NLP-based Mind Map Generation to Support Brainstorming Sessions

Stefan Hanisch

Technical University Carolo-Wilhelmina, Brunswick, Germany
stefan.hanisch@tu-bs.de

Abstract. Intelligent computer systems increasingly take on more tasks of humans, even in the context of creativity techniques. After an overview of current research approaches of support potentials of electronic creativity techniques and virtual moderators, this paper introduces natural language processing techniques to analyze brainstorming sessions. On this foundation, an artifact is developed that is able to generate mind maps in virtually real-time during electronic brainstorming sessions. The objective is to show mentioned terms to motivate the participants to continue own or foreign ideas and make user behavior visible.

Keywords: Brainstorming, Mind map, Natural Language Processing, Visualization

1 Introduction

Creativity techniques are an essential tool for the generation of new ideas, especially in the professional environment to generate new "products, processes and services" [25]. One common used technique is the collaborative method brainstorming [26]. The basic principle is to contribute as many (even unusual) ideas as possible to a given problem in a group project, without judging each other and the desirability of picking up and continuing foreign ideas [18].

Strohmann et al. deal in [26] with a support of this creativity technique by intelligent computer systems. They mention the use case of virtual teams, that are able to collaborate over multimedial systems, which additionally facilitates the generation of new ideas [12, 26]. Brainstorming benefits from an electronic support through a higher satisfaction of the participants [25, 10]. One advantage is that participants can express their ideas at any time and simultaneously on an electronic platform, in contrast to conventional methods without electronic support [19].

A challenge in brainstorming is the need for a moderator, that ensures the compliance of the principles [26]. This paper introduces related work of a digital implementation of moderator-like capabilities, responds to the need for visualization techniques and describes possibilities of natural language processing (NLP) techniques to develop an artifact that is evaluated in experiments to analyze the support potentials.

1.1 Related Work

A current research topic is the support of brainstorming sessions by a virtual moderator, who monitors the observance of the principles mentioned above and is able to actively support the generation of new ideas. In [25], Strohmann has developed guidelines for the functions that such a virtual moderator should have by conducting qualitative interviews with experts. This system should, among others, be able to understand natural language, recognize the context of the conversation (such as stagnation, criticism or deviation from the topic) and provide visualization techniques. Such a system was implemented in the brAInstorm project in [26], that allows the input and discussion of ideas and implements functionality of a virtual moderator by checking for unsuitable phrases and intervening "when the group drifts away from the topic or the group stagnates".

1.2 Problem Identification

As mentioned in the brainstorming guidelines, the continuation of foreign ideas is desirable. The evaluation of natural language, even in a written form, is a current research topic and was mentioned in the guidelines for the support of creativity techniques, in conjunction with documentation und visualization capabilities. [25]

The problem to be solved in this paper is the design and development of an artifact, that is able combine these two aspects by a visualization system that show the already mentioned ideas to motivate the brainstorming participants to continue their own or foreign ideas.

1.3 Research method

The research approach is based on the design science research method, introduced by Peffers et al. in [20]. The aim is to develop an artifact that solves the problem described above. Therefore, an objective is defined which is the basis for the artifact design and development. The artifact is afterwards evaluated with respect to the problem solving capability. In this case, laboratory experiments are executed to evaluate the support potentials of the system. Due to the interactive elements, a demonstration of the artifact is only shown in individual excerpts in the development section.

1.4 Objectives of a Solution

The objective is an online communication (chat) system, that provides an appropriate visualization technique for computer-based brainstorming sessions. For this, a mind map is used, which creates association lines between thematically related terms [6, 8]. Also, the related terms should be plotted close to each other and not thematically related terms should be far apart. This is called the cluster hypothesis [15].

Further guidelines by Strohmann [25] on the handling of virtual moderators should also be observed. The paper deals with the user point of view, with expectations and fears. One of the respondents mentioned the fear of being influenced by intelligent computer systems. Therefore, the system that is developed in this paper should be as simple and transparent as possible. In order to minimize any influence, more words should be displayed in the mind map than too few, so that the system does not make a selection of important words by itself. Also the ease of use was noted and the expectation that the underlying technique should not be visible and operate in the background to ensure a comfortable atmosphere that can reduce "stress to improve process success and perceived satisfaction". [25, 10]

2 Design and Development

This chapter describes an implementation of a system for the visualization of brainstorming sessions on the basis of NLP and apply techniques of search engines, that deal with related topics for the current task. Based on the objectives, the system should operate in the background and the operations should not be visible to the user. The only point of contact between the user and the system is the output in the form of a mind map. Therefore, a system architecture is designed, that uses a front-end web platform, operate on a server and not on the user devices, and transfer the resulting mind map in the end back to the user.

2.1 Theoretical Background

The statements of the user are, in their raw form, a kind of unstructured data, "that does not have clear, semantically overt, easy-for-a-computer structure". The subarea *information retrieval* (IR) of computer science deals with the task of transforming raw unstructered text data into searchable data structures to find "material (usually documents) [...] that satisfies an information need from within large collections (usually stored on computers)". [15]

In IR, users formulate a query with respect to their information need to find documents, which terms are similar to their query in a semantic matter. This techniques can be transferred to the current research question, to transform user statements into machine-processable structures. The statements users make during a brainstorming session can be considered as documents with attributes like user name, time stamp and the content.

The following sections explain these procedures to extract semantic interrelationships of statements in detail.

Text Preparation To evaluate a sequence of words in text data, the individual terms must be determined. This section explain the processing of words in the user statements to create the basis for extracting semantic information. Table 1 provides an overview of the explained processing steps in the following sections.

Table 1. Fundamental aspects to prepare text for further analysis [15]

Technique	**Description**
Tokenization	Extract of coherent characters and the corresponding challenges.
Stop words	The frequency of terms can give indications about the importance of a word.
Normalization	Unification of the syntax of terms and deals with grammatical extensions.
Stemming and Lemmatization	Unification of different forms (tenses or numerous) of a word.
Vocabulary	The vocabulary is the set of all underlying terms. This is growing rapidly and can be a limiting factor.

Tokenization This technique splits the text into a token for each word or several coherent words. A token is defined as "an instance of a sequence of characters in some particular document that are grouped together as a useful semantic unit for processing" and represents a single or multiple words in the original text, e.g. with respect to the selected tense by the author, or the numerus. Tokens are the basis to derive terms, which can be used for finding semantic relations in further processing steps. [15]

The result of the tokenization process is visualized in Figure 1.

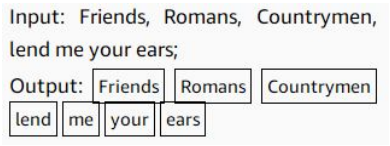

Fig. 1. Visualization of tokenization [15]

The generalization that all words, that are separated by whitespaces in the original text, corresponds to a token is not permissible. For example, the city name *New York* would loose its information value if both words are treated seperately and independently. To deal with city names, each user statement could be checked against pre-collected database tables of city names, but phrases like *New York library* do not give any information whether the library in New York is mentioned, or the new library in York. [15]

Another challenge in tokenization are compound words and their different spellings. For example, the words "lowercase", "lower-case" and "lower case" could have the same meaning. Other languages like German have their own difficulties for the task of tokenization by not using whitespaces in compound words, e.g. "Lebensversicherungsgesellschaftsangestellter" ("life insurance com-

pany employee") and contain several informations at once. The Chinese language does not use whitespaces between words or sentences at all. Also different meanings are possible depending on the context, e.g. "two [Chinese] characters can be treated as one word meaning 'monk' or as a sequence of two words meaning 'and' and 'still' ". [15]

Stop Words During processing words into tokens, not all tokens have the same information gain for the following analysis processes. Tokens, that are very common and present in nearly all sentences, like "a", "an", "and", "are", "as", "be", "to" etc. are called "stop words" and are viewed less descriptive for a user and the corresponding statement, in contrast to rare used words, like technical terms. An automatable method to determine stop words is the computation of the "collection frequency" for each term, which counts the existence for a term in the whole corpus. The smaller the "collection frequency", the more expressive is a term. [15]

If stop words are removed, the set of terms will be smaller, which leads to shorter computation times and the users statements could be evaluated with a more precise focus, which is important in further steps. The presence and the quantity of stop words should not influence the meaning of user statements.

On the other hand, statements whose contents are "stop words" lose their information, e.g. the phrase "to be or not to be". [15]

Normalization To map several tokens (different numerous, tenses, spellings, etc.) to one term, normalization steps are introduced. The most common technique is the creation of "*equivalence classes*, which are normally named after one member of the set". To return to the example mentioned above, "lower-case" and "lowercase" are member of the same equivalence class. Further normalization techniques are the processing of accents and diacritics. In languages, in which accents and diacritics have a small influence to the semantics, both can be removed from the token, e.g. "naive" and "naïve" map to the same equivalence class. In other languages, like Spanish, diacritics are more important and this generalization is not feasible, e.g. "*peña*" (cliff) is not similar to "*pena*" (sorrow). To deal with capitalization, all tokens can be mapped to a unified version, e.g. to lowercase. This technique has the disadvantage, that it is also not lossless, for instance the former president Bush is mapped to bush. [15]

Stemming and Lemmatization The stemming and lemmatization process goes beyond normalization and performs mapping to uniform word strains. For instance, the tokens "democracy", "democratic" and "democratization" have related meanings. [15] With respect to the current research task, if users use similar, disjoint terms of the same topic, the thematic proximity can be detected.

Stemming is a heuristic set of rules "that chops off the ends of words" to map different tokens to a word stem, but is not lossless, as shown in Figure 2. Lemmatization takes into account the morphological concept of a language and maps "am", "are" or "is" to the verb "be". But also lemmatization is not always

lossless, for instance if the token "saw" is used. This can be the past tense of "see" or the tool "saw". [15]

Rule			Example		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

Fig. 2. Application of a stemmer [15]

Improvements are possible by hand-crafted mapping structures, that consider synonyms, like "automobile" and "car". [15]

Vocabulary The complete set of terms in a corpus is called the vocabulary. The size is smaller as the amount of tokens (because several tokens are mapped to one term), but the size can have a significant impact on computing time. For this purpose, Manning et al. name various estimations in [15]:

- The Oxford Dictionary names more than 600,000 words for the English language, without consideration of persons, places or products.
- Heaps law: Estimation of the vocabulary size M for a given number of tokens k in a collection, with the parameters k, usually in the range of 30 to 100 and b approximately to 0.5: $M = kT^b$. This represents a linear relationship between the vocabulary size and the number of tokens.

Representation of Statements Based on the preprocessing steps mentioned above, each user statement consists of terms, that are derived from the raw token form. In the following steps, the *bag of words model* is introduced and transferred to *term-statement-matrices*.

Bag of Words Model This model represent texts by the contained terms with respect to the number of occurences, but regardless of the order. [15]

For example, the terms of the three statements result in the following term-statement-matrix (the literature speaks of term-document matrices in IR):

Statement 1: "car"; "automobile"; "traffic"; "road"
Statement 2: "car"; "car"; "manufacturer"; "traffic"; "sign"
Statement 3: "airplane"; "airport"

	Statement 1	*Statement* 2	*Statement* 3
car	1	2	0
$automobile$	1	0	0
$traffic$	1	1	0
$road$	1	0	0
$manufacturer$	0	1	0
$sign$	0	1	0
$airplane$	0	0	1
$airport$	0	0	1

If each statement come from different persons, the columns show who mentioned which terms and in the rows it can be seen which terms often occur together (the so called *term co-occurrences* [15]).

It can be seen that the authors of the first two statements mention terms that occur together, while the author of statement three uses terms that have not been mentioned by any other user. It suggests that the users one and two discuss the same topic, while user three mentions other topics.

Term Weights IR makes the assumptions for user queries to search engines, that the higher the frequency of query terms in a specific document, the more relevant is it for the information need of the user. Common metrics are the *term frequency* (tf), that measure the occurences of a term t in a given document d, the *document frequency* (df), which is "defined to be the number of documents in the collection [...] contain a term t". The metric *collection frequency* (cf) measures "the total number of occurences of a term in the collection" and can be used to normalize df values.[15]

This allows the computation of weighting factors, like the *inverse document frequency* (idf), that is defined as the logarithmig inverse of the df for a given term t, scaled by the size of the collection N. Figure 3 shows the relationship of df_t and idf_t values. [15]

$$idf_t = log \frac{N}{df_t} \quad [15]$$

term	df_t	idf_t
car	18,165	1.65
auto	6723	2.08
insurance	19,241	1.62
best	25,235	1.5

Fig. 3. df_t and idf_t values for the Reuters corpus. [15]

Based on the inverse document frequency, the tf-idf value is a common used weighting factor, that does not only consider the binary occurrence (whether

a term occurs in a document or not, as in the *idf*-value), but also deals with multiple occurrences (tf) and is defined as

$$tf\text{-}idf_{t,d} = tf_{t,d} * idt_t \quad [15]$$

with respect to a given document d. This has the consequence, that the term weight is high for term t if "t occurs many times within a small number of documents" and "lower when the term occurs fever times in a document, or occurs in many documents" and is "lowest when the term occurs in virtually all documents". [15]

This technique was mentioned in abstract form on the stop word classification in the text preprocessing step.

Transferred to the task in this paper, the entries in the term-statement-matrix can also be weighted by the term frequency. If a participant uses very rare terms, they are characterizing this person in a more detailed way.

Distance Measurement Each statement of the term-statement matrix can be considered as a vector in a *vector space model*, with an entry for each term dimension. This allows the computation of similarities by the *cosine distance* [15]. This distance metric, for the statements s_1 and s_2, is defined as:

$$sim(s_1, s_2) = \frac{s_1 * s_2}{||s_1||\ ||s_2||} = \frac{\sum_{i=1}^{n} w_{i,s_1}\ w_{i,s_2}}{\sqrt{\sum_{i=1}^{n} w_{i,s_1}^2}\ \sqrt{\sum_{i=1}^{n} w_{i,s_2}^2}} \quad [11]$$

Due to possible multiple occurrences, often used distance metrics like the Minkowski distance are not feasible in this context, because scaling factors influence the similarity. For example, if two statements s_1 and s_2 contain the same set of terms, but s_2 has for each term twice the term frequency (s_2 is a linear transformation of s_1), the similarity would be greater than zero. The cosine distance shown above normalizes each vector by its length and is therefore invariant to linear combinations.

The following paragraphs for the person and term similarity are based on the term-statement-matrix A.

Person Similarity The $A^T A$ matrix delivers in the "non-diagonal cells document-document cosine similarities" [11] which are in this context the statement similarities.

The example of the bag of words model with the corresponding matrix results in a statement similarity (and therefore in a person similarity, if each statement was mentioned by distinct individuals):

	Statement 1	*Statement* 2	*Statement* 3
Statement 1	1	0.56	0
Statement 2	0.56	1	0
Statement 3	0	0	1

Each statement vector was normalized by its length. The resulting matrix show, that each statement is similar to itself, and that the first two statements are similar to each other and dissimilar to the third statement.

Garcia presents in [11] further examples and calculations.

Term Similarity The similarity of terms can be computed by AA^T [15]. For the exemplary term-statement-matrix from the bag of words model, the term similarities (based on normalized vectors) are:

	car	*automobile*	$traffic$	*road*	$manufact.$	*sign*	*airplane*	*airport*
car	1	0.45	0.95	0.45	0.90	0.90	0	0
automobile	0.45	1	0.71	1	0	0	0	0
$traffic$	0.95	0.71	1	0.71	0.71	0.71	0	0
road	0.45	1	0.71	1	0	0	0	0
$manufact.$	0.90	0	0.71	0	1	1	0	0
sign	0.90	0	0.71	0	1	1	0	0
airplane	0	0	0	0	0	0	1	1
airport	0	0	0	0	0	0	1	1

Each term is completely similar to itself and the co-occurrences can be derived from the statements. This provides an overview of which terms often occur together and are therefore related to each other.

Word Embeddings In IR, term-document-matrices have a small number of nonzero entries and therefore are very sparse. Apart from stop words, many terms do not occur together in linguistic usage and produce many zero entries. Manning et al. provides an example of a search engine, that indexed 1 million documents, each with a length of one thousand words. This typically results in a 500,000 x 1,000,000 term-document-matrix, with a minimum of 99.8% zero entries. If the term similarities are precomputed on a large text corpus, the size of the matrix grows rapidly and no longer fits into the main memory. [15]

Word embeddings are a machine-learning-based technique to transform word vectors from the sparse term-statement-matrix into a dense representation of 50 to 300 dimensions [7]. They arrange word vectors in a way, that similar words are grouped together. Many of these embeddings additionally allow linear algebra computations, by calculating positions of words [17].

For example, the term "Madrid" relates to the term "Spain" like "Paris" to "France". The query for this example looks like "vec('Madrid') - vec('Spain') + vec('France')" and the result "is closer to vec('Paris') than to any other word vector". [17, 16]

Visualization To visualize the words with respect to their corresponding vector, dimensionality reduction techniques are necessary, that project the vectors onto two dimensions.

Latent Semantic Analysis (LSA) and Principal Component Analysis (PCA) These techniques based on a *singular value decomposition* (SVD), that decompose a given M x N term-document-matrix C into three matrices, so that the equation

$$C = U\Sigma V^T \quad [15]$$

is fulfilled, so that U "be the M x M matrix whose columns are the orthogonal eigenvectors of CC^T, and V be the N x N matrix whose columns are the orthogonal eigenvectors of C^TC. [15]

Both matrices are mentioned in the *distance measurement* chapter, that result in the term-co-occurrence-matrix and the statement similarities.

The matrix Σ is a r x r matrix with r as "the rank of the M x N matrix C" and has "the singular values on the diagonals". [15]

The SVD is the basis for the LSA, which is characterized by a *low rank approximation*, that approximates a matrix C_k of rank k with a "value of k that is far smaller than the original rank of C" and map C "to a k-dimensional space [...] defined by the k principal eigenvectors (corresponding to the largest eigenvalues)". [15]

The PCA is a generalized approach of the LSA. Gewers et al. provide a detailled derivation in [13]. In contrast to LSA, PCA "center the data before computing the singular value decomposition". [21]

T-distributed Stochastic Neighbor Embedding (t-SNE) Van der Maaten and Hinton introduce in [27] the non-linear dimensionality reduction t-SNE. Arora et al. discuss in [1], that linear techniques like PCA "are incapable of reducing dimension down to 2 in any meaningful way". Therefore, t-SNE uses non-convex optimization in conjunction with gradient descent and uses "two similarity measures between pairs of points - one for the high dimensional data and one for the 2-dimensional embedding". The 2-dimensional embedding preserve the interdependencies, to that the Kullback-Leibler divergence is minimized. [1] The Kullback-Leibler divergence is "a measure of the similarity of two known distributions" [5].

2.2 Own Implementation

This section introduces the distributed system, that implements the techniques mentioned above to evaluate user statements during brainstorming sessions to generate the mind map and plot the user positions. To obtain the user statements, the Slack application programming interface (API) is implemented, but the system has a generic structure and is easily adaptable to other interfaces.

For the NLP and data science tasks, the open source python frameworks spaCy[1] and scikit-learn[2] are introduced. The graphic representation is carried out by the python module matplotlib[3].

System Architecture The complete system consists of the components described in the following enumartion. The order shows the program sequence. Afterwards, the requirements are named.

[1] https://spacy.io
[2] http://scikit-learn.org
[3] https://matplotlib.org

Slack user interface The user interface is the environment, in which the user interact during the brainstorming session. In this environment, users can communicate with each other undisturbed. Figure 4 shows the appearance.

Slack API The Slack API allows the integration of own application and bot users. This API is used to retrieve the messages and transfer them to the database.

SendToDB module This self-written module implements the API calls, retrieve the messages in real-time and transfer them to the database.

Database system The database server is a SQL-based (*structured query language*) database, that contains one table, that collects the user ID, the corresponding channel, the statement and a time stamp.

EvaluateFromDB module This self-written module continuously retrieves all statements for a given channel from the database server and creates the mind map image.

Optional: Web server integration An optional web server can be used to upload the generated mind map image via the *file transfer protocol* (FTP) to make the mind map accessable to other participants over the internet.

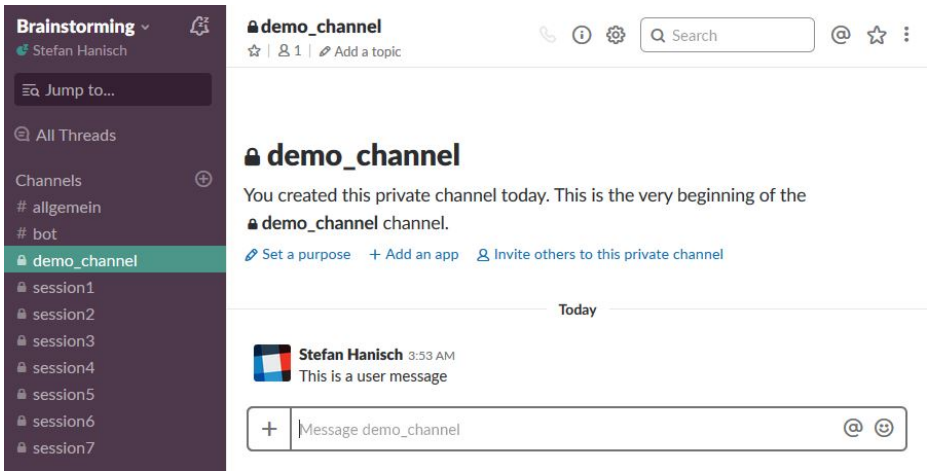

Fig. 4. Illustration of the Slack environment from the user's point of view

Requirements

Back-end server This server must be in operation when user write messages. Otherwise, the system is not able to transfer the statements to the database. This applies to the SendToDB script and the database. The evaluation script can be called as required.

Database server The implementation is designed for relational database systems. Other databases are possible, but require an additional adaptation. Queries to the database include the last message ID and a channel ID and retrieve the messages for the complete channel. This allows the query to be optimized with corresponding indexes with little effort.

Slack API The Slack API[4] provides functions for sending messages from a bot user, interact with the user by buttons and forms and react to events, like the registration of new users or specific messages. The developed applications can be published on the Slack platform and integrated in any workspaces. A workspace is a defined scope with various channels and a defined user group, as shown in Figure 4.

For the implementation, the *Real Time Messaging* (RTM) API is used to connect to a Slack workspace via a WebSocket[5], that "will provide a stream of events, including [...] messages [...]. This allows a client to easily maintain a synchronized local copy of all workspace data and messages." [22]

The authorization takes place using the OAuth 2.0 protocol[6], that creates 255 character long unique token for the workspace which is necessary to establish the connection. The has to be kept secret and enables the access to private user profiles in the complete workspace with their ID, names and images, in addition to the message retrieval.

SpaCy SpaCy is a NLP framework, that automates the pre-processing steps (tokenization, stemming and lemmatization with respect to morphological rules of the language and stop word filtering) and provide pre-trained word vectors with given term frequencies. SpaCy distinguishes between documents and tokens. A document is generated from raw input text of any length. [23] Afterwards, the individual tokens can be queried and provide, among other, the following attributes, as shown in Table 2.

Because of the word embedding, the tokens are arranged in a dense representation in the vector space and the similarity can be calculated quickly through the applied dimensionality reduction. SpaCy implements the cosine distance metric mentioned above in a similarity function, that is available for tokens and complete documents. For documents, the vector is averaged over the corresponding tokens.

The Figure 5 shows an example for the similarity values provided by spaCy based on the tokens from the theoretical background chapter, in conjunction with the values of the initial term-similarity-matrix.

SpaCy recognized the synonym "automobile" for the token "car" and returns the maximal similarity of one. It stands out, that the similarity is above zero

[4] https://api.slack.com

[5] A WebSocket is "standard way to open a long-lived bi-directional communication channel with a server over TCP." [22]

[6] https://api.slack.com/docs/oauth

Table 2. Important attributes of spaCy tokens [23, 24]

Attribute	**Description**
Text	The original word of the text.
Lemma	Result of the lemmatization process. Can be used to map several tokens to one term. Through the processing steps, the value does not longer matches the original text.
Part-of-speech tag	This tag provides information about the wort type, like noun, verb or number types.
Syntactic dependency	The interdependencies between tokens can be queried, like prepositions.
Is stop	This binary attribute indicates if the token is so common used, that it was classified as a stop word.
Vector	The vector is generated by a word embedding with 300 dimensions.
Probability	Logarithmic scaled token frequency. The value range is negative, but the higher the probability, the more common is a term in the underlying language model.

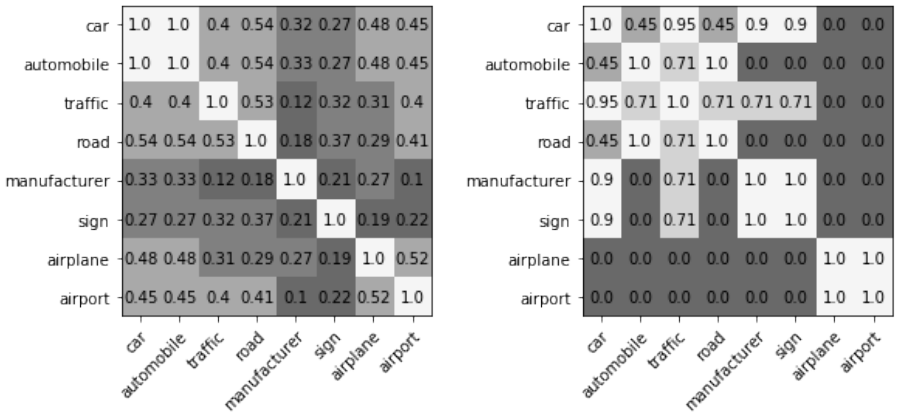

Fig. 5. Similarity values based on the spaCy corpus (left) and on the exemplary conversation of the theoretical background capter (right)

for all tokens. This can be explained by the fact that in the underlying corpus, with respect to a certain size, words appear in different constellations and are therefore variously related to each other. The word embedding may reinforce this situation.

But the similarities do not always coincide with the use in the considered conversation. For example, based on the discussion on the right side, the similarity between "car" and "airport" is equal to zero and for "car" and "traffic" equal to 0.95. On the other hand, the spaCy similarities on the left are opposed to them. The combination of "car" and "airport" is with a similarity of 0.45 much bigger and more likely in the underlying text corpus. The co-occurences of "car" and "traffic" are with a value of 0.4 less likely as in the example discussion. This is an aspect that needs to be considered. Spacy uses the vectors based on the underlying text corpus, that may not be equivalent to the usage of words in the brainstorming sessions.

SendToDB Module This module, shown in Appendix A, listens on the WebSocket with the Slack API and send retrieved messages to the database server.

The authorization data is stored in a *config.ini* file in the same directory. First, the RTM websocket is opened for the given token and leads to an infinity loop to listen constantly to the WebSocket. If new events (messages or a user starts to type a message) happen, the *rtm_read* function deliver all events since the last function call.

If new events are present, a connection to the database will be opened. An iterator considers all events and if such an event is a message type, the user ID, the text and the channel ID are extracted from the event dictionary and included into a SQL INSERT statement. Such a database connection is not is designed for a long-term connection, in contrast to WebSockets and therefore is initiated and closed after each usage. Afterwards, a *sleep* statement relives the processor and continue with the next *rtm_read* call and wait for new events.

EvaluateFromDB Module This module retrieves the data from the database and apply the NLP techniques to create the mind map during the brainstorming session. In contrast to the SendToDB module, this module can be executed at any time, even after the session finished. In this section, the program flow is summarized and design decisions are discussed. The code is available in Appendix B.

Because spaCy distinguishes only between *documents* (a text with several words) and *tokens*, in the following, the spaCy-like term *token* is used for words from the brainstorming session.

Initialization In the first step, the user names are downloaded from the channel with their corresponding profile image. This enables the appearance of names and images for the user vector in the mind map, that visualize the thematic location based on the corresponding statements.

Afterwards, an infinite loop checks the database server continuously for the new statements for a given Slack channel. If new statements are available, a new version of the mind map is generated.

Processing of User Statements Based on the spaCy framework, each statement is processed to a spaCy document object, that can be evaluated in its entirety to get the mean vector of the message, which represents the current thematic location of a user. Additionally, the tokens in all statements have to be evaluated, regardless of the author, to compute similarities. Therefore, data structures for both cases are implemented.

Token Requirements and User Vectors The user vector of the first statement in the conversation is the document vector (mean vector of all tokens). So that stop words do not affect the vector, a function checks each token for suitability with the following conditions:

- The token is no whitespace character,
- is not part of the spaCy stop word list,
- and is no digit (written out in full or as a number),
- has to be ASCII encoded to consider only valid characters,
- a length greater than one, to filter out a single character that may occur,
- the part-of-speech tag classifies the token as a noun
- and the token probability is within defined thresholds.

The current position of a user in the mind map can be a single vector of the latest statement or a course like a moving average, that allows the visualization of thematic transitions and can be plotted as a line in the mind map.

Because a moving average considers a defined number of vectors that are averaged, this implementation uses an alternative approach, that is often used as a learning rate in machine learning models, that updates the previous vector with a given weight. This allows a faster computation, because only two vectors are part of the calculation, which enables the use case for large scale applications. For example, the moving user vector is not limited to a single brainstorming session, but can (if intermediate vectors are stored) show the change in a long-term view across sessions.

Subsequent statements of a user updates the previous user vector with a given rate of adaption α. For a given user vector $\boldsymbol{v_t}$ and a statement vector $\boldsymbol{s}$, the following user vector $\boldsymbol{v_{t+1}}$ is calculated by:

$$\boldsymbol{v_{t+1}} \;=\; (1-\alpha)\boldsymbol{v_t} \;+\; \alpha\boldsymbol{s}$$

The higher α, the higher the adaption to the newer statements and the bigger the change in the mind map. The user similarity calculations are applied on the 300-dimensional vectors. After the dimensionality reduction is applied, the vectors and the corresponding courses can be visualized in the mind map.

Token Probability Thresholds The token probabilities provided by spaCy are logarithmic scaled and smaller than zero and equal to -20 or greater. The lower the probability, the rarer is a token, et vice versa. Table 3 show probabilities for exemplary tokens. The tokens "the" and "a" are very frequent and should not appear in the mind map, despite the fact that they can be filtered out by the stop word attribute. On the other hand, tokens like "car", "dog" or rarer should appear in the mind map. If a token is unknown, it gets a probability of -20. This can be very unusual tokens or typos. This implementation uses a minimum threshold of -19.9 to ignore typos and a maximum threshold of -8 per default.

Table 3. Exemplary spaCy token probabilities

Token	**Probability**
the	-3.53
a	-3.93
car	-8.37
dog	-9.0
computer	-9.18
algorithm	-12.06
deoxyribonucleic	-18.42

Dimensionality Reduction Token that meet the requirements mentioned above should be visualized in the mind map. In this context, is it important to ensure that only a set of lemmas (which is known from the lemmatization process and is a subset of all mentioned tokens) appear in the mind map. For example, if two participants use different forms of a word, for example a different numerous like "cars" and "car", only one form should be considered.

If a set of unique words is available, a dimensionality reduction can be applied to reduce the spaCy-given 300 dimensions into a two dimensional vector space for visualization. For an example, terms from three different topics are used (transportation, sciences, animals) and applied to the explained dimensionality reduction techniques LSA, PCA and t-SNE.

The results for each technique are provided in Figure 6. The LSA reduction separated the topic about sciences ("chemistry", "biology", "science") from the other tokens, but the distance of the transportation topic ("transportation", "ship", "car", "truck") to the animal topic ("cat", "dog", "animal") is partially lower (e.g. between "car" and "cat") than the intra-topic distances of the transportation tokens. With the given word embedding of spaCy, LSA may not be the best technique for an appropriate visualization.

The PCA technique seperates all three topics with a high distance, whereas the intra-topic distance is small, the tokens of a topic are grouped together in a semantically understandable manner.

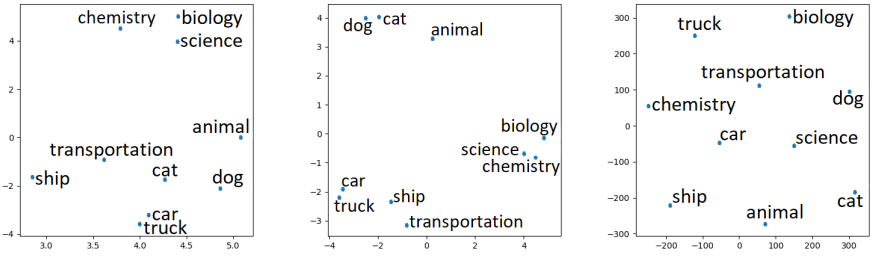

Fig. 6. Dimensionality reduction techniques: LSA (left), PCA (middle), t-SNE (right)

The t-SNE reduction seems to maintain an equal distance between the neighboring tokens. A semantic grouping is not recognizable. Therefore, t-SNE seems not to be appropriate for the current task.

The best results achieved the PCA reduction, which is chosen for the implementation.

Associations On the basis of the token similarity function, the semantic distance can be computed between two terms. This results in a term-matrix with quadratic run time and can be a limitation for sessions with many tokens (long session time or many participants).

To find an appropriate minimum threshold to paint associations, the tokens from the dimensionality reduction of three different topics are compared against each other in Figure 7, with two color gradations, to visualize intra- and inter-topic similarities (on the left side). It can be seen, that the science and the animal topic forms two groups with significantly higher similarities within and lower values outside of the topic. The minimal similarity of tokens to be within a group is around 0.6 (values are rounded), the maximal similarity to be outside of a group is around 0.4. A possible threshold for associations may be 0.5 and is the default value in this implementation. The corresponding mind map with association lines can be seen on the right side of Figure 7.

Associations between all tokens of the transportation topic is not permissible with a threshold of 0.5. The tokens "transportation" and "ship" are not connected to each other or to "car" or "truck", because of similarity values of 0.3 and 0.4. Would the threshold be lowered to 0.4, also associations between "truck" and "dog" would be would be drawn in the mind map, although they belong to different subjects.

Visualization of User Vectors and Courses The visualization of user vectors and corresponding courses can be seen in Figure 8. If no image is stored, the user name is used. The thickness of the line allows the identification of a chronological order. The thicker the dotted line, the more recent the user position is.

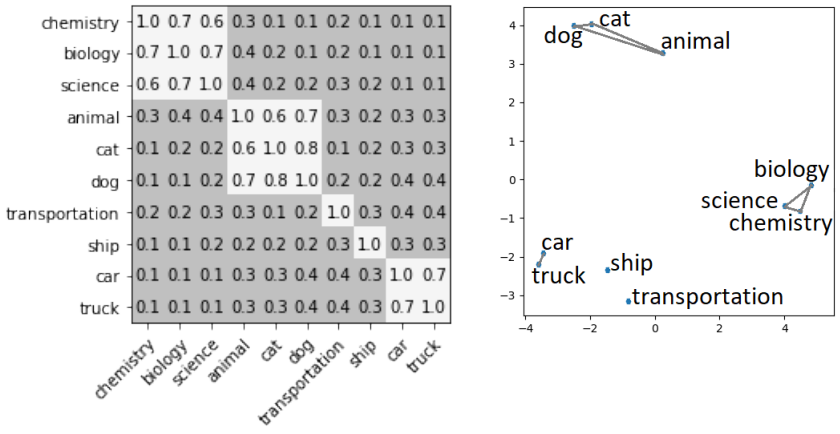

Fig. 7. Visualization of intra- and inter-topic similarities (left) and the corresponding associations in a mind map (right)

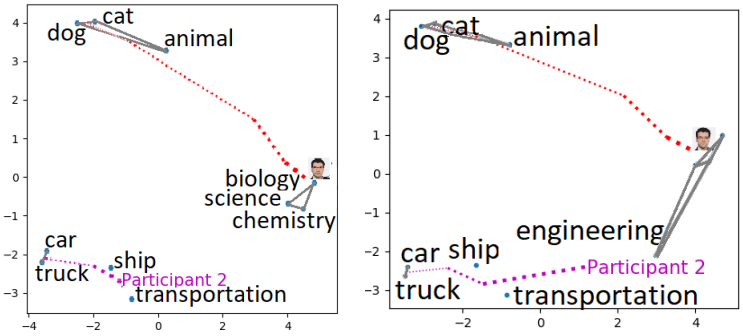

Fig. 8. Visualization of user vectors in the course of a conversation. Left: Mind map at time t_0, right: Mind map for t_1

The person of the red line started with the animal topic and moved on to the science topic. The person with the purple line start with tokens like "car" and "truck" and moved on to "ship" and "transportation" (left graphic). The distance of the user vectors in the mind map can also be used to determine the distance of the persons in the conversation. In this example, the topics animals and transportation are very different, but when the user of the red line moved on to the science topic, the distance gets smaller, with respect to the axis scaling.

In the graphic on the right side of Figure 8, participant 2 uses the token "engineering" and moves further up to the participant with the red line. For reasons of transparency, the science tokens are invisible for the illustration in this paper.

The mind map during the brainstorming sessions can have overlaps of labels if tokens are very similar. An often used technique to prevent overlapping elements of given coordinates is called *jitter*. A jitter plots the vector not on the exact coordinates, but with a small random distribution around the true value, so that similar elements are not inevitable overlapping.

With this procedure the mind map generation is completed. An optional upload via FTP is available, to transfer the mind map to a web server, so that participants can retrieve it over a network, an can be opened parallel to the Slack window on a computer, to track the mind map in real-time during the brainstorming session. A website can reload the image in regular intervals, so that the user always see the newest version.

3 Evaluation

For the evaluation phase, laboratory experiments were initiated. Based on data protection rules like GDPR[7] in the European Union, the participants must be informed about the data collection, processing and storing (in the current task the name or ID values and the conversational content) and must agree to this. Therefore, field experiments require strict legal requirements.

Due to the interactive components during the brainstorming session, the laboratory experiments have the character of qualitative experiments with open questions about expectations and experiences on brainstorming support systems.

This section introduces the experiment design with the corresponding questionnaire and analyze the given feedback from the participants.

3.1 Experiment Design

Subjects are selected randomly, without preference for e.g. technical affinity to consider different user groups with different technical backgrounds. As mentioned in the chapter introduction, the participants are informed about a system, that stores and process their data of a brainstorming session, but do not get details of functions or the implementation yet. For a participation, the participants have

[7] General Data Protection Regulation: https://gdpr-info.eu

to agree to the data storage and processing. The survey, on the other hand, is anonymous and does not allow conclusions to be drawn about a person.

The experiment starts with a first survey, that asks for their experiences with computer-based creativity technique support systems and expectations of such a system. Afterwards, the participants get access to a private channel on the Slack platform for the conversation and brainstorming about an arbitrary, but from the participants chosen topic for about ten minutes. A final survey asks for the made experiences about the ability of usefulness and the perceived creativity support. The asked questions are introduced in detail in the following subchapter.

3.2 Questionnaire

The questionnaire consists of two parts. The first part is filled out in advance by the test persons without knowing the exact procedure of the experiment. The questions are:

1. *Do you already have experience with computer-aided creativity techniques?*
   This yes/no question asks if participants have used similar systems.

2. *How likely do you think computer systems can support creativity techniques?*
   This question aims to the ability of usefulness on a scale from very unlikely to very likely in 5 levels.

3. *Are you familiar with the rules of brainstorming?*
   Yes/no question.

4. *What do you expect from a computer system that supports brainstorming?*
   This open question asks for the expectations of such a system in the sense of usefulness and ease of use.

5. *Which functions are particularly important to you?*
   This question aims at the expected functionality.

Afterwards, the participants are forwarded to the Slack workspace and conduct the brainstorming session and receive a link to the mind map available on the Internet.

After they finished the brainstorming session, they are pleased to rate their experiences by the following questions. The questions from number six to 14 are arranged in a scatter matrix with the following response options. An odd number of possible answers was chosen to allow a neutral category.

- Very unlikely / very bad / very unimportant
- Unlikely / bad / unimportant
- Not sure
- Likely / good / important

- Very likely / very good / very important

6. *Now that you have had experience with a computer-based brainstorming system: How likely do you think that such a system can support the brainstorming process?*
   Evaluation of the perceived ability of usefulness.

7. *How do you rate the productivity support?*
   Checking whether the system has a positive influence on productivity.

8. *How important is the traceability of the conversation to you?*
   This question asks how important it is to understand the course of the conversation.

9. *Referring to the previous question: how well do you rate the implemented traceability (course of a person)?*
   Evaluation of how well traceability is implemented from the user perspective.

10. *How good is it for you to recognize in a mind map which person has addressed which topics?*

11. *To what extent do you think the support of creativity for generating new ideas is given?*
    Ask if the system help to find new ideas

12. *How important would it be to you if the system independently made suggestions for new, unspecified terms?*

13. *At the beginning, you were informed that your data (name, profile picture if applicable, as well as your messages) would be stored in a database. Do you feel this affects your choice of words?*
    Asks if the user feels influenced in his choice of words. It should be noted that only the perceived influence can be queried.

14. *If the system identifies people who are very similar to you and your ideas, how likely is it that you will contact them to deepen your ideas together?*
    The user vectors allow similarity comparisons within a conversation as well as between conversations. Thus, persons that are important for each other (who are concerned with the same topics) can be identified and proposed to each other.

15. *Do you have any further comments or suggestions?*
    Possibility to enter additional characteristics that were not queried.

3.3 Analysis

This subchapter provide an analysis of the replies from the participants. A total of nine test persons could be acquired, eight of them use English as a foreign language and one person is a native English speaker. This may have an influence on the discussion behavior and influence the experience.

Introduction Seven of the nine participants had no experience with computer-aided creativity techniques. On average, the participants without prior experience are not sure about the ability of usefulness of computer systems for the support of creativity techniques. Participants with prior experience estimate the support potential as likely. Most probands are familiar with the rules of brainstorming (seven out of nine).

Expectations and Important Functionalities The most frequently mentioned expectations are an active support during the creativity technique, by suggestions of new terms or further information and that the system is able to group words by their semantic to provide an overview of the discussion. The system should also be able to identify possible vulnerabilities or problems. On the other hand, the system should not distract from the discussion to allow a free and creative thinking. The system should also support people who are not familiar with the topic of conversation. One participant mentioned the need for the system to detect when a person is too far away from the topic.

Experiences Most of the experiences were slightly positive, so that a support between "not sure" and "likely" is considered. The support of the brainstorming process and the productivity was "likely" in the most cases. The traceability of participants was rated as important, but usefulness the current implementation was rated as "not sure" and the ability of the recognition which participant deals with which topic was graded "bad" in the most cases and "not sure" on average. Only one participant graded the implementation of the course of a person as very good.

The support for new ideas is between "not sure" and "likely". The request for word suggestions is classified as important. The perceived influence in the choice of words was rated as "very unlikely" in the most cases and as "unlikely" on average.

For contacting proposed persons dealing with similar issues, the majority was uncertain whether they would make use of it. On average, establishment of contact is considered as likely.

Further Comments or Suggestions It has been noted, that it is an interesting experience to watch the mind map grow and that such a system "can be helpful in large companies" with a high fraction of communication in the everyday working life.

On the other hand, a few vulnerabilities have been identified. By the independent consideration of tokens, compounds like "rock melon" are not recognized as well as negations. It was noticed, that the arrangement of tokens is defined by a pre-computed or learned word distribution (the word embedding and the underlying corpus) and not always coinciding with the context in the conversation, what may causes confusion.

Suggestions for improvement are a more (pro-)active support (hints on what should be considered more closely), the dealing with British and American English (if both forms of a token are used, both appear in the mind map at the same location) and an improvement in representation (e.g. tokens should be colored with respect to a person, that mentioned it or a highlighting of common words). Also the courses of user vectors could be more clear.

4 Discussion

This chapter compares the objectives of a solution with the expectations and the experiences of the participants. Possible weaknesses and limitations are addressed in this context. Afterwards, further improvements are introduced, that may solve current challenges, also beyond the defined objectives, through further technical possibilities.

4.1 Comparison of the Expectations with the Objectives

The expectations of a visualization were formulated in such a way that important words are displayed and arranged according to topics. This was taken into account when formulating the objectives as the cluster hypothesis. Wishes that the system should not distract from the discussion coincide with the objectives based on the guidelines by Strohmann [25], according to which the system should operate in the background and the underlying operations should not be visible. The fear of manipulation by intelligent systems was not mentioned. On the contrary, an active support was desired, which should identify possible problems in the factual situation or suggest new idea or words.

4.2 Comparison of the Experiences with the Objectives

A productivity support was perceived as likely, but several points of improvement could be identified. The traceability of persons over the time during the session was rated as important, but the current implementation allows too few details for extensive traceability. At this point, highlighting terms in the colors of the users who mentioned them might be helpful, as a participants feedback revealed. The need for word suggestions that have not yet been implemented is considered as very important. The perceived influence in the choice of words was not detected by the participants. But as already mentioned in the questionnaire design, the true influence may differ from the perception and requires further studies.

4.3 Further Improvements

On the question of further word suggestions, it was criticized that each word is considered individually and therefore also negations are ignored. IR techniques can also deal with tokens, that spread over multiple words [15], but this result in a significantly higher vocabulary and requirements to the computing power. During experimental brainstorming sessions over a period of one hour with 78 statements, the quadratic run time of the term-similarity matrix was identified as a limitation, because the computing time no longer represents a virtually real-time evaluation.

Also the general pre-computed word embedding was criticized, that does not adapt to the context of the brainstorming session. It is possible to use several embeddings from which the appropriate one is selected according to the topic, in preparation for the session. Otherwise, Manning et al. provide machine learning techniques in [15], that allow the detection of topic within documents, and transferred to the current use case, within brainstorming sessions. Also it would be possible to apply the similarity calculations without a corpus and pre-trained embedding and group the tokens with respect to their usage during the session. But this makes it difficult to transfer the quality of the arrangement of words to other conversations and varies greatly. This may be an object for further studies. However, a pre-trained embedding would be necessary to implement unmentioned word suggestions. Because of the vocabulary size of an embedding, it may be a major challenge, to compare all mentioned tokens with the complete vocabulary, to find similar ones. Context-dependent lists with frequently mentioned terms would simplify this and could be used under the machine-learning-based topic detection.

Also, IR techniques can be used for sentiment analysis to classify statements, whether a person has expressed itself positively or negatively about a topic [15]. This techniques could be used to deal with negations.

Differential Privacy To deal with the influence on the choice of words due to the storage of complete statements in the database, the approach of Weggenmann and Kerschbaum in [28] is introduced, that combines word vectors (like tf-idf) with the differential privacy framework, "which currently serves as a 'gold standard' for privacy definitions". This framework adds a specific amount of noise to values, that enhance general interdependencies in the data to fulfill data mining tasks, but do not allow conclusions to be drawn as to whether an individual is present in the dataset or absent [28]. The removal of identifier attributes is not an appropriate technique to ensure privacy [28, 4].

Differential privacy uses a *privacy budget*, that determines the amount of noise in outcomes of queries. A data miner can choose between a very precise query or several queries with a bigger amount of noise. Regardless of the chosen privacy budget, all cases ensure privacy. This framework also protects against collusions between data miner [9]. Mathematical derivations can be found in the papers [9, 14].

On the other hand, papers like [3, 2] show weaknesses of the framework. In order to exclude an information leak, it may be necessary to add such a big amount of noise to the outcome of queries that either an evaluation is unusable or allow conclusions to be drawn about the underlying data based on the big amount of noise.

The evaluation of the approach by Weggenmann and Kerschbaum to "prevent authorship attribution" results in an effectively technique, while the general information could be preserved (using the example of classification task) [28].

5 Conclusion

The technical aspects for the objectives of a solution could be implemented. A visualization technique was introduced, that group similar words in similar regions on a mind map, to show the mentioned ideas during a brainstorming session. Although, the course of persons can be plotted as well as the potential to compute person similarities, to identify persons, that mention similar statements in brainstorming sessions and are presumed important for each other. Furthermore, the visualization of user vectors allow the identification of opportune and not opportune persons. For example, in a brainstorming session of n participants, $n-1$ may use the same words and are presumed opportune, the remaining person, that may not yet be convinced of the topic, could be identified and integrated into the talk. This can be done by the participants itself or offers further starting points for further research of virtual moderator functionality.

The evaluation of the experiments provides approaches to further objetives of studies, but basically the approach was perceived as helpful by the participants.

References

1. Arora, S., Hu, W., Kothari, P.: An Analysis of the t-SNE Algorithm for Data Visualization. (2018) arXiv: 1803.01768v2
2. Bambauer, J., Muralidhar, K.: A Response to the Criticisms of Fools Gold: An Illustrated Critique of Differential Privacy. Available at https://blogs.harvard.edu/infolaw/2016/05/17/diffensive-privacy/. Last accessed 21 Sep 2018
3. Bambauer, J., Muralidhar, K., Sarathy, R.: Fools Gold: an Illustrated Critique of Differential Privacy. (2013)
4. Barth-Jones, D.: The "Re-identification" of Governor William Welds Medical Information: A Critical Re-examination of Health Data Identification Risks and Privacy Protections, Then and Now. (2012)
5. Ben-David, A., Liu, H., Jackson, A.: The KullbackLeibler Divergence as an Estimator of the Statistical Properties of CMB Maps. (2015) arXiv: 1506.07724v2
6. Buzan, T., Harrison, J.: The Mind Map Book: Unlock Your Creativity, Boost Your Memory, Change Your Life. (2010)
7. Dutta, D.: A Review of Different Word Embeddings for Sentiment Classification using Deep Learning. (2018) arXiv:1807.02471v1
8. Elhoseiny, M., Elgammal, A.: Text to Multi-level MindMaps: A Novel Method for Hierarchical Visual Abstraction of Natural Language Text. (2014)

9. Friedman, A., Schuster, A.: Data Mining with Differential Privacy. (2010)

10. Gallupe, R., Dennis, A., Cooper, W., Valacich, J., Bastianutti, L., Nunamaker, J.: Electronic brainstorming and group size. (1992)

11. Garcia, E.: The Classic TF-IDF Vector Space Model. (2016)

12. Gera, S., Aneeshkumar, G., Fernandez, S., Gireeshkumar, G., Nze, I., Eze, U.: Virtual teams versus face to face teams: a review of literature. (2013)

13. Gewers, F., Ferreira, G., de Arruda, H., Silva, F., Comin, C., Amancio, D., Costa, L.: Principal Component Analysis: A Natural Approach to Data Exploration. (2018) arXiv: 1804.02502v2

14. Ji, Z., Lipton, Z., Elkan, C.: Differential Privacy and Machine Learning: a Survey and Review. (2014) arXiv:1412.7584

15. Manning, C, Raghavan, P., Schütze, H.: Introduction to Information Retrieval. (2008)

16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space. (2013) arXiv: 1301.3781v3

17. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed Representations of Words and Phrases and their Compositionality. (2013) arXiv: 1310.4546v1

18. Osborn, A.: Applied Imagination - Principles and Procedures of Creative Writing. (2013)

19. Paulus, P.: Electronic brainstorming research and its implications for e-planning. (2015)

20. Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S.: A Design Science Research Methodology for Information Systems Research. (2007)

21. Scikit-Learn Truncated SVD. Available at http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html. Last accessed 18 Sep 2018

22. Slack Real Time Messaging API. Available at https://api.slack.com/rtm. Last accessed 16 Sep 2018

23. Spacy Linguistic Features. Available at https://spacy.io/usage/linguistic-features. Last accessed 17 Sep 2018

24. Spacy Token API. Available at https://spacy.io/api/token. Last accessed 17 Sep 2018

25. Strohmann, T.: Virtual Moderation Assistance: Creating Design Guidelines for Virtual Assistants Supporting Creative Workshops. (2018)

26. Strohmann, T., Siemon, D., Robra-Bissantz, S.: brAInstorm: Intelligent Assistance in Group Idea Generation. (2017)

27. Van der Maaten, L., Hinton, G.: Visualizing Data using t-SNE. (2008)

28. Weggenmann, B., Kerschbaum, F.: SynTF: Synthetic and Differentially Private Term Frequency Vectors for Privacy-Preserving Text Mining. (2018) arXiv: 1805.00904v1

Appendix

A SendToDB module

```
# SendToDB.py
import time
import mysql.connector
from slackclient import SlackClient
from configparser import ConfigParser

config = ConfigParser()
config.read('config.ini')
token = config.get('Slack', 'BotToken')

sc = SlackClient(token)
if sc.rtm_connect(): # connect to a Slack RTM websocket
    while True:
        items = sc.rtm_read()
        if len(items) > 0:
            cnx = mysql.connector.connect(user = config.get('db', 'user'),
                password=config.get('db', 'pw'),
                host=config.get('db', 'host'),
                database=config.get('db', 'db'))
            cursor = cnx.cursor()

            # iterate over all events since the last rtm_read call
            for item in items:
                # decide between user messages and other interactive
                # messages with buttons
                # buttons containt the attribute 'subtype' and are not
                # supported in this version.
                if item['type'] == 'message' and not 'subtype' in item:
                    # Extract values
                    user = item['user']
                    text = item['text']
                    channel = item['channel']
                    print(user + ": " + text)

                    # Prepare Insert statement
                    query = ("INSERT INTO message(user, channel, text)
                        VALUES (%s,%s,%s)")
                    cursor.execute(query, (user, channel, text))
                    cnx.commit()
            cursor.close()
            cnx.close()
        time.sleep(1)
else:
    print('Connection Failed, invalid token?')
```

B EvaluateFromDB module

```
# sklearn
from sklearn.decomposition import PCA, TruncatedSVD
from sklearn.manifold import TSNE
#numpy
from numpy import dot
from numpy.linalg import norm
# spaCy
import spacy
from spacy.lang.en import English
#matplotlib
import matplotlib
matplotlib.use('Agg')
from matplotlib import pyplot as plt

import pandas as pd
import itertools as it
import random
import numpy as np

import matplotlib.image as image
import mysql.connector
from configparser import ConfigParser
import Helper
import time
import Token

# Constants for program flow
SHOW_USER_VECTOR = False # Show only the last user vector
SHOW_MOVING_USER_VECTOR = True # Show the moving average user vector
FTP_UPLOAD = False # Upload of mindmap to FTP server?

# Constants
CHANNEL = "GCM3LSGF8"
ALPHA = 0.6 # Rate of adaption of the user vector. The higher, the
# greater the deflections of the line
MIN_SIMILARITY = 0.4 # For association lines
USERS_WITH_IMAGE = ['UCBQG6T0W'] # Specify the user ID with an own
# (not Slack initial) image



# init
nlp = spacy.load('en_core_web_lg')
highestMessageID = 0
# Initialize user_dict and download user_images
user_dict = Helper.getUserNames()
Helper.downloadUserImages()
```

```
while(True):
    time.sleep(1)
    conn = Helper.getDBCursor()

    # Check for new messages in the specified channel
    latestMessageIDFromServer = Helper.checkForUpdates(conn, CHANNEL)
    if latestMessageIDFromServer is None:
        continue
    if latestMessageIDFromServer > highestMessageID:
        highestMessageID = latestMessageIDFromServer
        # new message available -> generate new mindmap
    else:
        continue

    #returns a List with tuple (user_id, message)
    messagesAndUsers = Helper.getMessagesOfUsers(conn, nlp, CHANNEL)

    #extract the messages
    messages = list()
    [messages.append(item[1]) for item in messagesAndUsers]

    tokens = Helper.getTokens(nlp, messages)

    # Compute user vector and append user to tokens,
    # so that the user mean vector can be visualized
    if SHOW_USER_VECTOR or SHOW_MOVING_USER_VECTOR:
        users = dict()
        for item in messagesAndUsers:
            user = item[0]
            message = item[1]
            # Only deal with the suitable tokens in the message,
            # not all words
            token_list_from_message = Helper.getTokens(nlp, [message])
            if len(token_list_from_message) > 0 :
                # extract vectors
                token_vectors = list()
                [token_vectors.append(item.vector_300_dim) for item in
                    token_list_from_message]
                message_vector = np.mean(token_vectors, axis=0)
                if not user in users:
                    users[user] = [message_vector] # init vector
                else:
                    weighted_old = np.multiply(users[user][-1], 1-ALPHA)
                    weighted_new = np.multiply(message_vector, ALPHA)
                    users[user].append(np.add(weighted_old, weighted_new))

    if SHOW_USER_VECTOR and not SHOW_MOVING_USER_VECTOR:
        for user in users:
```

```
            token = Token.Token(user)
            token.set_vector_300_dim(users[user])
            tokens.append(token)

    #Collect vectors for dimensionality reduction
    vectors_300dim = list()
    [vectors_300dim.append(item.vector_300_dim) for item in tokens]

    #Collect labels
    labels = list()
    [labels.append(item.token) for item in tokens]

    if len(vectors_300dim) < 2:
        continue

    #PCA dimensionality reduction
    X = np.array(vectors_300dim)
    pca = PCA(n_components = 2)
    vectors_2dim = pca.fit_transform(X)

    # Create instances of an own Token class for an easier similarity
    # comparison
    for i in range(0,len(tokens)):
        tokens[i].set_x(vectors_2dim[i][0])
        tokens[i].set_y(vectors_2dim[i][1])

    # Collect coordinates for visualization
    x_labels = list()
    y_labels = list()
    [x_labels.append(item.x) for item in tokens]
    [y_labels.append(item.y) for item in tokens]

    fig, ax = plt.subplots(figsize=(15,15))
    ax.scatter(x_labels, y_labels, marker = ".")

    # Plot tokens
    for i, token in enumerate(tokens):
        ax.annotate(token.token, (x_labels[i] + 0*random.uniform(-0.2,
            0.2)-0.5,
            y_labels[i] + 2*random.uniform(-0.2, 0.2)), fontsize=15)

    # Find associations
    similar_tokens = list()
    for token1 in tokens:
        for token2 in tokens:
            if token1.token != token2.token:
               # Only compute similarity if necessary
                sim = Helper.computeTokenSimilarity(nlp, token1, token2)
                if sim >= MIN_SIMILARITY:
                    similar_tokens.append([token1, token2, sim])
```

```
# paint associations
for item in similar_tokens:
    plt.plot([item[0].x, item[1].x], [item[0].y, item[1].y],
         color='k', linestyle='-', linewidth=0.2*item[2])

# paint moving vector of each user
if SHOW_MOVING_USER_VECTOR:
    for user in users:
        vector_list_300 = users[user]
        for i in range(0,len(vector_list_300) - 1):
            vectors_2dim = pca.transform(np.array([vector_list_300[i],
                 vector_list_300[i+1]]))
            plt.plot(
                [vectors_2dim[0][0], vectors_2dim[1][0]],
                [vectors_2dim[0][1], vectors_2dim[1][1]],
                color=Helper.userColor(user_dict[user]), linestyle=':',
                linewidth=5*i/len(vector_list_300))


        if user in USERS_WITH_IMAGE:
            # show user umage
            im = image.imread('user_images/' + user + '.jpg')
            ax.imshow(im, aspect='equal', extent=(
                vectors_2dim[1][0],
                vectors_2dim[1][0] + .6,

                vectors_2dim[1][1],
                vectors_2dim[1][1] + .6),
                zorder = -1)
        else:
            # for the last element: Annotate user ID
            ax.annotate(user_dict[user],
                (vectors_2dim[1][0] + 1*random.uniform(-0.2, 0.2),
                vectors_2dim[1][1] + 1*random.uniform(-0.2, 0.2) ),
                color = Helper.userColor(user_dict[user]),fontsize=15)

        ax.plot()
        ax.grid()


plt.savefig('mindmap.png')
plt.close()
print('New mindmap generated')

if FTP_UPLOAD:
    Helper.sendImage('mindmap.png')
```

C Helper module

```
from configparser import ConfigParser
import mysql.connector
import numpy as np
from sklearn.decomposition import PCA
import Token
import hashlib
from slackclient import SlackClient
from configparser import ConfigParser
from urllib.request import urlretrieve
import ftplib

def getDBCursor():
    config = ConfigParser()
    config.read('config.ini')
    conn = mysql.connector.connect(user = config.get('db', 'user'),
         password=config.get('db', 'pw'),
         host=config.get('db', 'host'),
         database=config.get('db', 'db'))
    return conn

def checkForUpdates(conn, channel):
    query = ("SELECT MAX(ID) FROM message WHERE channel = '" + channel +
        "'")
    cursor = conn.cursor()
    cursor.execute(query)
    for row in cursor:
        return row[0]

def getMessagesOfUsers(conn, spacy_nlp, channel):
    query = ("SELECT user, text FROM message WHERE channel = '" +
         channel + "' ORDER BY ID")
    cursor = conn.cursor()
    cursor.execute(query)
    message_list = list()

    for record in cursor:
        message_list.append((record[0], spacy_nlp(record[1])))
    return message_list

def getMessagesOfUsersMaxID(conn, spacy_nlp, channel, maxID):
    query = ("SELECT user, text FROM message WHERE channel = '" +
         channel + "' AND ID <= " + str(maxID) + " ORDER BY ID")
    cursor = conn.cursor()
    cursor.execute(query)
    message_list = list()

    for record in cursor:
```

```
            message_list.append((record[0], spacy_nlp(record[1])))
    return message_list

def getTokens(spacy_nlp, message_list):
    tokens = list()

    for message in message_list:
        for item in message:
            if checkForSuitability(item) \
                and not checkItemInList(item.lemma_, tokens):
                token = Token.Token(item.lemma_)
                token.set_vector_300_dim(spacy_nlp(item.lemma_).vector)
                tokens.append(token)

    return tokens

def checkForSuitability(token):
    if not (token.is_space and token.is_stop and token.is_digit) \
        and token.is_ascii and len(token) >= 2 and token.pos_ == "NOUN" \
        and -19.9 < token.prob < -8:

        return True
    else:
        return False

def checkItemInList(search_item, ls):
    for item in ls:
        if item.token == search_item:
            return True
    return False

def computeTokenSimilarity(spacy_nlp, token1, token2):
    return spacy_nlp(token1.token).similarity(spacy_nlp(token2.token))

def userColor(user):
    colors = ['b', 'g', 'r', 'c', 'm', 'y']
    return colors[abs(hash(user)) % len(colors)]

def getUserNames():

    # Create a dictionary with usernames with respect to a given ID
    user_dict = dict()

    config = ConfigParser()
    config.read('config.ini')
    botToken = config.get('Slack', 'BotToken')

    slack = SlackClient(botToken)

    users = slack.api_call("users.list")
```

```
    for user in users['members']:
        user_dict[user['id']] = user['real_name']

    return user_dict

def downloadUserImages():
    config = ConfigParser()
    config.read('config.ini')
    botToken = config.get('Slack', 'BotToken')

    slack = SlackClient(botToken)

    users = slack.api_call("users.list")
    for user in users['members']:
        urlretrieve(user['profile']['image_192'], 'user_images/' +
            user['id'] + '.jpg')


def sendImage(filename):
    config = ConfigParser()
    config.read('config.ini')

    user = config.get('ftp', 'user')
    password = config.get('ftp', 'pw')
    host = config.get('ftp', 'host')

    session = ftplib.FTP(host, user, password)
    file = open(filename, 'rb')
    session.storbinary('STOR ' + filename, file)
    file.close()
    session.quit()
```