Extracting Social Networks from Literary Text with Word Embedding Tools

Gerhard Wohlgenannt
Vienna University of Economics
Inst. for Inf. Business
Vienna, Austria
wohlg@ai.wu.ac.at

Ekaterina Chernyak
National Research University
Higher School of Economics
Moscow, Russia
echernyak@hse.ru

Dmitry Ilvovsky
National Research University
Higher School of Economics
Moscow, Russia
dilvovsky@hse.ru

Abstract

In this paper a social network is extracted from a literary text. The social network shows, how frequent the characters interact and how similar their social behavior is. Two types of similarity measures are used: the first applies co-occurrence statistics, while the second exploits cosine similarity on different types of word embedding vectors. The results are evaluated by a paid micro-task crowdsourcing survey. The experiments suggest that specific types of word embeddings like word2vec are well-suited for the task at hand and the specific circumstances of literary fiction text.

1 Introduction

Word embeddings are language modeling techniques that transform the vocabulary of an input corpus into a continuous and low-dimensional vector representation. Word embeddings have shown state-of-the-art performance as language technology (LT) tools esp. for word similarity estimations, but also for more sophisticated operations like word analogies and as input component to various natural language processing (NLP) tasks (Mikolov et al., 2013; Ghannay et al., 2016). Word embeddings use artificial neural networks for generating the vector representations. Neural networks have become very popular and successful tools in NLP in the last couple of years, esp. with recent improvements in the deep learning field.

The performance of word embeddings in various task when using huge corpora of unstructured text has already been demonstrated in previous work. Here, we study the suitability of different types of word embeddings as a LT tool to extract social networks from literary fiction, ie. to a specific task and domain, and a comparably small corpus size. More precisely, we apply word embeddings to the text from the "A Song of Ice and Fire" book series by George R. R. Martin. The goal is to find book characters with the strongest relations to a given input character, and to compare the results from word embeddings to a very intuitive system, which uses term co-occurrence to determine the relatedness of characters. Furthermore, we evaluate the results from different word embedding tools and from a method based on co-occurrence statistics with human judgements generated with crowdsourcing. In this study, we did not focus on the detection and merging of character names, which is an interesting topic by itself, discussed for example in (Vala et al., 2015).

In this publication, we want to address the following research questions:

(i) How well does a traditional method based on co-occurrence statistics, such as the one presented in (Rodin et al., 2016), perform against state-of-the-art LT tools such as word embeddings for the task of social networks extraction in literary fiction?

(ii) Are there any differences between various types of word embeddings in the particular task of social networks extraction in literary fiction?

(iii) Furthermore, how well is paid micro-task crowdsourcing suited to evaluate facts in a domain with a lot of background necessary, such as a book series in the fantasy novel domain.

Proceedings of the Workshop on Language Technology Resources and Tools for Digital Humanities (LT4DH), pages 18–25, Osaka, Japan, December 11-17 2016.

2 Related Work

A first step when extracting social networks from literary text, is the detection of characters. A state-of-the-art approach is presented by (Vala et al., 2015). Character detection often includes the usage of named entity recognition (NER) tools and methods for co-reference resolution. The authors present their own 8-step approach to character detection, and evaluate its usefulness.

An obvious next step is the actual extraction of social networks from novels. In the method by (Elson et al., 2010) the networks are derived from dialog interactions. Therefore their method includes finding instances of quoted speech, attributing each quote to a character, and identifying when certain characters are in conversation. They construct a weighted graph, where nodes correspond to actors, and the weights on the edges represents the frequency and amount of exchanges. In contrast to our work, (Elson et al., 2010) are solely focus on length and number of dialogues between persons to measure relatedness, whereas our approach looks at general co-occurrence or similarity as measured by LT tools which use word embeddings. Similarly, (Celikyilmaz et al., 2010) address a the problem related to the extraction of relations between characters. They attribute utterances in literary dialogues to actors, and apply the similarities in the language used to predict similarity and hidden relations between those actors. In contrast to our work, the approach also is restricted to dialogues between authors, and the evaluation of the method is of limited scope.

Another approach is presented by (Agarwal et al., 2013), who detect "social events" between persons (or groups of persons), where those persons interact. By using the connections from the social events, which help to form links between characters, the authors evaluate the extraction of social networks from literary text (*Alice in Wonderland*).

Our method of social network construction is more straightforward, and applies and evaluates existing word embedding tools. (Ghannay et al., 2016) did extensive evaluations to compare different kinds of word embeddings, such as word2vec CBOW and skip-gram, GloVe, CSLM and word2vec-f (see 3 for details on various algorithms). The different methods and tools perform very differently depending on the task. For NLP tasks, word2vec-f provided the best results, GloVe had the best performance in analogical reasoning, and CBOW/skip-gram were best at word similarity tasks. The authors also experiment with combinations of the methods in order to raise accuracy.

As already mentioned, we use crowdsourcing to evaluate the results produced by the various LT tools applied to the task of social network extraction. We selected paid micro-task crowdsourcing as scalable and in-expensive evaluation method, which has become popular in research only in recent years. There already exists a plethora of work on crowdsourcing in various fields, for example natural language processing (Bontcheva et al., 2014; Sabou et al., 2012), knowledge modeling (Wohlgenannt et al., 2016) or Bioinformatics (Mortensen et al., 2013). But, to our knowledge, it has not been applied in the digital humanities field on a similar task as social network extraction. In paid micro-task crowdsourcing the workload is usually split into small units, which are then presented to anonymous mass of crowd workers. A major issue is ensuring high quality results, typically measures in this direction are: (i) clear and extensive task description, (ii) careful worker selection, (iii) using test questions which workers need to pass before doing the real work, (iv) adequate worker remuneration, (v) assign work units to multiple workers and using aggregate results, etc.

3 Methods and Tools

In this section we briefly introduce and describe the methods used to find (the strongest) relations between the book characters. These include the co-occurrence based methods in Section 3.1 and the word embeddings tools in sections 3.2 to 3.4. Based on the results in (Ghannay et al., 2016) we picked three types of word embeddings to be applied: *word2vec, GloVe, and word2vec-f.* The configurations of the various methods used are found in Section 4.1 (Experiment Setup).

3.1 Co-occurrence based method

Our method is based on straightforward calculation of the so-called confidence coefficient. Given a text and two names, say A and B, we denote the frequency of name A by $F(A)$. The co-occurrence frequency

of A and B is then $F(A \cap B)$. There are several ways how we detect the co-occurrence of A and B: first, we can check whether A and B occur in the same book chapter. Secondly, we can check whether A and B occur in the same paragraph, which can be problematic, if the book is ill-formatted and the paragraph splits are not present. Finally, we can check whether A and B occur in the same sentence. The resulting confidence formula is the following:

$$conf(A, B) = \frac{F(A \cap B)}{F(A)}. \quad (1)$$

This formula can be interpreted this way: given A, how probable it is that B occurs, so that the coefficient is normalized into an interval between 0 and 1.

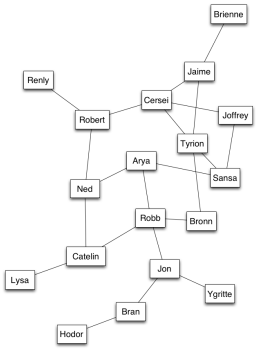

Figure 1: An example network, constructed using the co-occurrence-based method (on the paragraph level). Here, for the top 10 characters the top-3 connections are selected.

3.2 Word2vec

Word2vec (Mikolov et al., 2013) is a tool created by a team at Google led by Tomas Mikolov. Word2vec applies two-layer neural networks trained to reconstruct the linguistic contexts of words (or phrases). The input to word2vec is typically a large corpus (plain text), the output are word embeddings – which are continuous vector space representations of the words in the corpus. word2vec uses a dimensionality-reduced representation, usually with a vector length of 50 to 300. Proximity in vector space corresponds to similar contexts in which words appear. There are two model architectures to create the continuous vector representations: continuous bag-of-words (CBOW) or continuous skip-gram. With CBOW, the model predicts the current word by using a window of surrounding words. With skip-gram, the model predicts the surrounding window of context words by using the current word.

3.3 GloVe

Similar to word2vec, GloVe (Pennington et al., 2014) learns continuous vector representations of words. But it is not a predictive model, but rather a count-based model, using dimensionality-reduction on word-word co-occurrence statistics. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. We used the GloVe implementation from Stanford university[1].

3.4 Word2vec-f – Dependency-based Word Embeddings

Dependency-based word embeddings (Levy and Goldberg, 2014) are a modification of word2vec in order to generalize the skip-gram model with negative sampling to arbitrary contexts. Therefore, it is referred as the word2vec-f implementation[2]. In contrast to linear word contexts, dependency-based contexts are generated by a dependency-parser and produce markedly different embeddings. (Levy and Goldberg, 2014) expect "the syntactic contexts to yield more focused embeddings, capturing more functional and

[1] `http://nlp.stanford.edu/projects/glove,GloVeversion1.2`
[2] `https://levyomer.wordpress.com/2014/04/25/dependency-based-word-embeddings`

less topical similarity". In the evaluations by (Ghannay et al., 2016), word2vec-f was very effective in NLP tasks such as POS-tagging or Named Entity Recognition, but did not perform as well as word2vec on word similarity tasks.

4 Evaluation

In this section, we evaluate and compare the six methods to extract relations from text for a given set of input terms: (i) co-occurrence statistics on a chapter level, (ii) co-occurrence statistics on a paragraph level, (iii) co-occurrence statistics on a sentence level, (iv) word2vec, (v) GloVe, and (vi) word2vec-f – see Section 3 for details about the methods.

4.1 Evaluation Setup

Text corpus: For the evaluation we used the plain text versions of the first four books of the "A Song of Ice and Fire" (ASOIF) book series by George R. R. Martin. ASOIF is a series of fantasy novels. The action takes place in an fictional medieval-like universe. Although the number of character is immense, the are up to 40 main characters which communicate throughout the series. While narration is almost linear with minor flashbacks, the story is told in the first person. However there are different narrators telling the story from different viewpoints, i.e. different POVs. The raw books amounts to 6.9M of plain text, and contain 204 chapters with a mostly chronological storyline. There are 121098 sentences in total. Each chapter features a point of view character, which may live in any part of the ASOIF world. There are a few reasons behind our motivation to use ASOIF as the main source of data:

(i) it is popular nowadays, which gives the hope that the crowd will cope with the questions;

(ii) there is relatively large group of main characters, which interact intensively with each other in different circumstances, so that the social network might quite dense;

(iii) the book gives us more or less enough data to train selected word embedding models and conduct the powerful comparison.

Character detection: The problem of character detection was not a focus of our work, it has already been tackled for example by (Vala et al., 2015). We applied a very simple heuristic, which selects the 30 most frequent names of characters from the total list of characters – most frequent in the sense of counting the number of appearances per character in different chapters. If a character appears in various different chapters of the book series, this strongly hints at importance of the character to the story.

Relation selection: In order to make the results comparable for any of the three methods, we did the following: For any character in the list of 30 characters: get the *two strongest connections* to other characters on the list.

As described in Section 3, for method (i), (ii) and (iii) we selected the two characters with the strongest relation by co-occurrence between characters on a chapter, paragraph and sentence level, and for methods (iv)-(vi) we applied the different word embedding methods and tools.

For word embedding LT we used the gensim-word2vec toolkit. With Gensim, for any given character we compute the similarity to any other character – and then pick the two characters with the highest (cosine) similarity as strongest relations. Gensim[3] is a Python library (Řehůřek and Sojka, 2010), which provides tools for unsupervised semantic modeling from plain text, and also includes an implementation and extension of the original word2vec tool, which was written in C.

Method setup:

Co-occurrence: The computation of co-occurrence statistics $conf$ does not require any specific efforts. We introduced a threshold values: if B is among, say, top-3 candidates according to $conf$, we consider A and B similar and draw the corresponding vertex in the social network.

Word2vec: The corpus size of 6.9MB is a very small for word2vec standards, so it was interesting to see if word2vec will nevertheless produce good results. After a cleanup of the corpus (eg. removing

[3] http://radimrehurek.com/gensim

numbers and punctuation), we trained a CBOW model with 200 dimensions and a word-window size of 12. Those models are then used with gensim-word2vec, both for loading the pre-trained binary word2vec models, and for computing the similarity between terms (book characters).

GloVe: We applied the same basic settings as with word2vec – most importantly setting word vector length to 200 dimensions. For any other settings we kept the GloVe defaults. The resulting GloVe word embeddings were also used with Gensim, for this we adapted the following script[4].

Word2vec-f: Again, we trained vectors with 200 dimensions, but this time on the results of the Stanford Dependency Parser on the ASOIF books in CONLL-X format. After some tweaking, the trained model could be loaded with Gensim.

Crowdsourcing setup: The task of the user was the same as for the tool-based methods: Select the two characters with the strongest relation to the input ASOIF character. As options, we gave the users the whole set of candidates generated by the six methods to be evaluated, and also added a few random other characters to the list. Figure 2 shows a screenshot of a sample evaluation question posed to crowd

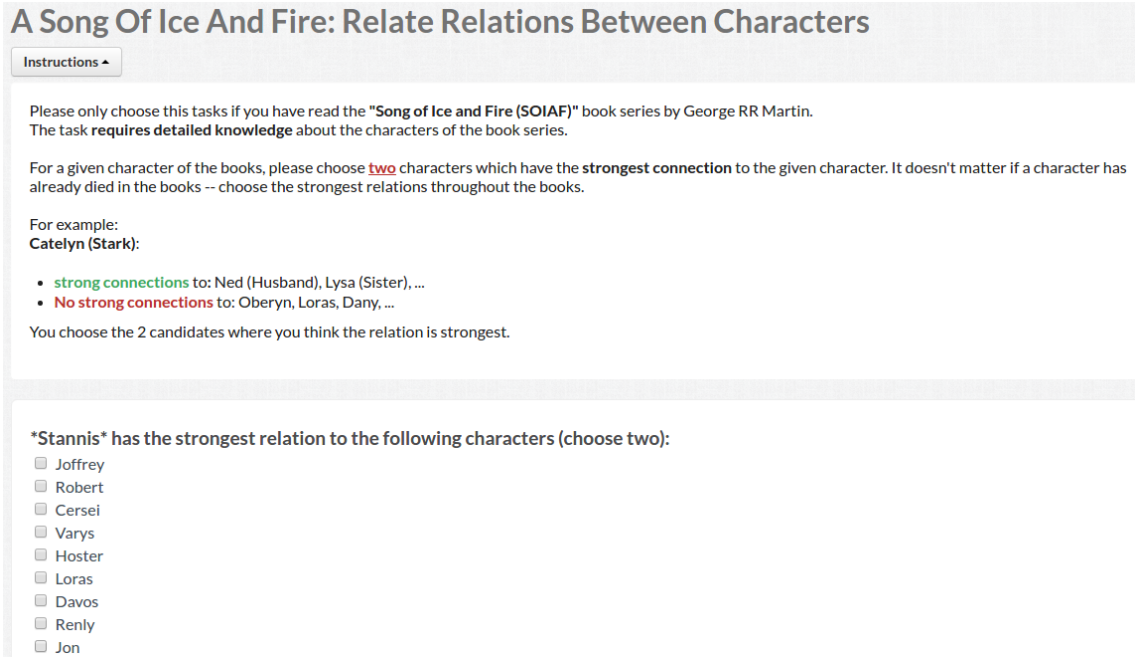

Figure 2: Screenshot of a CrowdFlower unit.

workers. We decided not to use all 30 available character options to be presented to the users, as this would be too many options to choose from, and be overwhelming and prohibitive for many users.

We used the following CrowdFlower settings: for all 30 units of work, we had 15 judgements each. We only allowed the highest quality workers (level 3), and we carefully designed test question to filter workers who lack knowledge about the ASOIF world. Workers had to answer at least 80% of test questions (gold units) correctly in order to be accepted to the job. The test questions were carefully created manually and test for general knowledge about the ASOIF universe.

4.2 Evaluation Results

We used the results of crowdsourcing as a gold standard, and compared them to the results for the automated methods (LT methods). We are aware that using results from crowd workers as gold standard is not without risk – so we also manually inspected the results retrieved to ensure high quality.

The crowdsourcing platform we used, CrowdFlower (CF), yields two types of results, the aggregated results, and the detailed results. In the aggregated results, CF gives exactly one ASOIF character which

[4] https://github.com/manasRK/glove-gensim/blob/master/glove-gensim.py

LT-Method	Top 1	Top 2
15 votes: Method i (Chapter Co-occ)	16.7%	50.0%
15 votes: Method ii (Paragraph Co-occ)	33.3%	63.3%
15 votes: Method iii (Sentence Co-occ)	33.3%	53.3%
15 votes: Method iv (word2vec)	36.7%	70.0%
15 votes: Method v (GloVe)	26.7%	53.3%
15 votes: Method vi (word2vec-f)	16.7%	20.0%

Table 1: Percentage where the suggested character of CF is also the No. 1 selection by the LT-method (Top 1), and where it is among the top 2 of automatically generated relations (Top 2).

has, according to the crowd workers, the strongest connection to the input character. And in the detailed results, CF gives all the single votes which where done by the individual crowd workers, which we then used to select the characters with the strongest connections. We used both aggregated and detailed results for evaluation, in Sections 4.2.1 and 4.2.2, respectively.

4.2.1 Aggregated CrowdFlower Results

As mentioned, in the aggregated results, the algorithms of CF select *one* character with the strongest connection to the input. As a first evaluation of the LT-generated relations we applied two scores:

1. The ratio of results where the character suggested by crowdsourcing is also the number one choice by the LT tool-based method (Top 1).

2. The ratio where the one character suggested by CF is among the top 2 of persons suggested by the LT method (Top 2).

Table 2 shows that paragraph-level and sentence-level co-occurrence, and also word2vec clearly outperform methods (i) chapter-level co-occurrence and (vi) word2vec-f. Agreement on the single most strongly related character (Top 1) is generally rather low, with values below 40%. But this is not unexpected, as it is a highly subjective choice if a book character has a stronger relation to his wife, his kids, or his best friend, for example. The *Top 2* score is much higher, up to 70%, which means that in 70% of cases the most related character selected by the crowd workers, is also in the top 2 picks of the LT methods. The best score here is provided by method (iv) word2vec, indicating that word embeddings can be very well suited for this task.

4.2.2 Detailed CrowdFlower Results

CF also provides all the individual votes of the crowd workers. As stated, we had 15 workers voting on each question. From the individual results, we selected the two characters that had the highest number of votes by the workers. Then we calculated the agreement between CF workers (the gold standard) and the tool-based methods with following formula:

$$score = Avg(\frac{A \cap B}{B}) \tag{2}$$

In this equation, A refers to the set of ASOIF characters suggested by the LT method, and B is the set of characters suggested by crowd workers. So, basically, we compute the average number of characters which are among the top two in the set of CF characters (our gold standard), and which are also in the top two characters suggested by the LT methods.

For example, if CF says that the set of (Renly, Davos) has the strongest connection to input character *Stannis*, and one of the LT tool-based methods suggests (Renly, Robert), then we have a 0.50 score on this single character. The final score then gives the average over all 30 input characters.

Similar to the results in Section 4.2.1, also with this score, methods (ii) to (v) are clearly more successful than chapter-based co-occurrence of characters and Word2vec-f. Again, word2vec had the highest score with a 58.3% match according to our metric. The GloVe word embeddings, and paragraph- and

LT-Method	Agreement
15 votes: Method i (Chapter Co-occ)	33.3%
15 votes: Method ii (Paragraph Co-occ)	53.3%
15 votes: Method iii (Sentence Co-occ)	51.3%
15 votes: Method iv (Word2vec)	58.3%
15 votes: Method v (GloVe)	53.3%
15 votes: Method vi (Word2vec-f)	26.7%

Table 2: Agreement between the sets of suggested character relations from CF workers and the LT methods, according to the score in Eq. 2.

sentence-level co-occurrence performed well, Word2vec-f and chapter-level co-occurrence are not suited for the job.

4.3 Discussion

The results are well in line with intuition and also with results from previous research. Confirming the results by (Ghannay et al., 2016), word2vec outperforms GloVe on word similarity tasks, while GloVe is superior on word analogy. For social network extraction, the word similarity feature is more important. Furthermore, as intuition suggests, chapter-level co-occurrence is not an optimal measure for relatedness between book characters.

Many interesting observations can be made about the method (vi) word2vec-f. As (Levy and Goldberg, 2014) argue, this method detects functional, not topical similarity, it gives words of same semantic type. For example for an input word like *go* it might suggest *run*, and *walk* as similar. In our task setup, this method is not well suited, as all input words are of same semantic type (book character) already. So word2vec returns words that associate with another, while word2vec-f suggests words that behave like one another. For extraction relations we seek primarily for associations between words.

Finally, with regards to research question (iii) and crowdsourcing itself, it is rather surprising how well crowdsourcing platforms like CrowdFlower seem to be suited even to address such specialized evaluation tasks such as relations between characters in the ASOIF book series. The high quality of results by crowdsourcing was confirmed by human inspection.

5 Conclusions

We considered the task of extracting a social network for literary texts and addressed a few main questions: do word embeddings outperform simple statistical similarity coefficients for our task? Which types of word embeddings are the most efficient? Is paid micro-task crowdsourcing suited to evaluate social networks extracted from literary texts? We came to the following results:

(i) To evaluate the quality of extracted social networks, we used the results of a crowdsourcing survey and the level of agreement between the crowd workers and the social networks extracted with language technology tools as the main quality measure. Although the social network of the highest quality is achieved by using the word2vec word embeddings toolkit, we can't say, that the co-occurrence statistics results are significantly worse, especially, when applied on paragraph level. Using GloVe embeddings we get a similar level of agreement, followed by the co-occurrence statistics applied to sentences. There are not drastic differences on the level of agreement, hence, we cannot say clearly, that one type of measures is significantly better than another;

(ii) There is a clear evidence, that word2vec-f embeddings are not suitable for the task.

(iii) Our results suggest that paid micro-task crowdsourcing is well suited to provide evaluation data even in such a specialized domain.

We have faced the following issues. First of all, there are some issues concerning the character names. Some character names have two or more forms (Dany and Daenerys, for example). Thus straightforward extraction of names will result in poor frequencies, and a tool for matching name forms should be applied. Some character names coincide (Jon Arryn and Jon Snow, for example). This fact can also spoil the

frequencies for the co-occurrence coefficients and for word embedding similarity. In future work we will train the model based on improved entity extraction and character name disambiguation. Secondly, since there is no clear algorithm for setting up the thresholds for any type for similarity measure, we struggle with the problem of choosing the number of possible edges for the given node of the network.

Our main future direction is to introduce the time axis in our experiments. Since there is a clear timeline in the SOIAF books, we can extract a dynamic social network, which will show how intensive the characters interact during different time spans. This will require from us the improvement of word embeddings similarity measures to a dynamic case and also a more complex design of the crowdsourcing validation.

6 Aknowledgements

The article was prepared within the framework of the Basic Research Program at the National Research University Higher School of Economics (HSE) and supported within the framework of a subsidy by the Russian Academic Excellence Project '5-100'. This work was also supported by RFBR grants 16-01-00583 and 16-29-12982.

References

Apoorv Agarwal, Anup Kotalwar, and Owen Rambow. 2013. Automatic extraction of social networks from literary text: A case study on alice in wonderland. In *Sixth International Joint Conference on Natural Language Processing, IJCNLP 2013, Nagoya, Japan, October 14-18, 2013*, pages 1202–1208. Asian Federation of Natural Language Processing / ACL.

Kalina Bontcheva, Ian Roberts, Leon Derczynski, and Dominic P. Rout. 2014. The GATE crowdsourcing plugin: Crowdsourcing annotated corpora made easy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 97–100. The Association for Computer Linguistics.

Asli Celikyilmaz, Dilek Hakkani-Tur, Hua He, Greg Kondrak, and Denilson Barbosa. 2010. The actor-topic model for extracting social networks in literary narrative. In *Proc. of NIPS 2010 – Machine Learning for Social Computing*, page 7pp.

David K. Elson, Nicholas Dames, and Kathleen R. McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 138–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sahar Ghannay, Benoit Favre, Yannick Estve, and Nathalie Camelin. 2016. Word embedding evaluation and combination. In Nicoletta Calzolari et al., editor, *Proc. of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France, may. European Language Resources Association (ELRA).

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308, Baltimore, Maryland, June. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jonathan Mortensen, Mark A Musen, and Natasha F Noy. 2013. Crowdsourcing the verification of relationships in biomedical ontologies. In *AMIA*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Ivan Rodin, Ekaterina Chernyak, Mikhail Dubov, and Boris Mirkin. 2016. Visualization of dynamic reference graphs. In *Proceedings of the 10th Workshop on Graph-Based Methods for Natural Language Processing*, NAACL'16, pages 34–38.

Marta. Sabou, Kalina. Bontcheva, and Arno. Scharl. 2012. Crowdsourcing Research Opportunities: Lessons from Natural Language Processing. In *Proceedings of i-KNOW '12*, pages 1–8. ACM.

Hardik Vala, David Jurgens, Andrew Piper, and Derek Ruths. 2015. Mr. bennet, his coachman, and the archbishop walk into a bar but only one of them gets recognized: On the difficulty of detecting characters in literary texts. In Llus et al. Mrquez, editor, *EMNLP*, pages 769–774. The Association for Computational Linguistics.

Gerhard Wohlgenannt, Marta Sabou, and Florian Hanika. 2016. Crowd-based ontology engineering with the ucomp protege plugin. *Semantic Web Journal (SWJ)*, 7(4):379–398.