Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2023”

Moscow, June 14–17, 2023

Remus, Lupin and Moony Walk in a Bar… Grouping of Proper Names Related to the Same Denotation in Large Literary Texts Collections

Zykova V. I.
National Research University
“Higher School of Economics” /
Pokrovsky Bulvar, 11,
Moscow 109028 Russia
vzykova2001@gmail.com

Klyshinsky E. S.
Keldysh Institute of Applied
Mathematics of Russian Academy of
Sciences / Miusskaya sq., 4
Moscow, 125047, Russia
klyshinsky@mail.ru

Abstract

In this article, we present a method of anaphoric proper names detection in fictional texts using Word2Vec model and algorithms of community detection on graphs. This method allows grouping different namings of a single entity and can be useful as a part of preprocessing texts for further analysis such as building social networks or training neural models. The method uses large text collection, related to the same domain. The foundation of the method is training of a Word2Vec model using information on direct characters interactions. This model allows building a social graph of characters. Than, the Louvain algorithm is used to divide the graph into communities containing different names of characters related to the same denotation.

Keywords: social network; graphs; word2vec; anaphora detection; fiction; Louvain algorithm
DOI: 10.28995/2075-7182-2023-22-1150-1157

Заходят как-то в бар Ремус, Люпин и Лунатик... Метод объединения имен собственных, имеющих общий денотат, в больших коллекциях художественных текстов

Зыкова В. И.
Национальный исследовательский
университет «Высшая школа
экономики» / 109028, Москва,
Покровский бул., д. 11
vzykova2001@gmail.com

Клышинский Э. С.
Институт прикладной математики
им. М. В. Келдыша /
125047, Москва,
Миусская пл., д. 4
klyshinsky@mail.ru

Аннотация

В этой статье мы представляем метод определения именований героев художественных произведений, относящихся к одному денотату, использующий модель Word2Vec и алгоритмы выделения сообществ на графах. Метод позволяет объединять различные названия одной сущности в группу с достаточно высокой точность и может быть полезен при использовании в качестве этапа препроцессинга текстов для дальнейшего анализа: построения графов социальных отношений или обучения нейросетевых моделей. Метод применяется к большой коллекции текстов, относящихся к одному домену. Основой метода является обучение модели Word2Vec с использованием информации о прямом взаимодействии героев. Модель служит для построения графа связей между героями, из которого при помощи Лувенского алгоритма выделяются сообщества, содержащие разные названия одного героя.

Далее проводится фильтрация разных героев, объединяемых моделью.

Ключевые слова: графы социальных отношений; графы; word2vec; определение анафоры; художественные тексты; Лувенский алгоритм

1 Introduction

Investigation of interaction among characters of literary works allows better understanding the overall plot of a masterpiece. One of the commonly used representations of such interactions is a graph. Its nodes represent literary characters, described in an investigated work, edges represent such interactions among characters as conversations, direct actions, and being in the same place. Investigation of large graphs allows describing two types of those: Barabasi-Albert [1] and 'Small World' [2]. The authors of [3] found that graph of character interactions has properties of the further and does not meet all the properties of the later.

There are several approaches for coupling nodes depending on the used information. In [4], an edge in the graph is created between two characters "if they were both syntactic arguments under the same predicate or appeared as two conjuncts". For the sake of community detection, the author used the Louvain algorithm [5]. Another approach is using mutual appearance in the same piece of text [6].

Investigation of drama corpora (e.g. [3]) has a big advantage. Such a corpora was preliminary tagged including information on phrases attribution by characters. Thus, it is easier to use the common appearance of characters in the same part of text or information of whom their phrases are addressed in order to build the social graph of a masterpiece.

However, researchers are faced with the problem of several names for a character. In different parts of a text, an author can use the name of a character, his or her surname, their combination, position, gender names (boy, girl, …), and other substitutive nouns. Thus, there is a problem of joining all of these names in the same cluster. Manually tagged corpora have an advantage that all these names can be unified by an assessor [6]. But that is not the case of automatically tagged large corpora. One of the solutions in the later case is preprocessing, i.e. selection of different names of the same character and automatic replacement of a name occurrence by the unified one. But it does not work in the case of a big number of unknown characters. Methods of anaphora resolution (see, e.g. [7]) could not help find different names for main characters.

In this paper, we introduce a new method for calculation of similarity between characters' names. The method is able to cluster names of the same character in a community of a social graph. The method is based on training a Word2Vec model on entity names co-occurrence, building a social graph according to similarity between those entities calculated by the model, and applying the Louvain algorithm for the sake of community detection.

2 Used Data

2.1 Text Collection

As a source data for our method, we need a collection of mono-domain texts, e.g. works created by one author or dedicated to the same fictional setting. We use a collection of fanfiction texts based on J.K. Rowling "Harry Potter" books. There are several reasons for such a decision.

- Authors of fanfiction texts use original setting or at least original characters as the basis of their work; so, all the texts sharing the same original subject and the same setting would share almost the same characters space and could be treated as a single domain.
- "Harry Potter" as a fiction is extremely popular, this fact provides us with a large amount of data. This, for its turn, gives us an opportunity to balance the imperfect performance of language models.
- Most of the fanfiction sites are open resources which can be easily and automatically crawled.

Considering all these facts, we chose the https://fanfics.me/ site. All the fanfiction texts belonging to the "Harry Potter" setting were crawled from this site in summer 2022 and then used in the algorithm described below.

The amount of data collected is 22252 texts which have around 336 million of tokens in total. This is a large collection of data so we have to take it into consideration while making a decision about preprocessing algorithms which will be discussed below.

2.2 Text Preprocessing

Our algorithm requires syntactically parsed natural texts. There are a variety of open source tools performing this task. Such tools can be multilingual and support Russian alongside many other languages or keep it as the only or almost only one available. For example, Spacy [8] and Stanza [9] provide models for more than twenty languages each while Natasha and DeepPavlov [10] are concentrated mostly on models for Russian language. There is also a UD-Pipe module but it is almost deprecated.

Besides the list of supported languages, mentioned models have different quality and capacity features: some of them are relatively precise but the level of their efficiency is quite low, some, in contrast, are surprisingly fast but their quality is below the acceptable level. Considering preprocessing as an important step which can dramatically affect all future calculations, we tend to prioritize quality over productivity. However, as it was already mentioned, we are working with a large amount of textual data processing which can require an inappropriate amount of time, thus we have to search for compromises. Because of all mentioned above, we decided to use the Spacy framework as it is relatively fast even without using a GPU and shows an appropriate, but not perfect, performance.

The next step of preprocessing is Named Entity Recognition (NER) which is an ongoing and difficult task itself; however, we need it to shape the initial list of characters. Note that on some level of precision, this list can be gained on the stage of syntactic parsing since most parsers have the PROPN tag in their tagset. Considering this, using NER is not so much about detecting entities in general but about detecting multi word namings as it is usually beyond the capabilities of syntactic parsers.

As a result of data preprocessing we should have a corpora of syntactically parsed texts with results of the NER algorithm. We used the one proposed by Spacy as it can be easily added to its pipeline.

3 Grouping of Proper Names Related to the Same Denotation

As can be seen from the task formulation, our algorithm can be divided into two parts. The first one is collecting of the characters co-occurrence information from the texts; so as it can be utilized as training data for the further model. The second part is the model training.

For the sake of model training, we need a list of pairs of characters which satisfies the condition of being “interacting”. Thus, the first stage can be divided in several subtasks too. In the beginning, we need to define what “character” and “interaction” are in terms of grammatical traits obtained from the preprocessing step; then the algorithm of applying these conditions to the process of list of pairs extraction should be described.

We define an entity (not necessarily named) or a character (if we are speaking about it in terms of fiction terminology) as a token which meets one of the following conditions: it is tagged as an entity by the NER model; it is tagged as a proper noun by the parser; it is tagged as an animate noun.

There are several reasons for such a decision. To begin with, we want proper names to be considered as an entity. However, neither syntactic parsers nor NER models have good precision, while we are speaking about them in terms of entity extraction from fictional texts. However, there is a chance that mistakes of the syntactic parser can be smoothed by NER model predictions and vice versa, so we decided to unite their decisions. Moreover, additional difficulties are provided by the fact that we need not only named entities but also their substitutive nouns to be detected. Here we hope that inanimate objects could not be considered as fictional characters (that is not the case of fairy tales, which are not the subject of this article). So, we select all animate nouns as hypothetical entities entailed with some of the characters.

After we proposed a definition of an entity, we now can discuss conditions of being interacting. Following [4] line of thought, we decided to use being descendants of one vertex in the dependency tree as a definition of connection existence. Unlike the approach described in the mentioned work where the author uses all predicates as the parenting vertices, we decided to make constraints on such vertices stricter: we consider only verbs being a predicate. Summarizing this with a part about entity definitions we can postulate the following definitions of a pair: it consists of the two entities which are the descendants of the same verb in a dependency tree.

We have found that a pair of characters can be considered as a context to each other. This allows us to train a language model treating pairs and entities as sentences and tokens respectively. In this project, we decided to train a Word2Vec model [11] in order to get vector representation (embeddings) for each

character. There are several reasons for choosing this architecture. On the one hand, Word2vec is a static language model that provides us with a single embedding for each entity from vocabulary. Since we want to get a method providing us with an absolute measure of semantic similarity between entities, the ability to calculate an average representation of a token becomes useful. On the other hand, considering usage of a contextualized model, we understood that the described way of representing context does not provide enough information for the stable learning of a contextual model.

Now we can now describe the process of obtaining the trained model. At the beginning all the entities should be extracted keeping their affiliation with sentences since we will need this information to make a decision about connection existence. Then for every possible pair of entities within a sentence it should be checked if these entities are the descendants of the same verb in the dependency tree. As a result of the described process, we have a list of pairs of characters ordered by their appearance in our collection of texts. Then it is used to train a word2vec model so that characters in every pair have only each other as a context (it is guaranteed by the window size parameter of a word2vec training algorithm).

Using the trained model, we then build a social graph of characters. At the first stage we select all named characters (entities which are marked by NER algorithm or are a proper noun) with frequency more than the reference minimal frequency (let be equal to 50) as initial nodes of the graph. Then for every node we extract top-100 nearest neighbors having cosine similarity more than 0.5 and frequency more than 20. Then we create an edge between two nodes if at least one of them is in the top-100 list for another even if they are both initial nodes. The result is a weighted graph where weights are designated by the cosine similarity between nodes.

As it noted in [13] “The weight of a link between two nodes in a social network can be used to represent the similarity of two characters. The larger the weight is, the more likely the two characters will be related to each other.” However, it appears that cosine similarity is not enough. There are named entities, which have a relatively high proximity but have to be divided, such as twin brothers Fred and George Weasley, who are always acting together, or Harry Potter and Hermione Granger, who share their adventures. However, there are entities which have smaller cosine similarity but have to be considered as a single entity. That is the case of different namings of one character whose frequencies differ dramatically.

In [4], characters which belong to the same predicate are considered as two different persons. In our case, if two extremely similar characters frequently appear as syntactical siblings in a parsed text, then they should be considered as two different persons. However, our experiments demonstrated that such an approach fails in case of less frequent characters as they often provide lesser statistics than we need for their separation. Note, that there are a variety of characters having both less and more frequent names. That is why we need here a community detection algorithm which will separate entities considering the whole picture of their relations.

Following [4], we decided to use the Louvain algorithm [5]. As a result of applying this algorithm to a social graph we obtain groups of entities which have closer relations. By changing the resolution parameter of a Louvain algorithm, we can control the density of achieved communities as its high value makes the algorithm favor smaller communities. In our case it means that we can decide if communities should contain names of different but narratively close characters or, if using higher values of resolution parameter, different name of the same character. Although all of the mentioned parameters (threshold of cosine similarity, minimal frequencies and resolution parameter) make the precision higher, we cannot just make them as high as possible since in the extreme case we would get an empty graph or a list of one-node communities which makes all the process meaningless.

4 Results of Experiments

We trained a model on 5 epochs and words of the minimal frequency 10. We were surprised to see that a model that was accidentally trained with pairs duplicates (pairs like “Harry Ginny” and “Ginny Harry” were both added to the list) showed a more stable performance and more representative results; so, we decided to use it as a basic approach in the further work. We used all entities lowercased and lemmatized; we also joined multiword entities extracted with NER model into single strings.

However, the first attempt of training the model showed that there are a lot of mistakes of lemmatization and pos-tagging. We decided to begin with solving the first problem. In order to map different forms of the one entity, we built a graph where an edge between two entities means that they have

Levenshtein distance [12] equal to 1 (see Fig. 1). After we got such a graph, we considered one connected component to be different names of a character; however, it happened to need a little manual correction. After that we mapped all the entities in a connected component to a single character's name which was the most frequent one in the original component. Then we retrained a model.

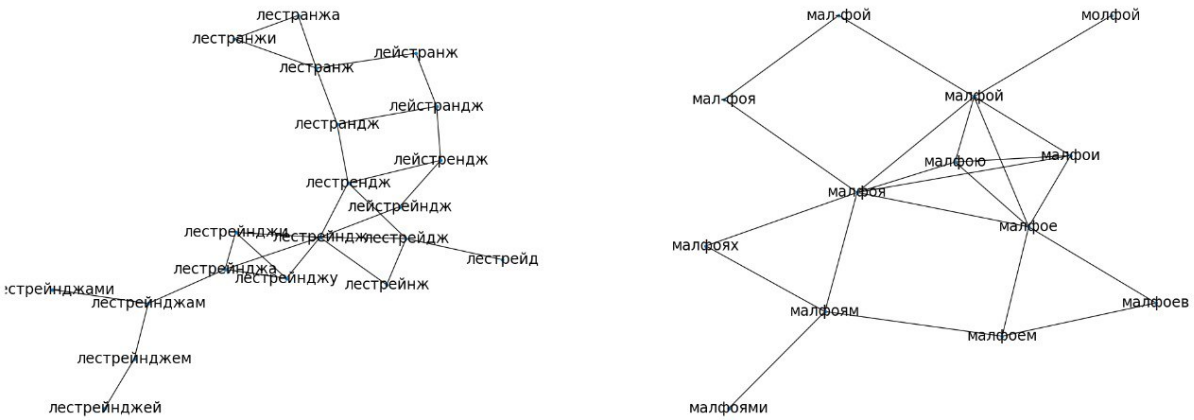

Figure 1: Graph of tokens having the Levenstein distance equal to 1

After retraining, the solution for the problem of incorrect pos-tagging (that actually means that we had adverbs or adjectives marked as entities) happened to be found. It appears that almost all incorrectly tagged words formed a single connected component (see Fig. 2), so we merely removed from the training data all the pairs including these tokens. Then we retrained the model again.

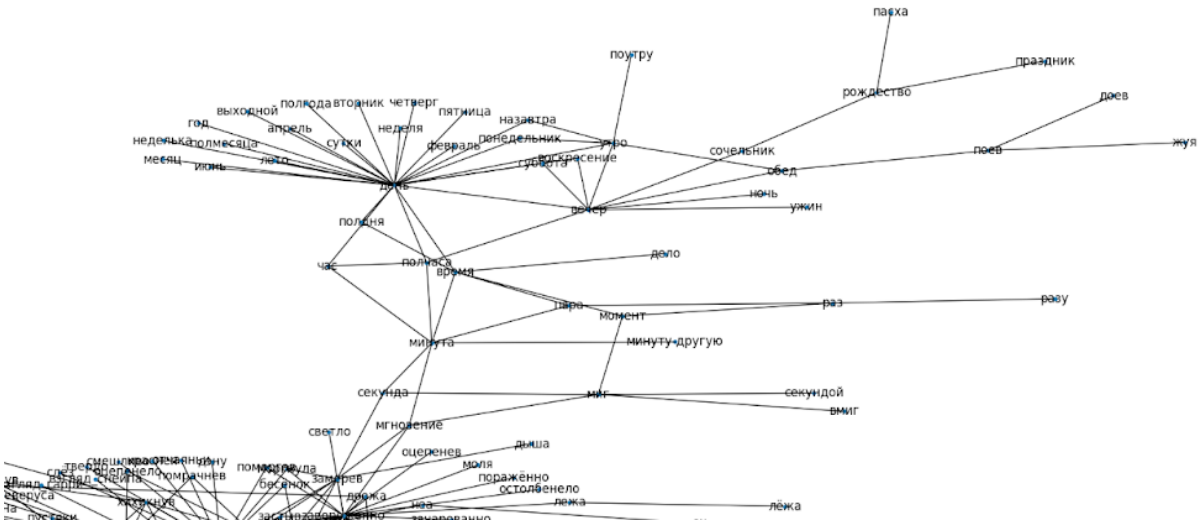

Figure 2: Graph of entities connections for the component with pos-tagging mistakes

Having a final model, we repeated all the process of graph building. The resulting graph had an interesting structure: there was a big connected component which contained almost all main characters and needed to be divided into parts (see Fig. 3) and a lot of small ones. As can be seen on the picture, the main connected component itself has several easily distinguished parts such as the one in the upper-left part which is devoted to Ministry of Magic characters or the bottom part whose components are all about geographical names both real and magical. Sadly, we still do have a small amount of mistagged words but there are already much less of them than there were in the previous iterations.

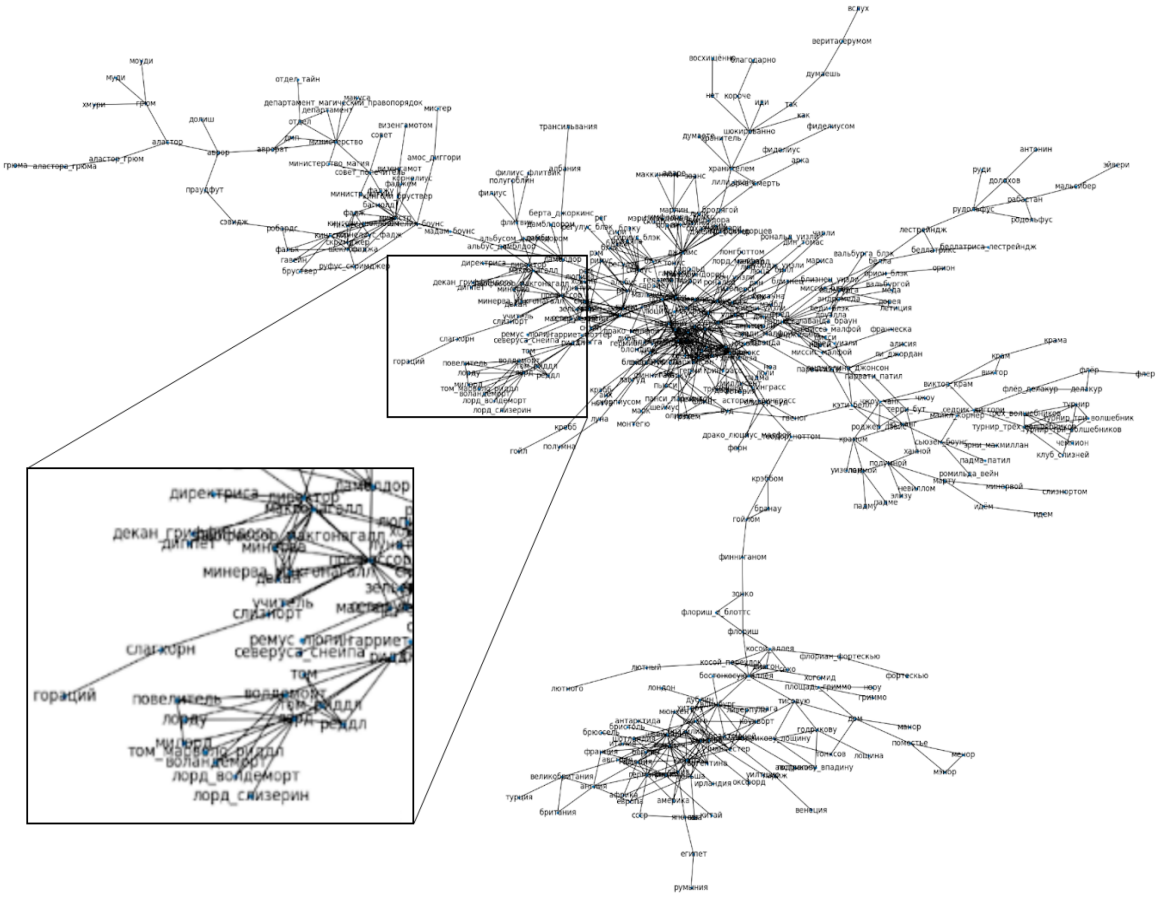

Figure 3: Graph of entities connections for the biggest connected component

As for the resolution parameter, we can start with 10 as the lower values of the algorithm does not really distinguish anything (it should be mentioned that this parameter has to be selected independently for every graph). However, this value appears to be too low and provides us with large groups of connected yet not the same characters (see Fig. 4).

министр магии, робардс, фаджа, сэвидж, фаджу, визенгамотом, шеклболт, фадж, корнелиус, скримджер, министр, кингсли, бруствер, визенгамот

америка, уэльс, испания, канада, шотландия, франция, германия, швейцария, греция, италия, ирландия, китай, бразилия

петрификус тоталус, инкарцеро, секо, экспеллиармус, петрификусом, протего, ступефаем, петрификус, ступефай, экспеллиармусом, непростительный

косой аллея, зонко, лютный, коукворт, лондон, манчестер, флориш, косой переулок, косую аллея, хогсмид, эдинбург

учитель, директор, профессор, флитвик, филиус, минерва, мастер зелий, минерва макгонагалл, диплет, декан, макгонагалл

риддл, том риддл, лорду, лорд, реддл, том, волдеморт, лорд Волдеморт, повелитель, воландеморт, милорд

джим, лили эванс, эванс, лили, сохатый, шмэри, марлин, адаре, джеймс поттер, джеймс, скорп

дин томас, уизли, рон, рон уизли, рональд, рыжик, уизел, дин, рональд уизли, перси

блэка, тонкс, блек, бродяга, нимфадора, сириус, блэку, блэк, сириус блэк, сири

чарли, джордж, фред уизли, анджелина, фред, кэти белл, джордж уизли, билл, кэти, близнец |

Figure 4: Biggest 10 communities detected by the Louvain algorithm with resolution parameter equal to 10

If we increase the value to 20, we will still not get enough precise groups (see Fig. 5), but they will become more clean.

министр_магии, фаджа, визенгамотом, фадж, корнелиус, скримджер, министр, фаджу, визенгамот
андромеда, друэлла, дорея, нарцисса, цисси, нарцисса_малфой, вальбурга, меда, нарси
совы, филин, букля, буклю, сова, сову, хедвиг, сов
блэка, блек, бродяга, сириус, блэку, блэк, сириус блэк, сири
джордж, фред уизли, анджелина, фред, кэти белл, джорджуизли, кэти, близнец
когтевран, гриффиндором, хаффлпафф, слизерин, пуффендуя, слизерином, рейвенкло, гриффиндор
долохов, мальсибер, руди, рудольфус, антонин, рабастан, родольфус, эйвери
кэрри, геланор, гарольд, гарриет, марьяна, эмили, гарри
дурсли, мардж, петунья, эйлин, туни, вернон, дурслей
косой аллея, зонко, косой переулок, косую аллея, лютный, хогсмид, флориш

Figure 5: Biggest 10 communities detected by the Louvain algorithm with resolution parameter equal to 20

Finally, the resolution parameter equal to 30 provides us with the best compromise between precision of the groups and the proportion of connections detected (see Fig. 6). As it can be seen on the picture, there are still groups which need further decomposition (such as the group with Harry which contains him and some marginal characters that are even not the part of Harry Potter original characters), but the fact that we can connect naming that are not connected in any way but semantically (such as Remus Lupin and Moony) make us consider this algorithm to have a pretty high precision and be at least a method of getting the baseline for more precise yet computationally complex models.

совы, хедвиг, филин, букля, буклю, сова, сову, сов
блэка, блек, бродяга, сириус, блэку, блэк, сириус блэк, сири
кэрри, геланор, гарольд, гарриет, марьяна, эмили, гарри
дурсли, мардж, петунья, эйлин, туни, вернон, дурслей
хогсмид, косой аллея, зонко, косую аллея, лютный, флориш, косой переулок
лунатик, люпина, люпин, рем, ремус люпин, ремус, римус
долиш, аврорат, министерство, отдел, отдел тайн, министерство магия, аврор
кричер, кикимер, добби, тилли, эльф, винки, домовик
министр_магии, фадж, фаджа, скримджер, корнелиус, фаджу, министр
лорду, лорд, воландеморт, милорд, волдеморт, лорд волдеморт, повелитель

Figure 6: Biggest 10 communities detected by the Louvain algorithm with resolution parameter equal to 30

5 Conclusion

In this article, we present a method of anaphoric proper names detection in fictional texts using Word2Vec model and algorithms of community detection on graphs. This method allows grouping different namings of a single entity and can be useful as a part of preprocessing texts for further analysis such as building social networks or training neural models.

We applied our method to a large collection of fanfiction texts devoted to the Universe of Harry Potter (22252 of texts, 336 mln of tokens) and obtained 109 groups consisting of 356 named entities, including characters names, nicknames, faculties, toponyms, and artifacts.

Note that results need manual post-processing since there are some misplaced entities in communities. Our algorithm needs a pretty big collection of texts on the same topic since the Word2Vec model needs a big amount of contexts. Thus, it can be useful for preliminary processing of big textual collections as a builder of lists of anaphoric links.

One of the limitations of our method is it requires a large collection to train a Word2Vec model. It is hardly applied to a single masterpiece, even a huge one.

In further research we are planning to formulate the requirements for a corpus more precise since it can be useful for understanding the ways of overcoming existing limitations. Also, the algorithm of contextualized detections should be invented.

References

[1] Barabasi A. Network Science. — Cambridge university press, Cambridge. 2016. 453 p.

[2] Watts D., Strogatz H. Collective dynamics of «Small-world» networks // Nature, 1998. — Vol. 393. — P. 440–442.

[3] Trilcke P. et al. Theatre Plays as 'Small Worlds'? Network Data on the History and Typology of German Drama, 1730–1930 // Digital Humanities 2016: Conference Abstracts — Kraków, Ploand, 2016. — P. 385–387.

[4] Skorinkin D. (2017) Extracting Character Networks to Explore Literary Plot Dynamics // Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2017" [Komp'yuternaya Lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoy Konferentsii "Dialog 2017"]. Moscow, pp. 257–270.

[5] Blondel V. D., Guillaume J., Lambiotte R., Lefebvre E. Fast unfolding of communities in large networks. — Journal of Statistical Mechanics: Theory and Experiment. — Vol. 10, p. 10008

[6] Stiller J., Nettle D., Dunbar R.I.M. The small world of Shakespeare's plays. — Human Nature. — Vol. 14, No. 4, pp. 397–408.

[7] Toldova S. Ju., Roytberg A., Ladygina A. A. at al. (2014) RU-EVAL-2014: Evaluating Anaphora and Coreference Resolution for Russian. // Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialog 2017" [Komp'yuternaya Lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoy Konferentsii "Dialog 2017"]. Moscow, pp. 681–694.

[8] Honnibal M., Montani I. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. 2017

[9] Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton and Christopher D. Manning. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In Association for Computational Linguistics (ACL) System Demonstrations. 2020.

[10] Burtsev M., SeliverstovA., Airapetyan R.et al. (2018) DeepPavlov: Open-Source Library for Dialogue Systems // Proc. of the 56th Annual Meeting of the Association for Computational Linguistics-System Demonstrations, pp. 1–6

[11] Mikolov T., et al. Efficient Estimation of Word Representations in Vector Space // In Proceedings of Workshop at ICLR. 2013

[12] Levenshtein, V. I. Binary codes capable of correcting deletions, insertions, and reversals // Soviet Physics Doklady. 10 (8), 1966, pp. 707–710.

[13] Fan C., Li Y. Network Extraction and Analysis of Character Relationships in Chinese Literary Works // Proc. of Computational Intelligence and Neuroscience. Vol. 2, 2022, pp. 1-10.