{aligned}
$$

Figure 5: Example for the creation of the initial word-vectors

'Foursquare' (*#4sq* and *#foursquare*) as well as Hashtags from automatically sent messages and blocked the UserIDs for a few very frequent bots, that transfer their geo-location. The remaining data consists almost exclusively of German human-composed Tweets (although a very few English Tweets have remained in the corpus).

We created balanced sets, where the amount of Tweets are the same for all regions. As the fewest Tweets came from the region 6, *Österreich* (8,787), we ended up with only 61,509 Tweets for all regions. In order to create a gold-standard, we subtracted 150 Tweets from each region, leaving us 60,459 for the training process.

Our assumption was that datasets of different sizes lead to different results, and we wondered in what way it would affect the accuracy to use the same data for the creation of the word-vectors in the geo-algorithm as we did for the main-algorithm. Therefore we created three balanced sets, one unbalanced set of all geo annotated Tweets and as a reference one huge set of unfiltered normal Tweets. Of course none of these sets do contain any Tweets from the gold-standard.

1. *balanced-21k* with 3,000 Tweets from each region.
2. *balanced-39k* consisting of the remaining 39,456 Tweets
3. *balanced-61k* combining the both previous sets.
4. *geo-175k* with all Tweets from the corpus that were sent from one of the defined regions. Not balanced.

5. *all-1500k* contains 1.5 million mostly not geo annotated Tweets.

3.3 Experiments

As the algorithms leave us five parameters to alter, we had to run a series of experiments to find the optimal value for them. In the following we present our procedure and the results.

3.3.1 Dataset combinations

In order to give a solid foundation which data we should use in further experiments, we started to compare the combination of the five datasets we created. We wondered in what way it would affect the accuracy if we trained for instance the geo-algorithm with a small set of geo-annotated data but used a very big set of Tweets for the learning of the word-vectors in the loop of the main-algorithm. Another question was what results we would get if the data in the main-algorithm contained Tweets that had been used in the geo-algorithm before.

Our expectation was in general that the accuracy would rise the more data we used and that the reuse of Tweets in the main-algorithm would lower it, because this data already had an effect and using it again would cause any change.

Unsurprisingly, using the smallest set, *balanced-21k*, for the training of the geo-algorithm nearly always produced the lowest accuracy. But our hypothesis about the reuse of data was disproved: The combination of the *balanced-39k* set for the learning in the geo-algorithm and *balanced-61k* for the main-algorithm generated the third best result with an accuracy of 0.345.

We also discovered that the use of balanced data in the geo-algorithm, where all regions are represented by the same amount of documents, is cru-

Geo dataset	balanced-21k	balanced-39k	balanced-61k	geo-175k	all-1500k
balanced-21k	0.319	0.308	0.304	0.307	0.308
balanced-39k	0.306	0.353	**0.345**	**0.368**	0.306
balanced-61k	**0.336**	**0.361**	0.344	0.360	0.332
unbalanced-175k	0.303	0.343	0.322	0.352	**0.340**

Calculation method: *Normalized*; Stopwords: *Top 200*; Loops: *1*; Guessed amount of non-regional Tweets: *60%*, leading to a similarity threshold between *0.999* and *0.991*, depending on the dataset.

Table 1: Comparison of the combination of all datasets

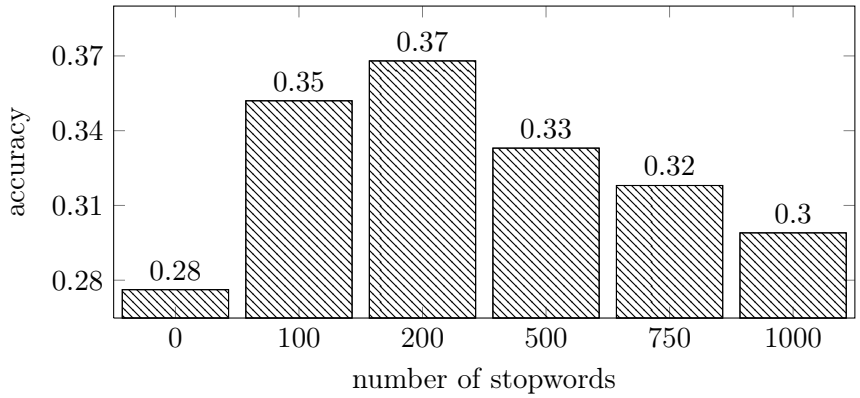
cial for good results. Having a look at the row for the *unbalanced-175k*, it produces a very bad accuracy. Even in combination with the set containing 1.5 million Tweets *(all-1500k),* it marks the second last rank.

This experiment showed us that we should use the *balanced-39k*-set for the training of the geo-algorithm and the *geo-175k*-set for the main-algorithm, because it generated the best accuracy of 0.365. We were surprised that the combination *balanced-39k*/*all-1500k* had a worse result, but interpreted it as a result of the unfiltered data of the *all-1500k*-set.

3.3.2 Comparison of different stopword-lists

Zipf's law tells us that a very small amount of tokens are found extremely often in corpora, while most tokens appear only a few times. As it is much more likely for a more seldom word to have a regional meaning, we decided to filter out the most frequent token.

We came up with the idea that it is not reasonable to use an existing stopword-list for the German language because people on Twitter could use specific words and symbols more often than in a standard German text. In addition, Tweets from spam bots or automatically sent messages that always contain the same words disturb the statistics and have to be filtered. So we used the Scheffler-corpus to create our own stopword-list using our modified version of Christopher Potts' tokenizer and counting the frequency of all tokens.



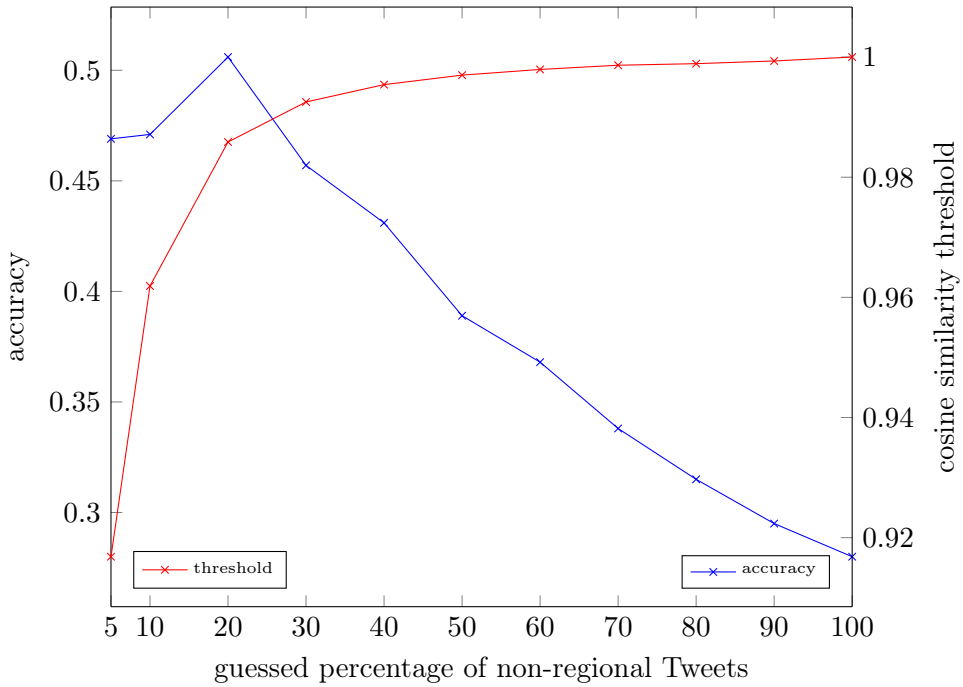
Calculation method: *Normalized*; Geo-dataset: *balanced-39k*, Main-dataset: *geo-175k*; Loops: *1*; Guessed amount of regional Tweets: *60%*

Figure 6: Accuracy of different stopword-lists.

We were unsure about how many token should appear on a stopword-list, so we created five of them, containing the most *100*, *200*, *500*, *750* and *1000* frequent words. Actually, we expected to have a better accuracy the more stopwords we use, but it turned out to be different.

Figure 6 shows clearly that more than 200 stopwords lead to worse results, probably because these words do have a regional meaning. On the other hand, only 100 stopwords have a lower accuracy than the list containing 200 entries. The fact, that the results of the run with no stopword filtering at all had the lowest accuracy, reveals the importance of using a stopword-list in general.

As the list containing 200 stopwords leads to the best accuracy of 0.37, we used this one in further experiments.



Calculation method: *Normalized*; Geo-dataset: *balanced-39k*, Main-dataset: *geo-175k*; Stopwords: *Top 200*; Loops: *1*;

Figure 7: Relation between the amount of regional Tweets and the accuracy.

3.3.3 Guessing the amount of regional Tweets

In section 2, *Algorithms* we talked about the advantages and disadvantages of filtering too ordinary Tweets using the cosine similarity metric. Summarized, we gain much better results considering the accuracy, but have to accept that we cannot classify a huge percentage of Tweets because they do not differ enough from the average tweet-vector.

We wanted to know what accuracy we could achieve, so we ran the program a few times, changing the parameter guessing the amount of regional Tweets. We started at 100% with no filtering at all and went down to an extreme of 5%.

As figure 7 illustrates, the results approved our hypothesis that the accuracy rises, the lower we guess. At the same time the similarity threshold lowers to values of 0.92 at a guess of 5%, showing that even these documents are very similar to the average tweet-vector.

The highest accuracy we could achieve had a value of 0.506, guessing that only 20% of all Tweets are regional salient. We were extremely happy about this result, because we had an accuracy lower than 0.2, when we started our first experiments in the regional-word attempt. Nevertheless, we have to keep in mind that 80% of all Tweet will be ignored, so for the majority of inputted Tweets we will not get any other results than that it is most likely written in standard German.

If we lower our guessing to 10% or 5%, the accuracy lowers again, probably because there are not enough Tweets left in the gold-standard to give a trustworthy result.

There is no ideal value for this parameter and eventually, people who will use this algorithm have to figure it out themselves, according to what they want to achieve: A higher coverage of data or a more reliable result. In the experiments to come, we decided to use a guessing of 20%, because our aim is to achieve the highest possible accuracy.

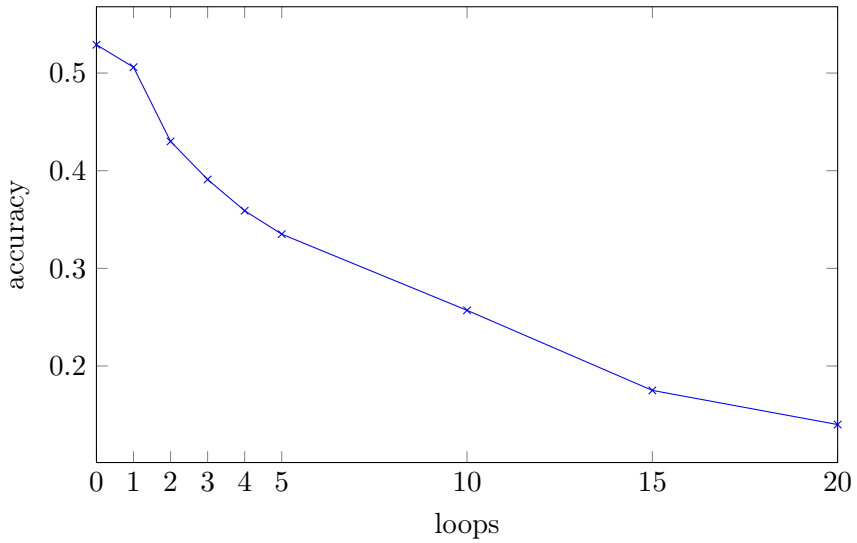
3.3.4 The right amount of loops

The idea behind the loops in the main-algorithm was to estimate the probability distinction from which region a Tweet could have been sent from as many words as possible. Especially for the regional-list attempt this was a crucial approach to prevent the sparse data problem.

In this attempt on the other hand, this problem is not urgent, because we create a very large list of word-vectors based on a geo-annotated corpus. Therefore we asked ourselves, which amount of loops would lead to the best

results. The more the better or the opposite?

To find an answer to this question, we took the parameters we had proven to be the best choice in the experiments before, but varied the value for the loop-parameter from 0 to 20 and measured the accuracy. We were a bit puzzled to find out, that the main loop, even if it was only entered one time, lowers the accuracy by 0.023. Our new record for the accuracy becomes 0.53, if we chose a value of 0. Please keep in mind, that even though we do not enter the loop, the calculation of the word-vectors is always performed one time before that.

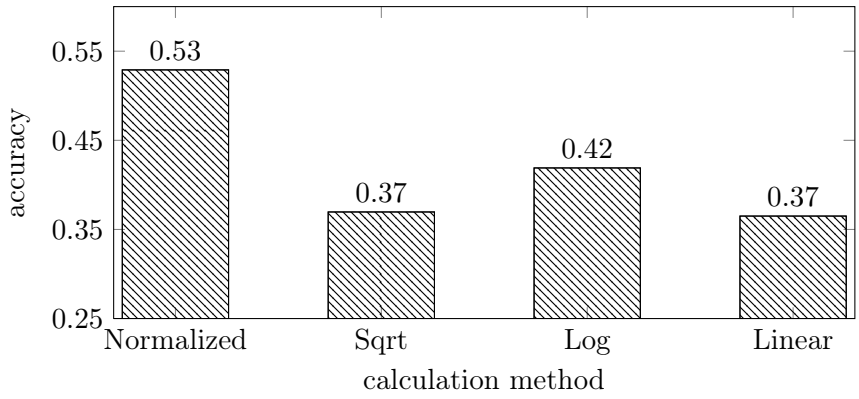


Calculation method: *Normalized*; Geo-dataset: *balanced-39k*, Main-dataset: *geo-175k*; Stopwords: *Top 200*; Guessed amount of regional Tweets: *20%*

Figure 8: Amount of loops in the algorithm and their effect on the accuracy.

If we have a look at the graph in figure 8, we can see clearly that the accuracy drastically decreases, the more loops we enter. So our assumption, that the loops in the main-algorithm would lead to better results has been falsified. We then came up with the explanation, that the more often we use a loop, the more do all word-vectors approach an average vector and in this way lose all signs of a specific region.

Since the importance of the dataset for the main-algorithm drops, if we choose not to use the loop, we repeated the experiment in section 3.3.1 with the loop = 0 setting. This corpus now acts only as a foundation for calculation the average tweet-vector for the cosine similarity and for calculation



Geo-dataset: *balanced-39k*, Main-dataset: *geo-175k*; Stopwords: *Top 200*; Loops: *0*; Guessed amount of regional Tweets: *20%*

Figure 9: Comparission of the four different calculation methods.

the word-vectors one time.

Fortunately, table 2 proves that the results were still the same, ranking the *balanced-39k* set for the geo-algorithm and the *geo-175k* corpus for the main-algorithm as the best combination.

	balanced-21k	balanced-39k	balanced-61k	geo-175k
geo-175k	0.308	**0.378**	0.371	0.374

Calculation method: *Normalized*; Stopwords: *Top 200*; Loops: *0*; Guessed amount of non-regional Tweets: *60%*.

Table 2: Comparing the accuracy of the combination between all geo-datasets and the *geo-175k* set for the main-algorithm with the loops=0 setting.

3.3.5 Comparison of different calculation methods

As mentioned in the 'Algorithms'-section before, we developed four different versions of the algorithms, that differ mostly in the way of normalization. We named our first attempt, that we also used in the previous experiments, *Normalized*, the other calculation methods are *Log*, *Sqrt* and *Linear*. For a discussion of the details of the algorithms, please have a look at the corresponding section.

The parameters for the calculation method is the last one we have to find an optimal value for. As recommended in the experiments before, we chose the *balanced-39k* set for the geo-algorithm, the *unbalanced-175k* dataset as a foundation for the calculation of the cosine similarity, which depends on our guess of the amount of regional Tweets which we decided to be *20%*. The amount of loops is *0* and we filtered using the *200* most used token as stopword-list.

The comparison of the accuracy in figure 9 reveals, that the previously used method is clearly the most successful one. Only the *Log*-attempt also reached a relatively good result of 0.42, while *Sqrt* and *Linear* have obviously been a bad choice.

3.4 Conclusion

In this section, we have developed an alternative way of constructing a list of initial word-vectors automatically learned from geo-annotated documents.

To find the best values for all five parameters in terms of accuracy, we tested step-by-step all possible combinations, leading to a final result of an accuracy of *0.53*. The best choice is the *Normalized* method in combination with the *balanced-39k* set for the learning in the geo-algorithm and the *geo-175k* corpus for the main-algorithm and the calculation of the cosine similarity metric, where we guessed, that only 20% of all Tweets have a regional influence. It seems to be the best to skip the loop in main-algorithm and we found out, that *200* is the best amount for the stopword-filtering.

Although an accuracy of 0.53 is very good, compared to a null hypothesis of 0.14, it is still not good enough to satisfy us completely. We believe, that a big corpus of several million Tweets, that are filtered and preprocessed in the way as the geo-datasets, will lead us to an even better accuracy, if we use it as data for the main-algorithm. Also, we think that a fine-tuning of the calculation method (or even a completely new one) could drastically improve the results.

4 Regional-word-attempt

4.1 Idea

The main idea behind the *Regional-word-attempt* is based on expressions of regional everyday language. This comprises expressions which are distributed somewhat complementary within the German-speaking area and are used by speakers in normal colloquial language, i. e. not necessarily just dialectal. Famous examples include *Samstag* versus *Sonnabend* and *viertel vor* versus *dreiviertel*. Since we restricted ourselves to unigrams for the sake of simplicity, the latter falls out of our examination's range.

4.2 Initial data

As a source for the regional expressions we used the *Atlas der Alltagssprache.* This is a collaborative project of the Universität Salzburg and the Université de Liège and is based on surveys of speakers from anywhere within the German-speaking area. The results were published in the form of maps along with explanations.

Collecting data from the Atlas der Alltagssprache we had to consider several factors. Of course only idiosyncrasies which are reflected in the written form came into consideration; data regarding for instance vowel qualities were out of the question. First of all we were looking for expressions which would divide the language area precisely as possible, i. e. show little to no overlapping. At the same time they should actually be regionally salient and not cover almost the whole language area like for instance *Backofen.* Our limit here was to assign a word to a maximum of four of our seven regions. Furthermore, homonyms and polysemes were inappropriate for our purposes, so for example most of the regional words for *attic,* including *Boden, Speicher* and *Bühne,* were ruled out. We also went without very short expressions like *wa* (Berlin dialect for *...right?*) because of the high chance of coincidence with abbreviations, cutted forms etc.

From the data we created the initial word vectors for this attempt in the form of a csv file *(comma separated values).* Doing this we assigned every word to n of our seven regions. For this regions the word vector got the value $\frac{1}{n}$, for the remaining ones it got the value 0.

4.3 Data accumulation

We used our threefold *Regional word corpus* (Regioword-corpus, Scheffler-Regioword-corpus, Scheffler-oneday-corpus). On base of our regional-word-

list and these corpora we created initial word vectors for each corpus. Obviously the chance of getting much smaller vectors is bigger for both the Scheffler-Regioword-corpus and Scheffler-oneday-corpus because many of our regional words are not within these corpora.

To avoid gross errors doing this, we removed stopwords beside usertags, hashtags and URLs from the training corpus. What's more, these derivative word vectors are calculated based on all found Tweets containing the respective word. So non-regional words, which naturally also occur in those Tweets, eventually get an average-like, regionally indifferent vector.

In order to actually gain enough data we thought of repeating this accumulation using the newly acquired word vectors. That's how our main algorithm came into being.

We used the results of the *Geo-location-attempt* and did not alter neither the number of stopwords (200) nor the percentage of guessed non-regional tweets (80%)

4.3.1 Expectations

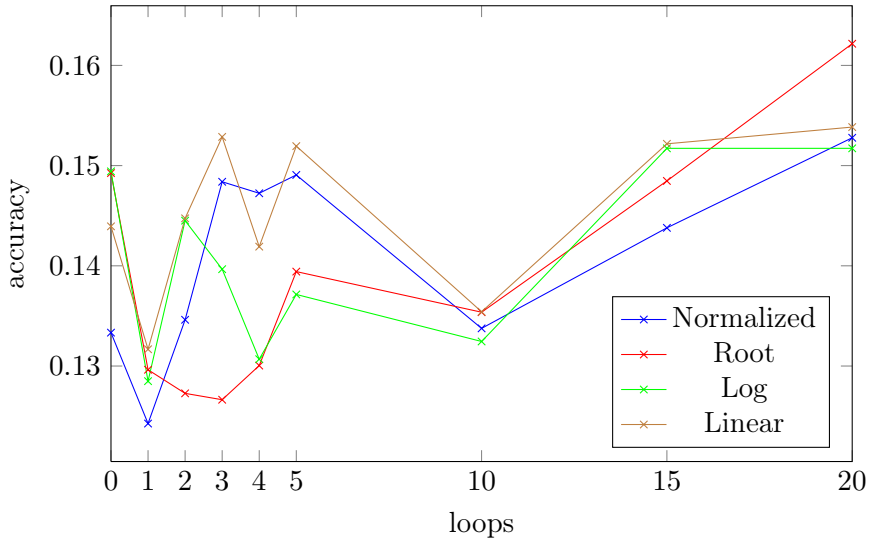
We expected the Regioword-corpus to lead to better results. We were convinced that the use of a initial list of regional words should have an enormous influence during the process of computing vectors. Only in this case we started with tweets which should contain regional words. And only in this case we could be somewhat sure about regional origins of the tweet-writer.

The opposite is the case for the two other corpora. We expected that the two corpora with arbitrary content should lead to arbitrary results (less than 0,14 % accuracy). Because we could not know which regional words are in these corpora (or if there are regional words at all) it also could be the case, that our algorithm is computing a arbitrary vector.

This should lead to a huge advantage of the Regioword-corpus over the two others.

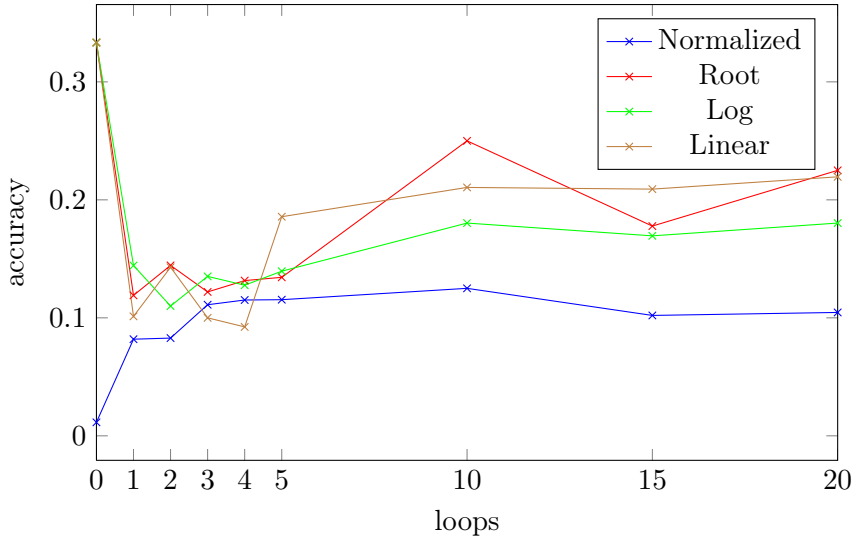
4.4 Experiments

We wanted to show the influence of the initial word vector for further calculation. So we tested our algorithm against our three corpora. Because the *Geo-location-attempt* lead to the result that the use of 200 stopwords and guessing 80 % of the tweets as non-regional the only parameter we changed was the number of loops and the methods of normalisation.



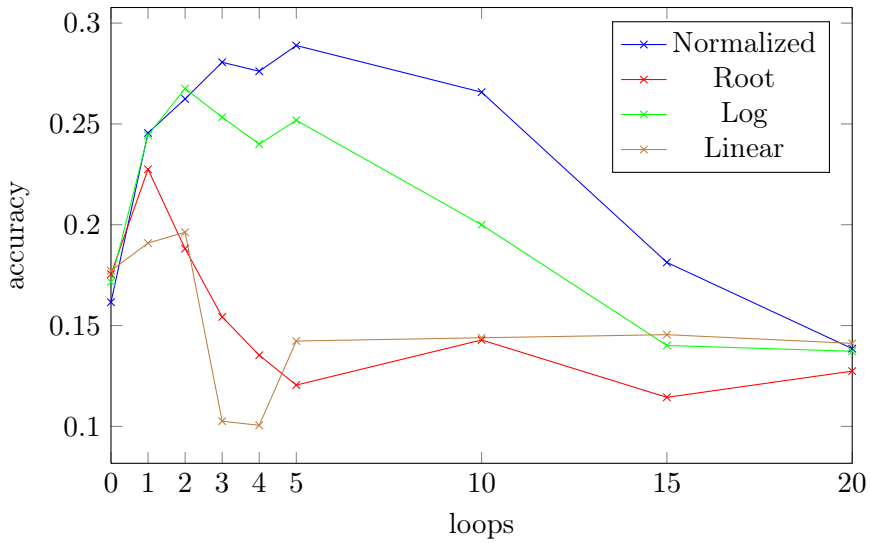
Stopwords: *Top 200*; Guessed amount of regional Tweets: *20%*

Figure 10: Amount of loops in the algorithm and their effect on the accuracy for Regioword-corpus



Stopwords: *Top 200*; Guessed amount of regional Tweets: *20%*

Figure 11: Amount of loops in the algorithm and their effect on the accuracy for Scheffler-Regioword-corpus.



Stopwords: *Top 200*; Guessed amount of regional Tweets: *20%*

Figure 12: Amount of loops in the algorithm and their effect on the accuracy for Scheffler-oneday-corpus.

4.4.1 Results

The results of our experiments were quite disappointing. Neither there could be found an advantage of the Regioword-corpus nor any systematic differences between our normalisation-methods.

Whilst in the *Geo-location-attempt* we found a steady decreasing value of accuracy with increasing number of loops this behavior could not be found during our experiments with the *Regio-words-attempt*. There is an up and down of accuracy rates for all corpora and with all normalisation-methods.

But the worst result of the *Regio-words-attempt* was the accuracy itself: Our expectations about a huge advantage of the Regio-word-corpus were totally demolished! Experiments based on this corpus had the worst results - accuracy rates of approx. 0,15% are only slightly (if at all) above the level of guessing! Even if some combination of loops and normalisation-methods with other corpora lead to slightly better results this results are fare from beeing useful.

4.4.2 Discussion

The results of the *Regio-words-attempt* were very disappointing. The most outstanding result was the fact that calculating word-vectors had no real influence or advantage over guessing the right region.

The fact that executing our algorithm based on our list of regional words lead to results which were even worse than based on arbitrary corpora was surprising. Probably this result did not mean that calculations based on a regional word list behave worse than calculations based on arbitrary material. The results only gave us a hint that *our* regional word list wasn't assembled adequate. Our word list contains many words that are expressions for foods and kitchen equipment. We believe that this very restrict compilation doesn't reflect the use of language of mostly young Twitter users. With our word list we probably caught the use of language of an older generation.

5 Conclusion

The results of our experiments were very surprising. Our expectation concerning the differences between *Geo-location-attempt* and *Regio-words-attempt* were confirmed. Nevertheless we were suprised by the very weak results of the *Regio-words-attempt*. Whilst the *Geo-location-attempt* in some case could reach accuracy rates of allmost 0,50 % the *Regio-words-attempt* results were disastrous. There was no advantage of this *Regio-words-attempt* over pure guessing.

Nevertheless we are still convinced that some of the disappointing problems of the *Regio-words-attempt* could be resolved by spending more time in carefully seeking for good candidates for the initial word list. We must accept that the language of social media differs from everdays language. This is the reason why simply adopting the results of other researchers did not fit our requirements.

Probably a mix of both *Geo-location-attempt* and *Regio-words-attempt* could be promising. This approach still was based on real geo-location data. But additionally to calculating the known word-vectors we could try to find regional words by simple comparing if the ratio of a word in one region is bigger than the ratio of this word over the whole corpus. This way we could create yet another corpus out of this data - this time based on real social-media data.

We know that the form of our regions are problematic. We divided the area with nativ german speakers to only seven regions. Maybe this is far too little for our purposes. In doing so we mixed together words and regions which better should be separated. On the other side we were confronted with the allready explained fact that Tweeting takes place only in some urban spots.

In any case there is still one big thing to do: Our attempts both did not take into account the probability of a word. So a word that appears only twice in the training corpus will be treated the same way a word with 10,000 appearances. The fact that maybe 9,000 of these appearances were found in one region is necessary to take into account. Because we did not find a way to weight this word due to lack of knowledge we are convinced that improvement of the algorithm is still possible.

Nonetheless this project is far from beeing unsuccessfull. We still believe that three crucial points appeared – all of them improvable: 1) The size of data was too small 2) The quality of our regional-word-list was not sufficient 3) Our knowledge depending specific problems has to be expanded. Often we had to handle problems which we were confronted with and we could take

a standard-algorithm out of our deposit. The deposit still holds too little solutions.

References

Arabic highest growth on Twitter (2010). Paris, France. URL: http://semiocast.com/publications/2011_11_24_Arabic_highest_growth_on_Twitter.

Brazil becomes 2nd country on Twitter, Japan 3rd (2012). Paris, France. URL: http://semiocast.com/publications/2012_01_31_Brazil_becomes_2nd_country_on_Twitter_superseds_Japan.

Jurafsky, Daniel and James H. Martin (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 2nd. Upper Saddle River, NJ, USA: Prentice Hall PTR. ISBN: 0135041961.

Lawhead., Joel (2011). *Point in Polygon 2: Walking the line*. URL: http://geospatialpython.com/2011/08/point-in-polygon-2-on-line.html.