Machine Learning tense classification in Dutch conditional sentences

Sep Keuchenius
6594190
Bachelor Kunstmatige Intelligentie, UU
Begeleider: Jos Tellings
Tweede beoordelaar: Maeghan Fowlie
7.5EC

02-07-2021

Abstract

In Time in Translation, a project currently researching the translation of tenses in conditional sentences, tenses are annotated manually to serve as data for the research. This paper researches the performance of Machine Learning algorithms to automate this classification, specifically, the classification of tenses of the antecedent part in Dutch conditionals. Four algorithms were trained using previously annotated conditionals and features that were found to be relevant to the tense, after which the algorithms were tested with some variation of parameters. MLP, K-NN and SVM performed the classification with an accuracy of around 93%, using cross-validation, while NB performed at 80%. The features that were found to be positively influential all measured the occurrences or presence of certain types of verbs in the antecedent, in some way. Consequently, the implication is made that these algorithms, requiring only a little data set and several simple features, can be used to classify the tenses for the research and that they might get better when provided with feedback.

Contents

1 Introduction

Text classification in natural language (also known as Natural Language Processing, NLP) is an area where much ground has been covered and much more is yet to be covered. Text classification is a task that, when done correctly, can help give insight to the semantic and syntactic difference between languages. This paper is concerned with the classification of tenses of sentences in Dutch. More specifically, the domain will be limited to conditional sentences.

Conditional sentences in Dutch have interesting structures, because they are made up out of two clauses, each with their own tense, that largely determine the function of the conditional. Whether it is a general truth (*if a bear is white, it's a polar bear*), or some future possibility (*it might get better, if we keep going*), the function is denoted by the use of the tense. Research in the department of tense translation is being done by a project called "Time in Translation" (Van Der Klis et al, 2017). This project analyzes the way that tenses in sentences are translated from language to language. This analysis helps us gain insight to the use of tenses and semantic difference between tenses in languages.

Currently, the project is focused on conditional sentences, mainly because the use of tenses is interesting in conditionals. Often, for example, past tense is used, when there is actually a reference to a point in the future, such as *if it were to rain tomorrow*. This incongruence between the use of tense, and the actual meaning of the sentence is unique. If it were a normal sentence (not a conditional one) it would have a past tense for a reference to the past: *it rained yesterday*.

The aim of the research of this paper is to find an efficient method of classifying Dutch tenses in conditional sentences to help speed up the study that Time in Translation is carrying out. Currently, the researchers are manually entering the tenses of the sentences is several languages to analyze that data later. The research of this paper aims to help them in automatically annotating those tenses efficiently in order to take away some of their effort.

This annotation can not be based on simple rules, considering the complexity of the sentences and the fact that language rules are often semantic, and the semantics of a sentence are hard for simple algorithms to interpret. Therefore, machine learning algorithms will be used to classify the tenses. Hence, the main question of this research:
How well can a machine learning algorithm perform in classifying the tenses of Dutch conditional sentences?

Conditional sentences are sentences that have the form *if this then that* (EN) - *als dit dan dat* (NL). They have an antecedent clause (the part after *this*, denoting the condition) and an consequent clause (the part after *that*, denoting whatever consequence the condition has), that both have a separate tense.

The data for this research will consist of sentences from the European Parliament (Europarl), where professionals write down the translation (or original sentence) of the sentence carried out in the Parliament. These sentences are structured in XML-form, along with some annotated features such as part of speech and lemma. The antecedent part of the sentence has been annotated (marked) along with its tense by the researchers of Time in Translation. The algorithms will be trained to determine the tense of the antecedent, assuming that it has been annotated which part of the sentence is the antecedent.

There has been done very little research in the area of Dutch tense classification. There has been, however, a similar research within Time in Translation. This research claimed to find the best method to automatically annotate the tenses of several languages with a decision tree (Westmijer, 2018). Note that these sentences were not conditional. In short, the decision tree was not optimized to work as well as humans could. The performance of the decision tree was significantly worse that the human performance. The author of that paper did give a suggestion as to how it might have been done better, insinuating the use of Neural Networks or other types of ML. This seems to be hopeful, given that Neural Networks are often used for Natural Language classification.

Another research concerning natural language classification (Narayanan et al, 2009) aimed to classify the sentiments of conditional sentences in English. The method used there was a Support Vector Machine (SVM), using several interesting features to distinguish the sentiment of the sentence. SVM's are known to require less data and parameter adjustment, and can perform quite well and quickly, where NN's take longer to train because they use more data. NN's are, on the other hand, much better - when used correctly - at the classification task at hand. To train the classifiers correctly, an efficient and relevant set of features must be acquired that the classifiers will use to distinguish the tenses by. These features must be quick to extract from the sentence and linguistically relevant to the types of tenses of the conditionals.

These two algorithms seem to be the best candidates for the classification task at hand, but there is still a lot to be found out about how well they can be used, and how they should be used. Therefore, the sub questions will be:

1 What are the best algorithms and their parameters for the classification task?

2 What are the most important features of the sentences in the classification task?

3 How well do the algorithms classify the tenses in comparison to humans?

In the coming chapters I will begin to analyze the way that conditional sentences are built and how they use tenses, to find out what features are best to use to train the algorithms on the input data. The results will be compared to human annotation and I

will conclude by evaluating the performance of the algorithms. Finally, the implications of this research and the further discussion will make way for further research in this area.

2 Linguistics

Dutch conditionals are a field without much background literature. Many papers about conditionals are concerned with very particular instances and occurrences of certain types of conditionals, such as conditionals that start with particular modals like "mocht" or "mits" (Boogaart, 2007; Daalder, 2009). These conditionals appear in the dataset that was used for this paper, but the domain is much bigger than that.

Conditionals in Dutch always have an antecedent and a consequent clause, that can appear in either order. To denote the start of the antecedent, sometimes words such as *als, mocht, mits, indien* are used, but they are not required. *Dan* can be used to denote the start of the consequent, but is also not required. Conditionals can form the entire sentence, which would mean that the sentence has an antecedent and a consequent and nothing else. Contrarily, they can also be embedded in a sentence, meaning that there are multiple clauses in addition to the antecedent and consequent clause, like so: *However, I agree with the previous speaker that, if paragraph 9 were challenged, this motion would clearly lose its substance.* In this example it is not hard to see that the conditional part of the sentence begins after another clause.

Dutch sentences can contain two types of clauses. The main clause (hoofdzin) and sub-ordinate clauses (bijzinnen). Main clauses contain verbs, and most of the sub-clauses do as well, while some do not (beknopte bijzin). In an antecedent or consequent could be multiple clauses that are not the main clause, and therefore do not determine its tense, while containing verbs. By merely knowing the verbs in the antecedent clause, one does not necessarily know the tense of the antecedent, because it is first required to know which verbs are relevant to the tense, and which are merely verbs of sub-clauses.

Another interesting aspect of Dutch conditionals, which does not apply to, for example, English conditionals, is that in both the antecedent and the consequent clause the word *zou* can appear. *Zou* is a modal verb, usually translated as *would* in English. The appearance of *zou* also partially determines the tense of the clause. The use of these modal verbs has been annotated in the data, so that can contribute to the classification of the tense.

Moreover, since that part of the data is already available as *modality* the tense annotation itself has been simplified as opposed to how tenses are usually denoted. The tenses for this paper will be annotated as one of the following:

OTT	Ik ben
OVT	Ik was
VTT	Ik ben geweest
VVT	Ik was geweest
VT	Ik zou zijn geweest
INF	Ik zou zijn

As mentioned above, not all of these are usually denoted as such. The last two traditionally have different labels, but combined with the *modality* of the antecedent, it can be deduced what tense is meant. VT stands for *Voltooide Tijd*, representing tenses that have a modal verb such as *zou*, and contain a *hulpwerkwoord* and a *participium*. INF stands for *Infinitief* and indicates the combination of a *hulpwerkwoord* and an *infinitief*.

English conditionals have distinct types that, in theory, denote the tenses of the antecedent and consequent and the function of the conditional (Imre, 2017). For example, a type I conditional is used for general truths, such as *if the sun shines, the sky is clear*. The tense in both the antecedent and the consequent is present simple. There are 3 known types that all have a certain function and a tense-combination (Imre, 2017). These combinations of tenses between the consequent and antecedent are beneficial, because when a certain tense appears in the consequent, it means that the antecedent tense has to be one of the tenses that the consequent tense combines with. Knowing that the tense of the consequent is present simple, for example, means that the antecedent's tense is most likely also present simple, because that combination appears often.

Unfortunately, in Dutch, conditionals don't follow this same principle. There are no widely-known types that can be distinguished. Antecedent and consequent tenses are not known to be correlated as much as in English conditionals, meaning they should not be as useful for classification tasks as in English, as will be examined later on in the study. This examination will include the use of data from the consequent, to help predict the tense of the antecedent.

Building a classifier to find the tenses of the antecedent clause of the conditional requires finding properties of the sentence that can help determine the antecedent clause tense. These properties have to be computationally available, meaning that the algorithm has access to them. This seemingly excludes the ability to distinguish features such as sentiment or semantics, because they are not so easy to classify and require much data, and the time scope for this project does not allow for such inquiries. Properties that are available and relevant are ones such as 'how many plural verbs does the antecedent contain'. Parts of speech are, as mentioned, part of the data and can be accessed easily.

These properties are to be features for the classifiers, meaning that the classifiers will use them to learn to distinguish between tenses. These features will have to be of numerical values, as they will be used to train classifiers that mostly use calculations of vectors.

The linguistic features of the conditional sentences that are relevant to the tense seem to be quite apparent. They mostly concern the verbs in the conditional sentences, mostly the ones in the antecedent, as that is the tense that we are after. If the antecedent and consequent tense are somewhat correlated after all, it would be valu-

able also to consider the verbs of the consequent tense. Mostly, the linguistic difficulty lies within distinguishing which verbs are important, and which are not. The ones that are not important are ones that don't contribute to the tense of the antecedent, mostly the ones that are in sub-clauses, therefore it seems valuable to be able to distinguish which parts of the sentence are main- and sub-clauses. However, to do this precisely it requires more information than just the tagged POS's. Sub-clauses can be introduced with comma's, conjunctions, connectives, which are not all tagged as such in the data. Humans could rather easily distinguish clauses if they knew what they were, because they understand the meaning of the sentence. With the data at hand, it seems nearly impossible to train a classifier to understand the meaning of the sentence.

There is of course the possibility to use other sources, such as corpora as NLTK, that have large amounts of data in them, and can parse sentences better and distinguish clauses of the sentence. However, unfortunately, there are not many of these parsers trained in Dutch, and considering that the data are quite specific, it seems to be more logical and interesting to train classifiers without help from big corpora.

3 Input Data

As there has not been done much research in this area, there is also not much data available. The data that is used for this paper consists of the data also used in Time in Translation. This is a relatively small dataset ($\approx$ 800) of sentences, extracted from a larger set from Europarl, structured in XML. The XML contains some linguistically relevant data per word, such as part of speech (POS), lemma (LEM) and, obviously, the word itself. The sentences were also manually annotated by researchers of Time in Translation and contain the following information:

1 Which part of the sentence is the antecedent

2 Antecedent tense

3 Conditional modality

Unfortunately, there is no data available that could help distinguish phrases or clauses in the sentences. If there were, it would be easier to determine what clause a verb belongs to. Consequently, finding the tenses will become significantly more difficult. Furthermore, the dataset is quite small for a ML algorithm to train on. Usually much larger sets are used, but these were not available. Of course, as the research of Time in Translation grows, so will their data. Considering the small amount of data, ML algorithms such as SVM or Naive Bayes (NB) should work best, as they require much less data. In addition, the research methods will be focused on enlarging the accuracy of the classification task, without deeming the possibility of perfection a realistic one.

The conditional modality of a sentence denotes what modal verbs are in the antecedent and the consequent, such as *zou* or *mocht*. The modality can, for example, be presented as *als zou, dan zou*, denoting that in both the antecedent and the consequent the modal verb *zou* is present.

Note that the output data consists of multiple possibilities (the tenses shown in the previous paragraph). These possibilities are known as classes. The classification algorithms aim to assign a class to each sentence.

4 Algorithms

Considering the available data, the most useful algorithms seem to be the ones that require few data and are easily trained. The data is labeled with the target output value, therefore a supervised algorithm can be used. The data has multiple possible output values, classes, so the algorithms do have to be able to classify to multiple classes, unlike some (such as linear regression, which is a binary classifier).

SVM. Following the article by Narayanan et al (2009), where conditional sentences are also classified, an SVM seems a reasonable candidate. An SVM is a vector-driven classifier that attempts to distinguish classes by making hyperplanes in n-dimensional spaces (where n is the amount of features). These hyperplanes are supposed to separate the classes that are, when the features are chosen correctly, divided into sections in the n-dimensional space (Jakkula, 2006). The kernel in an SVM determines the mathematical process by which the feature vectors, the list of features of a data point, are mapped into the n-dimensional space. For this classification task an *RBF* (radial basis function) kernel will be used, which maps the features in such a way that the SVM is able to distinguish the features not just by linear, but also curved hyperplanes.

K-NN. A similar approach to determining which class a vector belongs to in the space, is K-nearest neighbours, in which the algorithm places the vector in the space and calculates which k other nodes (vectors in the training data) are closest to it, to decide which class it belongs to. The biggest difference between K-NN and SVM is that SVM assumes that the distinction between the classes in the space is some boundary line, and draws a line there. K-NN makes no such assumptions and does not attempt to make any boundaries in the space. It looks at the training examples that are closest and counts which classes they belong to and finds the class that appears most often in the k examples.

Depending on the characteristics of the data and the geometric separability of the classes, K-NN could perform reasonably well (Kim et al, 2012).

NB. Naive Bayes, a probability based algorithm, will be tried as well. The algorithm is often used for simple text classification, and not expected to work as well as the others in this classification task. However, it will be interesting to see how the accuracy of this algorithm differs from the others. NB works by updating the probabilities of certain events occurring, based on the events in the training data, so that the probability of the all the possible classes, given some data point, can be determined.

MLP. Lastly an attempt should be made to train a Neural Network to classify the tenses. Neural Networks are known to require a large data set to be trained to have a deep understanding of the data and be able to very precisely predict the classes of test data. A Multi-Layered Perceptron (MLP) is an NN that takes as input a set of

features and has an x-amount of layers to process the numerical values of those features. Training involves updating weights of the edges that connect the nodes in the NN, until either some limit is reached or the weights are converging to some value (Jakkula, 2006).

5 Features

Perhaps the most important part of training a classifier is finding the right features to distinguish the output classes by. Features are attributes of the input data that can be translated to numerical values, that can be interpreted by the classifier. Depending on what type of classifier is used, the features will represent the first nodes, or a vector used to update weight values.

To find the most relevant and efficient features, it is important to consider their individual contribution to the accuracy of the classifier, but also how well they work together with the others. Two features that have a high correlation, don't both have to be used because they would both train the classifier the same patterns, simply put (Hall, 1999).

To determine the best set of features to maximize the accuracy of the algorithm, each of them needs to represent linguistically relevant data, be convertible to a numerical value and increase the accuracy of the classifier when tested together with the other features. To have a decent effect, each feature needed to at least increase the accuracy of the algorithm by 0.5%. To test this, an SVM was used with standard parameters (gamma = auto , test-size = 0.1, as will be explained in the Training & Testing section). The features below are distinguished by a positive (pos), negative (neg) or neutral (neu) effect on the accuracy, where respectively, they caused an increase $> 0.5\%$, $< -0.5\%$ or somewhere in between, in combination with the other so far selected features.

5.1 Antecedent Features

modality_contains_modal_verb (pos)
Modalities of conditionals contain information about the presence of modal verbs in the conditional. This feature extracts the binary value of whether or not there is an a modal verb in the antecedent, by looking at the annotated modality of the sentence. The presence of modal verbs limits the possibilities of tenses to a select group of classes.

modality_contains_zou (neu)
Likewise, it checks presence of "zou" in the first part of the antecedent, attempting to discover a correlation between a tense and the use of "zou", but found that it is too closely correlated to the former feature, resulting in a neutral influence. Note that it is not the same as the former feature, because it specifically focuses on the modality verb "zou", and not any of them.

n_ant_verbs (neu)
The amount of verbs in the antecedent, attempting to distinguish between sentences that seemingly have multiple sub-clauses and sentences that have only a simple main clause

n_verb_ant (pos) (4 features)
For each type of verb (present simple (pres), past simple (past), past participle (papa), infinitive (inf)) the occurrence in the antecedent was counted.

n_modal_verbs_ant (pos)
This feature represents the sum of the occurrences of certain verbs in the antecedent (*zullen*, *hebben*, *mogen*, *gaan*,

zijn). This feature was inspired by the occurrence of the modal verbs (*zullen, mogen*), but appeared to be more positively influential when expanded to include *gaan, hebben, zijn*. The verbs can appear in any form in the sentence, because they are compared to the LEM of all the words in the sentence.

verb_placement_ant (neu)
Attempting to find a relation between the placement of the verbs (beginning or end in the antecedent) and the tense, this feature represents whether or not the average index of the verbs, within the antecedent, is over halfway of the antecedent. This feature was also tried by outputting the average index as a number, instead of the binary value of whether it was over halfway. Neither attempts were positively influential.

5.2 Consequent Features

Considering the English types of conditionals, where the tense of the antecedent correlates with the tense of the consequent (Imre, 2017), some features attempting to represent data about the consequent were examined. Unfortunately none of those features had a positive effect. Several examples are demonstrated below.

n_cons_verbs (4 features) (neg)
Similar to n_ant_verbs, counting occurrences of types of verbs in the consequent

cons_modal_contains_modal_verb (neg)
Similar to modality_contains_modal_verb, but looking in the second part of the modality

cons_contains_papa (neu)
Whether or not there is a past participle among the verbs in the consequent. The past participle verb is a clear sign that the clause is of one of two tenses, narrowing the amount of possibilities for the consequent tense. This is an attempt to vary the *n_cons_verbs* which just counts the verbs (including papa), and make it binary, to make it easier for the algorithms to distinguish.

modal_cons_verbs (neu)
This feature calculates how many modal verbs are in the consequent clause of the sentence, by going through the whole sentence, checking whether a word is a verb, not in the antecedent and its lemma is one of the predefined modal verbs.

The fact that none of these features had any positive influence can be seen as evidence that the consequent is not correlated closely to the antecedent, in terms of tenses.

What must be mentioned is that the consequent clause was not annotated as the antecedent, meaning it was not clear from the data which part of the sentence was the consequent. So whenever a sentence contained more clauses than just the antecedent and consequent, the verbs from the remaining clauses were also counted as consequent verbs. Therefore the features might have been less precise than the antecedent features.

As mentioned in the Linguistics section, distinguishing which verbs belong to which clauses in the conditional sentence should be of great help to

the classification task. By determining which verbs are in the main part of the antecedent, it would be much easier to determine the tense of the antecedent. However, any attempts to include features that tried to distinguish the clauses resulted in a neutral or negative influence on the accuracy. Among these features were, for example, ones that counted only verbs that seemed to be in the main clause of the antecedent, by looking at the comma's and connectives of the sentence.

All features were, just as the data, processed in Python 3 using packages *Pandas* en *Numpy*.

6 Training & Testing

Extracting the features from the data requires an algorithmic approach to processing the data. In Python 3, some small methods were built to do this, and the features were placed in a pandas data frame (McKinney, 2011) as either numerical, categorical or binary values. These values were then processed by the *OneHotEncoder* algorithm to translate the binary and categorical values to numerical ones. These values were then split by a module from *sci_kit_learn* into *training* and *test* data, randomly, and using a value between 0 and 1, *test_size*, to determine the size of the partition that would be the test data.

Training data, when developing a classifier, is used to train the algorithm to classify the data, while the test data, the residual data, is used to see how well the algorithm performs. When putting the classifier to real practical use, the algorithm will be as well trained as it can be. Considering the size of the data set, the partition of the training set should be as large as possible to be able to train the algorithms well.

Unfortunately, making the training data partition large, would mean that the test set is very small, and the results of testing the algorithms would be variable, and highly dependent of which vectors were chosen as the test data. To balance this, all algorithms were tested by using varying partitions of test data, but were tested multiple times with different sets to ensure an accurate result. All results of the accuracy of the classifiers were stored in a list and the averages, as well as standard deviations were calculated to determine the exact accuracy of the algorithm. All algorithms were also tested using cross-validation, where the data is partitioned into n sets, after which all sets are used once for testing and $n-1$ times as training data.

Different sets of features should be beneficial to different algorithms. For example, using many features with many different categorical values can be exhausting for an algorithm such as SVM that classifies based hyper-planes or other mathematical-visual distinctions. For NN's however, these features can be very helpful, considering that the amount of nodes (neurons) can be incredibly high, and that the weights that determine the importance of a node can be adjusted to make the accuracy of the classification as high as possible.

Considering that the features have already been chosen, and that it would take too much time and much more data to produce a set of features that is more suitable to an NN, the NN will be tested with the same features as the other algorithms. The NN might benefit from more complicated features with more possible values, but that will be left to other studies. The goal of this paper is not to build an algorithm that understands the sentence, but rather to extract basic features of the sentence that can optimize the accuracy of the classification.

6.1 Varying parameters

Testing the algorithms requires some parameter settings that can alter the

outcome of the classification. Below is a list of parameters that will be varied during the testing.

Test-size. As mentioned, the partition of the data that is used for the testing of the algorithms can vary in size. Varying it can give insight to how much data is necessary to train the algorithm and how precise the algorithm is at that point. The test-size does need to remain big enough to be able to form a reliable accuracy result.

MLP: Max Iterations. In an NN, each time that a data point trains the network, the error of the outcome is measured and used to update the weights of the edges connection the neurons. These updates of the weights tend to converge to an optimal setting of the weights, however, they do not always do that (Jakkula, 2006). To ensure that the NN stops iterating, the max-iterations is set. This can also help find out when the NN starts to converge and stops to produce better results.

MLP: Hidden Layers. Apart from the input layer and the output layer, there are neurons in the network that are not defined as feature neurons, meaning they don't represent a specific feature. They are rather a point to which numbers from other nodes are mapped and summed up. These nodes are in layers that are called 'hidden layers'. The amount of hidden layers that a network has can influence how well the network can cope with the complexity of the data and provide a reasonable output and how long the algorithm takes to train (Panchal et al, 2011).

K-NN: k. In K-Nearest Neighbors the classifier looks for the k closest examples in the training set to determine which group the test data point belongs to. The more neighbors the algorithm has to check, the longer it takes for it to classify, but speed generally is not an issue for K-NN, it is the simplest of these algorithms. Following the article by Kim et al (2012), k will be tested around $k = 5$.

SVM: Gamma. Using the RBF kernel in the SVM means that the hyperplanes will be curved when segregating the classes in the vector space. The degree of the curves can be determined by the parameter Gamma, which will be varied.

SVM: C. C is another SVM parameter that influences the hyperplanes of the SVM. C is a regularization parameter that influences how much error can be permitted when training the algorithm. Usually allowing a large C means that much training error is permitted, but this can result in a greater accuracy when testing the algorithm.

7 Results

The performance of the algorithms was quite surprising. The best classifiers turned out to be the SVM and the MLP, which both scored around 93% accuracy at best. In table 1 the results can be found for the best mean score of the algorithms, tested using cross validation with 21 folds.

Alg	Mean	SD
SVM	92.9%	3.0%
MLP	93.1%	3.8%
K-NN	91.3%	3.3%
NB	80.0%	5.83%

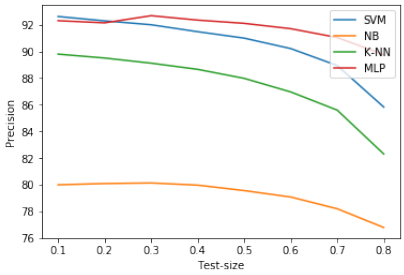
Table 1: Optimal algorithm results

These results were measured when testing the algorithms (with their optimal settings, as will be demonstrated) with cross-validation over 21 groups, using the features that were found to have a positive influence in the accuracy of the standard SVM algorithm. These features are:

1. modality_contains_x
2. first_ant_verb_pos
3. n_past_ant
4. n_pres_ant
5. n_papa_ant
6. n_modal_ant
7. n_inf_ant

The algorithm performing worst is the Naive Bayes algorithm, by a large margin. The standard deviation of the Naive Bayes is also visibly higher, meaning that it varies a lot, between iterations, how well the algorithm classifies, insinuating that it depends highly on what type of test data it is tested on.

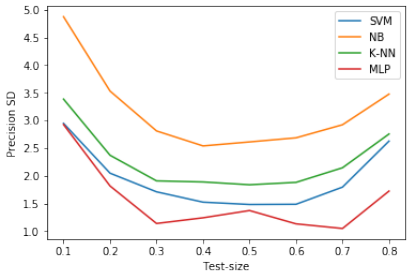
Test size: Mean. The results from the table are from cross-validation of 21 groups, meaning that the algorithms trained 21 times and tested 21 times on 21 groups of data in the data set. That means that the test size of the data is quite small, and the training set quite big, leading to the most precise classifications. This choice seems to be fair, considering the small data set that is used. When changing the test-size, the accuracy decreases in all algorithms, as to be seen below, where cross-validation was not used, but rather a test and training set randomly generated for a 1000 iterations, to get the mean accuracy.



Interestingly, the decrease is not so large, until the test size starts to be less than half of the total data set.

Test size: SD. To determine the degree that the algorithms are fitting the data, the same circumstances were tested, but this time the standard devi-

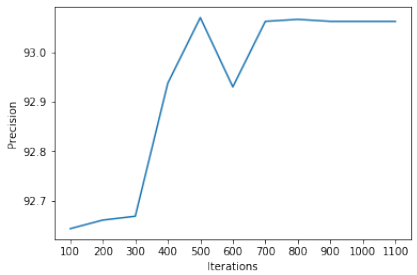
ations of the accuracy between all the iterations were inspected.



A low SD points to little variation in the accuracy of the classifications. This means that the algorithm is consistent in its classification and does not depend highly on which data is used to test it. SVM and MLP clearly have the lowest SD, insinuating that they are the best fitters for the data.

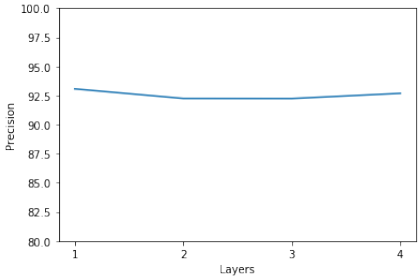
All the following tests were performed using cross-validation with 21 rounds.

MLP: Max. It. The MLP is surprisingly well at classifying the tenses, even though the data is not very elaborate. To maximize the accuracy of the MLP, two parameters were examined. The first parameter that was varied, was the maximum amount of iterations.



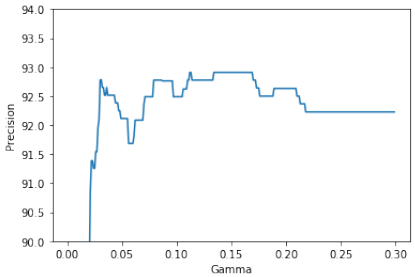
Clearly, after 800 iterations, the algorithms converges to weights in the network, meaning that the weight do not update anymore. This is also when the algorithm performs best, at 93.1%

MLP: Hidden Layers. The other parameter that is important to the algorithm is the amount of hidden layers. All hidden layers have 100 nodes. Following the former result, the MLP has been set to iterate 800 times to ensure convergence.



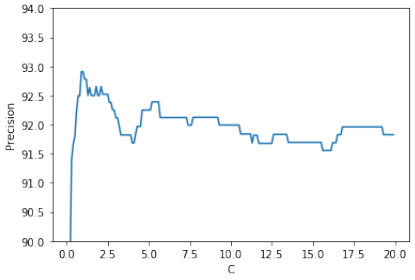
The amount of layers does not influence the accuracy, as long as it is higher than 0. It can be concluded that most likely the values of the features in the data are not so diverse that they can not be distinguished easily with one hidden layer.

SVM: Gamma. Varying gamma from 0.001 to 0.299



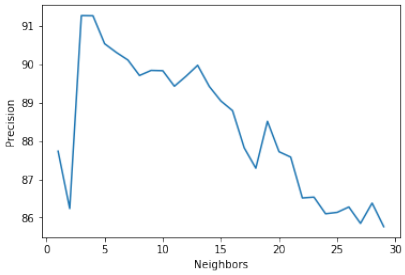
A clear correlation between gamma and the accuracy can not be spotted, but around 0.15, the accuracy is highest.

SVM: C Using the former result (gamma = 0.15 at the highest accuracy), the parameter C was varied from 0.1 to 30



The accuracy was highest at $C = 0.9$, concluding that SVM has its highest score at 92.9% with gamma = 0.15 and $C = 0.9$

K-NN: k Varying k from 1 to 29



K-NN is best when $k = 3$ and $k = 4$, at 91.3% accuracy. When k gets larger, the accuracy quickly decreases.

Having established the parameters for the highest accuracy of all the algorithms, and their corresponding accuracies, the difference can be calculated to find out whether there was a significant difference between the algorithms.

Performing paired t-tests on the scores of the algorithms using 21-fold cross-validation returns that none of the top-3 algorithms is significantly better than the others, because of the high variance between the validations. Even though it's visible that the MLP performs almost 2% better than the K-NN, it can not be concluded, after 21 validations with around 3% SD, that the MLP is significantly better than the K-NN, and the same goes for the SVM - MLP and SVM - K-NN combinations ($p > 0.05$ in all cases). Only when comparing to the NB algorithm, we can conclude that all three are significantly better than NB ($p < 0.005$ in all cases).

8 Conclusion

In this research several different feature sets and algorithms were tested to find out how well machine learning algorithms can perform in classifying tenses in Dutch conditional sentences. From the start it has been a challenging task, considering the size of the data set and the amount of annotated features that came with the data set. It has become clear that none of the machine learning algorithms, with the features that were deemed to perform best, could classify the tenses as well as any human at the Time in Translation project can. Below, all sub questions of this thesis are answered and concluded.

8.1 Features

Feature selection in text classification is closely related to the linguistic behavior of the data. The fact that none of the consequent features were successful leads to believe that the consequent tense does not determine, influence, or is even related to the antecedent tense. However, as mentioned, these features might not have been as precise as the others, due to a lack of annotation.

Also interesting was the result of the features that attempted to distinguish the clauses in the sentences, of which none were successful. The attempt was optimistic and may simply require much more data and perhaps pre-trained algorithms. However, if it had worked, it would have been incredibly valuable.

The features that turned out to be successful were none other than obvious. The occurrences of the types of verbs in the antecedents are closely related to its tense, logically. Counting the modal verbs and considering the modality of the antecedent also positively influenced the accuracy, leading to believe that they are highly correlated to the tense of the antecedent.

8.2 Algorithms

As it has been determined that the algorithms do not have a significant difference in their performance it can not be concluded that one is better than the others. However, it can be deduced that the differences in performances is of some anecdotal value, even if only to determine where to start in a next inquiry, perhaps with more data.

Having resolved that the top three algorithms performed significantly better than the last, it can be concluded that those three at least outperform any 'dumb' algorithm, and might perform even better, provided with more data.

MLP The parameters of the algorithms that performed best helped to gain accuracy and made, in some cases, a significant impact. The MLP performed best when most weights converged to a value, using 1 hidden layer. Logically, this means that the data was not that complex and the classes could be segregated rather easily. However, considering that the features do not directly determine the class (two sentences can have the same features, but not the same class), adding any hidden

layers could never result in a higher performance. To get the performance higher, the MLP would need to understand the data on a deeper level.

SVM The parameters in the SVM ended up only slightly changing the accuracy of the algorithm. In this case gamma $= 0.15$ made the algorithm perform best, but it was a close call. It does not have to be the case, when providing new data, that this will still be the best value for gamma. Looking at the parameter C, however, we can see that there is a slight decrease in the accuracy with a higher C value. A clear peak in the graph is spotted at $C = 0.9$, which is a relatively low C, meaning that not many errors in the training were permitted and the SVM hyperplanes were curved quite closely around the training data, which leads to believe that the SVM might have been overfitting.

K-NN The only parameter that was tested was the amount of neighbors considered when evaluating which class a test vector belongs to, k. The highest performance appeared when $k = 3, 4$, insinuating that the algorithm looked rather close by, when evaluating. This points to the same principle spotted with C in the SVM. The algorithms look quite close to the test vector when predicting a class.

8.3 Performance

All in all, the best feature, algorithm and parameter combinations performed at around 93% accuracy with a large, namely 3-4% SD. A rather high accuracy considering the data, but a high SD, insinuating some overfitting.

9 Discussion

This percentage alone may not suffice in answering the main question of this thesis, unfortunately. Machine learning *could* perform better when classifying tenses in Dutch conditional sentences, but considering the data at hand, it performed the way it did.

Implications. The fact that the algorithms mostly scored above 90% accuracy does indicate that classification can start to be used as a real help for the research that Time in Translation is doing. The sentences that the researchers annotate can be annotated much quicker than they were. While doing so, the researches could give feedback to the algorithm, supplying it with more data to train from, most likely increasing its accuracy.

Moreover, any research in the field of classification of Dutch sentences could draw information from this paper when choosing features, algorithms or their parameters to help with their research.

Limitations. The reliability of this research may be slightly limited due to the fact that the tenses were annotated by humans, and can still contain minor flaws. In addition, the tenses that were used for the classification do not exhaust all the existing tenses in Dutch language. Adding the missing tenses to the research could result in different results for what features, parameters or algorithms are considered to be optimal.

Future possibilities. Using bigger corpora, when available for Dutch conditional sentences, should result in a much higher accuracy, if a classifier is trained to classify using much more than just occurrences of verbs in the antecedent. It might be able to segregate the clauses and determine which verbs are of bigger importance to the tense than others. Moreover, it could perhaps look at embedding of the POS's in the antecedent, to determine after or before which POS a verb is places, or at what position, to find correlations between those features and the tense.

10 References

1. Van Der Klis, M., Le Bruyn, B., De Swart, H. (2017). Mapping the perfect via translation mining. In 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 (Vol. 2, pp. 497-502). Association for Computational Linguistics (ACL).

2. Westmeijer, B. M. (2018). Automatische toekenning werkwoordstijden (Bachelor's thesis). University of Utrecht, 2018.

3. Narayanan, R., Liu, B., Choudhary, A. (2009, August). Sentiment analysis of conditional sentences. In Proceedings of the 2009 conference on empirical methods in natural language processing (pp. 180-189).

4. Jakkula, V. (2006). Tutorial on support vector machine (svm). School of EECS, Washington State University, 37.

5. Boogaart, R. (2007). Conditionele constructies met moest (en) en mocht (en) in Belgisch-Nederlands en Nederlands-Nederlands. Neerlandistiek, 2007.

6. Daalder, S. (2009). Conditional constructions: The special case of modern Dutch mits. Journal of Germanic Linguistics, 21(2), 231-248.

7. Imre, A. (2017). A Logical Approach to English Conditional Sentences. Journal of Romanian Literary Studies, (12), 156-168.

8. Kim, J. I. N. H. O., Kim, B. S., Savarese, S. (2012). Comparing image classification methods: K-nearest-neighbor and support-vector-machines. In Proceedings of the 6th WSEAS international conference on Computer Engineering and Applications, and Proceedings of the 2012 American conference on Applied Mathematics (Vol. 1001, pp. 48109-2122).

9. McKinney, W. (2011). pandas: a foundational Python library for data analysis and statistics. Python for High Performance and Scientific Computing, 14(9), 1-9.

10. Panchal, G., Ganatra, A., Kosta, Y. P., Panchal, D. (2011). Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. International Journal of Computer Theory and Engineering, 3(2), 332-337.