UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Hele-Andra Kuulmets

Phrase similarity measures based on Word Mover's Distance

Masters's Thesis (30 ECTS)

Supervisor: Sven Laur

Tartu 2019

Phrase similarity measures based on Word Mover's Distance

Abstract:

Measuring semantic similarity between texts is necessary for successfully solving natural language document classification tasks. However, not always base the problems that can potentially be solved using semantic similarity on texts with the length of multiple sentences. Sometimes the decision has to be made only seeing a single sentence or a phrase from that sentence.

In this thesis, Word Mover's Distance (WMD), which essentially is a document similarity measure, is applied to three different problems where only short phrases are given. The first problem, predicting omitted word by the given context, is a made up problem and the goal is to assess the goodness of the measure and its suitability for such tasks. The results are good and show that it is possible to do some semantic separation of phrases using WMD.

Other two problems are examples of practical cases. Firstly, the method is used to detect adverse drug reactions from the patients' epicrises. Secondly, the method is applied to the analysis of syntax parser errors. The goal is to predict phrases that parser fails to tag correctly. For different reasons, which are also analyzed on this thesis, the results were not good for neither of the problem.

Keywords: natural language processing, semantic similarity, Word Mover's Distance

CERCS: P170 Computer science, numerical analysis, systems, control

Word Mover's Distance algoritmil põhinevad fraasisarnasusmõõdud

Lühikokkuvõte:

Loomuliku keele tekstide vahelise semantilise sarnasuse mõõtmisel on oluline osa tekstide klassifitseerimisülesannete lahendamisel. Samas probleemid, mida saaks potentsiaalselt lahendada kasutades semantilise sarnasuse mõõtmist, ei põhine alati pikkadel, mitmetest lausetest koosnevatel tekstidel. Mõnikord tuleb märgendamisotsus teha ainult ühe lause või fraasi põhjal.

Käesolevas töös kasutatakse tekstide sarnasuse mõõtmise meetodit Word Mover's Distance (WMD) kolme erineva probleemi lahendamisel, kus otsus tuleb teha lühikeste fraaside põhjal. Esimene probleem, milleks on puuduva sõna konteksti põhjal ennustamine, on välja mõeldud probleem, mille eesmärk on hinnata meetodi headust ja sobivust lühikestele fraasidele. Saadud tulemused on head ja näitavad, et WMD võiks sobida fraaside eraldamiseks semantilise sarnasuse põhjal.

Ülejäänud kaks probleemi ilmestavad meetodi praktilisi kasutusvõimalusi. Esimesel juhul kasutatakse seda patsientide epikriisidest ravimi kõrvalmõjude tuvastamiseks. Teisel juhul rakendatakse meetodit süntaksiparseri vigade analüüsiks. Viimasel juhul on eesmärgiks ennustada fraase, mida parser ei oska õigesti märgendada. Erinevatel põhjustel, mida samuti on käesolevas töös analüüsitud, ei olnud saadud tulemused kummalgi juhul head.

Võtmesõnad: loomuliku keele töötlus, semantiline sarnasus, Word Mover's Distance algoritm

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Contents

1 Introduction

Having a good semantic similarity measure between texts is crucial for successfully solving natural language document classification and clustering tasks. However, not always base the problems that could potentially be solved with measuring semantic similarity on texts with the length of multiple sentences. Sometimes the decision has to be made only seeing a single sentence or only a phrase from that sentence.

In this work we are searching for a similarity measure that would perform well on solving different phrase-based problems. More specifically, we are going to measure and apply semantic similarity for solving following problems:

1. separating words that are used in a similar context only by seeing the context;
2. detecting drug adverse effects from the patient epicrises;
3. analyzing errors made by syntax parser.

The first problem in the list should give us an understanding about how well the chosen methods work in nearly ideal conditions where the data is well-structured and contains fewer errors. The results should also give an intuition about what to expect from later experiments and also something to compare those results with.

In the second problem, we use contexts of the words that could potentially be drug adverse reactions to decide whether that word was related to some adverse effect or not.

The third problem is somewhat different from the previous two. Here we are doing error analysis of the syntax parser that does not always produce correct syntax tags. We are mostly interested in whether it is possible to separate correctly tagged phrases from incorrectly tagged phrases by the semantic meaning. We are trying to predict an output of the parser as well (whether the phrase will be tagged correctly or not).

All these three problems can be solved using the same methodology which consists of three steps:

1. defining and extracting phrases;
2. labelling some of the phrases;
3. extending the labelling using semantic similarity between phrases.

Having formulated the methodology for solving our problems, some questions still remain unanswered. Firstly, we need to give a formal definition to the semantic similarity that we are going to use in our work. Secondly, the goodness of that similarity measure must be evaluated somehow and the methods for that must be chosen. Finally, we need to decide how to get from the simple semantic similarity to the labelling extension decision. All these concerns will be discussed in this work as well.

2 Background

This section gives an overview of methods that our work bases on, including Word2Vec word embedding, Word Mover's Distance similarity measure and Support Vector Machine classification algorithm. It also provides an description of cluster analysis methods that are mainly used in our work to evaluate the goodness of similarity metric. Lastly, two dimensionality reductions algorithms are describes that we will be using for visualization purposes.

2.1 Word2Vec

Since many machine learning algorithms are incapable of processing natural language in the form of plain text, it must to be converted into more suitable form. One common approach is to map words to vectors of real numbers. These vectors are called word embeddings and there exist many different techniques for producing such vectors, for example Latent Semantic Indexing and Latent Dirichlet Allocation [DDF$^+$90, BNJ03] which are originating from topic modeling, and different neural models, for instance the one proposed by Bengio *et al*. [BDVJ03].

One popular method is Word2Vec that was described by Mikolov *et al.* [MCCD13]. The method became popular because it had much lower computational complexity than competitive neural models [BDVJ03, MKB$^+$10]. Their experiments also showed that Word2Vec models had as good accuracy of answering different types of semantic and syntactic questions as more complex models.

In their paper, Mikolov *et al.* proposed two model architectures for computing continuous vector representations of words [MCCD13]. Both models are similar to a shallow neural network with one hidden layer but the non-linearity from the hidden layer is removed, resulting in a log-linear model. The first architecture, continuous bag-of words model, is trained to predict a probability of a word given a context. Second architecture, continuous skip-gram model, is very similar to the first one but instead of predicting current word, it predicts the context given a work.

2.2 Word Mover's Distance

Word Mover's Distance (WMD) is a distance function between two lists of words such as text documents. It utilizes the property of word embeddings that distances between embedded word vectors are to some degree semantically meaningful. The distance between two documents is measured as the minimum cumulative distance that all words in one document need to travel to exactly match the other document. The metric was introduced by Kusner *et al.* [KSKW15].

Let $\mathbf{d}$ and $\mathbf{d}'$ be normalized bag-of-word (nBOW) vectors of two text documents. More precisely, if word i appears c_i times in the document, then $d_i = \frac{c_i}{\sum_{j=1}^{n} c_j}$. The

distance between $\mathbf{d}$ and $\mathbf{d}'$ is obtained by solving following linear program [KSKW15]:

$$\begin{aligned} \underset{\mathbf{T}\geq 0}{\text{minimize}} \quad & \sum_{i,j=1}^{n} \mathbf{T}_{ij} c(i,j) \\ \text{subject to} \quad & \sum_{j=1}^{n} \mathbf{T}_{ij} = d_i \;\; \forall i \in 1,...,n \\ & \sum_{i=1}^{n} \mathbf{T}_{ij} = d'_i \;\; \forall j \in 1,...,n \end{aligned} \tag{1}$$

In Equation 1, $\mathbf{T} \in \mathbb{R}^{n \times n}$ is a matrix where $\mathbf{T}_{ij} \geq 0$ denotes how much of word i in $\mathbf{d}$ travels to word j in $\mathbf{d}'$ and $c(i,j)$ is a cost associated with traveling from one word to another. Cost is defined as Euclidean distance between two word vectors in vector space, that is, $c(i,j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2$. The two constraints in Equation 1 ensure that the entire outgoing flow from word i equals its nBOW representation d_i, that is, $\sum_j \mathbf{T}_{ij} = d_i$ and incoming flow to word j must match d'_j, that is, $\sum_i \mathbf{T}_{ij} = d'_j$.

Kusner *et al.* also showed that WMD leads to unprecedented low k-nearest neighbour document classification error rates when compared to other state-of-the-art metrics [KSKW15]. These results are interesting in our context as well since we are looking for a metric that would represent semantic similarity between phrases.

The optimization problem that WMD aims to solve is a special case of the Earth Mover's Distance metric (EMD). The linear optimization problem underlying EMD is a well-studied transportation problem for which specialized solvers have been developed. However, the best average time complexity scales $O(p^3 \log p)$, where *p* is the number of unique words in the documents, thus prohibiting it's usage for large datasets, but this concern is not relevant for our problems since we are working with short phrases.

Word Mover's Distance is entirely unsupervised and not specific to any of the tasks it can be used for. Yet the measure of similarity could be very different for different tasks, for example, classifying news articles by sentiment or by topic. To address this problem, Huang *et al.* [HGS+16] proposed an algorithm that incorporates supervision into the WMD. Supervised Word Mover's Distance (S-WMD) is a metric learning algorithm that aims to improve the distance so that the documents that share the same label are close and those with different labels are far apart. The algorithm learns linear transformations of word embeddings together with word-specific importance weights by minimizing leave-one-out classification error rate under the WMD metric. The results of their experiments showed the superiority of their method across 26 baseline methods.

In this work, we will not implement S-WMD but the method itself is still relevant in our context. As we focus on short phrases, each word has a rather large influence on distance calculation and S-WMD could possibly provide adjustment.

2.3 Dimensionality Reduction

Dimensionality reduction is the process of reducing the dimensions of the feature set. It has two main goals. First is to prevent machine learning algorithms to overfit on the training data and second is to visualize high-dimensional data. In our work we use dimensionality reduction algorithms for visualization purpose.

2.3.1 Principal Component Analysis

Principal component analysis (PCA) is one of the most broadly used dimensionality reduction algorithm. Its common applications are dimensionality reduction, feature extraction and data visualization [Bis06].

The algorithm seeks linear combinations of variables such that the maximum variance is extracted from the variables. These uncorrelated linear combinations weighted by their contribution to explaining the variance in a particular orthogonal dimension are called principal components (PCs).

The first PC is chosen to minimize the total distance between the points and their projection onto the PC, or in other words, to maximize the sum of the squared distances from the projected points to the origin. The second and subsequent PCs are selected similarly but with the requirement that they must be orthogonal with all previous PCs.

In our work we would like to use PCA to perform qualitative evaluation of our distance measure but the problem is that we do not know the coordinates of our data points but only distances between them. This problem can be solved by using kernel-PCA [SSM99] with the kernel that requires only distances between instances in the input space, for example, RBF-kernel.

Kernel-PCA is an extension of PCA that maps the data into a higher dimensional feature space, which is nonlinearly related to the input space, and performs PCA in that space. It is also useful in cases where data is not linearly separable in input space as in higher dimensional space it will become linearly separable. However, this is not so relevant in our case.

2.3.2 Multidimensional Scaling

Multidimensional scaling (MDS) is a non-linear dimensionality reduction method that, given a table of distances between datapoints, maps these datapoints into (preferably low) p-dimensional space by preserving these distances as well as possible.

There are two types of multidimensional scaling: metric MDS and non-metric MDS. Metric MDS seeks to find an optimal representation of the data by directly comparing distances between the points in (low) p-dimensional space with the original dissimilarity. The goodness-of-fit is measured by a stress [BG05]:

$$\sigma_r(\mathbf{X}) = \sum_{i<j}(d_{ij} - \delta_{ij})^2 \ , \tag{2}$$

where $\mathbf{X}$ is a set of coordinates, d_{ij} is the distance between points $\mathbf{x}_i$ and $\mathbf{x}_j$ in *p*-dimensional space and δ_{ij} is the original dissimilarity. A popular method for minimizing stress is SMACOF algorithm, which was introduced by de Leeuw [Lee77].

Non-metric MDS, on the other hand, aims to find a configuration such that the new distances are in the same rank order as they were originally [BG05]. This method can be used in cases where dissimilarities between the objects are not known but their order is known.

In our work we are only using metric MDS to visualize distances between phrases.

2.4 Spectral Clustering

The decision to use spectral clustering bases on the work of Roosalu [Roo17], where it was concluded that this algorithm is best for clustering textual data.

Given affinity matrix $A = (a_{ij})$, the algorithm defines $D = (d_{ij})$ to be the diagonal matrix whose $d_{i,i} = a_{i1} + \cdots + a_{in}$ element in the sum of A's *i*-th row, and constructs normalized Laplacian matrix:

$$L = D^{-1/2}AD^{-1/2} \tag{3}$$

It then finds *k* largest eigenvectors of L and uses them to represent points in *k*-dimensional space where they are clustered using k-means clustering [NJW02].

There are several methods to transform given distance graph to affinity graph. One such method is ϵ-neighbourhood graph, where only points whose pairwise distance is smaller than ϵ are connected but without using weights. Alternatively, *k*-nearest neighbour graph can be used where a vertex has undirected weighted connection only with its *k*-nearest neighbours. Third option is to use the fully connected graph where all points are connected with positive similarity. It is common to use RBF kernel as a similarity function. In our work, however, we transformed distances to similarities using following equation after normalizing distances:

$$similarity = 1 - distance \tag{4}$$

During the analysis of clustering results we did some clustering experiments using RBF kernel as similarity function as well, to see whether it would improve the results, but it did not.

2.4.1 Methods for measuring quality of clustering

Purity and coverage. To compute purity, each cluster is assigned to the class which is most frequent in the cluster and then the accuracy is measured by counting the number of correctly assigned documents and dividing by total number of documents. High purity is easy to achieve when the number of clusters is large and thus cannot be used for choosing the number of clusters [MRS08].

Large number of clusters can be penalized with coverage which is the ratio of intra-cluster edges to the total number of clusters. As opposed to purity, coverage becomes 0 when each datapoint is assigned to its own cluster.

ROC curve and AUC. In our work the goal of clustering is to cluster the contexts in such a way that different semantic contexts are assigned to different clusters. That is, each cluster represents one semantic context. The question that we then would like to answer is that to what level are these groups of contexts word-specific.

We can measure this by calculating the ratio of positive instances for each cluster. We can then define some threshold and assign positive label to all instances that belong to a cluster above the treshold. Next, similarly to a binary classification task, we can count true positives and false negatives and visualize the tradeoff with ROC curve.

ROC curve is a common method for visualizing the performance of binary classifier with different thresholds. To do that, we need to rank our phrases by some score where higher score means higher evidence of positive instance, as described by Flach [Fla12]. We can use the ratio of positive instances in a cluster as a score and assign it to each of its phrase.

The ranking of phrases can be visualized with coverage plot by plotting positives and negatives to vertical and horizontal axes, respectively, on decreasing order. With this plot we can visualize the tradeoff of choosing different scores as thresholds by marking the results on the plot. These results form a curve called a coverage curve. After normalizing the axes of the plot to [0,1] we obtain a ROC plot and coverage curve becomes ROC curve.

The area under the ROC curve (AUC) is the ranking accuracy that can be seen as an estimate of the probability that an arbitrary positive-negative pair is ranked correctly Flach [Fla12].

Silhouette. Another way of detecting the quality of the clustering is using silhouettes. Silhouette score measures how similar is an instance to its own cluster compared to its neighbouring cluster. Silhouette of instance $\mathbf{x}_i$ is defined as following:

$$s(\mathbf{x}_i) = \frac{b(\mathbf{x}_i) - a(\mathbf{x}_i)}{\max(a(\mathbf{x}_i), b(\mathbf{x}_i))} \tag{5}$$

In the equation above, $b(\mathbf{x}_i)$ is the average distance of $\mathbf{x}_i$ to the data points in its neighbouring cluster and $a(\mathbf{x}_i)$ is the average distance to the points in its own cluster. The difference is normalized to obtain a number between -1 and 1.

Since it is also possible that $a(\mathbf{x}_i) > b(\mathbf{x}_i)$, that is, the average distance to neighbouring cluster is smaller than to own cluster, we take maximum value from $a(\mathbf{x}_i)$ and $b(\mathbf{x}_i)$ to get a normalized value [Fla12].

Silhouette scores can be visualized by sorting the scores and grouping by cluster. The method can also be used for determining the number of clusters in a data set.

Silhouette score for the entire dataset is obtained by taking the mean silhouette over all instances.

2.5 Classification

In this section we are going to describe the tools that we will use for solving classification tasks that our three problems essentially are. We will treat all our problems as binary classification tasks, where probabilities are not interesting to us. We will also describe how we are going to apply classification algorithms to distances instead of feature vectors as is usually required.

2.5.1 Support Vector Machine

Support Vector Machine (SVM) is a linear classification method that, if the classes are linearly separable, finds a separating model that maximises the margin - distance between the decision boundary and the closest instances. Following description of the algorithm is referred from Flach and Manning *et al.* [Fla12, MRS08].

In SVM the scoring classifier is defined as $\hat{s}(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - t$, which predicts positive class if $\hat{s}(\mathbf{x}_i) > 0$ and negative class otherwise. The margin *m* of an example is $c(x)\hat{s}(x)$ where $c(x)$ is +1 for positive examples and -1 for negative examples.

Therefore a true positive $\mathbf{x}_i$ has margin $\mathbf{w} \cdot \mathbf{x}_i - t > 0$ and a true negative $\mathbf{x}_j$ has margin $-(\mathbf{w} \cdot \mathbf{x}_j - t) > 0$.

For convenience it is assumed that the margin *m* of each example is at least 1. Examples with margin 1 are then the closest to the decision boundary and are called support vectors. There must exist at least one such example for each class. The margin of the classifier or, in other words, the width of separation between support vectors of two classes in such case is $\frac{2}{\|\mathbf{w}\|}$ which SVM aims to maximize. This is the same as minimizing $\frac{1}{2}\|\mathbf{w}\|$. The optimization problem has the following form:

$$\begin{aligned} \underset{\mathbf{w},t}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w} \cdot \mathbf{x}_i - t) \geq 1, 1 \leq i \leq n \end{aligned} \tag{6}$$

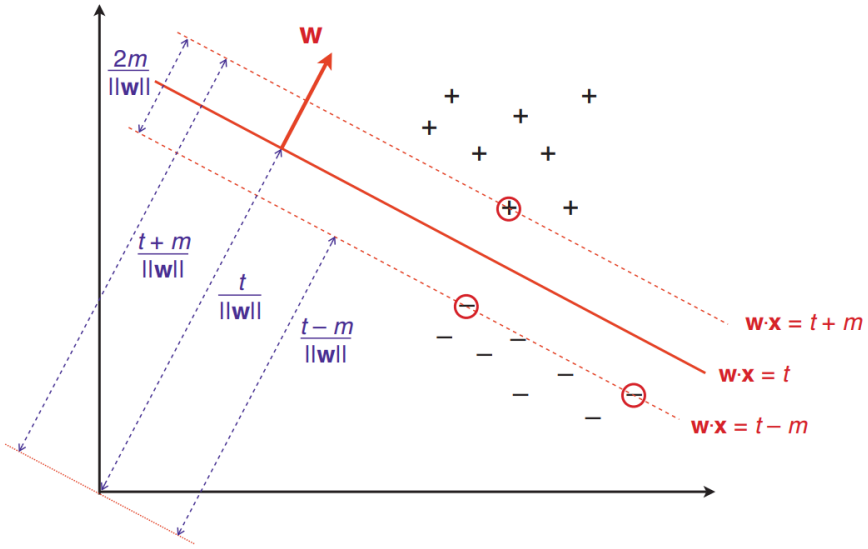


Figure 1. The geometry of a SVM classifier [Fla12]

This optimization problem is solved with the method of Lagrange multipliers and usually involves constructing and solving its dual form:

$$\begin{aligned} \underset{\alpha_1 \ldots \alpha_n}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j + \sum_{i=1}^{n}\alpha_i \\ \text{subject to} \quad & \alpha_i \geq 0, 1 \leq i \leq n \text{ and } \sum_{i=1}^{n}\alpha_i y_i = 0 \end{aligned} \tag{7}$$

2.5.2 Soft margin SVM

An extension to the SVM model, Soft margin SVM, can be used if the training data is not linearly separable. This allows decision margin to make a few mistakes, such as letting some points to be inside or on the wrong side of the margin. The cost of mistake is measured with a slack variable ξ_i and the optimization problem is changed as following:

$$\begin{aligned} \underset{\mathbf{w},t,\xi_i}{\text{minimize}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_i \xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}\cdot\mathbf{x}_i - t) \geq 1 - \xi_i, \text{ and } \xi_i \geq 0, 1 \leq i \leq n \end{aligned} \tag{8}$$

Here C is user defined parameter that trades off margin maximization against slack variable minimization. Higher value of C means a smaller margin will be accepted if it results in better training accuracy. A lower C allows more margin errors in order to achieve larger margin and thus a simpler decision function [Fla12].

In practice, proper C is chosen with parameter tuning.

2.5.3 Nonlinear SVM

While soft margin SVM allows some errors on the data that would otherwise be linearly separable, nonlinear SVM can be used on the data set that would not be possible to classify using linear classifier even with allowing errors. Nonlinear SVM maps the data onto a higher dimensional space where it becomes linearly separable and applies the linear classifier in that space.

As show in Equation 7, the SVM classifier relies on pairwise dot products $\mathbf{x}_i \cdot \mathbf{x}_j$ between training instances. If every datapoint is mapped into a high-dimensional space via some transformation $\Phi : \mathbf{x} \rightarrow \phi(\mathbf{x})$, the dot product becomes $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$. However, as computation of $\phi(\mathbf{x}_i)$ for each instance is not needed in order to train and classify datapoints, kernel functions are used to direcly calculate dot products in high dimensional space. A kernel function is some function κ that computes the dot product in higher dimensional space in terms of the original data [MRS08]:

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j) \tag{9}$$

The value of function in Equation 9 is then used in Equation 7 instead of $\mathbf{x}_i \cdot \mathbf{x}_j$.

With kernel functions it also becomes possible to use infinite-dimensional feature space as we never have to compute mapping $\phi(\mathbf{x}_i)$ explicitly.

2.5.4 SVM with RBF kernel

Radial basis function (RBF) kernel is given by

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \ , \tag{10}$$

where γ is a positive parameter that controls the width of the kernel.

In other words, γ controls how similar two instances are to each other in the kernel space. RBF function with large γ will result in low values and means that two instances will be considered different even they are close to each other in terms of the distance. Small γ means that even instances that are far from each other will be treated as similar.

In terms of SVM, γ defines how far the influence of a single training example reaches, with large values meaning 'close' and small values meaning 'far' [skla]. Therefore with large γ more instances are treated as support vector in order to classify training instances correctly. However, the model is able to correctly classify only instances very close to support vectors, resulting in overfitting.

On the other hand, with small γ the influence of a single training instance is large and less support vectors needs to be chosen which results in more simple decision function, however, the model can suffer from underfitting.

If $\gamma \to 0$, all instances will be considered equally similar and thus a majority class will be assigned to the test instance. If $\gamma \to \infty$, all examples will be considered equally different. Therefore, with sufficiently large γ values the model uses only some small number of closest instances to make the decision. From this it can be concluded that SVM with RBF kernel does a tradeoff between majority voting and KNN.

2.5.5 Handling class imbalance

The problem with SVM is that it is sensitive to the class imbalance. SVM with imbalanced data tends to produce sub-optimal models which are skewed towards the minority class and can cut the performance with respect to the minority class [HM12].

This weakness is caused by the fact that the same misclassification cost is assigned to all the training examples (see Equation 8). Since the density of majority class examples is also higher near the boundary region, the separating hyperplane is shifted towards the minority class, in order to reduce the total number of misclassifications [HM12].

There exist many different methods to handle the problem. For example, various data preprocessing methods can be applied before training SVM model including data under- and oversampling and ensemble methods, such as bagging.

Alternatively, some algorithmic modifications to the SVM algorithm have been proposed. In our work we use a method described by Veropoulos *et al.* for imbalanced data. It is a cost-sensitive learning method, where different miclassification cost is assigned to positive and negative instances [VCC99]. Besides that there exists a method called z-SVM that adjust the separating hyperplane of a model that is already trained on an imbalanced data so that the skew towards the minority class is reduced [IMTK06].

3 Hypotheses

Reasoning about the similarity of a set of words is a task that is actively being tried to solve using continuous vector-space representations of words. Some models, for example BART [LCH12], learn relations from the examples, and are successful in modelling first-order comparative relations, such as *larger* and *smaller*.

Alternatively, the representations of words themselves could encode semantic properties that can be used to infer relations between words. This has been shown to be to some extent true for word embeddings learned with neural networks, but as well for Word2Vec and GloVe embeddings [MYZ13, MCCD13, PSM14].

It has been demonstrated that using simple vector arithmetic, such embeddings can be used for answering analogy questions. For instance, the question *man is to king as woman is to ...* can be answered with operation

$$\mathbf{v} = \mathbf{v}_{king} - \mathbf{v}_{man} + \mathbf{v}_{woman}$$

and resulting vector $\mathbf{v}$ will be similar to $\mathbf{v}_{queen}$ [MYZ13]. However, it is still not well understood how and to what extent such semantic relations are encoded into the embeddings. Chen *et al.* evaluated different types of semantic relations and found that some relations are better captured than others [CPG17].

In our work we are going to lift this similarity problem to phrases, i.e. we want to capture similarity of phrases such that semantically more similar phrases are close. This would make it possible to detect phrases by some synonymous meaning. Since Estonian has free word order, that is, the meaning of the phrase does not depend on word order, the similarity measure should be invariant under permutations. We could take an average from all the word vectors but this is too crude. WMD, on the other hand, provides necessary alignment and can be visualized as well.

Kusner *et al.* showed in their experiments that WMD constantly outperforms other state-of-the-art document similarity measures in KNN classification task [KSKW15]. However, WMD gives a distance that does not have a simple representation in vector space. That is, there is no simple embedding that would capture WMD distance. We can overcome this limitation with RBF scaling which provides an infinity dimensional space to embeddings and with small kernel width reduces to KNN as well (see Section 2.5.4).

The problems that we are going to try to solve using WMD as a similarity measure are as follows.

Predicting omitted word. We are trying to predict omitted word, given only the contexts of the word. Our hypothesis is that, to some level, it is possible to predict the omitted word, even when the words are semantically very similar. However, as the phrases can be constructed in many different ways, we now formulate two hypotheses about phrase construction that we think will perform better than simple context window.

- Forming phrases using syntax tree fragments will improve the results achieved with using simple context window.
- Forming phrases by removing most common words and extending the context will improve the results achieved with using simple context window.

The hypotheses will be evaluated in terms of both, supervised and unsupervised learning methods. We have one additional hypothesis about classification algorithms:

- Kernel-SVM with RBF-kernel will have better classification accuracy than KNN.

Adverse drug reaction detection. In this experiment we are going to use contexts of the words that could potentially describe drug adverse reactions to decide whether the word was actually related to some adverse effect or not.

Analysis of syntax parser errors. Here we are going to analyse the phrases that are incorrectly tagged by the syntax parser. We are interested in whether it is possible to separate correctly tagged phrases from incorrectly tagged phrases by semantic meaning.

First problem is a made up problem to assess the suitability of the method for phrases. Last two problems are examples of practical cases. Therefore it only makes sense to apply WMD to last two problems if we solve the first problem successfully, that is, we show that it is possible to do semantic separation of phrases using WMD. In Section 4, where the results of the first problem are discussed, we see that to some level it indeed is possible to do that. Thus it is interesting to apply the method to the practical cases as well.

Other practical applications where method could be useful include inferring word sense from given context, detecting synonyms, extracting sub-meanings or automatically classifying word usages.

4 Predicting missing word

4.1 Word removal test

To carry out this experiment, we are going to use a special kind of method, which we will call a word removal test. Essentially, we are using this method to avoid manually labelling hundreds of phrases by some semantic meaning. Word removal test is a test where two different but semantically slightly similar words are removed from their contexts and the omitted word is used as a semantic label for the context. The phrases are then either clustered or used for learning a binary classifier that would predict the missing word.

In terms of cluster analysis, the expected result is that phrases are clustered by their semantic meaning and that each semantic meaning is relevant to only one word from two possible. That is, each cluster contains phrases that have the same label. This would mean that it is possible to decide which word was used in a context without knowing the word itself.

Ideally, all the clusters would be pure but we do not expect that to happen since the data will probably contain phrases that would be difficult to label even for a human. We are satisfied if easily labelable data is located in pure clusters and hardly labelable data is put into impure clusters. In other words, we are expecting to see a set of pure clusters together with a set of impure clusters. Ideally, the number of impure clusters would be as small as possible.

The shortcoming of this test is that there is no human baseline available to compare the results. It could be that accuracy of 55% for some cases is the best that can be achieved but we do not know that.

4.2 Data

The experiment is carried out on the data from etTenTen corpus [Mui16] which is an Estonian language corpus made up of texts collected from the Internet. We decided to use etTenTen because the language used there is less formal and contains many errors which is closer to medical texts that we will be also be working with. The corpus contains more than 600 000 text files that were converted into JSON format and linguistically analysed with EstNLTK [OPT+16] version 1.6_b [Ora18].

4.2.1 Defining word pairs

As word removal test bases on semantically similar word pairs, we need to somehow define those pairs. We decided to form the pairs by their level of semantic similarity: one pair of semantically very similar words, one of somewhat similar words and one of different words. We are assuming that it is more difficult to label contexts of similar

words and easier to label contexts of different words. Therefore we are expecting to see improvement in clustering quality while the level of similarity between words decreases.

To ensure that the results on a single word pair are not random and can be generalized to any word pair having the same characteristics, we chose three pairs of words for each level of similarity. The exact word pairs are shown in Table 1. As a result, we have three test sets, each containing words pairs with different similarity. We will hereafter refer to these testsets as *õun*-testset, *auto*-testset and *arst*-testset.

Very similar	Somewhat similar	Different
õun-puuvili	*õun-kivi*	*õun-auto*
auto-buss	*auto-lennuk*	*auto-koer*
arst-psühholoog	*arst-advokaat*	*arst-vend*

Table 1. Word pairs that were used in word removal tests

In *õun*-testset we have 597 phrases for each word, in *auto*-testset the number is 800 and in *arst*-testset it is 692. The number depends on how many sentences there are in the corpus for each word but is not larger than 800.

4.2.2 Constructing phrases

We used three different phrase generation methods in our experiments. All phrases were formed using lemmatized words, regardless of the method. Ambigous lemmas were left as they were, to the form of *lemma1|lemma2*, since Word2Vec model that we used for embeddings is trained on lemmatized data and also contains ambigous lemmas.

The consequence of using lemmatized forms instead of grammatical forms is that some semantic information gets lost. As it shortly can be seen, it is much more easier for a human to understand the meaning of a phrase when it is in its grammatical form instead of in lemmatized form. On the other hand, using grammatical forms would mean that we would have more words that are unknown to the model and must be left out.

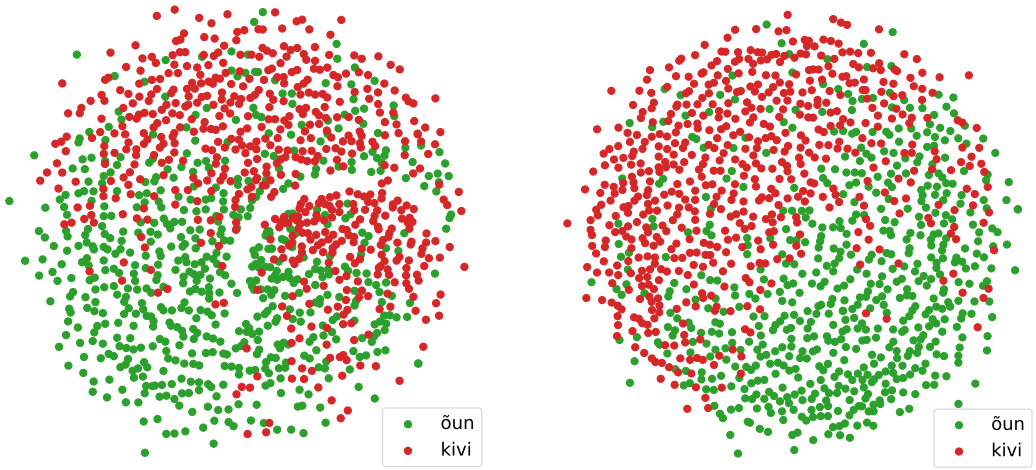
Context window based phrases. We used symmetric contexts windows with size 3 (three words from left and three words from right). Punctuation and conjunctions were omitted.

Syntax tree based phrases. Syntax parsing was done with EstNLTK MaltParser. We did not have much knowledge about what would be the best way to construct the phrases, so the goal was to explore syntactic structure of the sentences and find a meaningful way to do that.

We started by examining syntactic functions of test words. Since they were nouns, the distribution of syntax tags was similar for all words. By observing some impure clusters from baseline results we found that phrases that are difficult to separate tend to contain many attribute words that did not seem to help with deciding the right label and are missing words with other syntactic functions. Table 2 contains *õun-kivi* phrases from one such cluster.

One possible solution to that problem was to form the phrase from words that had more meaningful syntactic functions: subject, object, finite main verb, adverbial, and a root word. If the resulting phrase was longer than 6 words, as quite often happened, we chose 6 words that were closest to the test word. Examples of syntax based phrases are shown in Table 3. Sentences from which the phrases were extracted are in Table 3 same as in Table 2.

As it can be seen, some contexts have become much shorter and the words they contain carry less word specific meanings. On the other hand, some phrases now contain words that could possibly make it easiser to label the data correctly. Such words are, for example, word *laud* in the second sentence and word *veeretama* in the last sentence.

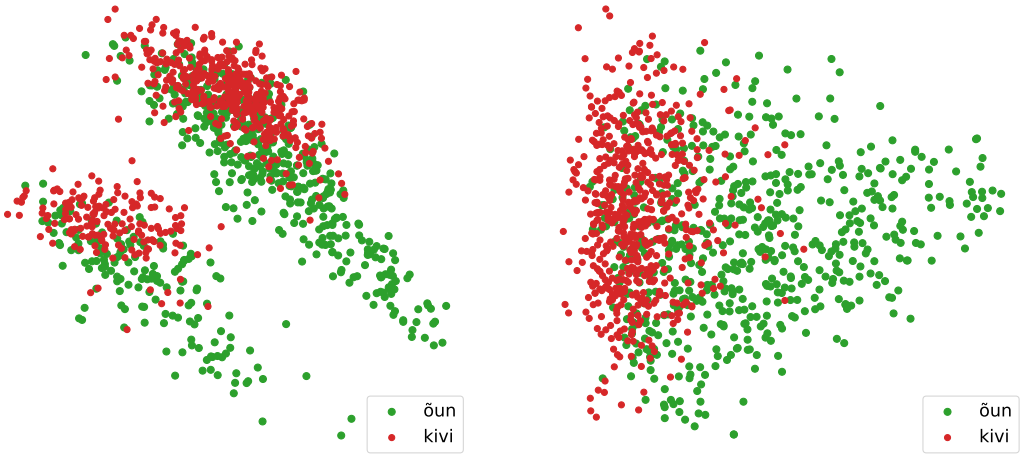


(a) baseline (b) no stop words

Figure 2. MDS visualizations of distances between *õun-kivi* phrases

Removing stop words. Another problem with the baseline phrases that was revealed after performing dimensionality reduction on the data and visualizing the results (Figures 2a and 3a), was that there was a large group of phrases that were notably closer to each other than to other phrases. It turned out that these were the phrases that contained one or more identical words. Examples of such sentences can be seen in Table 4.

Since the distance between identical words is zero, the distance between these phrases becomes smaller and they end up in the same clusters, regardless on which word these phrases originally contained. This suggested that by removing most common words, or stop words, as they are called, could improve the clustering quality. Figures 2b and 3b show that indeed removing stop words also removed the gap in distances.



(a) baseline (b) no stop words

Figure 3. Kernel-PCA visualizations of distances between *õun-kivi* phrases

In our experiment we used a predefined list of Estonian stop words [Uib18] to decide which words to remove and extended the context until the length of the phrase was up to six words. As a results, the phrases were not symmetric anymore, containing one to six words, as shown in Table 5.

It can be seen that, in some cases it is now much easier for a human to decide the correct label of a phrase because these phrases are containing words that are more label specific.

original phrase	lemmatized phrase
pani mannile ühe _ peo peale teised	panema mann üks _ peo|pidu peale teine
lk kõva köide _ passib patta panna	lk kõva köide _ passima pada panema
andma pool kuningriik _ teine pool kohuke	andma pool kuningriiki _ teise poole kohukese
ei jää ühtki _ teise peale mis	ei jääma üks _ teine peale mis
on üks suur _ väiksemate peale tõstetud	olema üks suur _ väiksem peale tõstetud|tõstma
joone tagant tuleb _ teise joone peale	joon tagant tulema _ teine joon peale

Table 2. Examples of *õun-kivi* baseline phrases that are in the same cluster

original phrase	lemmatized phrase
pööras ümber pani mannile _ peale teised tõstis	pöörama ümber panema mann _ peale teine tõstma
268 _ passib patta panna lauale kanda	268 _ passima pada panema laud kandma
olin andma _ eest	olema andma _ eest
tõesti jää _ teise peale mis kistaks	tõesti jääma _ tema peale mis kiskuma
peal on _ peale	peal olema _ peale
tõmmatakse kaks tagant tuleb _ peale veeretada	tõmbama kaks tagant tulema _ peale veeretama

Table 3. Examples of *õun-kivi* syntax based phrases

original phrase	lemmatized phrase
tõestanud kuidas tuleb _ toime nii raske	tõestama|tõestanud kuidas tulema _ toim|toime nii raske
tuleb katta rabarberite _ neile puistata pool	tulema katma rabarber _ tema puistama pool
küpsenud siis tuleb _ vars katki lõigata	küpsema|küpsenud siis tulema _ vars katki lõikama
sammuga on saapad _ liiva täis tuleb	samm olema saabas _ liiv täis tulema
hukkunute arv tuleb _ vaid sel juhul	hukkunu arv tulema _ vaid see juht
tuleb ära korjata _ hiljem oleks umbrohutõrjet	tulema ära korjama _ hiljem olema umbrohutõrje

Table 4. Examples of *õun-kivi* baseline phrases that are in the same cluster

original phrase	lemmatized phrase
teadus tõestanud _ toime raske ülesandega	teadus tõestama|tõestanud _ toim|toime raske ülesanne
põhi katta rabarberite _puistata pakki tordipulbrit	põhi katma rabarber _ puistama pakk tordipulber
punase õun küpsenud _ vars katki lõigata	punane|punas õun küpsema|küpsenud _ vars katki lõikama
paari sammuga saapad _ liiva täis kaldal	paar samm saabas _ liiv täis kallas
teksti hukkunute _ arv juhul teada täpne	tekst hukkunu arv _ juht teadma täpne
korjata _ hiljem umbrohutõrjet kergem	korjama _ hiljem umbrohutõrje kergem

Table 5. Examples of *õun-kivi* phrases without stop words

4.3 Results

We carried out three experiments in total. The baseline experiment was done with phrases formed using simple context window. Second experiment used phrases that were formed from syntax tree fragments and third experiment was carried out on phrases from which stop words were omitted.

4.3.1 Cluster analysis

Deciding the number of clusters. The number of clusters depends on the number of different meanings a word can have and the number of different contexts in which the word is used. The latter number is unknown to us and can be only guessed. We experimented with different number of clusters, ranging from 25 to 125 but none of these did not seem to be significantly better than others. Table 6 shows average silhouette scores for different cluster numbers from baseline experiment. The scores are close to 0 in all cases.

Silhouette score close to 1 for a single instance i would mean that i is much closer to instances in its own cluster than to closest neighbouring cluster. Therefore instance i is well-clustered as second best cluster is not nearly as good as current cluster [Rou87].

Silhouette close to -1 would mean that, on average, similarity to neighbouring cluster is much larger than to its own cluster and therefore instance i has not been assigned to an appropriate cluster [Rou87].

However, in our case the scores are close to zero. This indicates that average distance to instances in its own cluster is approximately equal to average distance to closest neighbouring cluster and it is not clear to which cluster i should have been assigned to, that is, what is the correct cluster for i [Rou87].

Intuitively, high score would mean that the phrases are assigned to clusters in such a way that each cluster contains phrases with some semantic meaning that is specific to only phrases in that cluster. In this case we would have found the number of clusters that corresponds to the number of different semantic meanings among the phrases (which is unknown to us).

No. of clusters	Avg. silhouette
25	0.024
50	0.018
75	0.022
100	0.012
125	0.008

Table 6. Average silhouettes

For final evaluation the number of clusters was chosen to be 75.

Quality. Table 7 compares purity scores of different word pairs from all three experiments. As it can be seen, removing stop words improved purity score, however, syntax based phrases did not give the expected improvement.

	Baseline	Syntax	No stop words
õun-auto	0.67	0.68	**0.72**
õun-kivi	0.70	0.65	**0.75**
õun-puuvili	0.67	0.67	**0.71**
auto-koer	0.66	0.64	**0.68**
auto-lennuk	0.63	0.63	**0.66**
auto-buss	0.61	0.63	**0.67**
arst-vend	0.71	0.68	**0.74**
arst-advokaat	0.69	0.70	**0.72**
arst-psühholoog	0.65	0.64	**0.70**

Table 7. Purity by word pair

We can also find the ratio of positive instances in each cluster and assign a positive label to all instances in clusters with above some threshold. This converts our clusters into a binary classifier and we can measure its performance with ROC curve and AUC score (see Section 2.4.1 for more details about converting clustering to a classifier). In our context, ROC curve is an important measure because it helps us to visualize the trade-off between true positives and false negatives as we do not know how well the data is clustered in terms of labelling and therefore what is a good choice for a threshold. AUC can be interpreted as an estimation of the probability that an arbitrary positive-negative pair is ranked correctly. Therefore it assess the choice of scoring.

Converting clustering into a classifier is especially relevant in practice, where the data is often not labelled, but still some binary decision has to be made. In this case, manual labelling of the whole dataset can be avoided by clustering the data and then manually ordering clusters by relevance by only looking at a small subset of instances in the clusters. Then a threshold can be chosen to separate the data into two classes. This provides a trade-off between the amount of manual work and accuracy of the labelling.

Respective AUC scores are shown in Table 8 and ROC curves of *õun*-testset are visualized in Figure 4. Again, it can be seen that there is not much difference in whether baseline phrases or syntax based phrases were used, and the best results were achieved with phrases without stop words.

There could be different reasons why syntax based phrases did not work as well as expected. Firstly, there are still clusters that are formed from phrases that contain some common word with little semantic meaning. Such words are, for example, *olema*, *tegema*, *võtma*, *eest*, *pärast*, *kes*. Secondly, perhaps by omitting words only on the basis of their syntactic functions we also lost some words that carried useful semantic information. As a result, some phrases could have become too general in order to be separated into different clusters.

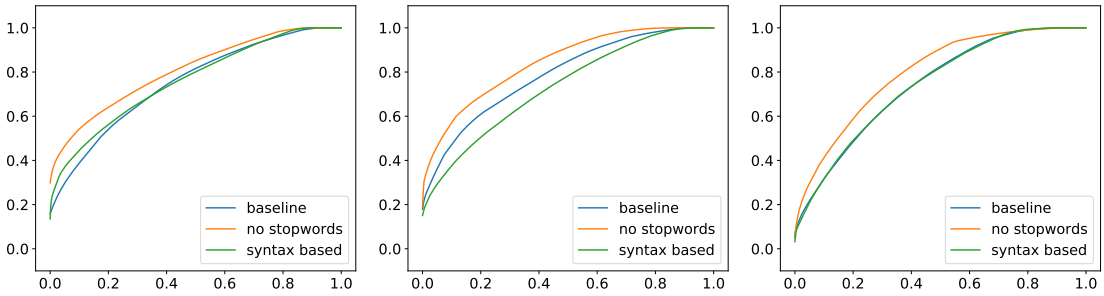
	Baseline	Syntax	No stop words
õun-auto	0.75	0.76	**0.81**
õun-kivi	0.79	0.73	**0.84**
õun-puuvili	0.74	0.74	**0.80**
auto-koer	0.73	0.71	**0.76**
auto-lennuk	0.70	0.70	**0.74**
auto-buss	0.67	0.69	**0.75**
arst-vend	0.79	0.74	**0.81**
arst-advokaat	0.78	0.79	**0.81**
arst-psühholoog	0.71	0.71	**0.78**

Table 8. AUC by word pair

Omitting stop words clearly solved both problems mentioned above but the problem of having clusters that are constructed from phrases that contain a mutual word stayed. Although this time the mutual words are more meaningful, for example, *aeg*, *vesi*, *korjama*, it results in smaller distances between phrases that might not otherwise be similar. This seemingly larger similarity can be misleading for clustering algorithm. For example, following two *õun-kivi* phrases *tikker pohl jõhvikas _ külmutamine keev vesi* and *põhjavesi tase purustama _ vesi tõstma ekskavaator* ended up in the same cluster although it is easy for a human to see that they have different labels.

4.3.2 Classification

In cluster analysis the labelling is not known for the algorithm and it must learn to group the data in a meaningful way without any supervision, that is, without labels that would indicate correct grouping. In classification tasks, however, labelled data is used to learn a function that would correctly output the label of an unseen instance. A classification algorithm analyzes available data with corresponding labels and produces a function that would output correct labels for those instances.



(a) õun-auto (b) õun-kivi (c) õun-puuvili

Figure 4. ROC curves comparing clustering quality of different experiments

Choosing the right parameters. Proper choice of penalty parameter C and RBF kernel parameter γ are essential to the SVM's performance. The best-performing combination of parameters was chosen using grid search. We used k-fold cross-validation with $k = 10$ on every word pair to train SVM classifier with given combination of parameters and evaluated the results on test set. Results of word pairs from *õun*-testset are shown in Figure 5. It can be clearly seen that the level of semantic similarity between two words affects the accuracy of the model. In the final model C was set to 1 and γ was chosen to be 0.1.

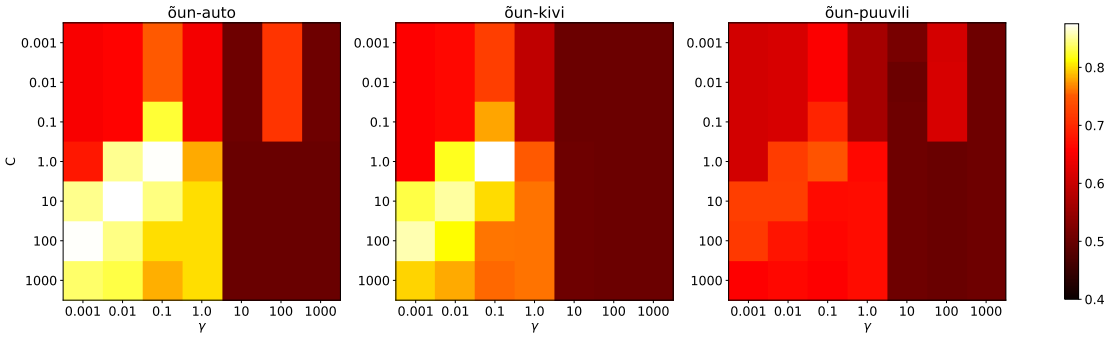


Figure 5. Choosing the proper parameters for RBF-SVM classifier

The number of neighbours in KNN algorithm was chosen to be 35 since it is approximately the square root of the train data size which is often used as a rule of thumb for choosing k. Cross-validation with different numbers of k showed that the performance of the algorithm was similar with any choice of k tested.

Results. Table 9 shows classification accuracy of SVM and KNN with different word pairs in all three experiments. The table shows that, with both algorithms, phrases without stop words achieved the best accuracy while results with syntax based phrases

were almost always lower than with baseline phrases. It is also clear that RBF-SVM outperforms KNN.

We achieved at least 80 percent accuracy with RBF-SVM on semantically different or somewhat different word pairs in all testsets. These results are promising and show that the distances between phrases obtained with WMD contain enough information in order to efficiently classify or cluster phrases by the semantic meaning they are carrying.

	SVM			KNN		
	baseline	syntax	no stop words	baseline	syntax	no stop words
õun-auto	0.87	0.83	**0.90**	0.81	0.77	**0.86**
õun-kivi	0.83	0.80	**0.86**	0.76	0.75	**0.82**
õun-puuvili	**0.74**	0.69	0.73	0.65	0.66	**0.67**
auto-koer	0.78	0.77	**0.82**	0.72	0.72	**0.80**
auto-lennuk	0.78	0.73	**0.80**	0.69	0.67	**0.76**
auto-buss	0.71	0.70	**0.76**	0.64	0.62	**0.71**
arst-vend	0.82	0.78	**0.85**	0.76	0.70	**0.83**
arst-advokaat	0.80	0.81	**0.86**	0.76	0.75	**0.83**
arst-psühholoog	0.74	0.71	**0.78**	0.65	0.65	**0.74**

Table 9. Classification accuracy by method and word pair

5 Adverse drug reaction detection

In this section we apply WMD to phrases that are extracted from patients' epicrises in order to automatically detect adverse drug reactions (ADRs).

In medicine, epicrisis is a summary of a medical case history. As such the epicrisis is a collection of structured data and free-text fields. Adverse drug reaction is an undesired harmful effect resulting from medication. Medical personnel is instructed to report about serious adverse reactions through structured diagnosis fields. However, sometimes the adverse reaction is in so mild form that it occurs only in textual description of the patient state. The textual description is also more nuanced than the rough formal categorisation. Hence, semiautomatic extraction of ADR phrases is a meaningful fact extraction task.

The data is provided to us by STACC who has already done sentence tokenization of the data (which is not a trivial task due to heavy use of domain-specific abbreviations and missing spacings). STACC is also responsible for tagging potential ADRs in the data. These potential ADRs were then manually labelled [Kre19].

5.1 The problem

We are going to treat ADRs detection problem as a binary classification task. Our guess is that phrases that describe ADRs are semantically different from other phrases in the texts and that dissimilarity is also reflected by WMD. This would make it possible to separate ADRs from other phrases with a binary classifier.

The task is especially interesting to researchers in Geenivaramu who seek for connections between ADRs and genetic data. Being able to detect ADRs automatically would reduce the amount of manual work that is done for the detection.

5.2 Data

The data contains 11 179 potential adverse effect mentions from 1052 different epicrises, all of which have diagnose codes which contain at least one adverse effect diagnosis. So we know that all the epicrises are related to the appearance of an adverse effect in some way. Some examples from the dataset are shown in Table 10.

For each mention the sentence that contains that word and a label is known. The data is manually labelled. For each tagged word it is known whether this specific word describes adverse effect or not. Multiple tagged words could be described by identical sentences since one sentence often contains more than one tagged word. Therefore the number of different sentences in the dataset it actually smaller which also reduces variability in phrases. In addition to that the original dataset contains duplicates due to being merged from different tables.

The data is heavily unbalanced. Around 10% of all the examples are positive instances, that is, they were used in the context of drug adverse effect. For the majority

sentence	ADR type	is ADR	drug
1 kuu tagasi tekkis ***lööve*** *käsivartele,*	*nahalööve*	yes	fastum
kasutas paikselt Fastum-geeli ***põlvevalu*** *tõttu /.../*	*liigesevalu*	no	
/.../ patsient adekvaatne, liigub osakonnas	*peavalu*	no	
ringi, käib suitsetamas.Kurdab ***peavalu****,*	*iiveldus*	no	
iiveldust*,* ***oksendamist*** *ei ole esinenud.*	*iiveldus*	no	
Tarvitanud Loratadini, leevendust saanud	*punetus*	yes	N/A
vähe.Vasema labajala dorsaalpinnal ***punetav***	*nahalööve*	yes	N/A
laik*.Plaanis UH ja biopsia, /.../*			

Table 10. Examples from drug adverse reaction dataset

of positive instances the drug that caused the adverse effect is also mentioned in the sentence, however, not always.

5.2.1 Preprocessing

During the preprcessing duplicates were removed and the data was tokenized and lemmatized. Similarly to word removal experiments punctuations and conjunctions were omitted, but also cardinal numbers and abbreviations since they are frequent words in the domain but carry little meaning in the context of our problem.

	total	no adverse effect	adverse effect	drug mentioned
original	11 179	9989	1190	883
no duplicates	7793	6969	824	602

Table 11. Dataset size and properties before and after removing duplicates

5.2.2 Constructing phrases

The phrases were constructed using the same method as with the word removal tests: using context window of size 3 as a baseline method and removing stop words for the comparison. The reaction itself was also included into the phrase in both cases. We decided to experiment with baseline phrases as well, because we are unsure whether removing stop words will improve the results as it did in word removal tests.

There are two reasons why it might not work. Firstly, the sentences are often very long and contain many semantically different ideas, however the ideas are described with rather short phrases. Extending the context too much could include words that are not

related to the problem and obscure the meaning of interesting words. Secondly, some most common stop words seem to be essential for deciding the label. For example, we might not want to remove last two words from the phrase *palavikku ei ole*.

5.2.3 Sentences

In addition to previous concerns, there is a problem with labelling. In fact, is not perfect for our task. The data is labelled so that, in case there is an adverse effect, all the related reactions in the epicrisis are labelled as positives. This means that the decision is not always made by looking at the local context but rather is global context used as well. Since epicrises are usually multiple sentences long, it is even possible that relevant information is not located in the same sentence.

In terms of our task it means that the data is faulty and contains false positives, as we are assuming that the decision can be made by only seeing a short context near the reaction. However, we do not know the extent of this problem and how this is affecting the results but it can increase the number of false positives returned by the model (with small γ). In fact, we do not even know what should be the size of the context that would contain all the relevant information. It could be that the whole epicrisis must be included into the decision making. However, we do know that the decision has always been made based on the information in the texts.

All in all, it is worth a try to predict the label for the whole sentence as well. To do that, we set the label of a sentence to positive if it contains at least one mention of adverse effect and negative otherwise. In total we have 5259 different sentences among which 500 are positive instances.

5.3 Results

We used SVM to classify the data. Grid search for choosing the parameters was done on half of the data using stratified k-fold cross-validation with k=10. The results were evaluated with F1 score and are shown in Figure 6.

We did not allow phrases from the same epicrises appear both in train and test/dev set, because such phrases are often very similar. The model in this case would predict the data it has already seen and the results would be misleading. Also, in real life we are interested in predicting the results of epicrises that we have not seen before.

The class imbalance problem was handled with assigning different values of penalty parameter *C* to positive and negative class. More specifically, we used balanced weighting where parameter *C* is multiplied with class weight. For each class *i* the class weight was calculated as following [sklb]:

$$weight_i = \frac{n_{samples}}{n_{classes} \cdot n_i} \ , \tag{11}$$

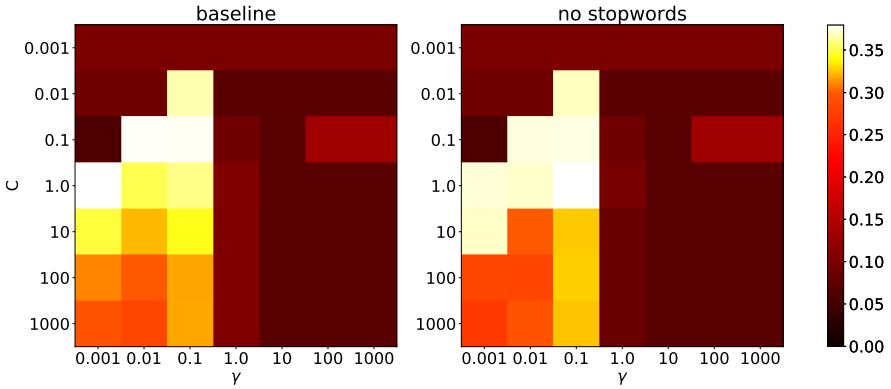


Figure 6. Cross-validation F1 score of RBF-SVM adverse effect classifier

where $n_{samples}$ is the number of samples, $n_{classes}$ is the number of classes and n_i is the number of instances from class i.

Figure 7 visualizes cross-validation γ values. Each matrix is a sum of results of 10 test folds. Since the data is heavily unbalanced, each class is normalized by the number of elements in that class for more visual interpretation. We can see that while model with lower gamma gives us more positive instances, it also increases the number of false positives, that is, recall is high but precision is low.

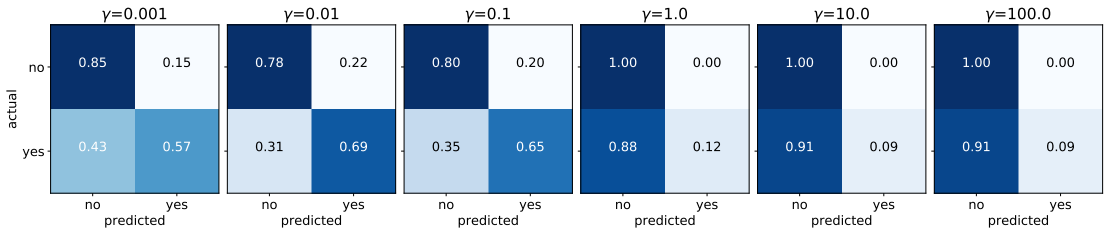


Figure 7. Cross-validation confusion matrices for different values of γ

Larger gamma, on the other hand, manages to identify only a small amount of positive instances but the number of false positives is also low, resulting in a high precision but low recall, in term of a positive class. It is also clear that the model in this case is overfitting. From these observations we can conclude that adverse drug reactions are not easily separable from other phrases.

The final model was trained on 90% of the data and tested on 10% of the data. Parameter C was chosen to be 1. Parameter γ, assuming we want to optimize recall, that is, detect as many positives as possible, was chosen to be 0.01. Results are shown in Table 12. Figure 8 shows ROC curves on the same data.

Results show that there is not much difference on whether we remove stop words or not. Observing the phrases revealed that removing stop words did not change the content

	accuracy	precision	recall
baseline	0.80	0.35	0.66
no stop words	0.80	0.35	0.65
sentences	0.87	0.38	0.71

Table 12. Results on test data

of the phrases as much as we had expected. This could mean that stop words are not used very frequently near the potential adverse effect mentions. Another explanation is that there is quite many words in the texts that are unknown to the Word2vec model and therefore were simply left out from the phrases. These words include, for example, some domain-specific words, misspelled words and drug names.

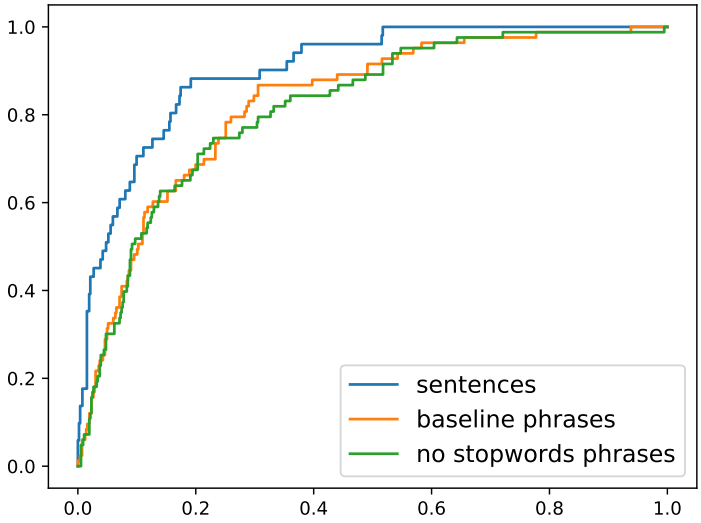


Figure 8. ROC curves on test data

We can also see that predicting labels for sentences performs better than predicting labels for phrases. This indicates that indeed larger context is needed for correct decision-making.

Because the labelling of the data is not perfect for our task, it is not so straightforward to reason about misclassification. However, one of the observation is that there exists a set of reactions that the model almost always predicts as positive. These reactions are, for example, *punetus*, *sügelus* and *lööve* and often appear in some combination, for instance, *punetav lööve*. In addition to that, phrases containing words *iiveldus* and *allergiline* are also mostly predicted as positives. All these words are among the most common adverse reactions. At the same time, some phrases that are very similar are labelled differently.

See Table 13 for some examples.

There is much less false negatives than false positives and it is hard to see any patterns other than some of the most common false negative ADRs are *turse* and *valu*. Finally, as can be seen in Table 13, given the short context it is very difficult to decide the label.

phrase	true label	predicted label
ülitugev iiveldus nõrkus jõuetus lisaprobleem ka	1	1
kui pearinglus tasakaaluhäire aeg-ajalt iiveldus	0	1
peavalu minestama peavalu püsima esinema iiveldus	0	0
viimane kuu süvenev jõuetus nõrkus turse jalg	0	0
kuu süvenev jõuetus nõrkus turse jalg anamnees	0	1

Table 13. Examples of potential ADR phrases

To sum up, it is clear that the extracted phrases do not contain information that is needed to make the correct decision. We also know that the information needed for that might not even be in the same sentence. Therefore it is not reasonable to predict label for each potential ADR. On the other hand, we saw that predicting label for the whole sentence somewhat improves the results.

We can see from Table 12 that for the majority of phrases the drug that caused adverse reaction is known, that is, the drug was mentioned in the same sentence. We could somehow try to make use of this information when extracting phrases, e.g, cut the phrase towards the mention of the drug as it seems that interesting context is located near and between the drug and the reaction.

However, as we do not want to predict label for each reaction, we could try to predict the label for each mention of the drug instead. After detecting drugs from the data they can be automatically labelled using the already available data. The disadvantage of this is that the data will again be unbalanced but on the other hand, context might be more meaningful for positive instances.

Alternatively, we could fix a list of potential ADRs and use WMD to extend that list. This is potentially useful because the same ADR can be expressed with different words. The phrases can then be compared with drug information. Comparing the relatedness of potential ADRs to different drugs would then give an answer about which drugs most often cause some specific ADR. As a result, we can say that some potential ADRs most probably are not actually ADRs. This would reduce the amount of manual work that has to be done to decide the final label.

6 Analysis of syntax parser errors

In this section we use WMD to analyse the errors made by MaltParser [NHN+07] syntax parser. We apply the distance metric to the phrases tagged by the parser and try to separate correctly tagged phrases from incorrectly tagged phrases.

Parsing algorithm in MaltParser consists of parser configurations and transitions. During parsing, transitions are used to map one non-terminal parser configuration to another configuration until the whole sentence is parsed [NHN+07]. What is interesting in our context, is that the choice of transition is nondeterministic, that is, there is normally more than one transition applicable. Therefore the system is supplemented with a classifier that predicts the next transition at each nondeterministic choice point. This classifier is learned from the annotated treebank.

MaltParser gets accuracy around 80-85%, however, it overfits because the training data is sparse. Because of that, it is infeasible to increase accuracy of the parser by extensive labelling, as the troublesome cases are also sparse. To overcome that, a new approach is needed that focuses on errors in the data. However, we can only work with potential errors, as there is no gold annotation. This is where WMD becomes useful as we could use it to try to separate true errors from false positives.

We already know that the potential errors are phrases with some specific structure. There is around 20 such cases where MaltParser often makes errors. Each case affects accuracy of the MaltParser around 0.5%, therefore successfully solving at least one of the problem gives an improvement in the accuracy. The problem analyzed in this work is one such case. More specifically, we only look at the errors with predicatives. Hence the task is to predict based on MaltParser output whether parser makes an error. We can evaluate the results with a small set of gold annotations [Sä].

6.1 The problem

The state-of-the-art MaltParser model for Estoninan language [Sä] quite often makes mistakes in analysing sentences and produces incorrect syntactic parses for the sentences. This does not mean that parse for the entire sentence is wrong, rather are some parts of the sentences parsed correctly and some incorrectly.

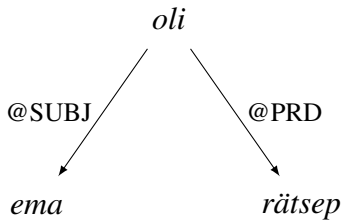
One such part of a sentence where MaltParser often makes mistakes, is a predicative. However, predicative is a single word and in order to analyze the problem with WMD, we need to define a phrase, some context in which the predicative appears in. Analysis of errors has shown that one problem with predicative is that MaltParser often confuses subject and predicative. Moreover, it is also known that problematic cases are phrases where both words are nouns or pronouns, that is, not adjectives. Thus we restrict our analysis of potential errors to only predicatives for which the subject also exists. For convenience, we define predicative fragment as a predicative in such specific context as follows.

Predicative fragment. Predicative fragment is a phrase that consists of a subject, a verb, and a predicative such that verb is a parent of other two.

Figure 9 shows an example of predicative fragment. Defining more general context for a predicative fragment would also significantly increase the noise among the possible predicatives found by the parser. This will become more clear after we have described the process of extracting phrases.

6.2 Data

The data contains 29767 sentences for which MaltParser syntax labelling and gold labelling is known. Gold labelling is manually verified and contains true syntax parses. MaltParser labelling is obtained through a cross-validation, where 9 folds are used for training and one for predicting and the process is cycled to get predictions for all folds. This is needed to counter extreme overfitting of MaltParser.



(oli (ema@SUBJ, rätsep@PRD))

Figure 9. Predicative fragment

6.2.1 Constructing phrases

There are three types of mistakes MaltParser can do while detecting predicative fragments. Firstly, it may tag a phrase as a predicative fragment when it actually is not. Secondly, it may find the correct fragment but might not able to detect predicate correctly. Thirdly, it may miss a predicative entirely by doing some nonsensical error. However, we can not consider third problem because that would require us treating all possible phrases as potential errors. We therefore restrict the set of possible errors to first two types of mistakes. The amount of phrases this set covers then depends on how we define second type of mistake.

On the basis of this discussion we define potential predicative fragment as follows.

Potential predicative fragment. Potential predicative fragment is a phrase that is extracted from the data that is tagged with MaltParser. It is a phrase that could potentially be a predicative fragment according to the true labelling. It consists of three words that could either be:

1. a verb, a predicative and a subject such that verb is a parent of other two;
2. a verb, and two subjects such that verb is a parent of other two.

As predicative can only appear together with some specific verbs, we further restrict that verb must be from a following list: *olema*, *paistma*, *tunduma*, *näima*. In case it is possible to form the fragment in both ways, the phrase matching the first rule is chosen.

Examples of potential predicative fragments can be seen in Tables 14, 15 and 16. Table 14 shows examples of potential predicative fragments that are not actually predicative fragments according to the gold labelling. As it can be seen, due do our definition of predicative fragment some predicatives will remain unrecognized.

MaltParser labelling	gold labelling
(pole (mõtet@PRD, spekuleerida@SUBJ))	(pole (mõtet@SUBJ (spekuleerida@<INFN)))
(on (Õunapuu@SUBJ, kunstnik@PRD))	(on (Õunapuu@ADVL, kunstnik@PRD))
(olema (täituvus@SUBJ, 50-55@PRD))	(olema (täituvus@SUBJ, 50-55@OBJ))

Table 14. Examples of potential fragments that are not actually predicative fragments

Table 15 shows examples of potential predicative fragments that indeed are predicative fragments according to the gold labelling but the labelling is incorrect, that is, predicate is not detected correctly.

MaltParser labelling	gold labelling
(on (Eesti@SUBJ, koht@SUBJ))	(on (Eesti@SUBJ, koht@PRD))
(on (Riigid@SUBJ, Austria@PRD))	(on (Riigid@PRD, Austria@SUBJ))

Table 15. Examples of fragments with incorrect labelling

Finally, Table 16 shows some examples of potential fragments that are indeed predicative fragments and MaltParser has also been able to detect predicative correctly.

As already mentioned and can be seen in Table 14, our extraction algorithm misses some cases where MaltParser can make errors about predicatives. One such case is phrases where subject is not given. Another case is phrases where verb is left out, for example, in sentence *Ta vist üsna mõistlik inimene.*

MaltParser labelling	gold labelling
(on (kes@SUBJ, lumeuurija@PRD))	(on (kes@SUBJ, lumeuurija@PRD))
(on (seda@PRD, kindlustajariik@SUBJ))	(on (seda@PRD, kindlustajariik@SUBJ))

Table 16. Examples of fragments with correct labelling

6.3 Results

We label each potential predicative fragment with a flag that indicates whether the potential predicative fragment is a true predicative fragment according to the gold labelling. We add two additional flags to potential fragments for which the first flag is true. First additional flag shows whether MaltParser has labelled subject and predicative correctly. Second additional label states the direction of the fragment according to the gold labelling - either subject appears before predicate in the fragment or vice versa. Table 17 shows flags for fragments that were used in Tables 14, 15 and 16. In Table 17 the direction of the fragment is labelled with an arrow, where $\rightarrow$ means that subject appears before predicate and $\leftarrow$ means that predicate appears before in gold labelling.

MaltParser labelling	PRD	correct	direction
(pole (mõtet@PRD, spekuleerida@SUBJ))	no		
(on (Õunapuu@SUBJ, kunstnik@PRD))	no		
(olema (täituvus@SUBJ, 50-55@PRD))	no		
(on (Eesti@SUBJ, koht@SUBJ))	yes	no	$\rightarrow$
(on (Riigid@SUBJ, Austria@PRD))	yes	no	$\leftarrow$
(on (kes@SUBJ, lumeuurija@PRD))	yes	yes	$\rightarrow$
(on (seda@PRD, kindlustajariik@SUBJ))	yes	yes	$\leftarrow$

Table 17. Labelling examples

Table 18 shows the number of positive and negative instances for each of the flag. As it can be seen, the data is quite unbalanced.

	total	yes	no
PRD	2437	1999	438
correct	1999	1421	578
direction is $\rightarrow$	1999	1815	184

Table 18. Dataset size

We use SVM with balanced class weight to predict first two flags. We can not predict direction of the phrase with WMD because it hides word order. What could be predicted is the dominant direction for the phrase. Parameter search was done from the same range as in previous experiments. Reasonable choices for γ were 0.01 and 0.1 from which latter was chosen.

Results are shown in Figure 10 and in Table 19. Precision and recall are calculated with respect to the minority class, that is, negative class. For the comparison we have added results of predicting same flags with extended context i.e., using the whole sentence. We evaluated extended context as well because it is possible, as it was for ADR prediction, that larger context is needed for correct decision-making.

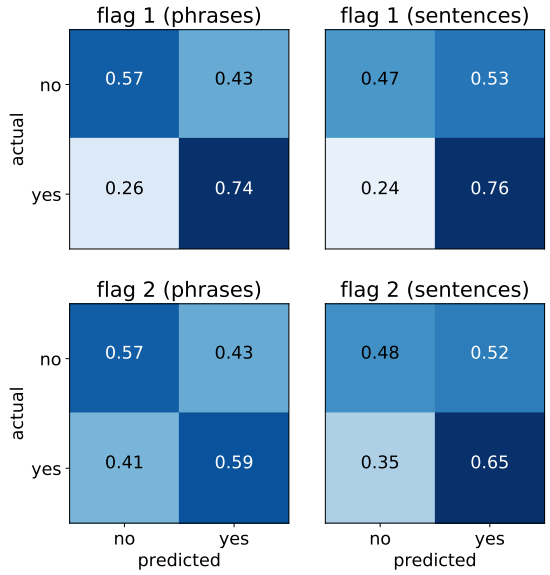


Figure 10. CV results of predicting first two flags (*PRD* and *correct*)

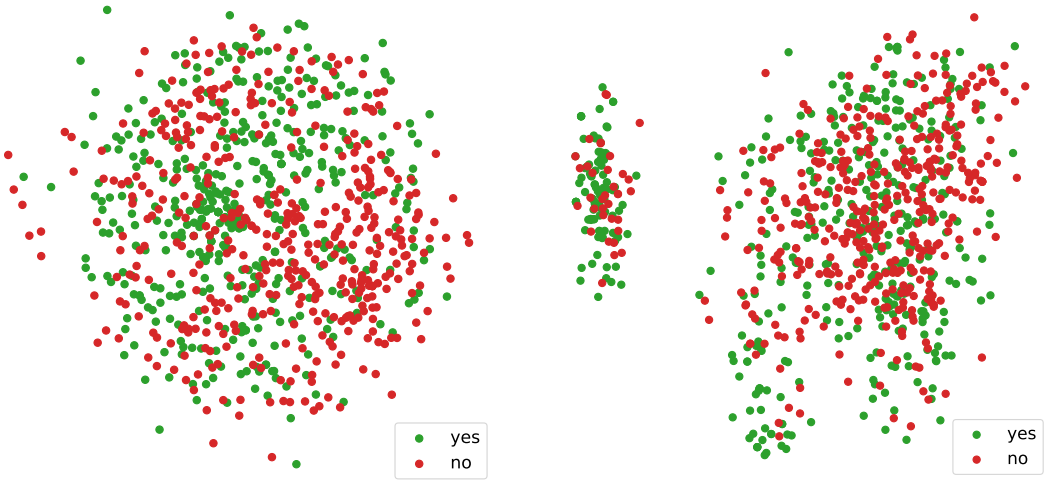
	accuracy	precision	recall
Flag 1 (phrases)	0.71	0.32	0.57
Flag 1 (sentences)	0.71	0.31	0.47
Flag 2 (phrases)	0.58	0.36	0.57
Flag 2 (sentences)	0.60	0.36	0.48

Table 19. CV results of predicting first two flags (*PRD* and *correct*)

Results show that with given configuration there is not much difference in whether we use phrases or sentences as results are not very good in neither of the cases. If we

want to detect true negatives, that is, phrases found by MaltParser that actually are not predicative fragments or are with wrong labelling, then phrases based prediction is better as recall is higher. In terms of precision there is no difference.

There are two possible reasons why the method did not work for this particular problem. Firstly, it could be, that phrases are close to each other by some semantic meaning which is too general and does not allow to make the decision about label. Figure 11 shows visualizations of undersampled phrases. It can be seen that there is no visual separation.



(a) MDS

(b) kernel-PCA

Figure 11. MDS and kernel-PCA visualization of the first label

Secondly, the data could be too sparse. To assess that, we found the closest neighbour for each instance and counted the times when labels of both instances were same and when they were different. For instances that are very close to each other, the ratio was 93:7, for instances with larger distance the ratio was not larger than 78:22. From the instances that were very close, 75% had zero distance and the ratio among those was 95:5. Therefore global context is needed to decide the label at least for 5% of cases. Among those with non-zero distance the ratio was 84:16. Such large difference in ratios means that the data is sparse. It could be that we would have achieved better results if that was not the case.

7 Conclusion

The aim of this work was to find a similarity measure that would work well with phrases. That is, we wanted to find a way to capture similarity between phrases such that semantically more similar phrases are closer, so we could detect phrases by some synonymous meaning.

We used Word Mover's Distance (WMD) as a similarity measure as it has state-of-the art performance in KNN document classification task. To overcome the limitation of not having vector representations of phrases available, we scaled WMD to an infinity dimensional space with RBF kernel. The measure was then used together with SVM to solve three different phrase base problems.

The first problem, predicting the omitted word given only the context, was a made up problem which we used to evaluate the goodness of the measure and its suitability for phrase-based tasks. Results showed that indeed it is possible to do semantic separation of phrases using WMD. We achieved best results with phrases where stop words were omitted. Forming phrases using syntax tree fragments, on the other hand, did not perform better than using simple context window.

We then proceeded with the task in medical text domain. The goal was to detect adverse drug reactions from the patients' epicrises using phrases that contain potential adverse drug reactions. The task in this time was more difficult because the language was more complex and the data itself heavily unbalanced. In addition to that, the labelling was imperfect for our task as the decision for the label was made using global context, however, our extracted phrases only included local context. We did not achieve good results and concluded that larger context is needed to correctly decide the label. We also suggested an alternative approach for solving the task with reasonable amount of additional work.

As a third problem, we were interested in whether it is possible to predict phrases that MaltParser has difficulties to parse correctly. However, we restricted the problem to only errors with predicatives that appear in a specific context. For each such potential predicative that MaltParser had found, we predicted two labels. Firstly, whether this potential predicative is a true predicative according to the gold labelling, and secondly, whether predicative in the phrase is tagged correctly. The results were not good for neither of the cases. For the comparison we predicted same labels using global context (sentence) as well instead of phrase but the results were slightly worse.

To conclude, we saw that it is possible to do some semantic separation of phrases using WMD, however, the success depends on how semantic similarity is defined and how the phrases are formed. Moreover, deciding what is the best way to form a phrase depends on the problem. Finally, one of the disadvantage of the method is that, as phrases are short, each word has a rather large influence on distance calculation. Therefore phrases containing some identical words are in terms of WMD more similar to each other than they might be by actual semantic similarity.

References

[BDVJ03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 2003.

[BG05] Ingwer Borg and Patrick J.F. Groenen. *Modern Multidimensional Scaling: Theory and Applications (Springer Series in Statistics)*. Springer, 2005.

[Bis06] Christopher Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3, 2003.

[CPG17] Dawn Chen, Joshua Caleb Peterson, and Thomas L. Griffiths. Evaluating vector-space models of analogy. *CoRR*, abs/1705.04416, 2017.

[DDF+90] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41, 1990.

[Fla12] Peter Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[HGS+16] Gao Huang, Chuan Guo, Yu Sun, Kilian Q. Weinberger, Matt J. Kusner, and Fei Sha. Supervised word mover's distance. *Advances in Neural Information Processing Systems*, 2016.

[HM12] Haibo He and Yunqian Ma. *Imbalanced Learning: Foundations, Algorithms, and Applications*. John Wiley & Sons, Inc., 2012.

[IMTK06] Tasadduq Imam, Kai Ming Ting, and Joarder Kamruzzaman. Z-svm: An svm for improved classification of imbalanced data. volume 4304, pages 264–273, 12 2006.

[Kre19] Kristi Krebs. Collection of anonymised ADR phrases, 2019. Available by request from Geenivaramu.

[KSKW15] Matt Kusner, Y Sun, N.I. Kolkin, and Kilian Weinberger. From word embeddings to document distances. *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 957–966, 01 2015.

[LCH12] Hongjing Lu, Dawn Chen, and Keith J. Holyoak. Bayesian analogy with relational transformations. *Psychological Review*, 119:617–648, 2012.

[Lee77] Jan De Leeuw. Applications of convex analysis to multidimensional scaling. *Recent developments in statistics*, 1977.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *Proceedings of the International Conference on Learning Representations*, 2013.

[MKB+10] Tomáš Mikolov, Martin Karafiát, Lukás Burget, Jan "Honza" Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. *Proceedings of Interspeech*, 2010.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[Mui16] Kadri Muischnek. Eesti veeb 2013 (etTenTen) korpus, morfoloogiliselt ühestatud. *Center of Estonian Language Resources*, 2016.

[MYZ13] T Mikolov, W.-T Yih, and G Zweig. Linguistic regularities in continuous space word representations. *Proceedings of NAACL-HLT*, pages 746–751, 01 2013.

[NHN+07] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135, 06 2007.

[NJW02] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2002.

[OPT+16] Siim Orasmaa, Timo Petmanson, Alexander Tkachenko, Sven Laur, and Heiki-Jaan Kaalep. Estnltk - nlp toolkit for estonian. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may 2016. European Language Resources Association (ELRA).

[Ora18] Siim Orasmaa. Veebikorpus13 korpus analüüsitud EstNLTK v1.6.b abil. *Center of Estonian Language Resources*, 2018.

[PSM14] Jeffrey Pennington, Richard Socher, and Christoper Manning. Glove: Global vectors for word representation. volume 14, pages 1532–1543, 01 2014.

[Roo17] Robert Roosalu. Context embeddings for natural language clustering. Master's thesis, 2017.

[Rou87] Peter Rousseeuw. Rousseeuw, p.j.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. comput. appl. math. 20, 53-65. *Journal of Computational and Applied Mathematics*, 20:53–65, 11 1987.

[skla] scikit-learn documentation. RBF SVM parameters. https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html. Accessed 15.05.2019.

[sklb] scikit-learn technical documentation. SVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. Accessed 15.05.2019.

[SSM99] Bernhard Scholkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING*, pages 327–352. MIT Press, 1999.

[Sä] Dage Särg. Experiments for training an Estonian MaltParser model with reduced features. EKRK repository. https://gitlab.keeleressursid.ee/dage.sarg/syntax_experiments/. Restricted access.

[Uib18] Kristel Uiboaed. Estonian stopwords. GitHub repository. https://github.com/kristel-/estonian-stopwords/, 2018. Accessed 15.05.2019.

[VCC99] Konstantinos Veropoulos, Colin Campbell, and Nello Cristianini. Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on Artificial Intelligence, Stockholm, Sweden (IJCAI99)*, pages 55 – 60, 1999. Other: Workshop ML3.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Hele-Andra Kuulmets**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Phrase similarity measures based on Word Mover's Distance,
   *(*title of thesis)

   supervised by Sven Laur.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Hele-Andra Kuulmets
16/05/2019