Visualizing User Story Requirements at Multiple Granularity Levels via Semantic Relatedness

Garm Lucassen(✉), Fabiano Dalpiaz, Jan Martijn E.M. van der Werf, and Sjaak Brinkkemper

Utrecht University, Utrecht, The Netherlands
{g.lucassen,f.dalpiaz,j.m.e.m.vanderwerf,s.brinkkemper}@uu.nl

Abstract. The majority of practitioners express software requirements using natural text notations such as user stories. Despite the readability of text, it is hard for people to build an accurate mental image of the most relevant entities and relationships. Even converting requirements to conceptual models is not sufficient: as the number of requirements and concepts grows, obtaining a holistic view of the requirements becomes increasingly difficult and, eventually, practically impossible. In this paper, we introduce and experiment with a novel, automated method for visualizing requirements—by showing the concepts the text references and their relationships—at different levels of granularity. We build on two pillars: (i) *clustering techniques* for grouping elements into coherent sets so that a simplified overview of the concepts can be created, and (ii) state-of-the-art, corpus-based *semantic relatedness algorithms* between words to measure the extent to which two concepts are related. We build a proof-of-concept tool and evaluate our approach by applying it to requirements from four real-world data sets.

1 Introduction

Natural language (NL) is the most popular notation to represent software requirements: around 60 % of practitioners employ NL as their main artifact [12]. Moreover, the trend in agile development has boosted the adoption of the semi-structured NL notation of *user stories* [11,14,31]: "As a ⟨*role*⟩, I want ⟨*goal*⟩, so that ⟨*benefit*⟩". Recent research [14] shows that 90 % of practitioners in agile development adopt user stories.

NL requirements are easy to read but have a major drawback: as their number increases, the quantity of the involved concepts grows rapidly, making it increasingly harder for humans to construct an accurate mental model of those concepts. A possible solution is the (semi-)automated generation of an explicit conceptual model [6,10,19].

Inspired by these works, we have previously proposed the *Visual Narrator* tool that automatically extracts conceptual models from sets of user stories with satisfactory accuracy (80 %–90 %) [23]. However, our evaluation with practitioners indicated that the extracted models quickly become too large to be effectively explored by analysts.

I. Comyn-Wattiau et al. (Eds.): ER 2016, LNCS 9974, pp. 463–478, 2016.
DOI: 10.1007/978-3-319-46397-1_35

The problem of model comprehensibility can be generalized to the conceptual modeling field [3, 17]: humans' working-memory capacity restricts the ability to read models and leads to *cognitive overload* when the same model includes too many concepts.

To tackle this problem, we build upon Shneiderman's *visual information seeking* mantra: *"overview first, zoom/filter, details on demand"* [26]. We propose and experiment with a mechanism for improving the visualization of conceptual models that are generated by the *Visual Narrator* from user stories. We make use of clustering techniques to group the concepts so to obtain an initial *overview*.

We go beyond existing clustering approaches in literature (see Sect. 5) by leveraging on state-of-the-art, corpus-based *semantic relatedness* algorithms based on neural networks to determine the similarity between concepts [8, 28]. The significant improvements in accuracy of these recent approaches trigger our experimentation of these techniques for guiding the clustering of the concepts in user stories.

Our approach is novel in that it does not require additional documentation about the system under development; indeed, it relies on publicly available corpuses of data from the Web. Moreover, we focus on conceptual models generated from user stories, which have a limited expressiveness compared to full-fledged conceptual models.

Specifically, this paper makes two contributions:

- We devise a method for creating an overview of the concepts in a conceptual model by creating clusters that contain semantically related concepts;
- We propose a proof-of-concept tool for generating clusters and for zooming them; we evaluate its feasibility by applying it to user story sets from 4 real-world cases.

The rest of the paper is structured as follows. Section 2 outlines our background: the *Visual Narrator*, semantic similarity, and clustering techniques. Section 3 presents our method. Section 4 applies our proof-of-concept tool to real-world user story data sets. Section 5 reviews related work, and Sect. 6 presents our conclusions and future directions.

2 Background

To ease constructing a mental image of a software system, we propose to take requirements expressed as user stories and derive a conceptual model that enables inspecting system functionality with different degrees of granularity. We review the background work that will later be combined as part of our method in Sect. 3.

2.1 From User Stories to Conceptual Models

User stories are a textual language for expressing requirements that uses a compact template. A user story captures *who* the requirement is for, *what* it is expected from the system, and (optionally) *why* it is important [33]. Although many different templates exist, 70 % of practitioners use the Connextra template [14]: *"As a* ⟨*type of user*⟩*, I want* ⟨*goal*⟩*, [so that* ⟨some *reason*⟩*]"*. For example: *"As an Event Organizer, I want to receive an email when a contact form is submitted, so that I can respond to it"*.

In previous work, we created a tool that automatically extracts a conceptual model from a set of user stories using NLP heuristics: the *Visual Narrator*. This tool is itself built upon a conceptual model of user stories that distinguishes between *role*, *means* and *ends* parts of a user story [13]. By parsing the user stories with SpaCy's part-of-speech tagging[1] and applying eleven state-of-the-art heuristics, the Visual Narrator creates conceptual models of a user story set with up to 86 % recall and 81 % precision. The output of Visual Narrator is an OWL 2 ontology or a Prolog program including the concepts and relationships extracted from a set of user stories.

2.2 Novel Approaches to Semantic Similarity

As we aim to group concepts in order to facilitate comprehension, we need to identify concepts that are similar or related to one another. We rely on the *semantic similarity* (more precisely, *semantic relatedness*) of word pairs. This is a number—typically in the [0,1] range—that captures the distance between the two words, with 0 being no relatedness and 1 being full relatedness. For any given word, this technique can be used to identify a list of similar words or to calculate its semantic similarity score with a collection of words. If the process is repeated for all words in a collection C, one obtains a matrix that defines the pairwise similarity between all concepts in C.

Among the many approaches to calculating semantic similarity [9], we focus on a family of novel, state-of-the-art algorithms: *skip-gram* by Google [16] and *GloVe* by Stanford [20]. Both algorithms parse huge quantities of unannotated text to generate *word embeddings* without requiring supervision. A word embedding maps some attributes of a word to a *vector* of real numbers that can then be used for a variety of tasks, similarity being one of them. Both skip-gram and GloVe adhere to the distributional hypothesis: *"linguistic items with similar distributions have similar meanings"*.

These techniques constitute the most accurate state-of-the-art and provide significant improvements even on other word embedding approaches [16, 20]. Moreover, the innovative vector-based approach of word2vec enables basic "semantic arithmetics" on words: vector("King") - vector("Man") + vector("Woman") results in a vector which is most similar to vector("Queen"). These new methods have not yet been applied in the conceptual modeling and requirements engineering literature, while they are slowly but steadily being adopted in industry, also thanks to their excellent performance.

[1] https://spacy.io/.

2.3 Clustering Algorithms

Clustering refers to the process of taking a set of concepts and grouping them so that concepts in the same group are similar and concepts in different groups are different. We aim to adopt clustering in the context of user stories and on the basis of the semantic similarity/relatedness between concepts. Since word embeddings and semantic similarity scores are expressed as numbers, we can easily use pre-existing tools to apply the existing variety of clustering algorithms.

The go-to algorithm for most clustering needs is k-means , but through experimentation with many of the available algorithms we found it to be less applicable for our use case. This is mostly due to the randomness of the algorithm: the resulting clusters differentiate too much between runs.

Instead, we choose an algorithm that leads to similar accuracy results but uses a consistent approach: Ward's minimum variance method [32]. Ward's algorithm starts by assigning all concepts to their own cluster and then iterating over the cluster collection until it finds the two clusters that, when merged, lead to a minimum increase in within-cluster variance of the collection of clusters. It keeps repeating this step until k clusters have been formed. Although Ward's method is slower than k-means, the impact is negligible for the relatively small data sets that one extracts from a set user stories, and in extreme cases the clustering over the entire data set can be executed once the tool starts.

3 Visualization Method for User Stories

We describe our approach to visualizing concepts and relationships between user stories based on the theory introduced in Sect. 2. Our method features three main functionalities: the generation of an overview (Sect. 3.1), zooming in and out mechanisms (Sect. 3.2), and filtering techniques (Sect. 3.3). To illustrate, we use a publicly available set of 104 user stories from the Neurohub project[2], an information environment for neuro-scientists developed by three British universities.

3.1 Overview Generation

The purpose of an overview is to *"provide a general context for understanding the data set"* [5]. By abstracting from all the details of the data, filtering extraneous information and highlighting specific patterns and themes in the data, the overview supports the end-user in understanding the information. To achieve this goal for a user story set, we propose a 6-step process visualized in Fig. 1 and elaborated below.

1. **Extraction from User Stories.** The Visual Narrator extracts a set of relevant concepts C from the user stories and a set of relationships $R \subseteq C \times C$ between those concepts. In our example, the output consists of 124 concepts

[2] http://neurohub.ecs.soton.ac.uk/index.php/All_User_Stories.

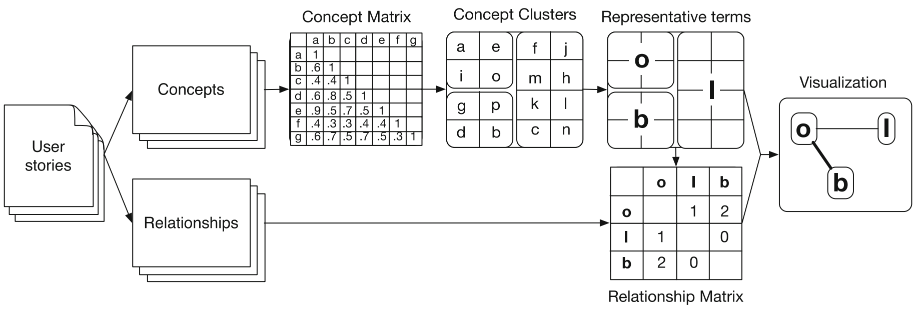

Fig. 1. Our method for generating the overview

in C and 144 relationships between these concepts in R as shown in the following Prolog lines:

```
concept('Neuroscience').
concept('Researcher').
...
isa(concept('Book Page'),concept('Page')).
rel(concept('Researcher'),'Create',concept('Book Page')).
```

2. **Concept Similarity Calculation.** We use the skip-gram implementation *word2vec*[3] to calculate the semantic similarity/relatedness scores—in the range $[0, 1]$—for each concept with all other concepts in the list C. As explained in Sect. 2, the use of skip-grams combines efficiency and accuracy. This step results in a similarity matrix SM of size $|C| \times |C|$ such that $\forall i, j \in [0, |C|)$. $SM_{i,j} = skipgram(c_i, c_j)$. In the following example, it is possible to see how some couples of words are much more semantically related than others: compare researcher and neuroscience (0.5134) with neuroscience and booking (0.1667).

SM	neuroscience	researcher	book	booking	...
neuroscience	1.0000	0.5134	0.3446	0.1667	...
researcher	0.5134	1.0000	0.3362	0.2055	...
book	0.3446	0.3362	1.0000	0.2301	...
booking	0.1667	0.2055	0.2301	1.0000	...
...	...	...	...	...	...

3. **Concept Clustering.** We utilize Ward's clustering algorithm to group all the concepts according to their similarity in SM. This results in a high-level disjoint clustering WC that forms the basis for our visualization. In our experimentation, inspired by the cognitive principles by Moody [18], we generate nine clusters.

[3] https://code.google.com/p/word2vec/.

```
0: ['acceptance test', 'acceptance', 'analysis', 'behaviour', 'dependency', 'description', 'drug
        response', 'experiment description', 'experiment', 'input', 'knowledge', 'neurohub
        dependency', 'neuroscience', 'paper', 'provenance', 'research paper', 'research', 'response
        ', 'search result', 'search', 'southampton neuroscience', 'test result', 'test', 'work', 'worm
         analysis']
1: ['control system', 'equipment booking', 'equipment', 'installation', 'lts machine', 'lts', 'lab
        administrator', 'lab member', 'lab', 'mri operator', 'mri', 'neurohub installation', '
        neurohub workspace', 'spreadsheet', 'system administrator', 'system', 'workspace']
2: ['behaviour video', 'calendar', 'directory', 'google calendar', 'google', 'inventory', 'link', 'log
        ', 'machine', 'meta', 'reference', 'script', 'table', 'tag', 'type', 'ups', 'version', 'video', '
        web', 'worm', 'write ups', 'write']
3: ['browser', 'client', 'interface graphics/colour', 'interface', 'mendeley client', 'neurohub
        node', 'node', 'operator', 'protocol', 'user', 'web browser']
4: ['booking', 'control', 'cost', 'drug', 'event', 'field', 'forward', 'minimal', 'others', 'period', '
        release', 'result', 'run', 'share', 'sharing', 'southampton', 'time', 'track', 'what']
5: ['data file', 'data', 'file type', 'file', 'html tag', 'html', 'information', 'keywords', 'meta data'
        , 'metadata', 'minimal information', 'share data', 'template']
6: ['administrator', 'engineer', 'investigator', 'member', 'release engineer', 'researcher', '
        supervisor']
7: ['graphics/colour', 'mendeley', 'neurohub']
8: ['book entry', 'book page', 'book', 'entry', 'log book', 'neurohub page', 'page']
```

Note how the clusters are of different sizes: for example, while cluster 0 has size 25, cluster 7 has size 3. Also note how cluster 6 neatly relates roles/professions such as administrator, engineer, etc. This is one of the key differences of employing corpus-based similarity as opposed to looking at the graph structure.

4. **Representative Term Selection.** From each cluster c_i in WC, we identify the concept which is most similar to the collection of concepts in cluster c_i. We do so by using the analogy capabilities of skip-gram as introduced in Sect. 2.2: we compute the sum of the word vectors swv of the concepts in a cluster; then, we set the cluster label by choosing the name of concept in the cluster whose vector model is most similar to the vector swv. For example, consider a cluster with concepts administrator, visitor and individual. We compute the sum vector swv = vector("Administrator") + vector("Visitor") + vector("Individual"). Among these concepts, the word whose vector is closest to the sum of the vectors is individual. To avoid meaningless labels, we remove stop words—such as he, a, from, ...—before we execute this step. For the Neurohub case, we obtain the following results:

0: analysis	1: lab	2: web	3: user	4: time
5: data	6: engineer	7:	8: book	

Note that for cluster 7 no label could be assigned because word2vec does not have any of the cluster's terms in its dictionary.

5. **Inter-cluster Relationships Matrix Generation.** Since the concepts in a cluster are represented by one term, intra-cluster relationships do not need to be visualized. Starting from the list of relationships R, we derive a matrix ICR of size $|WC| \times |WC|$ that determines the strength of the relationships between the clusters by counting how many relationships exist between the concepts in those clusters. Formally, $\forall i, j \in [0, |WC|)$:

$$ICR(c_i, c_j) = \begin{cases} 0, \mathit{if}\ i = j \\ |r(x, y)| \in R.\ (x \in c_i \wedge y \in c_j) \vee (x \in c_j \wedge y \in c_x), \mathit{else} \end{cases}$$

In our example, we obtain the following matrix:

ICR	c0	c1	c2	c3	c4	c5	c6	c7	c8
c0	0	2	2	4	5	0	3	1	0
c1	2	0	2	3	3	1	5	2	0
c2	2	2	0	8	0	4	4	0	1
c3	4	3	8	0	7	10	1	3	2
c4	5	3	0	7	0	2	5	0	0
c5	0	1	4	10	2	0	14	0	0
c6	3	5	4	1	5	14	0	0	1
c7	1	2	0	3	0	0	0	0	1
c8	0	0	1	2	0	0	1	1	0

The remarkably large number of relationships between 5 and 6 is caused by the concentration of role concepts in cluster 6 who perform an action on the concepts related to data in cluster 5.

6. **Visualization Drawing.** Each cluster $c \in WC$ becomes a vertex (a circle) with the representative term as its label. The diameter of the circle increases with the number of concepts in the cluster. Lines are drawn for each inter-cluster relationship in ICR; the width of a link increases with the number of relationships between the connected clusters. An example of the generated overview is shown in Fig. 2.

3.2 Zooming

The purpose of zooming is to reduce the complexity of the data presentation by having the user adjust the data element size and selection on the screen [5]. We propose two zooming views that enable exploring distinct details of the overview; these views are accessed by clicking on either (1) a concept or (2) a relationship.

When a user clicks on a concept, that concept will be zoomed in and the steps outlined in Sect. 3.1 are re-run on the concepts within the cluster. The

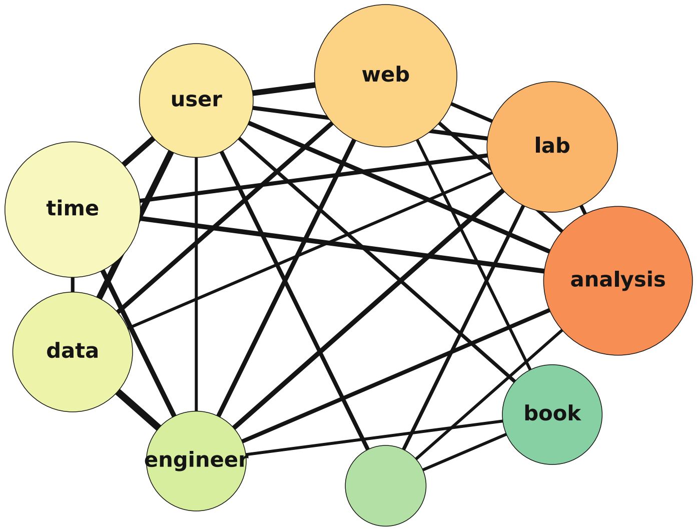

Fig. 2. Example overview of the user stories from the Neurohub project

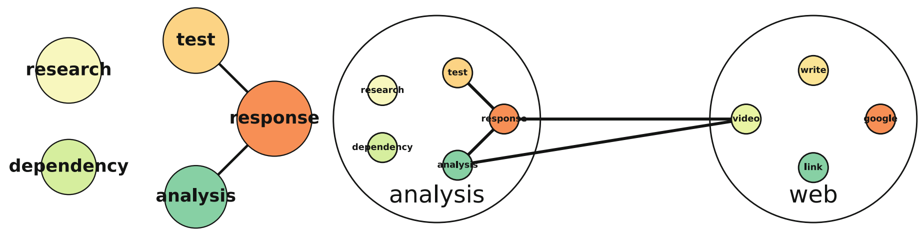

Fig. 3. Zooming examples for Neurohub: concept zoom on the *analysis* cluster (on the left), and relationship zoom between the *analysis* and the *web* clusters (on the right)

only difference is that we set the number of sub-clusters to the square root of the number of elements within the cluster[4]. The outcome is a more granular view of the concepts in the cluster and showing new inter-cluster relationships that were previously hidden as intra-cluster relationships. See the left image of Fig. 3 for an example. In case the number of concepts in that cluster is lower or equal than 9 (see [18]), all concepts will be shown.

Clicking on a relationship will simultaneously zoom in on the two clusters that the relationship connects, showing the same more granular view of the concepts as when clicking on a concept. Furthermore, zooming in on the relationship displays all underlying relationships individually and connects the smaller underlying concept clusters on both sides. See the right image of Fig. 3 for an example.

3.3 Filtering

The purpose of filtering is the same as zooming: reducing the complexity of the presented data. However, instead of selecting a specific region, filtering controls enable the user to change whether the data points with a given attribute are visible [5]. In the context of user stories, we propose four filters that enable the user to further simplify a data view or to further explore some specific details:

1. *Relationships:* by default any view is drawn with its relevant relationships. Optionally, the user can turn this off the relationships clicking, allowing complete focus on the concept clusters. Alternatively, it is possible to filter out specific relationship types, e.g., visualizing only or hiding is-a relationships.
2. *Role:* a central and prominent aspect of user stories, roles are the most frequently occurring concepts in any user story set. Indeed, 96 of the 123 relationships in the Neurohub example (78 %) connect a role to some concept. We propose two ways of filtering roles: (i) removing all roles from the set of concepts C, enabling the user to focus on relationships between other concepts; (2) selecting a specific role to focus on, removing all concepts and relationships that are not related to that role.

[4] Determining the number of clusters is still a work-in-progress part of our approach.

3. *Search:* users can query for concept terms to find the cluster related to that concept. For example, searching for *file* will highlight the data cluster and its relationships while slightly blurring all unrelated concepts and relationships.
4. *Agile Artifacts:* In agile software development, user stories are organized into meaningful chunks: epics, themes and sprints. The user can select any combination of these in order to explore specific parts of the system (via epics and themes) or to focus on certain development periods (sprints).

4 Prototype Demonstration

We demonstrate the feasibility of our approach by applying a prototype implementation to three real-world case studies. The prototype is available online including the Neurohub user stories[5]. Unfortunately, we cannot release the confidential case study user stories. For each case, we present four views of their user story concepts and relationships: overview, concept zoom, relationship zoom and a role filter. Finally, we evaluate and discuss the output.

4.1 Case 1: CMSCompany

The company developing this complex CMS product for large enterprises is located in the Netherlands, serving $\tilde{1}$50 customers with 120 employees. Their supplied data consists of 32 syntactically correct user stories, which represents a snapshot of approximately a year of development in 2011. Visual Narrator extracted 96 concepts and 114 relationships, exemplifying that despite the small size of the user story set, the use of long user stories with non-trivial sentence structuring means many concepts and relationships are present.

Applying the prototype to *CMSCompany*'s user stories results in Fig. 4. Upon examination of the overview on the left, one thing immediately stands out: the clusters are highly interrelated and none of them clearly has the majority of relationships. This is likely a consequence of the long, non-trivial structuring of these user stories. Furthermore, some of the representative terms are highly relevant to the CMS domain: site, marketeer, text, business & analytics are important aspects of CMSCompany's product.

By contrast, the concept zoom of the *result* cluster has no intercluster relationships at all. In fact, none of the subclusters contain such a relationship. Intuitively, the authors believed this to be a bad result but upon closer examination this phenomenon actually turns out to be the ideal situation. Indeed, there actually *are* relationships between concepts in this subcluster but they are all within their own subsubcluster, demonstrating that at this level the semantic clustering approach is very effective at grouping related user story concepts.

[5] https://github.com/gglucass/Semantic-Similarity-Prototype.

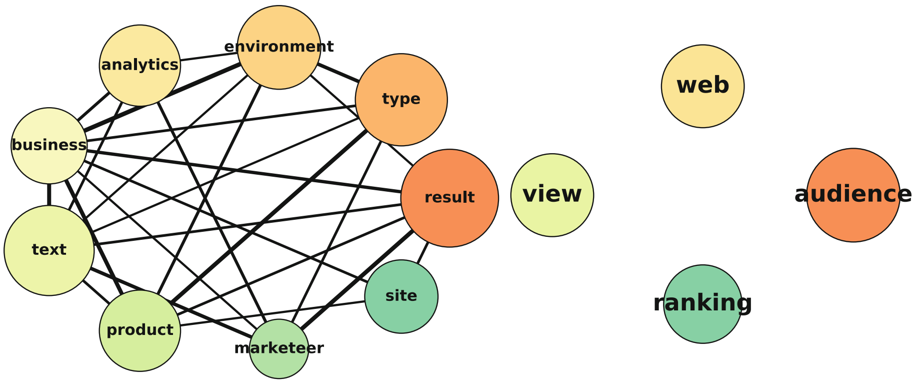

Fig. 4. Overview for CMSCompany user stories and concept zoom on the *result* cluster

4.2 Case 2: WebCompany

This is a young Dutch company that creates tailor-made web applications for businesses. The team consists of nine employees who iteratively develop applications in weekly Scrum sprints. WebCompany supplied 98 user stories covering the development of an entire web application focused on interactive story telling that was created in 2014. Although the data set is 3× as big as CMSCompany's, these user stories are very simple, concise and contain very few complex sentence structures. Because of this, Visual Narrator extracts just 106 concepts and 123 relationships.

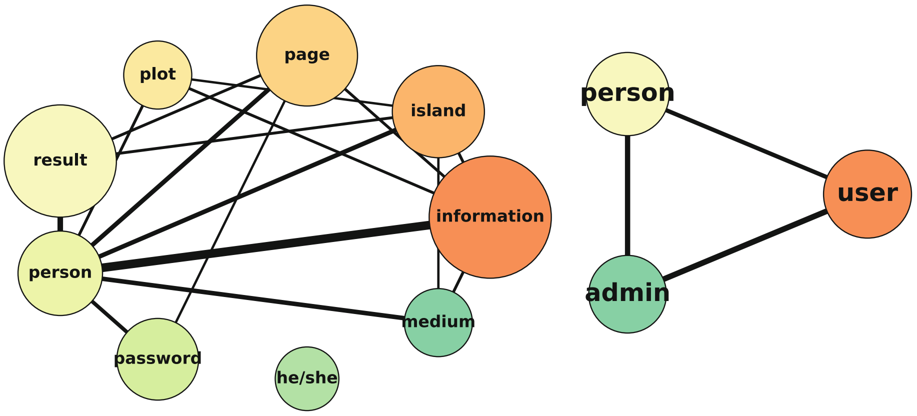

Fig. 5. Overview for WebCompany user stories and concept zoom of *person* cluster

In CMSCompany's overview in Fig. 5 the *person* cluster clearly has the most relationships with other clusters. This is a direct consequence of two factors:

(1) the person clusters contains all roles defined in the user stories and (2) because the user stories are simple, the majority of relationships in this set are role(action, object). However, for this case not all representative terms are meaningful. In particular, *result* is strongly related to merely 3 or 4 out of 22 concepts in that cluster. This exemplifies the approach's weakness of selecting very general terms for large, less coherent clusters because they are at least somewhat related to many of the terms in the cluster.

In some cases, previously intracluster relationships do become intracluster relationships when zooming in on a cluster. The subclusters in *person* are all related to one another in some way. Because this user story concerns an entire web application, this is to be expected. Indeed, if admin was not related to a user a human analyst should be triggered to investigate if the user story set is incomplete. This exemplifies a possible real-world use case of the prototype output.

4.3 Case 3: SCMCompany

This case concerns stories from a company that delivers a leading Supply Chain Management (SCM) suite for large companies in the logistics, healthcare, technology and consumer sectors. To support development in keeping up with double digit revenue growth, the company has started to embrace user stories. This set consists of 54 high quality user stories of moderate size and complexity. In total, Visual Narrator extracted 91 concepts and 114 relationships.

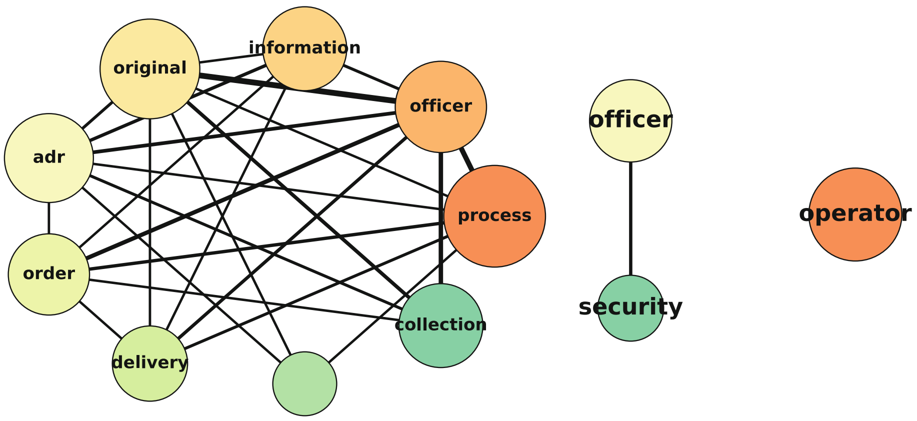

Fig. 6. Overview for SCMCompany user stories and concept zoom of *officer* cluster

Nearly all of the representative terms in the overview in Fig. 6 are strongly related to the SCM domain. Furthermore, the relationship between concepts *security* and *officer* in the concept zoom is actually an accurate representation of the relationship in the source data. Considering that the sub clusters contain multiple concepts, this example demonstrates a positive result of our approach for selecting a representative term.

5 Related Literature

We review the relevant literature about RE visualization, clustering techniques for generic conceptual models, and extraction of conceptual models from requirements.

5.1 RE Visualization

Requirements engineering visualization (REV) is concerned with creating effective visualizations of RE artifacts. Cooper et al. [4] review the papers appeared in the REV workshops between 2006 and 2008. They distinguish between different types of visualizations: tabular, relational, sequential, hierarchical, and metaphorical/quantitative. The most relevant categories for our work are the relational—i.e., (hyper-)graphs—and hierarchical—decomposing a system into its parts. While many relational approaches exist, very few focus on hierarchal aspects, which are the key in our work.

We deliberately exclude from this section the vast body of literature on requirements modeling languages: this important family of requirements visualization approaches goes beyond the scope of our paper, as we aim at visualizing the main concepts that can be extracted from NL requirements.

Reinhard et al. [22] propose an improved Fisheye zoom algorithm for the visualization and editing of hierarchical requirements models. The most interesting feature of their algorithm is that is guarantees stability of the adjusted layouts and runs in linear time. We may exploit this algorithm in future work. Gandhi and Lee [7] use visualizations in the context of requirements-driven risk assessment. They extract a concept lattice that relates risk-related concepts such as assets, threats, vulnerabilities and countermeasures. The lattice is a possible criterion for a zoom-in/zoom-out mechanism.

To date, ReCVisu+ [21] is the most effective tool for requirements visualization. ReCVisu+ supports different visual exploration tasks and, like our approach, is based on clustering techniques and semantic similarity. While RecVisu+ determines similarity based on the frequency of co-occurrence in system documentation, we rely on corpus-based techniques that do not require the existence of additional system documentation. Moreover, we do not consider only concepts but also relationships.

5.2 Conceptual Modeling Clustering

The conceptual modeling and databases community is well aware that (E)ER diagrams are often large and cluttered. Teorey et al. have studied this problem already in the late 80s [27]: they proposed collapse/expand mechanisms that can be used to hide/view entities and relationships that are secondary to some primary entities; for example, 'journal address' and 'journal note' can be collapsed into a cluster labeled 'journal'.

Akoka and Comyn-Wattiau [1] propose automated clustering techniques that can be used to realize Teorey's vision and that derives non-overlapping clusters. They experiment multiple distance indicators with different semantics (visual, hierarchical, cohesive, etc.) and compare their strengths and weaknesses. Moody and Flitman [18] combine and refine principles from previous work and include cognitive aspects, such as the maximum size of a cluster being nine elements, in order to facilitate human comprehension. Tzitzikas and Hainaut [29] use link analysis from web searching to generate a smaller diagram that includes only the major entity and relationships; they also propose an automated 2D visualization that uses force-directed drawing algorithms.

Summarization techniques exist for database schemas. Yu and Jagadish [34] formalize the notion of a schema summary and define its quality in terms of complexity (number of elements), importance (inclusion of the most important elements), and coverage (are all major chunks of the original schema represented?). Based on these notions, they propose algorithms that optimize each of these quality criteria. Yuan et al. [35] go further by proposing elaborate metrics to determine table similarity and importance.

All these techniques inspire our work. The main novelty of our proposal is that we focus on conceptual models that represent requirements, and that we use novel corpus-based techniques to determine similarity. The significant advances that these algorithms provide make it possible for us to experiment clustering based on the co-occurrence of words in corpora of data on the Web with promising results.

5.3 Extracting Conceptual Models from Requirements

There is a long tradition in generating conceptual models from NL requirements. Already in 1989, Saeki et al. [24] proposed a method for the automatic extraction of verbs and nouns from NL. The ideas of this method were operationalized by numerous (semi-)automated tools, including NL-OOPS [15], CM-Builder [10], CIRCE [2], aToucan [36], and the tool by Sagar and Abirami [25].

All these approaches use NL processing algorithms such as tokenization, part-of-speech tagging, morphological analysis and parsing. These tools show promising precision and recall—comparable if not better than human experts—, although they often require restricted NL to do so. In previous work [23], we leveraged these tools and proposed an approach that is specifically suited for requirements expressed as user stories.

6 Discussion, Conclusion and Future Work

This paper explored the potential of applying semantic relatedness algorithms for visualizing user stories. After studying and experimenting with state-of-the-art algorithms such as skip-gram, we presented a novel, automated method for visualizing user stories at different levels of granularity. We applied a prototype implementation of this method to four real-world user story sets, studied the output and observed that:

- The generated visualizations are capable of highlighting relevant information classifications for the system. For example, the central role of the person cluster in the WebCompany overview is easily recognizable.
- For the majority of clusters, the generated representative term is meaningful and relevant within the application domain.
- When an intercluster relationship is present on the zoom level, it is generally relevant within that (sub-)domain. The analysis subcluster of Neurohub for example relates test, response and analysis.
- On the overview level, too many intercluster relationships are visible, effectively rendering them useless for further human analysis.
- The prototype tends to select irrelevant common denominator terms for large clusters with low internal coherence.
- Word2Vec does not include all words in meaningful clusters, resulting in residual clusters that cannot be assigned any label.

Based on these observations, we envision possible applications for our visualization approach to include: (1) discovering missing relationships between clusters that may result in further user stories; (2) teaching system functionality by exploring simplified, manageable chunks; and (3) analyzing expected system changes after introducing new sets of user stories (e.g., new epics). However, further practitioner evaluation is necessary to confirm the validity of our observations and the potential of these applications.

In future work, we intend to experiment with these possible use cases as well as investigate how to substantially improve the generated output. A necessary next step is to combine and compare our work with existing state-of-the-art clustering techniques that do not rely on semantic relatedness. Additionally, future work should incorporate state-of-the-art group structure visualization techniques [30]. We expect this to produce outputs that are even more usable in real-world scenarios. Additionally, we are investigating the potential benefits of applying machine learning to enhance the accuracy of semantic relatedness scores for specific application domains.

References

1. Akoka, J., Comyn-Wattiau, I.: Entity-relationship and object-oriented model automatic clustering. Data Knowl. Eng. **20**(2), 87–117 (1996)
2. Ambriola, V., Gervasi, V.: On the systematic analysis of natural language requirements with CIRCE. Autom. Softw. Eng. **13**(1), 107–167 (2006)
3. Aranda, J., Ernst, N., Horkoff, J., Easterbrook, S.: A framework for empirical evaluation of model comprehensibility. In: Proceedings of MiSE (2007)
4. Cooper Jr., J.R., Lee, S.W., Gandhi, R.A., Gotel, O.: Requirements engineering visualization: a survey on the state-of-the-art. In: Proceedings of REV, pp. 46–55 (2009)

5. Craft, B., Cairns, P.: Beyond guidelines: what can we learn from the visual information seeking mantra? In: Proceedings of IV, pp. 110–118, July 2005
6. Du, S., Metzler, D.P.: An automated multi-component approach to extracting entity relationships from database requirement specification documents. In: Kop, C., Fliedl, G., Mayr, H.C., Métais, E. (eds.) NLDB 2006. LNCS, vol. 3999, pp. 1–11. Springer, Heidelberg (2006). doi:10.1007/11765448_1
7. Gandhi, R.A., Lee, S.W.: Discovering and understanding multi-dimensional correlations among certification requirements with application to risk assessment. In: Proceedings of RE, pp. 231–240 (2007)
8. Goldberg, Y., Levy, O.: Word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. arXiv preprint arXiv:1402.3722 (2014)
9. Harispe, S., Ranwez, S., Janaqi, S., Montmain, J.: Semantic Similarity from Natural Language and Ontology Analysis. Morgan & Claypool Publishers, San Rafael (2015)
10. Harmain, H., Gaizauskas, R.: CM-Builder: a natural language-based CASE tool for object-oriented analysis. Autom. Softw. Eng. **10**(2), 157–181 (2003)
11. Kassab, M.: The changing landscape of requirements engineering practices over the past decade. In: Proceedings of EmpiRE (2015)
12. Kassab, M., Neill, C., Laplante, P.: State of practice in requirements engineering: contemporary data. Innov. Syst. Softw. Eng. **10**(4), 235–241 (2014)
13. Lucassen, G., Dalpiaz, F., van der Werf, J.M., Brinkkemper, S.: Improving agile requirements: the quality user story framework and tool. Requir. Eng. **21**, 383–403 (2016)
14. Lucassen, G., Dalpiaz, F., Werf, J.M.E.M., Brinkkemper, S.: The use and effectiveness of user stories in practice. In: Daneva, M., Pastor, O. (eds.) REFSQ 2016. LNCS, vol. 9619, pp. 205–222. Springer, Heidelberg (2016). doi:10.1007/978-3-319-30282-9_14
15. Mich, L.: NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. Nat. Lang. Eng. **2**, 161–187 (1996)
16. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, vol. 26, pp. 3111–3119 (2013)
17. Moody, D.: The "Physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Softw. Eng. **35**(6), 756–779 (2009)
18. Moody, D.L., Flitman, A.: A methodology for clustering entity relationship models — a human information processing approach. In: Akoka, J., Bouzeghoub, M., Comyn-Wattiau, I., Métais, E. (eds.) ER 1999. LNCS, vol. 1728, pp. 114–130. Springer, Heidelberg (1999). doi:10.1007/3-540-47866-3_8
19. Omar, N., Hanna, J., McKevitt, P.: Heuristics-based entity-relationship modelling through natural language processing. In: Proceedings of AICS, pp. 302–313 (2004)
20. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Proceedings of EMNLP, pp. 1532–1543 (2014)
21. Reddivari, S., Rad, S., Bhowmik, T., Cain, N., Niu, N.: Visual requirements analytics: a framework and case study. Requir. Eng. **19**(3), 257–279 (2014)
22. Reinhard, T., Meier, S., Glinz, M.: An improved fisheye zoom algorithm for visualizing and editing hierarchical models. In: Proceedings of REV. IEEE (2007)
23. Robeer, M., Lucassen, G., Van der Werf, J., Dalpiaz, F., Brinkkemper, S.: Automated extraction of conceptual models from user stories via NLP. In: Proceedings of RE (2016)

24. Saeki, M., Horai, H., Enomoto, H.: Software development process from natural language specification. In: Proceedings of ICSE, pp. 64–73. ACM (1989)
25. Sagar, V.B.R.V., Abirami, S.: Conceptual modeling of natural language functional requirements. J. Syst. Softw. **88**, 25–41 (2014)
26. Shneiderman, B.: The eyes have it: a task by data type taxonomy for information visualizations. In: Proceedings of VL, pp. 336–343 (1996)
27. Teorey, T.J., Wei, G., Bolton, D.L., Koenig, J.A.: ER model clustering as an aid for user communication and documentation in database design. Commun. ACM **32**(8), 975–987 (1989)
28. Trask, A., Michalak, P., Liu, J.: sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. arXiv preprint arXiv:1511.06388 (2015)
29. Tzitzikas, Y., Hainaut, J.-L.: How to tame a very large ER diagram (using link analysis and force-directed drawing algorithms). In: Delcambre, L., Kop, C., Mayr, H.C., Mylopoulos, J., Pastor, O. (eds.) ER 2005. LNCS, vol. 3716, pp. 144–159. Springer, Heidelberg (2005). doi:10.1007/11568322_10
30. Vehlow, C., Beck, F., Weiskopf, D.: The state of the art in visualizing group structures in graphs. In: Borgo, R., Ganovelli, F., Viola, I. (eds.) Eurographics Conference on Visualization (EuroVis) - STARs. The Eurographics Association (2015)
31. Wang, X., Zhao, L., Wang, Y., Sun, J.: The role of requirements engineering practices in agile development: an empirical study. In: Zowghi, D., Jin, Z. (eds.) Requirements Engineering. CCIS, vol. 432, pp. 195–209. Springer, Heidelberg (2014). doi:10.1007/978-3-662-43610-3_15
32. Ward, J.H.: Hierarchical grouping to optimize an objective function. J. Am. Stat. Assoc. **58**(301), 236–244 (1963)
33. Wautelet, Y., Heng, S., Kolp, M., Mirbel, I.: Unifying and extending user story models. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 211–225. Springer, Heidelberg (2014). doi:10.1007/978-3-319-07881-6_15
34. Yu, C., Jagadish, H.: Schema summarization. In: Proceedings of VLDB, pp. 319–330 (2006)
35. Yuan, X., Li, X., Yu, M., Cai, X., Zhang, Y., Wen, Y.: Summarizing relational database schema based on label propagation. In: Chen, L., Jia, Y., Sellis, T., Liu, G. (eds.) APWeb 2014. LNCS, vol. 8709, pp. 258–269. Springer, Heidelberg (2014). doi:10.1007/978-3-319-11116-2_23
36. Yue, T., Briand, L.C., Labiche, Y.: aToucan: an automated framework to derive UML analysis models from use case models. ACM Trans. Softw. Eng. Methodol. **24**(3), 13:1–13:52 (2015)