Quantification of Coherence in Spoken Language as an Indicator for the Schizophrenia Spectrum Disorder

Author: Michel Doré
Supervisors: Dr. Hugo Schnack, Alban Voppel
Second evaluator: Dr. Benjamin Rin
Field: Bachelor of Artificial Intelligence, UU
15 ECTS

August 11, 2019

Utrecht University

Abstract

A low level of coherence in spoken language is one of the positive symptoms in the schizophrenia spectrum disorder. Previous studies have developed methods that can quantify the coherence level in spoken and written language. These methods are helpful for an objective analysis of discourse. In this paper two existing methods that quantify coherence in spoken language are re-implemented and applied to a new data set consisting of interviews with patients and with a healthy control group.

Besides, we present an additional method to quantify coherence of spoken language whose results differ significantly between the patients in the schizophrenia spectrum disorder and healthy controls. Furthermore, an effect size showed a medium positive correlation between the PANSS positive symptoms and the coherence level outputted by new model. Our research has been an improvement to this field due to the use of a larger data set compared to other studies. Hopefully our findings will lead to improvements trying to diagnose more objectively in the schizophrenia spectrum disorder in other studies.

Contents

1 INTRODUCTION **3**

2 METHODS **5**

2.1 Participants 5
2.2 Data (pre)processing 5
2.3 Coherence Measures 5
2.3.1 Incoherence Model 6
2.3.2 Tangentiality Model 6
2.3.3 Word Best Representing Sentence Model 6
2.4 Statistical Analysis 7

3 RESULTS **8**

4 DISCUSSION **10**

5 ACKNOWLEDGEMENTS **10**

References **10**

Appendices **12**

A Figures **12**

1 INTRODUCTION

Schizophrenia spectrum disorder is a mental disorder with drastic impact on a patients life. Total cure of the schizophrenia spectrum disorder rarely happens. However, cognitive behavior therapy and drug treatment helps to reduce the severity of psychosis, being one of the main symptoms of schizophrenia. Due to the improvements of the description of the schizophrenia spectrum disorder over time its diagnosis has become highly reliable (Tandon et al., 2013) (Association et al., 2013).

Commonly, symptoms of schizophrenia spectrum disorder are classified in 3 categories: negative, cognitive, and positive symptoms. Negative symptoms involve diminished emotional expression, avolition, and emotional withdrawal (Simpson, Kellendonk, & Kandel, 2010), (Tandon et al., 2013). Cognitive symptoms consist of working memory dysfunction and attention deficit among other things (Cirillo & Seidman, 2003). The third category, on which we focus in this paper, are the positive symptoms. The third category consists of the following symptoms: hallucinations, delusions, grandiosity, hostility, suspiciousness, excitement, and disorganized speech.

In this paper we investigate the relationship between positive symptoms in schizophrenia and the coherence of spoken language with a new set of data acquired by a research team at the University Medical Centre Utrecht. Previous research has shown that there is a significant relationship between positive symptoms in the schizophrenia spectrum disorder and the level of discourse coherence (Elvevåg, Foltz, Weinberger, & Goldberg, 2007), (Iter, Yoon, & Jurafsky, 2018). Therefore, we expect to find a significant relationship between positive symptoms and the coherence level in spoken language in this research. Furthermore the most important reason for this research is the need to find more features that indicate symptoms in the schizophrenia spectrum disorder. Reliable indicators could help professionals construct a more objective diagnosis. Even though the objectiveness of psychiatric diagnosis is similar to that of most medical specialities (Pies, 2007), the field of psychiatric diagnosis still contains subjective decisions.

Both of the cited studies ((Elvevåg et al., 2007), (Iter et al., 2018)) make use of Latent Semantic Analysis (LSA) among other techniques like machine learning. (Landauer, Foltz, & Laham, 1998) In short, "Latent Semantic Analysis (LSA) is a theory and method for extracting and representing the contextual−usage meaning of words by statistical computations applied to a large corpus of text". (Landauer & Dumais, 1997)
A technique comparable to LSA is word2vec (Goldberg & Levy, 2014) (Mikolov, Chen, Corrado, & Dean, 2013). Both techniques create vector representations of texts.
In the word2vec technique a model is trained on an existing corpus. During the training process, a two-layer neural network is trained on a given corpus that can later on be used to generate word embeddings. Text is represented as a list of vectors in word2vec, where every vector represents a single word with 300 dimensions. Based on the advise of the supervisors of this research, an existing word2vec model was used during the analysis of the supplied data.

One of the main advantages of using vector representations like word2vec is the ability to measure semantic similarity between words. Semantic similarity is of importance when quantifying discourse coherence because semantic similarity contains one of the properties which define coherence; the relationship between ideas. To clarify, in figure 1 an example is given of a sentence in which semantic similarity in combination with word2vec is used.

Furthermore, during this study we use the cosine similarity. The cosine similarity is a measure of equality between two vectors. For vectors A and B the cosine similarity is calculated as follows: the dot product of A and B is taken, which then divided by the multiplication of the Frobenius norm of each vector as shown in figure 2.
The outcome of the cosine similarity is a value between -1 and 1, with values closer to 1 meaning vectors A and B are more semantically similar and values closer to -1 meaning vectors A and B are

less semantically similar. Both vectors A and B must be nonzero vectors.

[I (0.70), am (0.71), walking (0.65), home (0.47)]

Figure 1: Example sentence word2vec usage. Every word in the sentence is paired with its distance to the mean of that sentence.

$$\text{cosine similarity} = \frac{A \cdot B}{||A||_f * ||B||_f}$$

Figure 2: The cosine similarity used to calculate the difference between two vectors, using the Frobenius norm.

The example sentence in figure 1 was generated with the word2vec model trained by Google on a data set of news articles (Mikolov, Sutskever, Chen, Corrado, & Dean, 2013). For this sentence a mean vector was calculated using the vector representation of each word. Subsequently, for each word the cosine similarity between its own vector and the mean vector of the sentence was being calculated. These are the values that are shown next to each word in figure 1. Note that the word 'home' has the least similarity with the mean vector of the sentence. Based on the data of the news based word2vec model our hypothesis therefore is that the words which lie furthest from the mean vector are of greater importance to the conversation than the other words in a sentence. This will hopefully leave us with only useful words for our coherence measure and will in its turn increase the accuracy of the coherence model.

The analysis conducted in this research is based on three previous papers and the models used in the papers that quantify coherence in spoken language (Elvevåg et al., 2007) (Bedi et al., 2015). The models used are the *Tangentiality Model* and the *Incoherence Model*. The application of those two coherence quantification models on a new data set will hopefully lead to a better understanding of those models. Furthermore, in this research a third model will be developed to further improve analysis. For equivalence and comparison reasons all three methods will be implemented and tested on the same data. All mentioned models will be explained in detail in the methods section.

We will now focus on the background of the previously developed methods.
One of the first papers concerning the use of vector-based word representations of speech produced by schizophrenia spectrum patients was written by Brita Elvevåg and her team (Elvevåg et al., 2007). In the introduction of the paper by Elvevåg et al. their definition of discourse coherence and coherence of speech is respectively defined as follows: "Discourse is perceived coherent when ideas relate to a global theme and follow a logical sequence determined by ones knowledge of the world. ... We define coherence of speech as the semantic similarity or relationship of ideas to other ideas." Especially the last definition, in which semantic similarity is used in the definition of coherence of speech, is of importance for our research. This is of importance because the representation of the word2vec model is purely semantic. The aim of the quantification of coherence of speech in this research is solely done by combining different techniques of calculating semantic similarity between ideas and other ideas. The method developed in (Elvevåg et al., 2007) paper that quantifies coherence is the *Tangentiality Model*. Another influential research that has been carried out contains the *Incoherence Model*, which is the other method to quantify coherence that is used in our research (Bedi et al., 2015). In the *Incoherence Model* the minimum coherence between two sentences in the text and the mean coherence between all sentences in the text negatively correlated with disorder level of subjects. Sentences were established with a fixed window size in the model.
Our paper also emphasizes on the fact that psychiatric diagnosis lacks objective tests which is claimed in (Bedi et al., 2015) as "routinely used in other fields of medicine".

In "Automatic detection of incoherent speech for diagnosing schizophrenia" by Iter et al. the two previously described papers (Bedi et al., 2015), (Elvevåg et al., 2007) are combined and analyzed (Iter et al., 2018). In this paper by Iter et al. the previously mentioned models were appointed their corresponding names: *Incoherence Model*, *Tangentiality Model*.
The research (Iter et al., 2018) is the main inspiration for our research since the analysis of the different used techniques and data is unique.

Due to the advent of natural language processing reliable quantification of discourse coherence became possible through the use of vector representation techniques such as Latent Semantic Analysis (LSA) and word2vec. (Landauer et al., 1998) (Mikolov, Sutskever, et al., 2013) Here, we aimed to use the natural language processing innovations to create a tool which can be used to assist with early diagnosis in the schizophrenia spectrum disorder. This will hopefully lead to an assistive tool for psychiatrists which provides as an indicator on the same level of reliability and objectivity as for example an MRI in neuroscience (Ewers et al., 2006).

Characteristics of spoken natural language like incoherence are symptoms of schizophrenia. Developments in the field of natural language processing make it possible to quantify these characteristics. We use it as a basis and aim to improve on previous research in this field in the following structure. In Section 2 the methods of the used techniques from previous research and those developed in this research are examined as well as the data which is used for conducting this research. In Section 3 the results of different statistical tests are put together and the different methods used are compared to each other. Then, in Section 4 a conclusion to the paper is drawn.

2 METHODS

2.1 Participants

The data for this paper was obtained by an ongoing research at the University Medical Centre Utrecht. In that research spoken language was acquired from an interview of a maximum of 45 minutes.
The interviewed group consisted of 50 patients diagnosed in the schizophrenia spectrum disorder by a fully qualified psychiatrist and 50 controls. In comparison, 34 participants were interviewed in (Bedi et al., 2015), 9 patients and 5 controls in (Iter et al., 2018), and 26 patients and 25 controls were used in (Elvevåg et al., 2007). The level in which the symptoms occur in each patient was estimated in the Positive and Negative Symptom Scale (PANSS) (Kay, Fiszbein, & Opler, 1987). All subjects were interviewed by trained researchers, using the same semi-structured, neutral topic interview. On average each subject used 1550.16 words in each interview with a standard deviation of 590.33 words. More information on the data can be seen in Table 1.

2.2 Data (pre)processing

For the processing of the data a 300 dimensional word2vec model was trained on a corpus of transcribed spoken Dutch language (Oostdijk, 2000). The answers from the interviews were transcribed and converted to vectors using the word2vec model. In the transcripts only the answers given by the interviewees were transcribed. The PANSS scores per patient for the positive scale and the negative scale can range from 7 to 49. For the general scale the score ranges between 16 and 112. For this research only the positive PANSS scale was investigated.
For anonymity only the vector representations of the transcribed interviews were used in this research.

2.3 Coherence Measures

In this subsection the three models quantifying coherence measures are explained. The code of these models used in this research can be found in the appendix.
Next, a few details and terms used to explain the models will be defined. This makes it easier to refer

back to these terms when explaining the models.
First, the window size. The window size is the length of the list of words on which each model calculates its coherence score. The window size is used to represent a sentence in spoken language, since no punctuation was added during the transcribing process.
In the models a moving window is applied, which means that the models calculate their coherence scores from sentence to sentence. Each model used takes a list of vectors and a chosen window size as input. This list of vectors is a list of words which are represented in the semantic space as vectors. The models each return a value in the range of -1 to 1.
We chose the window size that gave the best results in previous research, which is 8 (Elvevåg et al., 2007).

2.3.1 Incoherence Model

The first model used in this research is the *Incoherence Model* which is based on previous research (Bedi et al., 2015) (Iter et al., 2018). The model quantifies coherence by calculating the cosine similarity between each adjacent pair of sentences and then takes the minimum cosine similarity between two sentences in the interview. For each sentence in the text a mean vector is calculated and the cosine similarity between the mean vector of each adjacent pair of sentences is taken.
In this research we will take the mean of the coherence level of each document instead of the minimum level because this makes the results of the model constructed by Bedi et al. more comparable to other models used in this research.

2.3.2 Tangentiality Model

The second model used in this research is the *Tangentiality Model*. The purpose of this model is to measure how far the conversation with the interviewee has drifted off during the interview from the beginning of the interview up to the end. This measurement of "drift" has been calculated by means of the cosine similarity between the start and the end of the response of the interviewee.

The model as defined in (Elvevåg et al., 2007) was slightly altered. The reason for this minor modification is that the transcribed interviews differ in form from the transcribed interviews in the original research.

In the data of the aforementioned research the text of the interviewer and interviewee is separated: the text of the interviewee of every separate answer is being used for the analysis (Elvevåg et al., 2007). In the research in hand however the questions and answers are separated too but the answers of interviewees are presented as one full text without being set apart in different answers.

The next step in the research has been, as mentioned, the calculation of the cosine similarity between the start and the end of an interview in order to encapture a tangential value and coherence level.

This means that in our version of the model tangentiality is measured over the course of the entire interview, instead of over the course of one response. Therefore the linear regression, as used in the original method, has no use for this implementation.

2.3.3 Word Best Representing Sentence Model

For the third coherence measure a new model has been developed. This model was created to broaden the research and try another approach at quantifying the coherence in spoken language.
The core idea of the model is the semantically most valuable word of a given window. For this model there are two approaches. Both of them calculate the semantically most valuable word of a window size in a unique way and these approaches will be explained in their corresponding sections *Max* and *Min*.

2.3.3.1 Max

The first approach to calculate the semantically most valuable word is called the *Max* function. For each window in the interview the mean vector is calculated, which is the mean vector of all vectors in the given window. Then for each word in the window the cosine similarity is calculated between it and the mean vector. Subsequently the word which has the greatest distance from the mean and therefore has the lowest cosine similarity is chosen as the semantically most valuable word for each window and added to a list. Next, the cosine similarity between each adjacent pair of words in the list of semantically most valuable words is calculated and a new list is constructed with each of those cosine similarity values.
Finally the variance is calculated over the list of cosine similarities, that is the value which is returned as the coherence level by this function. The idea behind this last step is to capture the spread of coherence over all sentences.

2.3.3.2 Min

The second way of calculating the semantically most valuable word in a window can be seen as the *Min* function. What that means is the following.
First the mean vector of a window is calculated. Then, for each word in that window the distance to the mean is calculated using the cosine similarity. Just like for the *Max* version of the model. Next, the word whose semantic vector has the least distance to the mean of the window is chosen as the semantically most valuable word in this window. The last step of the calculation which includes the variance is done in the same manner as for the *Max* function.
In contrast to choosing the mean vector of a window as a semantically most valuable word this results in the exact vector of a given word in a window which has semantically more value for our calculation. So, the main difference between the two approaches is that for each window the *Min* function takes the word which is closest to the mean vector and the *Max* function takes the word which is furthest from the mean vector.

Furthermore, the reason behind using the *Max* and *Min* functions which are used in the same model but calculate the opposite of each other is that they both try to capture the subject of a window. For the *Max* function the idea is that the word that has the least semantical correspondence with the mean of the window is the word which is the most irregular in that window and therefore represents the subject of that window best. As for the *Min* function the idea is that the word which has the most correspondence with the mean of the window represents the window the best.
Both ideas seemed legitimate and where therefore implemented and compared.

Important to mention here is the fact that the two models which were implemented from other studies were not replicated exactly in our research. This was mainly due to difference of the data format. Even though these models were tweaked due to this obstacle, our effort was to preserve their main goal, namely quantifying coherence correctly.

2.4 Statistical Analysis

All statistical analysis conducted in this research was written and executed in Python 2.7.15. The code for these function can be found in the same repository as where the models are in (link to code). The statistical functions for the t-test and the linear regression were imported from existing libraries. The t-test function was imported from the scientific Python library (Jones, Oliphant, Peterson, et al., 2001–). The linear regression package was imported from the scikit-learn library (Pedregosa et al., 2011). Furthermore, the NumPy library was used for vector processing and calculations (Oliphant, 2006–).

3 RESULTS

An overview of the information on the subjects used in this research can be found in Table 1. In general the subject and control group did not significantly differ in age or sex.

	Patients (n=50)	Controls (n=50)
Men / Women	76% / 24%	84% / 16%
Age (SD)	29.2 (9.0)	31.4 (12.3)
Words used in interview (SD)	1261.1 (607.2)	1839.2 (401.5)
PANSS positive (SD)	11.3 (4.3)	

Table 1: Information on the subjects used. Gender, age, word usage including their corresponding standard deviation (SD), and PANSS positive score (not applicable to the control group)

The control group differs less in word count per interview than the patient group and the control group has a lower average word count than the patients. No positive symptom values are shown for the control group since they did not have any symptoms.

Analysis using linear regression showed no correlation between the positive symptom score and the coherence level. The linear regression was fitted to the data set of PANSS positive scores and the variance in coherence level using the Word Best Representing Sentence model using both the min and the max approach of the model as can be observed in Figure 3 and Figure 4. The linear regression resulted in a correlation coëffient of 0.02 for the model with the min function and 0.04 for the model with the max function.

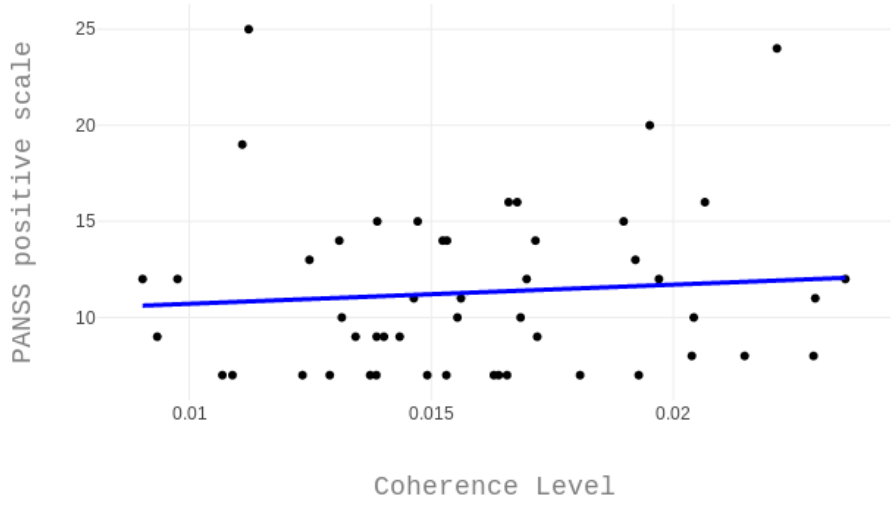

Figure 3: Linear regression line fit to the variance in coherence level and PANSS positive scale using the min function on the Word Best Representing Sentence model

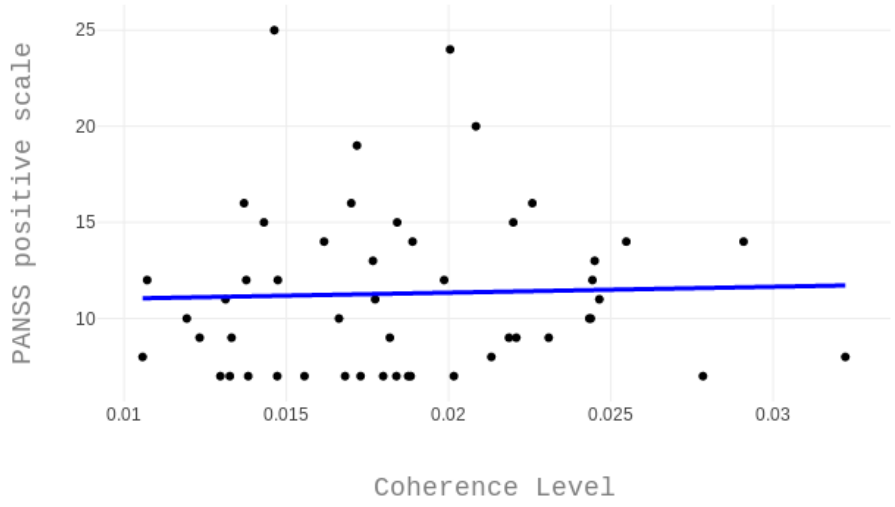

Figure 4: Linear regression line fit to the variance in coherence level and PANSS positive scale using the max function on the Word Best Representing Sentence model

Even though no correlation was found between the positive symptoms and the quantified coherence level, further analysis showed a significant difference between the patient and control group using the Word Best Representing Sentence model in with the minimum function. For this analysis an independent t-test was used as well as a Pearson's correlation coëfficient. These tests were conducted on all of the models, visible in Table 2.
The Word Best Representing Sentence model outperformed all other models used, showing a significant improved between almost all models. It furthermore showed that there is a medium positive correlation between the patient and control group for the Word Best Representing Sentence model with the min function, whereas in the other models there is no sign of a significant positive or negative correlation. Additional figures that show the difference in coherence level per model can be found in the appendix.

Model:	p value	t value	r
Incoherence Model	0.361	0.91	-0.05
Tangentiality Model	0.596	0.53	0.04
Word Best Representing Sentence (max)	0.182	1.34	0.09
Word Best Representing Sentence (min)	0.001	3.37	0.31

Table 2: Comparison of the three coherence quantification models. An independent t-test was between the control and patient group and their corresponding results from the model. P and t value are the results from the independent t-test, Pearson's r is shown in the third column, a correlation between the coherence level given by each model to the control and patient group where values can range from -1 to 1.

4 DISCUSSION

This research was performed to study the effect of the coherence level in spoken language on positive symptoms in the schizophrenia spectrum disorder. We expected to find a significant relationship between positive symptoms and the coherence level in spoken language in this research. After a thorough analysis of all data this hypothesis has been rejected; we have not represented evidence during the research to suggest that there is an existing relationship between a quantified coherence level in spoken language and the PANSS positive symptom level. This is the result of multiple statistical tests and different approaches. In contradiction to previous studies (Elvevåg et al., 2007) (Bedi et al., 2015) no connection between the coherence level in spoken language and psychiatric symptoms were found. One of the reasons for this could be the lack of a more diverse dataset. Which could mean that because of the relatively low PANSS positive average in the patient group compared to previous findings the data set did not provide a total overview of the language used by patients in the schizophrenia spectrum disorder (Kay et al., 1987).
The research though was not without any significant results; we have measured a significant difference between the coherence level of spoken language of the control group and patient group. This difference has been found using a model developed in this research; the so called Word Best Representing Sentence model.
An interesting angle that can be explored with this research is the implementation of a sliding window. In contrast to a moving window where the window moves up one window size at a time a sliding window moves up one word at a time. In addition to the sliding window, the research could be improved with the entire model defined in (Bedi et al., 2015) applied to the data set used in this research. Right now the textual features developed in (Bedi et al., 2015) are not implemented in this study due to the fact that only the vector representations were available in this research. Repeating (Bedi et al., 2015) exactly should result in comparable results to the original research.
Furthermore, the research could be repeated with a new data set. Different conversation techniques could be applied and compared like in (Elvevåg et al., 2007).
This study has re-implemented methods from previously conducted research which involve coherence quantification in relation to the schizophrenia spectrum disorder. However due to a larger data set compared to the previous studies, we thought that a stronger relationship would be the result of that larger data set, especially in comparing the different models when applying them on the same data set. The larger data set implemented on the models does not result in a stronger relationship between the coherence level and PANSS positive score but it does confirm the existing relationship found in previous studies.

5 ACKNOWLEDGEMENTS

We would like to thank the University Medical Centre Utrecht for their data and supervision during the research process, especially to A.E. Voppel and H.G. Schnack.

References

Association, A. P., et al. (2013). *Diagnostic and statistical manual of mental disorders (dsm-5®)*. American Psychiatric Pub.

Bedi, G., Carrillo, F., Cecchi, G. A., Slezak, D. F., Sigman, M., Mota, N. B., ... et al. (2015). Automated analysis of free speech predicts psychosis onset in high-risk youths. *npj Schizophrenia*, *1*(1). doi: 10.1038/npjschz.2015.30

Cirillo, M. A., & Seidman, L. J. (2003). Verbal declarative memory dysfunction in schizophrenia: from clinical assessment to genetics and brain mechanisms. *Neuropsychology review*, *13*(2), 43–77.

Elvevåg, B., Foltz, P. W., Weinberger, D. R., & Goldberg, T. E. (2007). Quantifying incoherence in speech: an automated methodology and novel application to schizophrenia. *Schizophrenia research*, *93*(1-3), 304–316.

Ewers, M., Teipel, S., Dietrich, O., Schnberg, S., Jessen, F., Heun, R., ... Hampel, H. (2006). Multicenter assessment of reliability of cranial mri. *Neurobiology of Aging*, *27*(8), 1051 - 1059. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0197458005002034` doi: https://doi.org/10.1016/j.neurobiolaging.2005.05.032

Goldberg, Y., & Levy, O. (2014). word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, *abs/1402.3722*. Retrieved from `http://arxiv.org/abs/1402.3722`

Iter, D., Yoon, J., & Jurafsky, D. (2018). Automatic detection of incoherent speech for diagnosing schizophrenia. In *Proceedings of the fifth workshop on computational linguistics and clinical psychology: From keyboard to clinic* (pp. 136–146).

Jones, E., Oliphant, T., Peterson, P., et al. (2001–). *SciPy: Open source scientific tools for Python.* Retrieved from `http://www.scipy.org/` ([Online; accessed ¡today¿])

Kay, S. R., Fiszbein, A., & Opler, L. A. (1987). The positive and negative syndrome scale (panss) for schizophrenia. *Schizophrenia bulletin*, *13*(2), 261–276.

Landauer, T. K., & Dumais, S. T. (1997). A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, *104*(2), 211.

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse processes*, *25*(2-3), 259–284.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, *abs/1301.3781*. Retrieved from `http://arxiv.org/abs/1301.3781`

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (pp. 3111–3119).

Oliphant, T. (2006–). *NumPy: A guide to NumPy.* USA: Trelgol Publishing. Retrieved from `http://www.numpy.org/` ([Online; accessed ¡today¿])

Oostdijk, N. (2000). Het corpus gesproken nederlands.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Pies, R. (2007, Oct). How "objective" are psychiatric diagnoses?: (guess again). *Psychiatry (Edgmont)*, *4*(10), 18–22.

Simpson, E. H., Kellendonk, C., & Kandel, E. (2010). A possible role for the striatum in the pathogenesis of the cognitive symptoms of schizophrenia. *Neuron*, *65*(5), 585596. doi: 10.1016/j.neuron.2010.02.014

Tandon, R., Gaebel, W., Barch, D. M., Bustillo, J., Gur, R. E., Heckers, S., ... Carpenter, W. (2013). Definition and description of schizophrenia in the dsm-5. *Schizophrenia Research*, *150*(1), 3 - 10. Retrieved from `http://www.sciencedirect.com/science/article/pii/S0920996413002831` (DSM-5) doi: https://doi.org/10.1016/j.schres.2013.05.028

Appendices

A Figures

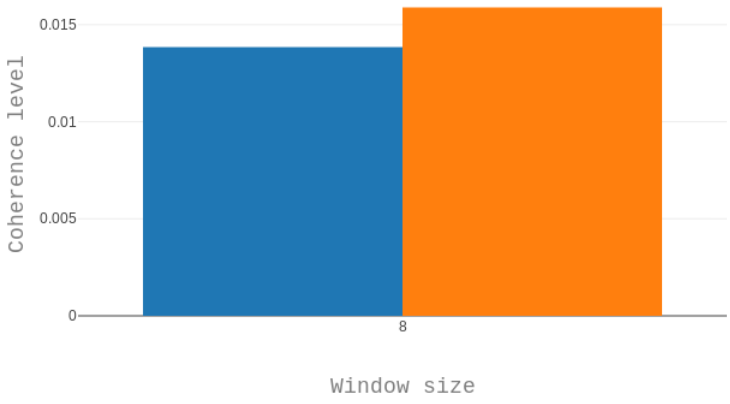

Figure 5: Variance in coherence level for a window size of 8 using the Word Best Representing Sentence model with the min function. Orange represents the patient group, blue represents the control group.

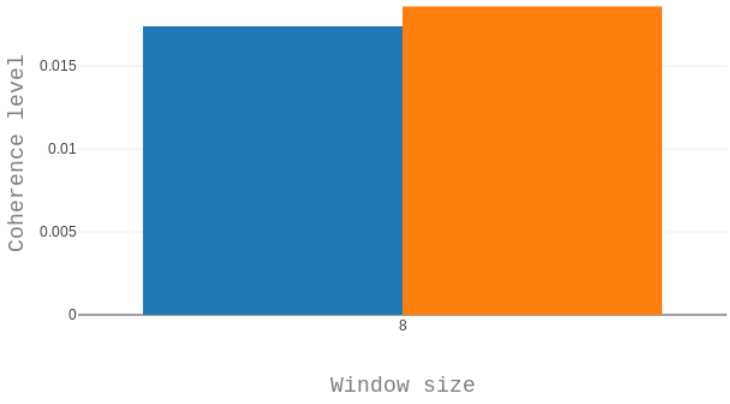

Figure 6: Variance in coherence level for a window size of 8 using the Word Best Representing Sentence model with the max function. Orange represents the patient group, blue represents the control group.

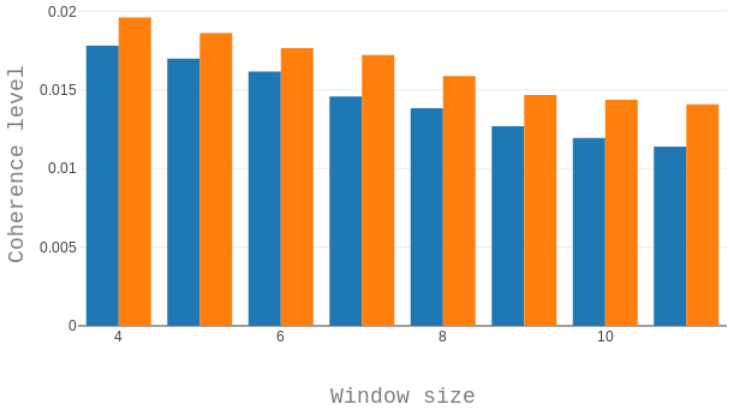

Figure 7: Coherence level per window size using the Word Best Representing Sentence model with the min function. Orange represents the patient group, blue represents the control group.

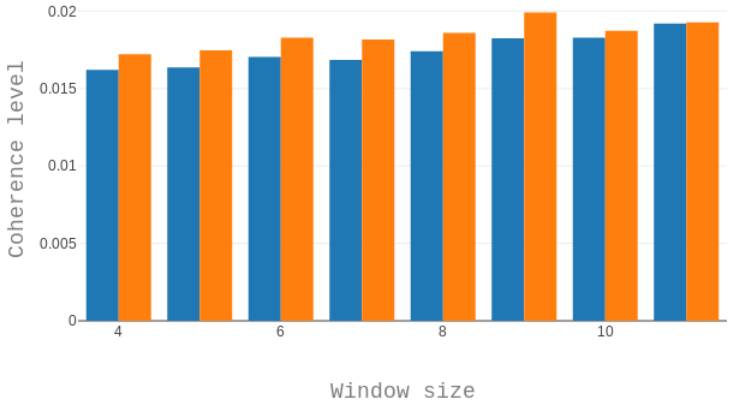

Figure 8: Coherence level per window size using the Word Best Representing Sentence model with the max function. Orange represents the patient group, blue represents the control group.

Listing 1: Coherence Quantification Functions

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 27 15:20:20 2019
@author: michel
"""

#import requirements
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import logging
import random
logging.basicConfig()
import numpy as np
from IPython import embed #to be able to use embed() for troubleshooting

#functions
def cosine_similarity_vector (word1,word2):
    """
    requires 2 numpy vectors from the model; then gives back the cosine similarity
    """

    value = np.dot(word1, word2)/(np.linalg.norm(word1)* np.linalg.norm(word2))
    return value

def first_order_coherence (vectors, window_size=8):
    if len(vectors) < (window_size*2):
        return 0

    result = np.mean([cosine_similarity_vector(np.mean(vectors[i*window_size:i*window_size +
        window_size], axis=0),
                                               np.mean(vectors[i*window_size + window_size:i*
                                                   window_size + 2*window_size], axis=0))
                                               for i in range(0, int(len(vectors)/(window_size*2)))
                                                   ])
    return result

def tangentiality_model(vectors, window_size=8):
    return cosine_similarity_vector(np.mean(vectors[:window_size], axis=0), np.mean(vectors[-
        window_size:], axis=0))

""" Function which returns the centroid of a group of vectors """
def centroid(vectors):
    return np.mean(vectors, axis=0)
```

```
""" Function which returns the nearest centroid according to the
    cosine similarity given a vector and a list of centroids.
"""
def nearest_centroid(vector, centroids):
    min_centroid_distance = -1
    min_centroid = None
    for centroid in centroids:
        if cosine_similarity_vector (centroid, vector) > min_centroid_distance:
            min_centroid_distance = cosine_similarity_vector (centroid, vector)
            min_centroid = centroid
    return min_centroid

""" The k-means clustering algorithm. The algorithm requires at least a list of lists (vectors) as
     input.
    Another setting can be passed in the other inputs of the method, which is the amount of
        clusters 'k',
    set to 7 as default.
"""
def k_means_clustering(vectors, centroids = [], k=7, first_run = True, difference = 0):
    # If the algorithm is executed for the first time, then the centroids are chosen randomly
    if first_run : centroids = [[-0.5+random.random() for i in range(300)] for c in range(k)]

    # Initiate the dictionary for the division of vectors so they can be grouped by centroid
    centroids_dict = {}
    for c in centroids:
        centroids_dict [str(c)] = []

    for vector in vectors:
        centroids_dict [str(nearest_centroid(vector, centroids))].append(vector)

    new_centroids = [centroid( centroids_dict [vs]) for vs in centroids_dict .keys()]

    # The differences between every corresponding centroid of the previous run is calculated and
        the
    # mean of those values is used as a comparison
    diff = np.mean([cosine_similarity_vector(new_centroids[i], centroids[i]) for i in range(k)])

    if(abs(difference - diff) <= 0.02 and diff > 0.8):
        k_means_clustering(vectors, new_centroids, first_run =False, difference = diff)
    else:
        return centroids_dict

def max_words(vectors, window_size=8):
    if len(vectors)<window_size: return []
    words = []
    max_word = []
    distance = 1
    for i in range(0, int(len(vectors)/window_size)):
```

```
        sentence = vectors[i * window_size:(i * window_size) + window_size]
        summary = np.mean(sentence, axis=0)
        for word in sentence:
            temp_distance = cosine_similarity_vector(word, summary)
            if temp_distance < distance:
                distance = temp_distance
                max_word = word
        words.append(max_word)
        max_word = []
        distance = 1
    return words

def median_words(vectors, window_size=8):
    if len(vectors)<window_size: return []
    words = []
    max_word = []
    distance = -1
    for i in range(0, int(len(vectors)/window_size)):
        sentence = vectors[i * window_size:(i * window_size) + window_size]
        summary = np.mean(sentence, axis=0)
        for word in sentence:
            temp_distance = cosine_similarity_vector(word, summary)
            if temp_distance > distance:
                distance = temp_distance
                max_word = word
        words.append(max_word)
        max_word = []
        distance = -1
    return words

def coherence_measure(vectors, window_function, window_size):
    words = window_function(vectors, window_size)
    distance_words = []
    for i in range(len(words)-1):
        distance_words.append(cosine_similarity_vector(words[i], words[i+1]))
    return np.var(distance_words)
```

Listing 2: Statistical tests in Python

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*-
"""
Created on Wed Feb 13 14:24:21 2019
@author: alban, michel
"""

# import requirements
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import logging

logging.basicConfig()
import numpy as np
from IPython import embed # to be able to use embed() for troubleshooting
import os, glob
import csv
import pickle
import coherence
import figure
import pandas as pd

def print_all():
    print("Max results: ")
    run_tests("max")
    print("Median results: ")
    run_tests("median")

def run_tests(word_function, ws=8):
    if word_function=="max":
        func = coherence.max_words
    else:
        func = coherence.median_words

    # %% running the function on all the pd4 files
    subjects = {}
    for filename in glob.glob("data/*.pd4"): # select all vectorized pickles
        with open(filename, 'rb') as fp:  # load in .pos file
            subjects[filename[5:-7]] = pickle.load(fp)  # save as a vector
    legenda = pd.read_csv("data_legenda.csv") # information on each subject


    control = {}
    control["first_order"] = []
    control["tangentiality"] = []
```

```
control["most_significant_word"] = []
psychose = {}
psychose[" first_order "] = []
psychose[" tangentiality"] = []
psychose["most_significant_word"] = []
window_sizes = []

window_sizes_most_significant_word_control = []
window_sizes_most_significant_word_psychose = []

pvalues = {}
pvalues[" first_order "] = 0.
pvalues[" tangentiality"] = 0.
pvalues["most_significant_word"] = 0.
tvalues = {}
tvalues[" first_order "] = 0.
tvalues[" tangentiality"] = 0.
tvalues["most_significant_word"] = 0.
effect_size = {}
effect_size [" first_order "] = 0.
effect_size [" tangentiality"] = 0.
effect_size ["most_significant_word"] = 0.

most_significant_word_scores = []
PANSS_positive_scores = []


# For each subject the scores for all the statistical tests are saved in their corresponding
    dictionaries
for i in range(len(legenda)):
    most_significant_word_score = coherence.coherence_measure(subjects[legenda["subject"][i ]],
        func,
                                                              window_size=ws)
    if legenda["Groep"][i] == "Controle":
        control[" first_order "].append(
            coherence. first_order_coherence (subjects[legenda["subject"][ i ]], window_size=ws))
        control[" tangentiality "].append(
            coherence.tangentiality_model(subjects[legenda["subject"][ i ]], window_size=ws))
        control["most_significant_word"].append(most_significant_word_score)
    else:
        psychose[" first_order "].append(
            coherence. first_order_coherence (subjects[legenda["subject"][ i ]], window_size=ws))
        psychose[" tangentiality "].append(
            coherence.tangentiality_model(subjects[legenda["subject"][ i ]], window_size=ws))
        psychose["most_significant_word"].append(most_significant_word_score)

    if not np.isnan(legenda["PANSS_positive"][i]): # Check whether the PANSS score exists
        most_significant_word_scores .append(most_significant_word_score)
```

```
        PANSS_positive_scores.append(legenda["PANSS_positive"][i])

from scipy.stats import ttest_ind, pearsonr
tvalues["first_order"], pvalues["first_order"] = ttest_ind(control["first_order"], psychose["
    first_order"])
tvalues["tangentiality"], pvalues["tangentiality"] = ttest_ind(control["tangentiality"],
    psychose["tangentiality"])
tvalues["most_significant_word"], pvalues["most_significant_word"] = \
    ttest_ind(psychose["most_significant_word"], control["most_significant_word"])
effect_size["first_order"], _ = pearsonr(control["first_order"], psychose["first_order"])
effect_size["tangentiality"], _ = pearsonr(control["tangentiality"], psychose["tangentiality"])
effect_size["most_significant_word"], _ = pearsonr(control["most_significant_word"], psychose["
    most_significant_word"])

window_sizes.append(ws)

window_sizes_most_significant_word_control.append(control["most_significant_word"])
window_sizes_most_significant_word_psychose.append(psychose["most_significant_word"])
control["most_significant_word"] = []
psychose["most_significant_word"] = []

# Coherence graph
figure.multiple_bar_plot(window_sizes, [np.mean(x) for x in
    window_sizes_most_significant_word_control], "Control", window_sizes,
                        [np.mean(x) for x in window_sizes_most_significant_word_psychose], "
                            Psychose",
                        "Window size", "Coherence level","Coherence Graph "+word_function
                            +".html")

# Coherence level and PANSS positive scale + linear regression figure:
from sklearn import datasets, linear_model
regression_X = np.array(most_significant_word_scores)[:, None]
regression_Y = np.array(PANSS_positive_scores)

model = linear_model.LinearRegression()
model.fit(regression_X, regression_Y)

from scipy.stats import pearsonr
pcor, _ = pearsonr(most_significant_word_scores, PANSS_positive_scores)
print("Pearson's correlation: " + str(pcor))

figure.linear_regression_plot(regression_X, regression_Y, model, x_title="Coherence Level",
                            y_title="PANSS positive scale", filename="Linear Regression
                                Graph "+word_function+".html")

# T-Test and P values:
for (keyP, valueP), (keyT, valueT), (keyE, valueE) in zip(pvalues.items(), tvalues.items(),
    effect_size.items()):
```

```
        print("+-----------------------------+")
        print(keyP)
        print("+-----------------------------+")
        print("P-value: " + str(valueP))
        print("T-value: " + str(valueT))
        print("Effect size (Pearson): " + str(valueE))
    print("+-----------------------------+")
```

Listing 3: Code to generate figures

```
import plotly as py
import plotly.graph_objs as go


def bar_plot(x, y, x_title, y_title, title, file):
    data = [go.Bar(
        x=x,
        y=y
    )]

    layout = go.Layout(
        annotations=[
            dict(
                x=0.5004254919715793,
                y=-0.16191064079952971,
                showarrow=False,
                text=x_title,
                xref='paper',
                yref='paper'
            ),
            dict(
                x=-0.04944728761514841,
                y=0.4714285714285711,
                showarrow=False,
                text=y_title,
                textangle=-90,
                xref='paper',
                yref='paper'
            )
        ],
        autosize=True,
        margin=dict(
            b=100
        ),
        title=title,
        xaxis=dict(
            autorange=True,
            type='linear'
        ),showlegend=False
```

```
    )

    fig = go.Figure(data=data, layout=layout)
    py.offline.plot(fig, filename=file)

def multiple_bar_plot(x1,y1,n1,x2,y2,n2, x_title , y_title ,filename):
    trace1 = go.Bar(
        x=x1,
        y=y1,
        name=n1
    )
    trace2 = go.Bar(
        x=x2,
        y=y2,
        name=n2
    )

    data = [trace1, trace2]
    layout = go.Layout(
        barmode='group',
        xaxis = go.layout.XAxis(
            title =go.layout.xaxis.Title(
                text=x_title,
                font=dict(
                    family='Courier New, monospace',
                    size=18,
                    color='#7f7f7f'
                )
            )
        ),
                yaxis = go.layout.YAxis(
            title =go.layout.yaxis.Title(
                text=y_title,
                font=dict(
                    family='Courier New, monospace',
                    size=18,
                    color='#7f7f7f'
                )
            )
        ),showlegend=False
    )

    fig = go.Figure(data=data, layout=layout)
    py.offline.plot(fig, filename=filename)

# Code obtained from: https://plot.ly/scikit-learn/plot-ols/
def linear_regression_plot (X_test, Y_test, model, x_title , y_title , filename):
    def data_to_plotly(x):
```

```
    k = []

    for i in range(0, len(x)):
        k.append(x[i][0])

    return k

p1 = go.Scatter(x=data_to_plotly(X_test),
                y=Y_test,
                mode='markers',
                marker=dict(color='black')
                )

p2 = go.Scatter(x=data_to_plotly(X_test),
                y=model.predict(X_test),
                mode='lines',
                line=dict(color='blue', width=3)
                )

layout = go.Layout(xaxis = go.layout.XAxis(
        title=go.layout.xaxis.Title(
            text=x_title,
            font=dict(
                family='Courier New, monospace',
                size=18,
                color='#7f7f7f'
            )
        )
    ),
            yaxis = go.layout.YAxis(
        title=go.layout.yaxis.Title(
            text=y_title,
            font=dict(
                family='Courier New, monospace',
                size=18,
                color='#7f7f7f'
            )
        )
    ),
    showlegend=False, hovermode='closest')

fig = go.Figure(data=[p1, p2], layout=layout)

py.offline.plot(fig, filename=filename)
```