TU WIEN Informatics

Generating Knowledge Graphs with Specified Ambiguities

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Peter Kittenberger, BSc
Registration Number 01425768

to the Faculty of Informatics

at the TU Wien

Advisor: MSc PhD Marta Sabou, MSc PhD
Assistance: Ao. Univ.Prof. Dr. Stefan Biffl
Prof. Masaomi Kimura, PhD

Vienna, 12th June, 2021

Peter Kittenberger	Marta Sabou

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Peter Kittenberger, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 12. Juni 2021

Peter Kittenberger

Acknowledgements

I want to thank everyone who helped me over the course of not only this thesis, but also during the years of education necessary to write it.

My family, who supported me with their knowledge, opinions, curiousness and ideas. I'm especially grateful for my parents who allowed me to complete my university education without too much time- or financial pressure.

My professors, advisors and tutors, all researchers whose knowledge, feedback and work I was able to use and include in my work. Without the continuous feedback and insight into genuine research my advisors were able to give me, my work would not be what it is now.

My friends for the time and effort they put into my thesis, be it for answering questionnaires, correcting my writing or listening to my ideas and worries.

My university, for allowing me to organise my studies and thesis freely and not only enabling me to go and study abroad, but also for contributing financially.

Thank you!

Abstract

Large scale knowledge graphs are commonly used in software products ranging from web applications to the control software of self-driving vehicles. Due to their size, these graphs are usually built by either employing a crowd of people to build them or by scraping already existing information on the web. Both approaches require the collected data to be validated and improved before being suitable to be used in production ready systems. While much current research aims to explore and improve the algorithms required for this task, it is hampered *by the lack of annotated datasets containing typical human mistakes (or ambiguities)* such as those caused by ambiguous questions or answers. This problem intensifies if graphs have to follow certain restrictions to be of value (eg. containing specific relation types or classes of nodes as used by an existing system), and may even be impossible to solve if specific expert-graphs are required whose contents non-experts would struggle to comprehend. In addition to that, there is currently no existing solution capable of leveraging the structure of a knowledge graph as basis for artificial generation of mistakes.

To address this issue, in this thesis we propose an vector embedding based approach called "AmbiVec" to enrich arbitrary graphs with generated, human-like mistakes similar to those made by crowd workers or web scraping approaches. To this end, the adopted methodology includes (1) relying on *literature study* to investigate the most prevalent sources of ambiguities during crowdsourcing and categorise the mistakes that are caused by them; (2) based on these findings, the *design and implementation of an approach*, "AmbiVec", to generate configurable amounts of artificial mistakes, using vector embeddings and leveraging similarity between elements in the graph, so mistakes can then easily be used for research; (3) an *evaluation of the approach* using a crowd sourcing method.

Our evaluation shows that our approach works well for mistakes of a small severity that are commonly caused by existing crowd based approaches. User ratings of the severity correlate well with configured severity and workers categorised a portion of our generated ambiguities as being human-like.

Contents

CHAPTER 1

Introduction

In the introduction we will define the problems our thesis strives to solve and what research questions we have to answer to achieve a solution. We then explain our methodology and the main output we expect as a result of our work.

1.1 Problem definition

Large-scale knowledge graphs [EW16] are widely used in modern applications like semantic search, question answering systems, recommender systems, expert systems and personal assistants [GAS16] but their creation remains problematic for individual humans due to their size. To work around this problem graphs are often generated with the help of software. This software uses human knowledge as the basis for knowledge graph construction, either through a crowd sourcing or web scraping approaches, or graphs may be further refined using crowd algorithms [Pau17].

Since the knowledge comes from multiple persons with high likeliness of different answers or different interpretations of the objects and concepts queried about [IHvH+17], some low quality results are bound to be generated when merging their answers to one definitive statement, even when using optimal aggregation algorithms for specific datasets [ZLL+17]. Additionally, there could be other sources for ambiguity or mistakes. These sources and the resulting types of ambiguity in the knowledge graph are not yet well documented in literature and hard to categorise by their nature.

It is possible to use algorithms to detect and correct such mistakes [d'A09], but in order to develop and test these algorithms, knowledge graph based datasets containing relevant mistakes and including a gold standard are required. While we found signs of their existence, these datasets were not available due to various reasons (e.g., dead links, inadequate internal and external references, paywalls), even less in the required quantity and quality. In this master thesis, we address this general lack of knowledge graphs

with human-like ambiguity mistakes and propose an algorithm for generating them, thus supporting research on the development of algorithms for identifying such mistakes.

1.2 Research question

Our research question thus asks whether it is possible to generate datasets that fulfil the following requirements: (i) the ambiguities should resemble human mistakes closely enough to be of value for other algorithms and to not be easily recognisable as non-human by other humans; (ii) it is important to be able to vary the severity of the ambiguities to simulate everything from a small mistake to random answers; (iii) the generated ambiguities have to follow the format of a knowledge graph and make sense in that highly linked context; (iv) the introduced ambiguities also should be quantifiable in absolute numbers and percentual in relation to the size of the knowledge graph.

To reiterate in one sentence, the main research question of this thesis is:
"How can one introduce a well defined, structured and measurable amount of adequately human-like ambiguities as defined by our research into a pre-existing knowledge graph?".

Concretely, this overall research question was investigated through three more refined research questions as follows:

- **RQ1: Ambiguities and their sources** How can ambiguities in the area of crowdsourcing be defined and distinguished from similar concepts, what sources do these ambiguities stem from and what characteristics are shared between them?
- **RQ2: Creation of ambiguities** How can we best generate ambiguities in an efficient, robust and generalisable way for arbitrary knowledge graphs?
- **RQ3: Evaluation of ambiguities** How can we measure the quality of our generated ambiguities? Do they resemble human-generated ambiguities enough for use in practical applications and does our configuration work?

1.3 Methodology and approach

The thesis uses different methodologies for each research question, but basic literature research is employed for all research questions. Our research will focus on solutions that are at least partially similar to ours, with a focus on work simulating human mistakes, sources of mistakes and their qualitative measurement as well as knowledge graphs especially suited to demonstrate our approach. We will not include work that is purely focused on psychology. In addition to the methodology used for all research questions we will also focus on specific methodology for each research question.

- **RQ1 - Literature research** For our RQ1 additional literature research will define the kinds of ambiguity we are considering for our approach and further focus on the

reasons and causes for ambiguity caused by humans to combine them into sources. Based on those sources we then conclude what characteristics these ambiguities share and how to best describe them.

- **RQ2 - Algorithm design and implementation** Using the knowledge acquired in RQ1 we develop the "AmbiVec" workflow with the aim to emulate the found characteristics of ambiguities. The methodology used for this will follow the design science research as proposed by Hevner et al. [HMPR04].

- **RQ3 - Experimental evaluation** To ensure that our approach is able to achieve its goals we design and conduct an experimental evaluation using human computation and crowdsourcing. The results are then evaluated and summarised in the conclusion.

1.4 Main outputs

Our research will provide a number of outputs for each of our research questions.

- **RQ1 - Literature research** For this research question one output will be the definition of what constitutes ambiguities and the distinction to similar concepts (eg. bias). Another output will be the sources that lead to ambiguities, each named, including a definition and some examples. The third output will be a classification of the characteristics the individual ambiguities stemming from these sources show and which are the basis for the work done to answer RQ2.

- **RQ2 - Algorithm design and implementation** Based on the outputs of RQ1 we then propose the "AmbiVec" workflow as output of RQ2. This workflow is able to generate configurable, human-like ambiguities based an arbitrary knowledge graphs. It can easily be extended or adapted for specific graphs and offers the possibility to check and correct results of intermediate steps in order to improve quality and reduce computation time.

  For easier visualisation the workflow will also be shown by two diagrams, one focused on the users and one focused on the code. Both diagrams are necessary because required interactions vary depending on the view and both usage and modification of the code are equally important.

  To allow for easier modification, extension and configuration we also define the structure of configuration. This way interoperability between various methods can be ensured while keeping an intuitive interface for the user.

  We will provide methods as required by our literature research on ambiguity sources and additional methods we use for testing purposes. The methods will be easily expandable and both existing methods and the process used for including additional methods will be described in the thesis.

We also include a knowledge graph that we use for demonstrations and testing. This graph will contain all special cases we use for testing and some mock data for manual tests. The size of this graph will be small enough to complete the calculations required for the workflow almost instantly.

- **RQ3 - Experimental evaluation** As output of our RQ3 we propose the design for an experimental evaluation using crowdsourcing. We also run it, evaluate its results and conclude whether our outputs were able to satisfy the goals set by our research questions. Due to the unavailability of similar datasets containing a gold standard we also take care to preserve all of our outputs for future research.

In this chapter we stated that the availability of annotated knowledge graphs, while more relevant than ever, is still an open problem. To alleviate this we defined three research questions, the methodology necessary to answer them and what their outputs will be composed of. The first research question asks what ambiguities and their sources look like and what characteristics we can generate, the answers to this questions will be provided by a literature study. The second research question concerns itself with the creation of ambiguities and will be answered using the design science research methodology. The third research question aims to provide a conclusion on the quality of our generated ambiguities and will be answered using a questionnaire.

This thesis is further structured as follows. Chapter 2 introduces areas relevant to our research starting with insights on how our literature research was planned (see section 2.1). We then provide deeper insights into knowledge graphs (see section 2.2), the availability of datasets required for our research (see section 2.3), embedding technologies (see section 2.4) and crowdsourcing & human computation (see section 2.5).

In chapter 3 we first define what we consider to be ambiguities in contrast to similar concepts (see section 3.1) and then compile a list of sources these ambiguities can stem from (see section 3.2). We can then deduce the common emulatable characteristics that the individual ambiguities coming from those sources show (see section 3.3) and conclude our first research question.

In chapter 4 we start by explaining how artificial ambiguities can be generated using the "AmbiVec"-workflow (see section 4.1), the data structures required for it (see section 4.2) and the methods we use for the generation (see section 4.3). As the workflow allows for manual checks to recognise low quality output early in the process (see section 4.4) we explain those as well as the metrics used to measure the quality of generated ambiguities (see section 4.5).

In chapter 5 we introduce our reference implementation of the "AmbiVec"-workflow. We explain design decisions (see section 5.1), what libraries we used (see section 5.2) and how our methods can be extended (see section 5.3). We also briefly explain the main functions used by our implementation (see section 5.4) and provide some performance measurements (see section 5.5) as well as access to our repository (see section 5.6). At the end of this chapter we can conclude our second research question.

In chapter 6 we evaluate the output of our approach using a crowdsourcing platform. Starting with the purpose (see section 6.1) and users (see section 6.2), going over the tasks (see section 6.3), setup (see section 6.4) and procedure (see section 6.5) and ending with the analysis (see section 6.6) and conclusion (see section 6.7) all parts of our evaluation are explained. In the conclusion we are able to answer our third research question.

In chapter 7 we can then draw a conclusion on the outcome of all our research questions (see section 7.1). As we are subject to some limitations we expand on them (see section 7.2) and finally give some insight into further work that could improve our workflow in various areas (see section 7.3). This chapter marks the end of our thesis.

CHAPTER 2

Background and related work

In this chapter we first define the selection-criteria for literature we investigate and then explain the topics we identified as relevant (see section 2.1). Since these topics build upon each other we will start with knowledge graphs (see section 2.2), explain why finding a suitable graph was a challenge and list some graphs we investigated (see section 2.3). Next we will explain what embedding technologies are and what kind we chose to use in our thesis (see section 2.4). The last section will explain what crowdsourcing is and how it is used in our thesis (see section 2.5).

2.1 Literature research

Through extensive literature research existing solutions in the area of knowledge extension and the use of human computation are analysed and iterated on, this ensures a novel and well engineered solution that brings improvements to the state of the art and that provides a basis for further research. For each conference we investigated at least all works listed in the most recent proceeding and for journals we investigated submissions going back at least one year at the time of writing. Additional featured as well as unrelated works were investigated as part of an explorative literature research.

Since our research mainly concerns itself with two research areas we will list proceedings and journals grouped by those areas.

Human computation:

- HComp [CK18]
- CSCW [10.18]
- CHI [CHI19]

- IJHCS[1]
- JHC[2]

Semantic web:

- EKAW [ZGNT18]
- K-CAP [10.17]
- ISWC [VBSF+18a][VBSF+18b]
- ESWC [HFJ+19]
- BHCC[3]
- SWJ[4], JWS[5]

2.2 Knowledge graphs

Knowledge graph [EW16] is the name of a specific data structure that was originally coined by Google with the intention of modeling large, interwoven representations of real world concepts. We use this data structure as the starting point of our approach and make use of the highly interconnected structure to generate our ambiguities. Färber et al. [FBMR17] uses the following definition:

"We define a Knowledge Graph as an RDF graph. An RDF graph consists of a set of RDF triples where each RDF triple (s, p, o) is an ordered set of the following RDF terms: a subject $s \in U \cup B$, a predicate $p \in U$, and an object $U \cup B \cup L$. An RDF term is either a URI $u \in U$, a blank node $b \in B$, or a literal $l \in L$."

An example of this structure is shown in Figure 2.1 where blue circles represent nodes and arrows represent relations.

These graphs contain a network of information and can be queried and manipulated using "SPARQL", a query language made for the RDF format the graphs are often saved as. This format represents two nodes and one relation as a triple, one node is the subject and the relation (called predicate in this context) connects it to the object, the second node.

[1] International Journal of Human-Computer Studies. https://www.journals.elsevier.com/international-journal-of-human-computer-studies

[2] Journal for Human Computation. http://hcjournal.org/ojs/index.php?journal=jhc

[3] Symposium on Biases in Human Computation and Crowdsourcing. https://sites.google.com/sheffield.ac.uk/bhcc2019/home

[4] Semantic Web journal. http://www.semantic-web-journal.net/

[5] Journal of Web Semantics. https://www.journals.elsevier.com/journal-of-web-semantics

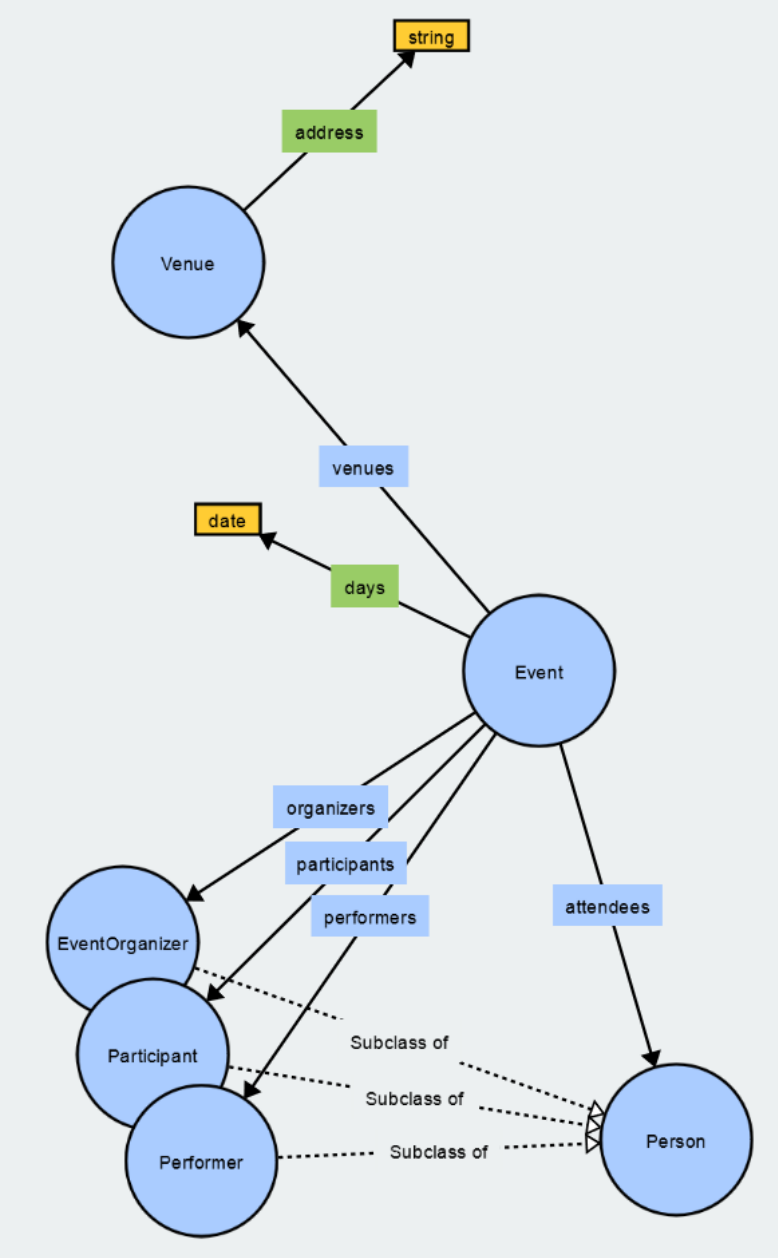


Figure 2.1: Knowledge Graph Example

Knowledge graphs are used for a wide variety of services, for example recommender systems, question answering systems, semantic search, expert systems, medical applications, education and personal assistants. There are multiple freely available large-scale knowledge graphs, for example DBpedia which contains knowledge from Wikipedia, YAGO, Freebase and Wikidata [HBC+20].

2.3 Availability of datasets containing human-made mistakes

While there is a lot of research done on knowledge graph refinement and error detection [d'A09] [JPJJ19], there is very little work on generating authentic erroneous datasets. In addition, there were no relevant human-generated knowledge graph based datasets including a gold-standard made available in any related papers. Finding such a dataset is required to both research and evaluate the human-likeness of original and generated mistakes. In our case the requirements for a dataset were the following:

- Between 1000 and 20000 triples in the graph
- At least 15 distinct nodes and 15 distinct relation types
- Enough similar nodes and similar relation types to allow for ambiguities that are close to their correct answer (this was only required since we wanted to evaluate how well our workflow works for close ambiguities of both nodes and relations in a single run)
- Individual human answers and the respective correct answer are available
- The content is in the general area covered by Wikipedia so that enough nodes can be resolved in the dictionary

Among the unusable datasets we found, most chose to not share their created gold standards or links to the data were no longer functional. We list the most relevant datasets and explain why we couldn't use them.

Quality Assessment for Linked Data: A Survey

Description This paper [ZRM+15] does an extensive quality assessment of linked data resulting in 18 quality dimensions and 69 metrics. Dataset selection according to various quality metrics would've been possible.

Constraints The availability of a gold standard was not considered as criteria for dataset quality. Correctness was considered on the syntactic and semantic level, but not in depth on the content level. Only two of the investigated papers [SLA+13][ZKS+13] used crowd sourcing as a means to check content correctness and are described in detail below.

Crowdsourcing DBpedia Quality Assessment

Description This paper [SLA+13] used DBpedia as knowledge graph and built a gold standard for parts of it[6]. It also compared crowdsourcing using random and expert users' answers.

[6] Website: http://people.aifb.kit.edu/mac/DBpediaQualityAssessment/experiments.html

Constraints While the gold standard was made available, it only contains the labels of the respective nodes but neither unique identifiers nor the complete graphs that were used. Re-linking this to the full DBpedia dataset is error-prone due to the possibility of duplicate names and recent changes to the graph. The gold standard and user answers are also spread across the whole graph which is too big for us to use.

User-driven quality evaluation of DBpedia

Description This paper [ZKS+13] created a gold standard using crowdsourcing with 58 users and assessed a total of 521 resources.

Constraints The created gold standard is not shared completely, only detected errors are made available[7].

SABINE: A Multi-purpose Dataset of Semantically-Annotated Social Content

Description This paper [CFG+18] linked sentiment statements to DBpedia nodes.

Constraints This paper builds a gold standard only for sentiment but not for the knowledge graph data itself.

BetterRelations: Detailed Evaluation of a Game to Rate Linked Data Triples

Description This paper [DHRB+11] creates a gold standard for the order of items by importance. This data could provide additional insights especially for the "subjective ambiguity"-source.

Constraints The data is not made available and the correctness of content of the used knowledge graph itself is not explicitly evaluated.

GuessWhat?! Human Intelligence for Mining Linked Data

Description This paper [MV10] used user answers from a game to build an ontology which could be used as a gold standard. It also increases the available information users have to decide on a node in multiple steps and could therefore provide insights in the "no knowledge"-source.

Constraints The data was originally made available to download, but the provided link was not accessible any more.

[7] Website: `http://aksw.org/Projects/DBpediaDQ.html`

Probabilistic Error Detection Model for Knowledge Graph Refinement

Description This paper [JPJJ19] aimed to increase classification accuracy in knowledge graphs using neural nets to simulate human judgements reinforced by a small crowdsourced gold standard. The datasets used were subsets of NELL and YAGO.

Constraints The created gold standard as well as the used subsets of the knowledge graphs were not made available.

Toward multiviewpoint ontology construction by collaboration of non-experts and crowdsourcing: The case of the effect of diet on health

Description This paper [ZGEBI16] conducted a large-scale crowdsourcing experiment about the effect of diet on health. It also compared the answers of subjective and objective questions.

Constraints The data was originally made available to download, but the provided link was not accessible any more. In addition to that, there was also no response to our emails.

Truth Inference in Crowdsourcing: Is the Problem Solved?

Description This paper [ZLL+17] compares 17 algorithms on 5 datasets with the intent to simplify algorithm selection for further work.

Constraints The data was originally made available to download, but the provided link was not accessible any more. In addition to that, there was also no response to our emails. We were later able to recover the broken link and download the data, but unfortunately only relational datasets were used and conversion was not possible.

A Task-based Approach for Ontology Evaluation

Description This paper [PM04] compares answers from a task based system to a gold standard to achieve incremental improvements of the knowledge graph.

Constraints The dataset and gold standard were not available for download.

Using the wisdom of the crowds to find critical errors in biomedical ontologies: a study of SNOMED CT

Description This paper [MMJ+14] compared answers from experts and crowdsourcing to find critical errors in a medical ontology. They made their generated gold standard available as PDF.

Constraints The dataset that was used in the paper is locked behind a paywall and its usability for our research remains unclear.

2.4 Embedding technologies as basis for semantic relation generation

Vector embedding allows us to represent an element (eg. a word in a sentence or a node in a knowledge graph) as a vector. In our approach these vectors are used to select and rank all candidates and choose the best suited ones for ambiguities. We will investigate relevant approaches and explain the advantage of this representation.

2.4.1 word2vec

The word2vec word embeddings [Ris18] lie at the core of our approach. They are needed to determine and quantify closeness between single words by substituting them with high dimensional vectors. These representations are captured in dictionaries where they are easily accessible.

The vectors in the dictionary are learned from a corpus of sentences. Any words not in the corpus also can't be present in the dictionary, so an extensive corpus is necessary to generate an all-purpose dictionary. We chose a Wikipedia-based dictionary since it provides a good balance of common and specialised vocabulary and is of adequate size. As there are multiple techniques to train the dictionary, we focused on dictionaries built using word2vec and GloVe, another way of creating the vector representation [PSM14].

The vectors encode the meaning of the words, their direction and length determine how words are related to each other and by representing each word with a vector we can also use arithmetic operations on them. Words that are similar in meaning also point in a similar direction, but their length will vary depending on their significance. To determine the closeness of two words the euclidean distance or cosine similarity is used.

For example, the similarity or distance between two vectors v_A and v_B can be calculated using equation 2.1.

$$\text{dist}(v_A, v_B) = \cos(\theta) = \frac{v_A \cdot v_B}{\|v_A\| \cdot \|v_B\|} \tag{2.1}$$

The result is a value between -1 and 1, where 1 means the direction of the vectors is identical and they point in the same direction, -1 means the direction is identical but they point in the opposite direction and 0 means they are orthogonal to each other.

Even though this seems to allow us to determine the inverse of a word this is actually not possible. For example we can't determine the inverse of "white" to be "black" because this would assume the context of the inverse still being a colour. Depending on the dictionary the result is actually a word not even closely related.

2.4.2 RDF2Vec

RDF2Vec aims to vectorise a given knowledge graph without the need of additional inputs by using random (and in subsequent versions also other) walks over the knowledge

graph [RRN+19]. It was originally inspired by the word2vec approach.

Since an additional dictionary is not required and the approach is generally content agnostic RDF2Vec might be at an advantage over wordvec approaches for some datasets, but it also has various downsides when compared to word2vec:

- Big graphs are often split into multiple files. While it is usually possible to combine them beforehand word2vec can operate on split files with only a small decrease in quality for some calculations, as the vectors have a global orientation. This is not the case in RDF2Vec where the general vectors orientation can change with every file.
- RDF2Vec has to be trained for every graph, depending on the size of the graph this can take a lot of time. Some pre-built graph representations eg. for DBpedia are available for download, but custom graphs always have to be trained. In word2vec pre-built dictionaries can be quickly swapped and dictionary/graph compatibility is defined by a shared vocabulary which is likely to be available even for most custom graphs. The downside of word2vec is a less accurate representation for multi-word nodes which can be mitigated.
- Word2vec also works on text-based literals like labels, descriptions or user-comments and custom vector computation for special literals is possible more easily that it is for RDF2Vec.
- Word2vec can have more accurate representations on very small or weakly linked graphs due to the pre-trained dictionaries independence of the graph size.

2.5 Crowdsourcing and human computation

In contrast to classical digital computation including AI based approaches who rely on predefined programs and models, human computation and crowdsourcing make use of individual humans or a crowd of people (possibly distributed across the world and connected over the internet) to solve tasks [LA18]. When using crowd algorithms such as majority voting to combine individual answers coming from laymen into an accredited final statement, the result becomes comparable to that of an expert [MMJ+14]. In our thesis this is one of the main causes for ambiguities in datasets the we want to simulate as well as our chosen way to evaluate the results of our approach. The concept of human computation using crowds is shown in Figure 2.2.

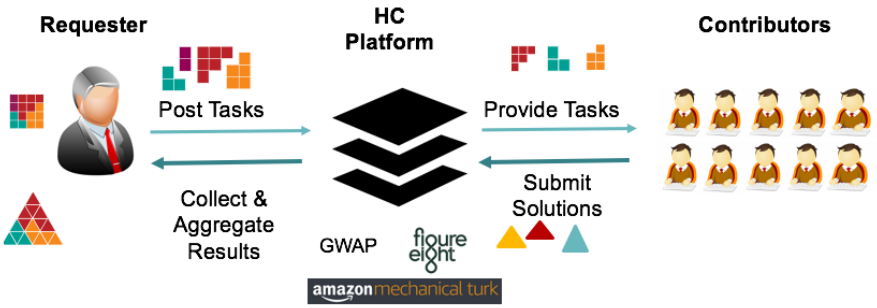


Figure 2.2: A blueprint of the HC process, adapted from Mao et al. (2017).

This offers vast possibilities, as areas that are still troubling for classical computation can easily be solved by humans (eg. image classification, language understanding). As even complicated work can be split into a number of simple tasks, this approach can be used to create or validate massive amounts of data in short times. A crowd connected over the internet can also act at a global scale unfazed by the time of day.

Amazon mechanical turk[8] is a crowdsourcing website that allows users (requesters) to create tasks that other users (workers or contributors) can then answer in exchange for payment. These tasks can be used to answer surveys, do content moderation and build, clean or validate datasets for machine learning and other automated computation. Workers that do tasks usually get a small monetary compensation set by the task creator.

For our thesis especially the surveys are of interest. It is possible to let workers classify generated data to custom classes. It is also possible to record data like the time it took workers to complete the tasks. This allows us to let workers finely rate generated data and correlate it with our input parameters.

In this chapter we provided insight into the literature research process (see section 2.1) and explained relevant background areas like knowledge graphs (see section 2.2), embedding technologies and crowdsourcing (see section 2.4) which are necessary for our research. We also expanded on the challenges to availability of existing annotated datasets and listed some papers who were almost able to fulfil our requirements (see section 2.3).

[8]Website: `https://www.mturk.com/`

CHAPTER 3

Ambiguities

In this chapter we will research ambiguities with the goal of arriving at a definition we can use for generation, as required to answer our first research question. We first define the term of ambiguity in contrast to similar concepts (see section 3.1) to establish the boundaries required for our next step, the investigation of sources of ambiguities (see section 3.2). By exemplifying the different sources, giving general explanations and concrete examples we are able to understand what characteristics individual ambiguities coming from all of those sources show (see section 3.3). As these characteristics can be mimicked and thus used as basis for generation we can conclude our first research question at the end of the chapter.

3.1 Ambiguity categories

Before we can define the sources of ambiguity, we have to clarify what characteristics cause the data in the knowledge graph to be considered as ambiguous. To achieve a clear distinction of what data is relevant for our approach we define three categories that nodes and relations can fall into: ambiguity, bias and garbage.

3.1.1 Ambiguity

Ambiguity is caused by user-decisions being made with either missing or low quality information [Zel18] in regards to lexical or syntactical definitions. For knowledge graphs this can happen at any point during the process of creation (eg. crowd-sourcing [AD17], majority voting [DAW18]), merging of graphs [Kle01] or validation (eg. crowd validation). The lack of information forces the users to assume the missing information to be able to come to a decision, and therefore causes ambiguous results. While this can in some cases affect multiple nodes or relations, it is in practice usually constrained to only one of those elements. These ambiguities can be minimised during the mentioned processes but

it is hard to identify and correct or remove them after the respective process is completed and only the final knowledge graph is available.

Example:
(highway_A23 − increasesWorthOf → vienna) could be (highway_A23 − decreasesWorthOf → vienna) instead if it is affected by an ambiguity. In this example the 'incrasesWorthOf' relation is the ambiguous part as it doesn't obviously state what variables the worth should be measured by. Concerning the people who have to live with the noise and pollution in the immediate area it might decrease the worth of their living area, but for the industry dependent on the highway it might increase the worth of the city as an industrial location.

3.1.2 Bias

Bias is a systemic skewing of the results and can affect all or a relatively large, specific part of the knowledge graph. It is created when the cause for the ambiguities affects all or a portion of triples regardless of the contained nodes and relations. This could happen when selecting users of specific groups (eg. only students) [NFG+20], when attempting to run corrective algorithms (eg. crowd validation) [Dem19] on all or parts of the data or it could already be embedded when using existing knowledge graphs [FPCM20]. Biases can be minimised during the process of creation, merging or validating too, but just like with ambiguities it is hard to identify and correct or remove them after the process is completed and only the final knowledge graph is available.

Presumed biases that are spanning only a relatively small and interrelated proportion of the knowledge graph therefore better fit the criteria for ambiguities rather than those for biases, even though they might have been created by sources of biases.

Example:
(region_tibet − partOf → country_tibet) could be (region_tibet − partOf → country_china) depending on the nationality of the workers answering the questions. If the workers nationalities are not monitored and distributed with this in mind a bias on this and also similarly motivated answers might be the result, the distinction to an ambiguity being the influence of this effect on other, related answers as well as this one.

3.1.3 Garbage

Garbage is caused by triples that don't follow any discernable underlying rules in how they are created and therefore information restoring algorithms can't achieve better results using this data than if they were not using this data in the first place. Possible sources for garbage include faulty data conversions, originally malformed data (eg. due to web crawling), restored data (eg. merging multiple non-linked knowledge graphs together) or if it is not possible to categorise data as either correct, ambiguous or a bias due to the severity of misinformation or lack of context.

Example:
(coffee − from → brazil) could be (coffee − from → 2020) instead if it is garbage. In this example the relation is so highly ambiguous that it doesn't restrict the possible answers to either a place or a time (of harvest, production, sale, or any other) and the result might be a mix of all possible interpretations. It is impossible to discern exactly how the answer was chosen without additional information due to the high potential for severity of this ambiguity.

3.2 Ambiguity sources

Using the ambiguity definitions from section 3.1 we create a list of the most prevalent sources of ambiguity, based on reviewed literature. This list is meant to cover the most common sources that are present in existing knowledge graphs created by crowdsourcing or web scraping approaches. For every source we state a definition, we aim to create a clear and understandable name for the source and give examples for situations that could lead to these ambiguities.

3.2.1 Subjective ambiguity

Definition Statements where users commonly have different opinions or that can by their nature only be viewed subjectively.

Explanation Subjective ambiguity is prevalent in areas where the answer to a question depends on the opinion of the worker because there are multiple relevant and potentially equally valid answers [DMA+18]. These ambiguities are created because the questions wrongly assume there is a single correct answer regardless of context.

Examples

- Borders in war areas/conflicted regions
  eg.: (region_tibet − partOf → country_tibet) and (region_tibet − partOf → country_china)
- Artists who influenced each other
- Opinions of members of different political parties
- Beliefs of different religious groups
- Personal taste on food
- Ranking a list by importance

3.2.2 Ambiguity of specialisation

Definition Statements are resulting from merging two graphs created for different contexts.

Explanation These ambiguities are introduced when answers of different specialisation levels are mixed. While a specialist and a normal user might also have different answers to the same question due to differences in available information [SHHQ16] it is also possible that the same person changes their mind between specialised and general answers while answering.

Examples

- Created by professionals vs random crowd
  eg.: (cpu − isCalculatingPartOf → computer) and (arithmethic_logic_unit − isCalculatingPartOf → computer)
- Specific ontology vs universal ontology
- Statements in same area coming from multiple professional backgrounds
- Privileged information not generally available
- Unclean entity linking during merging

3.2.3 Inherited ambiguity

Definition The correct answer to the statement depends on the users interpretation of either an ambiguous question or ambiguous answers.

Explanation Inherited ambiguity is not caused by the user but rather by an ambiguously formulated or overloaded question or ambiguous answers [WDK17] [Alt98]. While the user can minimise the severity of the ambiguity it's impossible to give a completely correct answer.

Examples

- Automatic generation of questions/answers
  eg.: question asking the number of legs all kinds of animals have, (snail − numberOfLegs → 0) and (snail − numberOfLegs → 1) could both be considered kind of correct if only numbers are allowed as answers, but both are most likely missing the point of the original question
- Automatic translation of questions/answers [PHg14]
- Missing information in either question or answer
- Question asking more than one thing at the same time
- Constrained set of answers that don't fit well enough

3.2.4 No knowledge

Definition Users answer questions that are out of their area of expertise.

Explanation When workers answer questions they don't know enough about, their answers can become ambiguous. This can be reduced if workers are allowed to skip questions but might still happen if the worker wrongly thinks he knows the answer. While this source of ambiguity can be corrected with majority voting and other crowdsourcing algorithms in small numbers, this doesn't work if the majority of workers provide unqualified answers. If the majority of the workers is wrong on a majority of questions, the result is garbage rather than ambiguity.

Examples

- Pays good and doesn't seem to hard, try to guess it
  eg.: (videoDecoding − computedOn → GPU) and (videoDecoding − computedOn → CPU) might depend on the context of this specific question (eg. whether the device in question has a dedicated GPU or only one integrated into the CPU) and could therefore be misunderstood by a user without sufficient knowledge while guessing either of them could still lead to the answer being accepted
- Answers with low confidence scores

3.2.5 Most common answers of other users

Definition Users don't provide their own answer but provide the presumed answer of a different group of people.

Explanation If users are asked for the presumed answers of another group of users this can improve quality in some cases but might reduce it in others because the user overly assumes the other groups standpoints. This would usually affect only single users (if the user differs from the other users) or single answers.

Examples

- Unknown change of answering person
  eg.: Two persons using a shared computer and account to answer questions on AMT
- What would the average person answer
- What would another ideological group answer

3.2.6 Temporal changes

Definition Different answers for questions depending on time of the questioning.

Explanation When asking an old set of questions and answers to another or the same group of users months or years later their answers for some questions might differ because of changed perspective, knowledge or vocabulary. In knowledge graphs that are continually expanded this mix leads to ambiguities [LC18]. This effect can also occur with changing time of day, season, day of the week and other cyclic timeframes.

Examples

- Changing information, perspective, knowledge, vocabulary of users
  eg.: An user who answered (conference − hasVenue → conference_hall) before the Corona virus might have changed their mind and answer with (conference − hasVenue → homeoffice) now as they are not willing to accept large crowds any longer
- Changing group of users

3.2.7 Failed attempts to correct data

Definition Effort to correct in assumption of some error but either under- or overachieve in doing so.

Explanation There exist a lot of algorithms aimed to correct faulty knowledge graphs, but these algorithms might just as well cause ambiguities if used incorrectly or in general because of their nature.

Examples

- Effort to correct bias but over-/underachieve goal
  eg.: A survey in France might result in the specific (white_wine − fitsToFood → cheese_brie), but the researcher tries to remove this regional bias in favour of the more general European answer of (white_wine − fitsToFood → cheese). During this process other answers to specific variants of cheese might be changed to cheese in general as well, even if the specific variant is required
- Overlooked irregular mistakes caused by algorithm
- Reduced multiple differing answers to one ambiguous answer [SSSF16]

3.2.8 Malicious behaviour

Definition Users might maliciously change or add answers that are not or only partially correct.

Explanation Users may want to overly include their viewpoint into datasets regardless of legality [OdV18] [CBD20]. This can take the form of intentionally wrong answers, automated answering, flooding with fake users or exploits in the code.

Examples

- Answering more extreme than own belief to shift the result
  eg.: Individual answers on a scale (eg. 1 for low agreement, 5 for high agreement) might use the mean or median to come to a single conclusive answer. By answering (artificial_intelligence − isSusceptibleToBias → 1) in sufficiently high numbers this result could be tampered with

- Creating fake users

The material from this section is summarised and synthesised in the next section.

3.3 Ambiguity source categories

While there are a lot different of sources of ambiguities, most individual ambiguities created by them show similar characteristics. Only when investigating multiple mistakes in combination the sources start to be discernable by their distributions and total numbers. This similarity enables us to define one or more methods to generate ambiguities for every category, each mimicking one of those characteristics and thus allowing us to answer our first research question. By combining multiple such methods we would then even be able to simulate multiple sources if we knew their internal distributions. The ambiguities generated by our identified sources fall into three categories in most cases.

Similar answers Often the users will pick the closest answer if the correct one is either not an option or unknown to the user. These answers are very similar in meaning, but usually no completely equivalent answers are given as options. This similarity can be calculated using vector embeddings (see section 2.4).

Example:
If the question is "What colour does the sky have?" and the possible answers do not include the obvious "blue" the given answers might be "dark blue", "grey", "white" and "black", but answers like "rain clouds", "air plane" or "quantum computation" would be to far off to be considered a viable answer. Seen on the scope of all possible answers the graph allows the few acceptable answers are all very similar, as they are all colours commonly seen in the (night-)sky just as "blue" would be.

Opposite answers It is also common that the user disagrees with the correct answer and instead chooses a contradicting answer. These answers usually still are in the same general area or context as the correct answers but opposite to it inside this areas.

Example:
The user might be asked to select the most intense colour and answer it with "black" or "white" which are usually considered opposite colours, but he won't answer it with "bright" or "pepperoni" because these, while possibly intense, are not considered to be close enough to the expected answer of a colour.

Different viewpoints It is also possible that users have a different concept of how close answers are to each other. While this still causes the same ambiguities as in the similar answers category (this would best be represented by a different dictionary, but can be adequately approximated trough similarity, see section 2.4) or the opposite answers category from an objective standpoint, they will still differ form other users.

Example:
Continuing with the question of the most intense colour the reasoning behind why one user chooses "black" or "white" might depend on their understanding of how these colours are composed. If pigments are mixed to derive the colour black (as a mix of all pigments) might be considered more intense, but if light is used white (as a mix of all coloured light beams) might be preferred.

In this chapter we first defined what kinds of incorrect data qualify as ambiguities in contrast to similar concepts (see section 3.1) and then compiled a list of eight sources that are able to cause such ambiguities (see section 3.2). Based on the individual ambiguities these sources can produce we then identified and described the mutual characteristics, namely similar and opposite answers, as well as different viewpoints which result in both of those characteristics (see section 3.3). Since this definition allows us to continue with research question two we can conclude that we proficiently answered research question one.

CHAPTER 4

Generating ambiguities with "AmbiVec"

In this chapter we will present our "AmbiVec" workflow used to generate ambiguities following the characteristics identified in the last chapter. We will first present the workflow both from a user-perspective as well as a code-perspective and then dive deeper into the most complex steps required for generation (see section 4.1). Data structures required for this process will be explained next, including examples from our test-graph (see section 4.2). Since each characteristic allows for at least one method to generate it we will explain the three methods based on vector embeddings and similarity calculation and the random method included in our thesis (see section 4.3). To allow researchers to spot and prevent low quality output as soon as possible during the workflow we will also provide a list of manual checks (see section 4.4) that can be performed and explain the metrics (see section 4.5) required to do so.

4.1 Workflow

We created a workflow allowing human interaction at various stages to generate high quality human-like ambiguities with our methods, using vector embeddings and similarity calculations to identify suitable ambiguities. The diagrams use colours to mark elements that have specific characteristics. Blue elements have to be provided by the user and green elements are outputs of the workflow.

4.1.1 User workflow

The diagram shown in Figure 4.1 shows what parts of the workflow have to be performed by the user and what parts are executed by the code. The user starts by providing the original knowledge graph and the dictionary. The knowledge graph is then vectorised

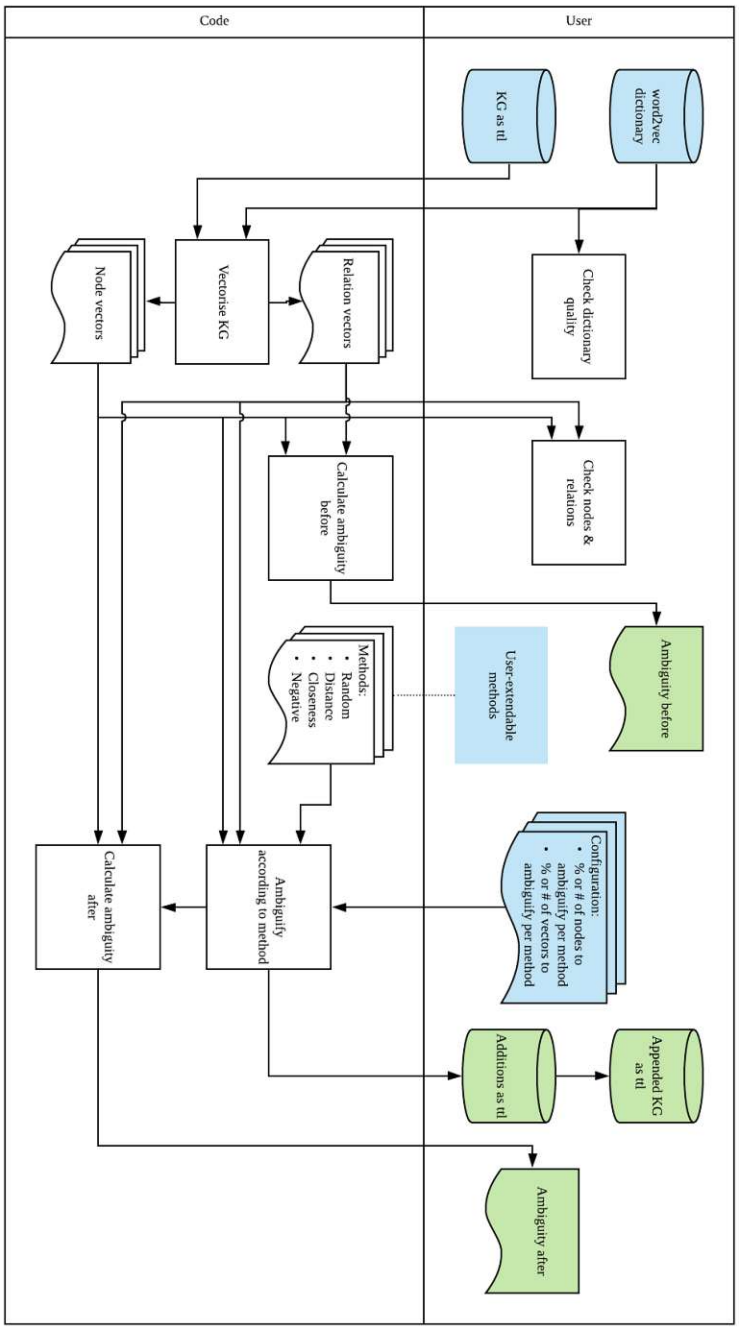


Figure 4.1: User Workflow Overview

to calculate the node vectors and relation vectors (see section 4.1.3). For every distinct node and relation these data structures contain the average vector that represents them as well as additional information needed by the user to do manual checks (see section 4.4). At this point the total ambiguity of the original knowledge graph can be calculated (called "ambiguity before") (see section 4.5.7). In the next step additional ambiguities are generated according to the user-provided configuration using the respective methods (see section 4.1.4). The ambiguity including these additions is called "ambiguity after" and the additions can be saved alone or saved together with the original knowledge graph.

If any of the manual checks done by the user don't yield the expected outcomes, the workflow can be aborted at this point and changes to the knowledge graph or intermediate data can be made to guarantee high quality outputs.

4.1.2 Code workflow

The code workflow shown in Figure 4.2 shows the workflow from a code-focused viewpoint. It's content is identical to the user workflow overview, but the split between user and code is removed for ease of reading.

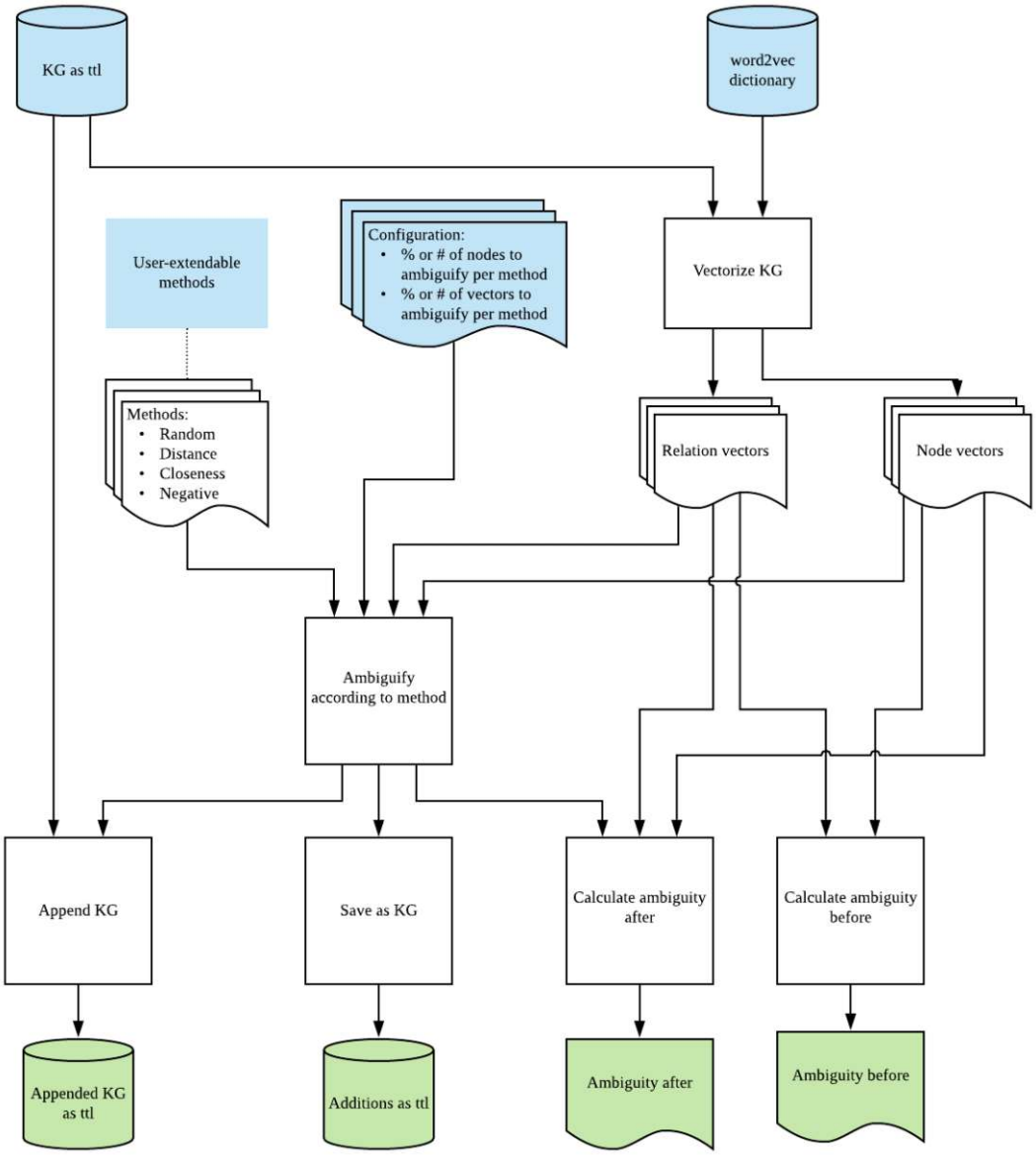


Figure 4.2: Code Workflow Overview

4.1.3 Step: Vectorise KG

Figure 4.3 shows the inner workings of the "Vectorise KG"-step. Vecotrising the knowledge graph starts with reading and converting the triples from the input file to a data structure (see section 4.2.1), this is needed to easily access and modify the content later. For every node the matching vector is looked up in the dictionary and the relation vector between the nodes is calculated as shown in equation 4.1. If the node's name or label consisted of multiple words this information is saved, as well as how many of the words were found. If both the object and subject are resolved to the same vector the resulting relation vector consists of only zeroes and is marked as such.

$$v_{rel} = v_{obj} - v_{sub} \text{ where } \forall\ v \in \text{fullVector} \tag{4.1}$$

In the next step the relation vectors (see section 4.2.2) are calculated. This is done by using the mean of all available relations of the same type as shown in equation 4.2. If a relation can't be calculated it is saved as a lost relation (see section 4.2.3) for analysis and excluded from further calculations. The usable relations are saved with their resulting vector, the total number of individual relations, the amounts that were lost or zero vectors and the number and percentage of relations used for their calculation. Since there can be a lot of variation depending on the relation, the minimum, maximum and mean of the distance the individual vectors have to the mean vector, as shown in equations 4.3, 4.4 and 4.5, is included.

$$\text{relationVector}_{rel,\ preType} = \underset{\forall v \in \text{fullVector:}\ v_{pre=preType}}{\text{mean}} (v_{rel}) \tag{4.2}$$

$$\text{relationVector}_{meanDist,\ preType} = \underset{\forall v \in \text{fullVector:}\ v_{pre=preType}}{\text{mean}} (dist(\text{relationVector}_{rel,\ preType}, v_{rel})) \tag{4.3}$$

$$\text{relationVector}_{minDist,\ preType} = \underset{\forall v \in \text{fullVector:}\ v_{pre=preType}}{\text{min}} (dist(\text{relationVector}_{rel,\ preType}, v_{rel})) \tag{4.4}$$

$$\text{relationVector}_{maxDist,\ preType} = \underset{\forall v \in \text{fullVector:}\ v_{pre=preType}}{\text{max}} (dist(\text{relationVector}_{rel,\ preType}, v_{rel})) \tag{4.5}$$

Using the relation vectors, an estimated position in the form of a vector is calculated for every node. This is done by summing the node vectors and respective relation vectors of all connected nodes and averaging the result as shown in equations 4.7, 4.8 and combined

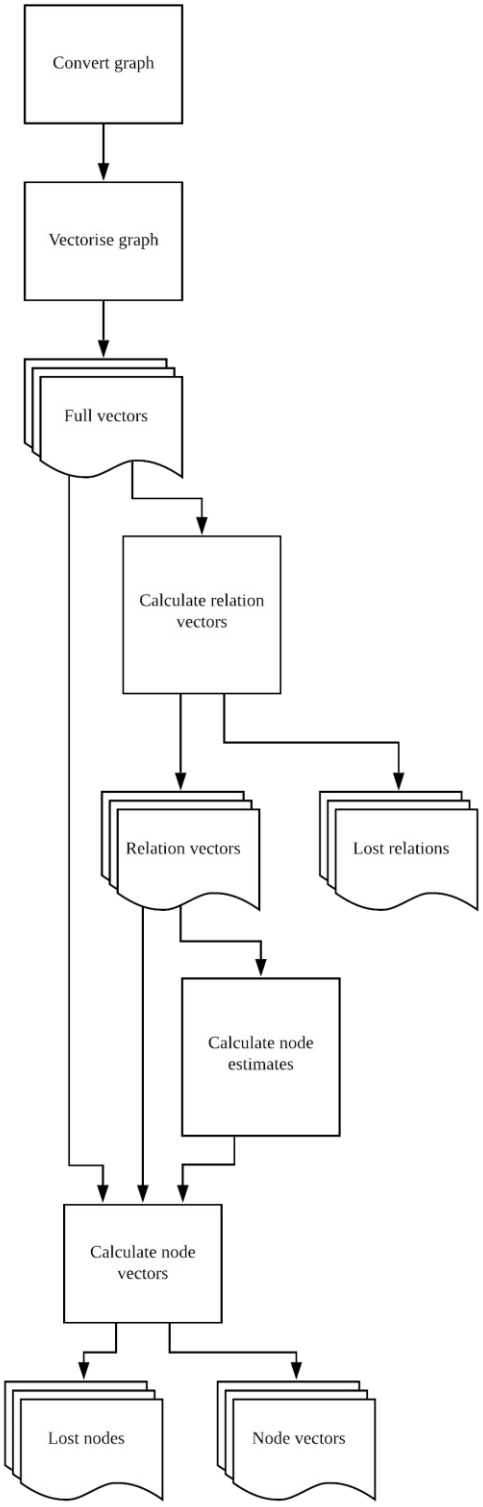


Figure 4.3: Step: Vectorise KG

in equation 4.6. The distance of this estimate to the original node is then calculated as well.

$$\text{nodeVector}_{est,\ nodeType} = \text{mean}(\text{nodeVector}_{objEst,\ nodeType}, \text{nodeVector}_{subjEst,\ nodeType}) \tag{4.6}$$

$$\text{nodeVector}_{objEst,\ nodeType} = \underset{\forall v \in \text{fullVector: } v_{obj=nodeType}}{\text{mean}} (v_{subj} + v_{rel}) \tag{4.7}$$

$$\text{nodeVector}_{subjEst,\ nodeType} = \underset{\forall v \in \text{fullVector: } v_{subj=nodeType}}{\text{mean}} (v_{obj} - v_{rel}) \tag{4.8}$$

Since the node vectors and some additional data are already available in a data structure they are combined into a new data structure (see section 4.2.4), containing one entry per node. The average estimated position as well as the minimum, maximum and mean distance to the average vector are saved as well. Unresolvable nodes are saved separately for analysis (see section 4.2.5) and excluded from further calculations.

4.1.4 Step: Ambiguify according to method

Figure 4.4 shows the "Ambiguify according to method"-step. Ambiguification is done by taking a number of random nodes or relations according to the configuration (see section 5.3). For each of those an ambiguified version is calculated using the configured methods (see section 4.3). To form triples out of these ambiguified nodes and relations they are randomly inserted into triples containing their original versions, but the original versions also remain in their unchanged form. The new and ambiguified triples are then made available as their own data structure for other parts of the workflow.

4.2 Data structures

The main data structures organise the data in a way that is easy to access and analyse. There are six main data structures users can interact with during the manual checks (see section 4.4). We will give an overview and explain the values they contain, as well as selected example output from a small knowledge graph containing both regular data and special cases. The example data will also contain the original index that we will use to reference individual entries in the text. The numeric values of some fields are restricted to an interval, these restrictions are given in mathematical notation. Vectors are always identical in dimensions to the vectors retrieved from the dictionary.

4.2.1 "fullVectors" data structure

The "fullVectors" data structure shown in Table 4.1 contains the complete triples with some additional information, but it is not needed for manual analysis. It is used to

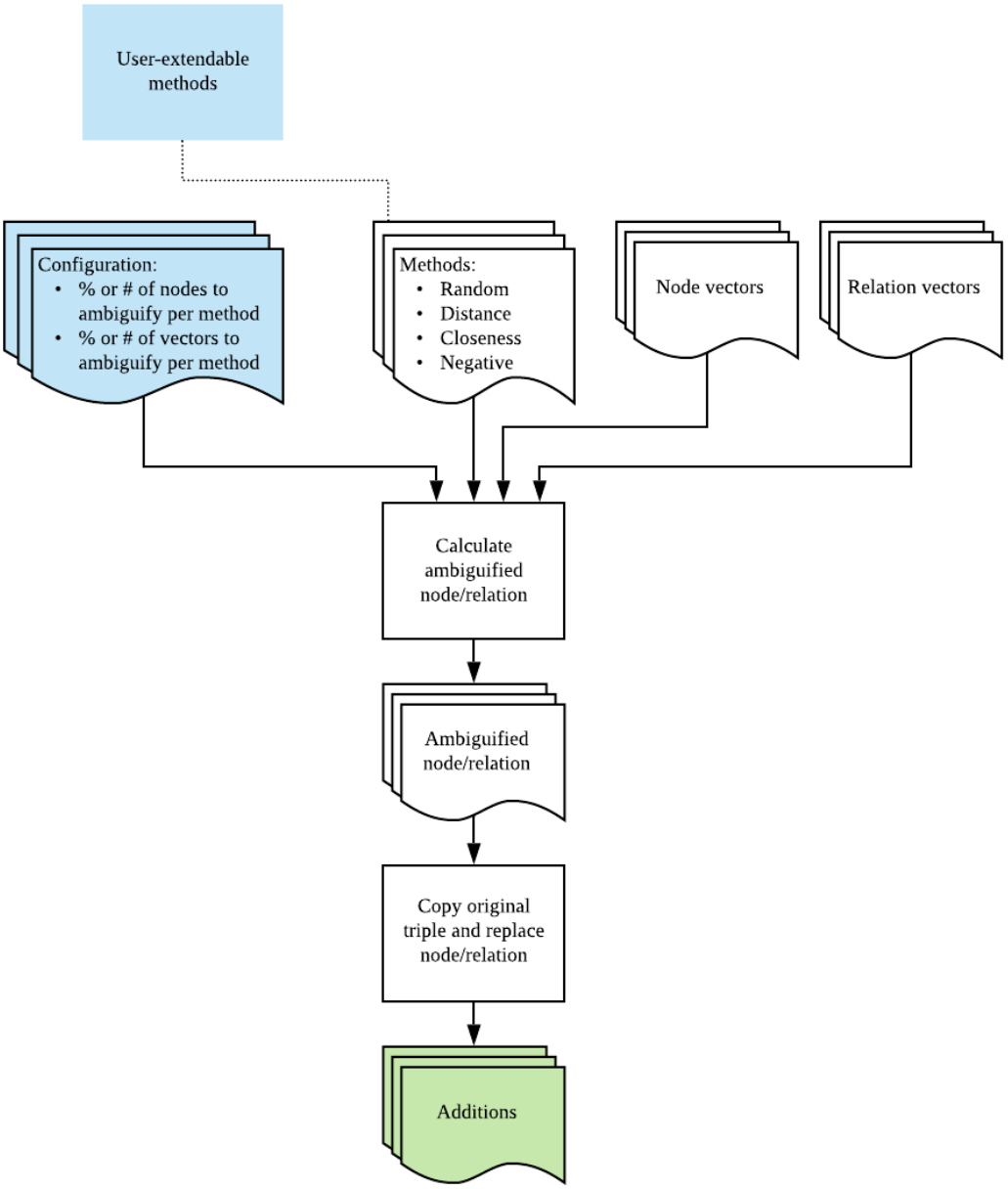


Figure 4.4: Step: Ambiguify according to method

read in the file containing the full knowledge graph, calculate the "nodeVectors" (see section 4.2.4) and "relationVectors" (see section 4.2.2) data structures and to calculate the ambiguity.

The data structure contains the identifiers of the elements as well as their vectors. For the nodes information on whether they consisted of multiple parts, what percentage of these parts was able to be resolved in the dictionary, the estimated vectors and the distance between the vector and the estimate are also saved. In the case of relations it is recorded whether the relation had a zero-vector or not.

fullVectors data structure		
Name	Type	Description
s	String	The subject of the triple
p	String	The predicate of the triple
o	String	The object of the triple
s_vec	Vector	The vector of the subject
s_is_multipart	Bool	Whether the subject consists of multiple parts
s_multipart_%	Float (0..1)	Percentage of the subject found in the dictionary
o_vec	Vector	The vector of the object
o_is_multipart	Bool	Whether the object consists of multiple parts
o_multipart_%	String	Percentage of the object found in the dictionary
r_vec	Vector	The calculated vector of the relation
is_zero_vector_relation	Bool	Whether the r_vec is a zero-vector
s_est	Vector	The calculated estimated vector of the subject
s_est_dist	Float (-1..1)	The distance between s_vec and s_est
o_est	Vector	The calculated estimated vector of the object
o_est_dist	Float (-1..1)	The distance between o_vec and o_est

Table 4.1: fullVectors data structure

The example shown in Table 4.2 shows a zero-vector on index 0, there is also an unresolvable object on index 15 and a complete entry on index 17.

fullVectors example			
index	s	p	o
0	http://example.org/countries/France	http://...#prefLabel	France
15	http://example.org/countries/Austria	http://example.org/r/undefinable	http://example.org/languages/Random
17	http://example.org/countries/France	http://example.org/r/speaks	http://example.org/languages/French

fullVectors example						
index	s_vec	s_is_multipart	s_multipart_%	o_vec	o_is_multipart	o_multipart_%
0	[0.4359, ...	False	False	[0.4359, ...	False	False
15	[0.2758, ...	False	False	None	False	False
17	[0.4359, ...	False	False	[0.027048, ...	False	False

fullVectors example							
index	r_vec	is_zero_vector_relation	s_est	s_est_dist	o_est	o_est_dist
0	[0.0, 0.0, 0.0, ...	True	None	None	None	None
15	None	False	None	None	None	None
17	[-0.408852, ...	False	[0.20744851, ...	0.931187	[0.2554948, ...	0.936883

Table 4.2: fullVectors example

4.2.2 "relationVectors" data structure

The "relationVectors" data structure shown in Table 4.3 contains the relations, their vectors and some additional information. This data structure is used to calculate the node estimates, to calculate the ambiguity in the whole graph and for manual analysis.

The data structure contains the identifier and vector of the relation. It also contains the number of all relations of this type, how many were lost or zero-vectors, the percentage of usable triples and the minimum, maximum and mean distance of the individual relation vectors to the average vector.

relationVectors data structure		
Name	Type	Description
relation	String	The name of the relation
vec	Vector	The calculated vector of the relation
total	Int	The total number of occurrences of this relation in the knowledge graph
lost	Int	The number of lost triples using this relation
zero_vector	Int	The number of zero-vector triples using this relation
quality	Float (0..1)	Percentage of usable triples using this relation
min_dist	Float (-1..1)	The minimum distance-value between the individual vector and the mean vector of any triple using this relation
max_dist	Float (-1..1)	The maximum distance-value between the individual vector and the mean vector of any triple using this relation
mean_dist	Float (-1..1)	The mean distance-value between the individual vector and the mean vector of any triple using this relation

Table 4.3: relationVectors data structure

The example shown in Table 4.4 shows a zero-vector on index 1, a lost vector on index 2 and a full entry on index 0. Both relations were still resolvable, but have a reduced quality.

relationVectors example									
index	relation	vec	total	lost	zero_vector	quality	min_dist	max_dist	mean_dist
0	http://.../isCapital	[-0.07769306,...	6	0	0	1.000000	0.438275	0.851050	0.735537
1	http://.../isOn	[-0.18347017, ..	6	0	1	0.833333	0.527029	0.849935	0.688621
2	http://.../speaks	[-0.1804005, ...	8	1	0	0.875000	0.498719	0.876127	0.760343

Table 4.4: relationVectors example

4.2.3 "lostRelations" data structure

The "lostRelations" data structure shown in Table 4.5 contains the relations for which no vector could be calculated and some additional information. This data structure is used for manual analysis. It contains all relations that couldn't be resolved.

This data structure contains the identifier of the relation as well as the amounts of total, lost and zero-vector triples of this relation.

lostRelations data structure		
Name	Type	Description
relation	String	The name of the relation
total	Int	The total number of occurrences of this relation in the knowledge graph
lost	Int	The number of lost triples using this relation
zero_vector	Int	The number of zero-vector triples using this relation

Table 4.5: lostRelations data structure

The example shown in Table 4.6 shows two intentionally lost relations in index 0 and 1, one due to lost vectors and one due to zero-vectors. Index 2 shows the label which was lost due to both, but mainly due to the identical name and label of the nodes.

lostRelations example				
index	relation	total	lost	zero_vector
0	http://example.org/r/undefinable	1	1	0
1	http://example.org/r/zerovec	1	0	1
2	http://www.w3.org/2004/02/skos/core#prefLabel	22	1	21

Table 4.6: lostRelations example

4.2.4 "nodeVectors" data structure

The "nodeVectors" data structure shown in Table 4.7 contains the nodes, their vectors and estimated vectors and some additional information. This data structure is used for calculating the ambiguity and for manual analysis.

This data structure contains the identifier of the nodes, their vector, whether they consist of multiple parts and the percentage of parts that could be resolved in the dictionary, their estimated vector and the distance between the estimated vector and the dictionary-vector as well as the mean, minimum and maximum distances. It also contains the total amount of nodes of this type.

nodeVectors data structure		
Name	Type	Description
node	String	The name of the node
vec	Vector	The dictionary vector of the node
is_multipart	Bool	Whether the nodes name consists of multiple parts
multipart_%	Float (0..1)	Percentage of the nodes name that was found in the dictionary
est	Vector	The calculated estimated vector of the node
est_dist	Float (-1..1)	The distance between vec and est
total	Int	The total number off occurrences of this node in the knowledge graph
mean_est_dist	Float (-1..1)	The mean distance-value between the nodes vector and the individual estimated vectors of any triple using this node
min_est_dist	Float (-1..1)	The minimum distance-value between the individual estimated node vector and the mean estimated vector of any triple using this node
max_est_dist	Float (-1..1)	The maximum distance-value between the individual estimated node vector and the mean estimated vector of any triple using this node

Table 4.7: nodeVectors data structure

The example in Table 4.8 shows a node consisting of multiple parts on index 6. Please note that while the identifier doesn't use a whitespace in it's name the label does, and is used instead of the identifier if present. On index 0 a full example is shown.

nodeVectors example					
index	node	vec	is_multipart	multipart_%	est
0	http://example.org/countries/France	[0.4359, ...	False	False	[0.6985833, ...
12	http://example.org/continents/Europe	[0.1716, ...	False	False	[0.5543091, ...

nodeVectors example					
index	est_dist	total	mean_est_dist	min_est_dist	max_est_dist
0	0.587461	2	0.917151	0.917151	0.917151
12	0.772506	4	0.938034	0.927703	0.949444

Table 4.8: nodeVectors example

4.2.5 "lostNodes" data structure

The "lostNodes" data structure shown in Table 4.9 contains the nodes for which no vector could be found in the dictionary, their estimated vector (if it was possible to calculate it) and some additional information. This data structure is used for manual analysis. It contains all nodes that couldn't be resolved.

This data structure contains the nodes identifier, whether it consists of multiple parts and the percentage of parts that could be resolved in the dictionary as well as the estimated vector and the total number of nodes of this type.

lostNodes data structure		
Name	Type	Description
node	String	The name of the node
is_multipart	Bool	Whether the nodes name consists of multiple parts
multipart_%	Float (0..1)	Percentage of the nodes name that was found in the dictionary
est	Vector	The calculated estimated vector of the node
total	Int	The total number off occurrences of this node in the knowledge graph

Table 4.9: lostNodes data structure

The example in Table 4.10 shows two lost nodes, one in index 0 for which an estimated position could be calculated and one in index 1 where this wasn't possible.

lostNodes example					
index	node	is_multipart	multipart_%	est	total
0	http://.../Random	False	False	[-0.20528017, ...	2
1	RandomTndsfkjnebkjabd	False	False	None	1

Table 4.10: lostNodes example

4.2.6 "changes" data structure

The "changes" data structure shown in Table 4.11 contains the triples that were selected to generate the ambiguities, the method and configuration used (see section 4.3), the type (whether it is a node or relation) and both the current and new value of the element that will be substituted. This data structure is used for manual analysis.

This data structure contains the name of the method used for generation as well as the relevant configuration. The configuration is saved in a machine-readable object, but can also be read by humans. The type of the source (whether it was a node or a relation),

the element that is replaced (both before and after) and the full triple are also contained in this data structure.

changes data structure		
Name	Type	Description
method	String	The method that was used
config	Object	The configuration that was used
source_type	String	The type of the element (whether it is a node or relation)
source	String	The original element that was replaced
target	String	The element used as replacement
s_orig	String	The original subject of the triple
p_orig	String	The original predicate of the triple
o_orig	String	The original object of the triple

Table 4.11: changes data structure

The example in Table 4.12 shows a set of generated ambiguities. The configuration featured two methods, random (index 0 for nodes) and distance (index 40 for relations).

changes example					
index	method	config	source_type	source	target
0	random	{'amount': {'num': 5}}	node	http://.../Europe	http://.../HalfRandom
40	dist	{'amount': {'num': 5}, 'param': {'dist': 1}}	relation	http://.../speaks	http://.../isOn

changes example			
index	s_orig	p_orig	o_orig
0	http://.../France	http://.../isOn	http://.../Europe
40	http://e.../France	http://.../speaks	http://.../French

Table 4.12: changes example

4.3 Methods and their configuration

The functions that we use to calculate ambiguities are called methods. There are four default methods described in this thesis: random, distance, closeness and negative. They are further explained in sections 4.3.1 to 4.3.4.

The methods are configured by the "config" object (see section 5.3) which is internally split into nodes and relations. Methods can be individually configured for the addition of ambiguities to both nodes and relations. The method-configuration includes the amount which can set as number or percentage and depending on the method also additional method-specific parameters.

4.3.1 Distance method

The distance method selects the replacement based on the number of elements that are closer to the original element. A distance of zero always returns the original element, a distance of one the element closest to it, a distance of two the 2nd-closest element and so on. The maximum distance is the length of the dataset minus one. This approach is able to generate ambiguities because it can simulate one of the ambiguity source categories (see section 3.3).

This method has to calculate the distance (see section 2.4) between the original element and all other elements for every replacement and generating a large number of replacements or using a large dataset can slow down the selection significantly.

For this method the distance can be passed as parameter.

```
config = {
    'nodes': {
        'dist': [{
                    'amount': {'num': 5},
                    'param': {'dist': 1}
                }]
    }
}
```

4.3.2 Closeness method

The closeness method selects the replacement based on the specific distance (see section 2.4) it has to the original element. It selects the element closest to the set distance, no matter how far off this element is. The minimum value that can be configured is 0 (which is equal to the input element) and the maximum is 2 (which is the element most different from the input element). This approach is able to generate ambiguities because it can simulate one of the ambiguity source categories (see section 3.3).

This method has to calculate the distance between the original element and all other elements for every replacement and generating a large number of replacements or using a large dataset can slow down the selection significantly.

For this method the closeness can be passed as parameter.

```
config = {
    'nodes': {
        'closeness': [{
                        'amount': {'num': 5},
                        'param': {'closeness': 0.2}
                      }]
    }
}
```

4.3.3 Random method

The random method selects a replacement randomly from the source with no consideration to any distances. The possible replacements include the original element and can result in the same element being chosen multiple times as a replacement.

There are no additional configuration-parameters available for this method.

4.3.4 Negative method

The negative method inverts the vector and then selects the nearest element. This usually produces seemingly random results and not what would be considered a logical opposite of an element.

There are no additional configuration-parameters available for this method.

4.4 Manual checks

Manual checks can be performed at some points in the "AmbiVec" workflow to ensure the processed data meets the quality requirements. Since quality requirements can vary widely depending on their intended use, we will state what cases can cause quality to be reduced but not define a universal breaking point. For small knowledge graphs the checks can be done after generating the ambiguities, but for big graphs it is suggested to perform the checks as early as possible to not waste processing time on low quality data. Please note that it is not possible to set concrete values for all metrics to check against as they are dependent on the used knowledge graph and dictionary.

In case the expected quality is not achieved users can choose to drop individual nodes or relations from their data structures (see section 4.2)before generating the ambiguities to prevent them from being used.

4.4.1 Quality of the dictionary

For our proposed implementation we chose to use a pre-trained dictionary, but in case of a custom dictionary the quality has to be ensured. Even though our approach is able to mitigate this, a low quality dictionary will decrease the quality of the generated ambiguities, especially of nodes. There are multiple ways to validate a dictionaries' quality, for example confusion matrices, analogy sets or other methods to measure perplexity.

4.4.2 Relation metrics

The relation metrics of the "relationVectors" data structure (see section 4.2.2) provide insight into the quality of their respective relation. How many relations were lost due to insufficient data? Were they relevant for the structure of the graph? Are any of the distances for any resolved relation unusually high?

4.4.3 Node metrics

When assessing the "nodeVectors" data structure (see section 4.2.4) we have to check if enough nodes could be resolved from the dictionary and why the other nodes were lost. Are they words not contained in the dictionary or do they contain special characters? For the resolved nodes the percentage of resolved words in multi-part names can show low quality vectors. How far are the nodes distanced from their estimated positions?

4.4.4 Output triples

This is the final check, do the ambiguified triples fulfil the requirements for their intended application? Are there any triples that used irrelevant relations or nodes or elements that were of significantly lower quality than the other ones?

4.5 Metrics

Metrics are calculated values that can be used to understand the quality of certain elements. Since they depend on the knowledge graph and the dictionary the concrete values that cause low quality output can't be predefined.

4.5.1 Knowledge graph size

Knowledge graph size (the number of triples in the graph) and composition (the number of unique nodes and relation types) is the first important metric. Especially very small graphs might not contain enough distinguished nodes or relation types to generate ambiguities of adequate quality from them. If that is the case the distances of elements to their mean or estimated position will be relatively high on average.

4.5.2 Percentage of resolved words

The next metric is the percentage of resolved words in multipart node names. Since these nodes already need calculations instead of a dictionary lookup to arrive at their vector, they are more sensitive to unresolvable words but will also remain usable if at least a single resolvable word remains. One example of this could be a node with the name 'chocolate cake'. For this example we'll assume that 'chocolate' could be resolved but 'cake' couldn't (a reason for this could be a small or incomplete dictionary, or one in a different language). This node would then use the vector of 'chocolate', which is still acceptable in most cases, but also not really correct since the influence of 'cake' would be missing. Low quality entries like this will still be partially corrected by the calculations averaging across all occurrences, but if most linked node names of a relation consist of multiple parts the relation vector can still degrade in quality.

4.5.3 Zero-vector relations

Zero-vector relations are a special case of lost relations and the result of a relation between nodes with an identical vector. This usually happens in multipart node names where the identifying word can't be resolved and they will not be used when calculating the relation vector for all relations of this type. If a relation mostly consists of these vectors be aware that the relation vector might degrade in quality because of the low number of remaining relations.

4.5.4 Distances

The given mean, minimum and maximum distances can help to estimate the quality of an element. High distances can be a sign of low quality elements either caused by the graph structure itself or by the calculations using low quality elements. For nodes this is calculated as the distance of each individual estimated position to the average estimated position and for relations it is calculated as the distance of each individual relation to the average relation. Comparing to the average positions of all elements reduces the influence of low quality elements or outliers.

4.5.5 Lost nodes and relations

The number of resolved nodes and relations in comparison to the number of lost nodes and relations can give insights on how well the vectorisation worked. While usually most relations should be resolvable this might not be the case for most nodes as they are directly looked up in the dictionary.

4.5.6 Relation quality

The quality metric is calculated as the percentage of individual relations that were used to calculate the mean and can be used to quickly identify potential low quality relations.

4.5.7 Ambiguity before & after

An ambiguity score is calculated for the whole knowledge graph before and after the ambiguous triples were calculated and added to the graph. While the value of the original graph can be used to estimate its quality the increase in this score after generation can be used to estimate the additional impact of ambiguities on the graph.

In this chapter we presented our "AmbiVec"-workflow for generating ambiguities. We started by explaining the workflow including complex steps (see section 4.1), continued with the data structures (see section 4.2), the methods and their configuration possibilities (see section 4.3) and ended with the manual checks (see section 4.4) and the metrics they use (see section 4.5). To be able to conclude our research question two and three we need a reference implementation which we will be describing in the next chapter.

CHAPTER 5

Reference implementation of "AmbiVec"

To test and evaluate our research we developed a reference-implementation of the "AmbiVec"-workflow in python using jupyter as both are widely used in related research. We first explain what design decisions we had to take as well as the reasoning behind them (see section 5.1) and continue with the libraries we chose to use (see section 5.2). As our thesis is meant to be a basis for further research including modifications to our workflow we also explain how the methods can be configured and extended (see section 5.3), what the main functions of our implementation are as well as their purposes and where in the workflow-graphs they are positioned (see section 5.4). To ensure proficient computation speed we also measure the performance of our implementation (see section 5.5) and provide future researchers with access to the full repository (see section 5.6). The chapter concludes with the answer to our second research question.

5.1 Design decisions

The code features some areas which handle specific behaviour depending on the context. They are listed here and the reasoning behind decisions is explained.

The main data structures are saved as pandas dataframes as these allow for easy access and manipulation of the content as well as the use of integrated graphical functions for the evaluation.

The vector length in the dictionary is not restricted by the code, it can handle vector lengths of any size compatible with the gensim library. Please note that especially large vectors may cause increased computation time and memory usage.

If the graph contains labeled nodes the English label will be used where present. Special characters ('_' and ',') will be ignored and handled like spaces. If multiple words are

present in the label the mean of the resolvable vectors is used. The 'getPreferredTitle(..)' and 'toVector(..)' function are responsible for handling these cases and can be adapted as needed.

Instead of using the mean of resolvable vectors for every node other strategies like using the estimated position or dropping them completely are possible, but since these options either require more computation or have the potential to greatly reduce the quality we decided against them.

5.2 Used libraries

While we tried to not use libraries unnecessarily we also had to rely on the functionality of some of them. here we give a short overview of the most important used libraries and what we use them for.

Pandas is a library made for data manipulation and analysis, which is also what we use it for. All our dataframes are created and transformed using pandas.

Numpy is a library providing us with basic mathematical operations and various tools for arrays and is also required by pandas.

Scipy, while quite extensive, is only used to calculate the p-values of the correlation.

Gensim is a library made for natural language processing. We use it to fetch the dictionary and for all calculations regarding the vectors. This library enables the vectorisation and most methods in a performant way.

Rdflib is a library made for working with RDF. It allows us to read and write ttl files and to convert the triples contained in them to pandas dataframes.

Tqdm, **Logging**, and **Tempfile** are used only in a few instances or for minor functions and self-explanatory. Tqdm allows us to show progress bars for calculations, logging is used for easier debugging and tempfile enables us to save pandas dataframes instead of re-calculating them on every startup.

5.3 Methods and configuration

The configuration provides an uniform interface to the methods. While some methods allow for additional, method-specific configuration the amount is saved in the same way for all methods. A combined example configuration for all methods in our implementation is given here.

```
config = {
    'nodes': {
        'random': [{'amount': {'perc': 0.2}}],
        'dist': [{
```

```
                    'amount': {'num': 5},
                    'param': {'dist': 1}
                }],
            'closeness': [{
                        'amount': {'num': 5},
                        'param': {'closeness': 0.2}
                    }]
        },
        'relations': {
            'random': [{'amount': {'num': 5}}],
            'dist': [{
                    'amount': {'num': 5},
                    'param': {'dist': 1}
                }],
            'negative': [{'amount': {'perc': 0.2}}]
        }
}
```

The shown default methods (see section 4.3) can also be extended with custom functions by adding function callbacks to the "methods"-object. The original triple, the source from which it will take the replacement and the configuration parameters will be automatically forwarded to the function. It is also possible to use different functions for nodes and relations, but the default methods work for both types.

```
methods = {
    'nodes':{
        'random': rand,
        'dist': dist,
        'closeness': closeness,
        'negative': negative
    },
    'relations': {
        'random': rand,
        'dist': dist,
        'closeness': closeness,
        'negative': negative
    }
}
```

5.4 Main functions

In case users want to extend the code and methods or want to repurpose parts of it we give a brief overview of the most important functions, their parameters and outputs.

5.4.1 convertedGraph = convertGraph(g)

This function is shown in Figure 4.3 as "Convert graph" and converts the graph 'g' created by reading the ttl file using rdflib to a pandas dataframe 'convertedGraph' containing the identifiers of subject, predicate and object.

5.4.2 fullVectors = vectorifyGraph(convertedGraph)

This function is shown in Figure 4.3 as "Vectorise graph" and reads in the 'convertedGraph' dataframe, calculates 's_vec', 's_is_multipart', 's_multipart_%' and the equivalent columns for the object and resolves any multiword-nodes and labeled nodes in the process. It then calculates 'r_vec' and 'is_zero_vector_relation' using the vectors and returns the 'fullVectors' dataframe containing all these columns.

5.4.3 relationVectors, lostRelations = generateRelationVectors(fullVectors)

This function is shown in Figure 4.3 as "Calculate relation vectors" and calculates the complete 'relationVectors' and 'lostRelations' dataframes using the 'fullVectors' dataframe.

5.4.4 fullVectors = calculateNodeEstimates(fullVectors, relationVectors)

This function is shown in Figure 4.3 as "Calculate node estimates" and adds the 's_est', 's_est_dist' and the equivalent columns for the object to the given 'fullVectors' dataframe using the 'relationVectors' dataframe and returns the changed 'fullVectors'.

5.4.5 nodeVectors, lostNodes = generateNodeVectors(fullVectors)

This function is shown in Figure 4.3 as "Calculate node vectors" and calculates the complete 'nodeVectors' and 'lostNodes' using the 'fullVectors' dataframe. Please note that the fields added by the 'calculateNodeEstimates(..)' function are required for this.

5.4.6 ambiguity = calculateAmbiguity(nodeVectors, relationVectors)

This function is shown in Figure 4.1 as "Calculate ambiguity after" and "Calculate ambiguity before" and calculates the ambiguity as percent of the maximum ambiguity in the graph using the 'nodeVectors' and 'relationVectors' dataframes.

5.4.7 res = ambiguify(config, nodeVectors, relationVectors)

This functionis shown in Figure 4.1 as "Ambiguify according to method" and creates a temporary dataframe called 'res' according to the 'config' object using the 'nodeVectors' and 'relationVectors' dataframes. It contains the columns 'method' (the name of the method used to generate this triple), 'config' (the configuration used for the method

generating this triple), 'source_type' (whether a node or the relation from the original triple was changed), 'source' (the original node or relation) and 'target' (the ambiguified node or relation).

5.5 Performance

While a reasonable performance is required by our research question for our workflow to be usable in practice it was not a focus of our research. While our workflow performs well enough for our purposes we also proposed multiple improvements that could substantially speed up computation (see section 7.3.1). In this section we will measure how performant the workflow is currently.

To measure the performance we first have to define what part of the workflow is measured and what performance-relevant variables are varied. We decided to split the workflow into two parts for this purpose. The first part encompasses the preparations of the data structures (see section 4.2) for the generation while the second part will continue with the generation. For each investigated combination of inputs we will run the measurement three times to minimise the influence of factors we could not control (eg. background processes) and the resulting runtimes will be recorded.

The measurements will be taken on a personal computer equipped with an Intel i9-9900K processor running at 3.6GHz and 32GB of RAM. The PC is running Windows 10 in the 64bit version and the workflow will only use a single core and none of the available graphic cards. During measurement all other software was stopped if possible.

We use two dictionaries for our tests, the "glove-wiki-gigaword-100" dictionary and the "glove-wiki-gigaword-300" dictionary[1]. We also created 6 graphs in different sizes using the LIMIT-functionality of SPARQL (see section 6.3 for more information on this query) as well as example configurations to generate a specific number of nodes and relations. The parameters used are described for each measurement.

The query we used for the selection of the graphs:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>

construct {
  ?s ?p ?o .
}
WHERE {
```

[1] Website: https://github.com/RaRe-Technologies/gensim-data

```
  ?s (dbo:starring |
    dbo:writer |
    dbo:musicComposer |
    dbo:director |
    dbo:producer) ?o .
  ?s ?p ?o
}
LIMIT <varying per measurement>
```

For the preparation we will start with pre-loaded graphs and dictionaries, meaning the measured computations start with the "convertGraph(g)"-function and ends with the "generateNodeVectors(fullVectors)"-function. We will run two tests in this scope, in one we will test the "glove-wiki-gigaword-100" and "glove-wiki-gigaword-300" dictionaries with multiple graph sizes to investigate the influence different dictionaries have and in the other we will only vary the graph size with the "glove-wiki-gigaword-100" dictionary to investigate what influence the graph size has.

For the generation we will start after completing the preparation and measure the starting with the "ambiguify(config, nodeVectors, relationVectors)"-function and ending with the "populateAdditions(changes, g2)"-function. We will run one test to investigate the influence the number of generated nodes and relations has on the runtime.

5.5.1 Influence of different dictionaries on preparation

To investigate the influence different dictionaries have we will compare the two dictionaries on multiple differently sized graphs. The individual measurements will be shown in Table 5.1 and visualised in Figure 5.1.

numFullVectors	numDimensionsOfDictionary	timesMedian
2000	100	4.634643
2000	300	4.639974
5000	100	11.593553
5000	300	11.593975
7000	100	16.203833
7000	300	16.206599

Table 5.1: Runtime of preparation with different dictionaries

As expected we can observe a linear increase of the runtime in regard to the graph size and almost no variation between individual runs. The impact the dictionary has on the runtime is irrelevant in all investigated graphs.

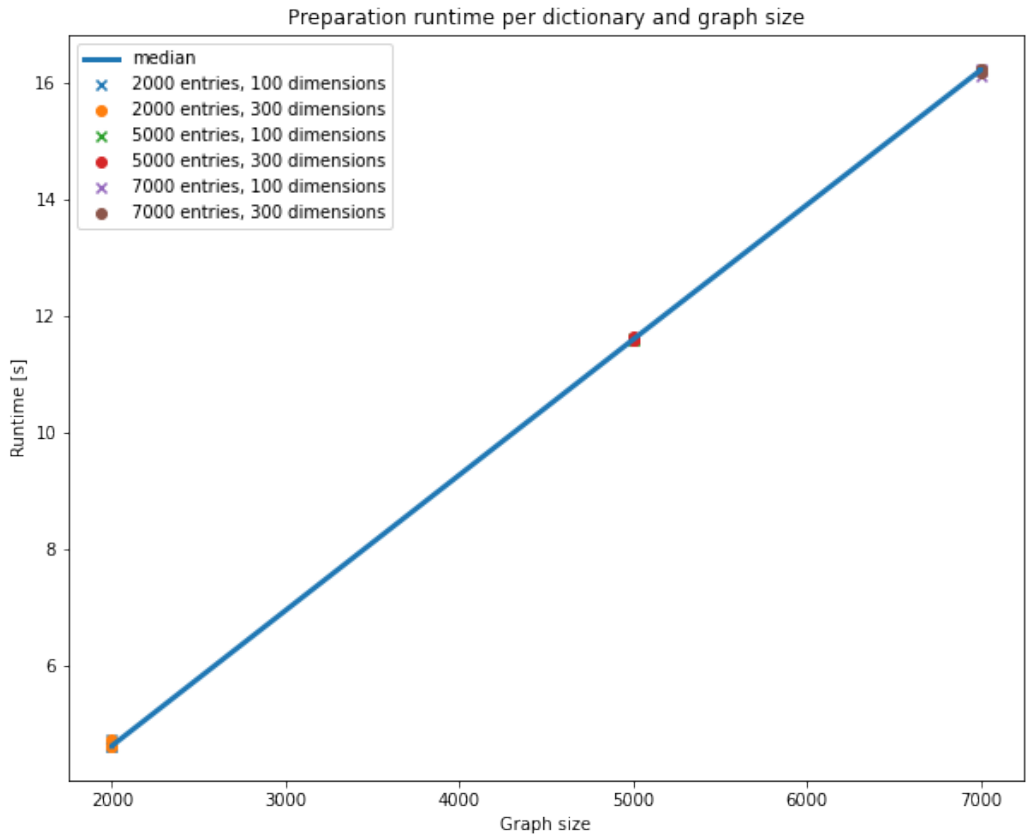


Figure 5.1: Runtime per dictionary and graph size

5.5.2 Influence of different graph sizes on preparation

To investigate the influence graph size has we will compare multiple differently sized graphs using the "glove-wiki-gigaword-100" dictionary. The individual measurements will be shown in Table 5.2 and visualised in Figure 5.2. The datasets are characterised by the number of full vectors (or triples) they contain.

numFullVectors	numNodeVectors	numRelationVectors	timesMedian
2000	545	26	4.814626
3000	773	29	6.978133
4000	1008	32	9.268512
5000	1233	34	11.588389
7000	1663	38	16.129400
10000	2348	45	23.190552

Table 5.2: Runtime per graph size

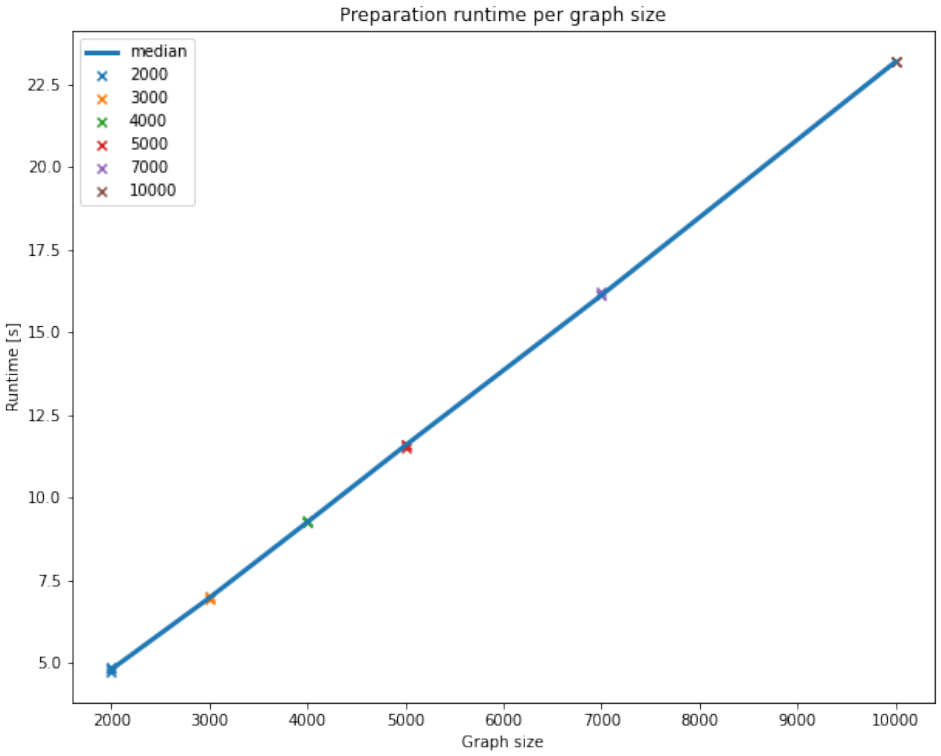


Figure 5.2: Runtime per graph size

We can observe the same expected linear increase of the runtime as when inspecting the influence of different dictionaries, and again almost no variation between individual runs. We can also observe that neither the number of distinct nodes or distinct relations had any noteworthy digressions in their relation to the total number of triples. The runtime of even the largest graph easily fulfils the requirements for practical research.

5.5.3 Influence of the number of generated nodes and relations on generation

To investigate the influence the number of generated nodes and relations have we will compare multiple configurations on a graph with 2000 triples and the "glove-wiki-gigaword-100" dictionary. The individual measurements will be shown in Table 5.3 and visualised in Figure 5.3.

numGeneratedNodes	numGeneratedRelations	timesMedian
5	5	4.086421
10	10	7.533231
15	15	10.977045
20	20	14.290249
25	25	17.731888

Table 5.3: Runtime per generated nodes and relations

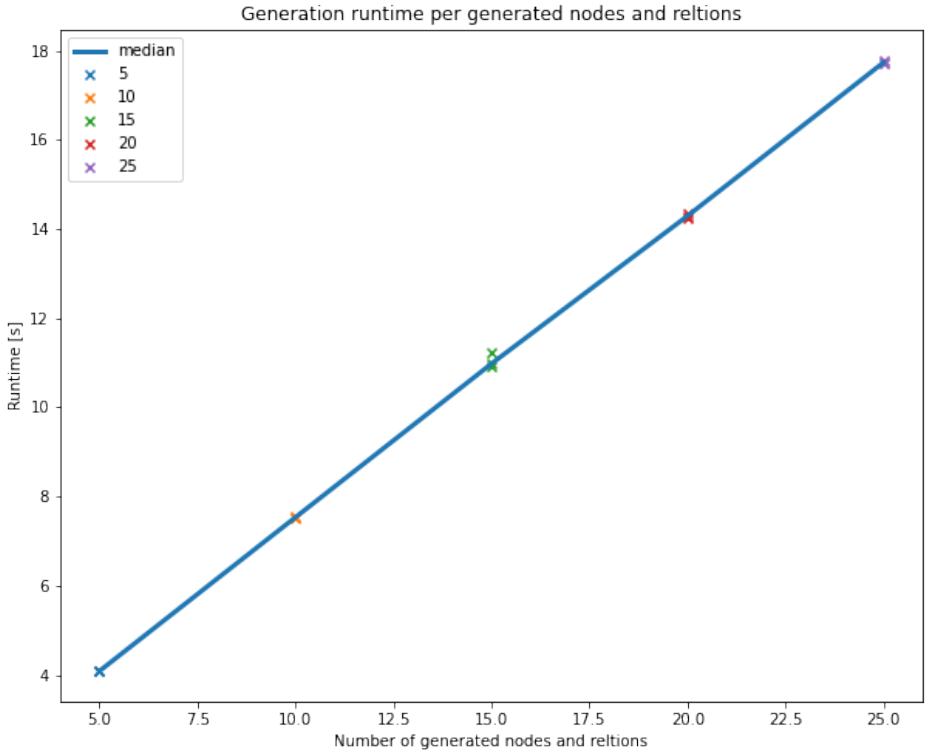


Figure 5.3: Runtime per generated nodes and relations

As expected we can observe a linear increase of the runtime with increasing number of generated nodes and relations. While there is a bit of variation between individual runs the median is not influenced by them. Even generating a total of 50 nodes and relations fulfils our requirements for practical research.

5.6 Repository

Since we noticed many issues with the availability of data and code used in papers during our literature research (see section 2.3), we decided to take care and prevent similar issues for our thesis. Our code is made available together with all our used and created data in a github repository[2].

The complete graphs of user- and code-workflow (see section 4.1) are also provided in the repository, but were too large to include in the thesis directly.

In this chapter we described our reference implementation written in python. We explained design decisions and their reasoning (see section 5.1), used libraries (see section 5.2), how to extend the exiting methods (see section 5.3), the main functions the code is comprised of (see section 5.4), the performance measurements (see section 5.5) and finally share the repository (see section 5.6) containing all software artefacts that allow us to conclude our second research question. As we are able to generate configurable amounts and severities of ambiguities in almost arbitrary knowledge graphs (see section 7.2 for limitations) we can conclude that we were able to adequately answer the second research question.

[2]Repository: `https://github.com/Malspherus/AmbiVec`

CHAPTER 6

Evaluation

In this chapter we evaluate the output of our "AmbiVec"-workflow. For this purpose we chose to do a questionnaire on Amazon Mechanical Turk (see section 2.5). The evaluation is structured following the guidelines discussed in Pesquitas' "A Framework to Conduct and Report on Empirical User Studies in Semantic Web Contexts" [PILL18] starting with the purpose (see section 6.1) and users (see section 6.2), going over the tasks (see section 6.3), setup (see section 6.4) and procedure (see section 6.5) and ending with the analysis (see section 6.6) and conclusion (see section 6.7). In the conclusion we are able to answer our third research question.

6.1 Purpose

Pesquitas' paper distinguishes between four non-exclusive categories in regards to the purpose of empirical user studies, namely: "exploration" (search for information not clearly defined from the start), "search" (data examination with focus on specific information), "creation" (the data itself is the output that was required) and "management" (data validation, assessment or modification of existing data). Our questionnaire best matches with the "search" category of the paper as we specifically want to answer our third research question but also includes some aspects of the "exploration" category as additional interesting insights might be found.

Human-likeness
The first purpose of the evaluation is to find out whether our generated ambiguities match human-like ambiguities closely enough for real-world applications as asked by research question 3. Since we don't have access to human-made ambiguities usable as a baseline we are only able to investigate whether our approach is in principle capable of generating ambiguities that are perceived to be human-like, but not compare them to the baseline. For this we can formulate two hypotheses:

H_0: we are unable to generate any ambiguities that are perceived to be human-like
H_1: we are able to generate at least some ambiguities that are perceived to be human-like

Correlation
The second purpose is to ensure that the severity parameter of our configuration (see section 5.3) (which is based on the calculated similarity between the original element and the ambiguified element) is able to influence the perceived amount of ambiguity as asked by our third research question. For this we can formulate two hypotheses:
H_0: the configured severity has no influence on the perceived severity
H_1: the configured severity has some influence on the perceived severity

Our third research question can be proficiently answered by those two hypotheses. During this evaluation the findings of research question 1 ("What do human ambiguities look like?") and research question 2 ("Can we artificially generate such ambiguities?") will also be tested indirectly. Should our findings for one of those questions prove faulty it will be reflected in the findings of the evaluation as well.

6.2 Users

Our users (usually called workers in the context of crowdsourcing) are not chosen directly by us, but instead we can restrict the population of all available workers by introducing restrictions. In our case we chose to only use one restriction, namely that workers should be "masters" according to AMT. This is meant to filter out new and inexperienced workers or workers known to produce untrustworthy results, but the exact rules are not disclosed by Amazon.

Apart from our restriction the workers choose which questionnaires they want to answer and can stop doing so at any moment. In return the requester is able to accept or reject individual answers if they decide the workers did not answer them honestly or correctly. In general the answers received from AMT come from a paid layman population without restrictions of gender, age, background and other criteria if done with a high enough number of workers. This would match exactly with our intended population.

For our tasks we choose to collect 5 answers from different workers per question and paid 0.25$ per accepted answer. Collecting answers from significantly more users was not possible due to budget restrictions. We dropped the "master"-restriction after we got 213 of the 300 answers, as we weren't able to get any more responses, but even after that no additional responses were submitted.

To summarise, we commissioned a total of **300 answers** which could be given by a minimum of **5 distinct workers** if they answered all 60 questions. In actuality we received **213 individual answers** from **11 workers**. Our workers were recruited by **self-selection over AMT** and while we didn't require specific expertise content-wise, we did require the workers to have the "master" qualification.

6.3 Tasks

Our tasks' content was generated using the "AmbiVec" workflow (see section 4.1) but with a slight modification. We made sure that the same triples were selected for ambiguification (one set for all nodes and one set for all relations) to be able to compare the influence different amounts of severity have. Usually every configured method would select a new group of triples.

For the dataset we had the choice to either use existing datasets or to select a subset of a big, publicly available knowledge graph using SPARQL. We inspected various datasets with the goal to select one that is both roughly considered to be within general knowledge and features a high number of close nodes and relations in the dataset to be able to generate close ambiguities. Since most pre-made datasets couldn't be considered to be within general knowledge or didn't feature enough similar relation types to be of value for the evaluation we chose to select a subset of DBpedia. We investigated the areas of music and associated people, food and ingredients, countries and entities within them and movies and the people making them. We chose the movies subset as it featured the most amount of similar relation types. The query selects all triples where the relation matches one of the types dbo:starring, dbo:writer, dbo:musicComposer, dbo:director or dbo:producer. The subset was limited to 5000 entries and is saved under the name "KGmovies.ttl".

The query we used for the selection of the DBpedia subset was:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>

construct {
  ?s ?p ?o .
}
WHERE {
  ?s (dbo:starring |
    dbo:writer |
    dbo:musicComposer |
    dbo:director |
    dbo:producer) ?o .
  ?s ?p ?o
}
LIMIT 5000
```

As dictionary we chose to use the "glove-wiki-gigaword-100"[1] provided via the gensim API. This dictionary is built using the data provided by Wikipedia and is a good fit for our dataset because our dataset was originally built from the same data.

For the evaluation we also chose to use the distance-method (see section 4.3.1) as it is the most straight-forward method to generate the minimal amount of ambiguity. We generated 5 triples for every distance between and including 0 and 4 for both nodes and relations. We also generated 5 triples for nodes and relations using the random-method (see section 4.3.3) with the intention of using them as a baseline.

After generating the ambiguities we transformed the data to the necessary format for AMT and added the code required by the questionnaire. The resulting file was given the name "amt.csv". Some examples of this output are shown in Table 6.1 where the original subject, predicate and object are compared to the same triple with one generated subject, predicate or object.

index	orig. s	orig. p	orig o.
12	The Hills Have Eyes 2	director	Martin Weisz
32	Way of the Warrior (video game)	gameArtist	Jason Rubin

index	new s	new p	new o
12	The Hills Have Eyes (1977 film)	director	Martin Weisz
32	Way of the Warrior (video game)	backgroundArtist	Jason Rubin

Table 6.1: Examples of generated ambiguities

6.4 Setup

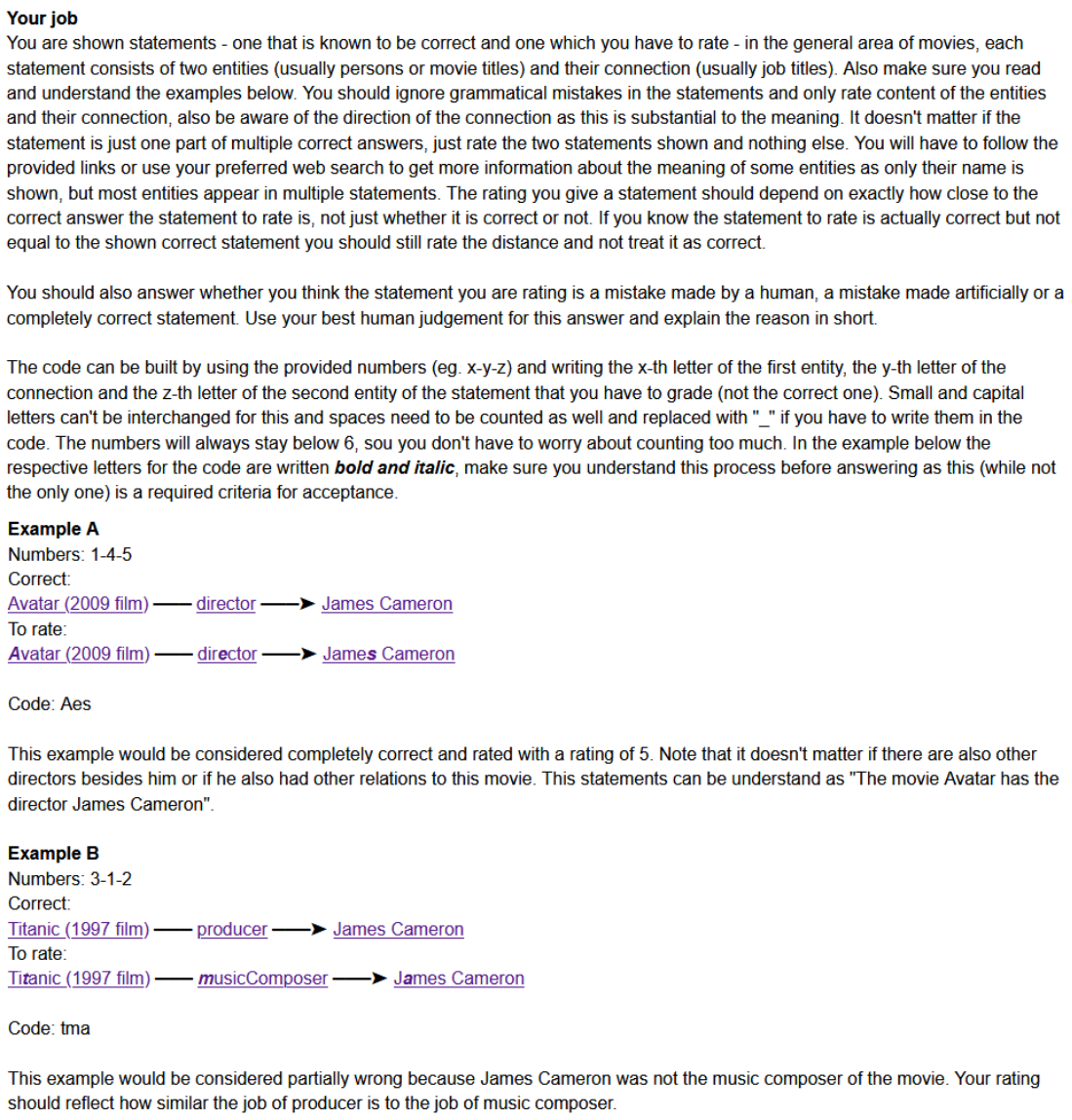
We chose to create our own questionnaire layout on AMT as none of the available AMT task templates fit our requirements well enough. This questionnaire was refined in two pilot studies with three and one selected workers respectively.

The questions were all requested via the default AMT interface, but as we don't have any influence or insight into the environment in which users answer our questions we can't know any details. The design of the instructions is shown in Figure 6.1 and one example question is shown in Figure 6.2.

Workers had to answer four non-optional question and could answer one optional question for every ambiguity. They first were shown the correct statement as well as the ambiguified statement they should investigate. In the first question they had to quantify how close the correct statement is to the ambiguified statement on a scale from 1 to 5 with 10 steps, which translates to the commonly used 5-star scale with half stars. In the second

[1] Website: `https://github.com/RaRe-Technologies/gensim-data`

question they had to state whether they think the shown ambiguity was caused by a human mistake or artificially generated. For cases in which both correct statement and ambiguified version were identical there also was the option to answer this question as "correct answer". In the third question workers had to briefly explain why they made this choice in text form. These answers were only used to identify problems with the questionnaire and not further investigated once no such problems were found. In the last mandatory question they had to build the correct code according to the shown numbers. The optional question allowed workers to enter a textual comment, but no feedback was given.



Your job

You are shown statements - one that is known to be correct and one which you have to rate - in the general area of movies, each statement consists of two entities (usually persons or movie titles) and their connection (usually job titles). Also make sure you read and understand the examples below. You should ignore grammatical mistakes in the statements and only rate content of the entities and their connection, also be aware of the direction of the connection as this is substantial to the meaning. It doesn't matter if the statement is just one part of multiple correct answers, just rate the two statements shown and nothing else. You will have to follow the provided links or use your preferred web search to get more information about the meaning of some entities as only their name is shown, but most entities appear in multiple statements. The rating you give a statement should depend on exactly how close to the correct answer the statement to rate is, not just whether it is correct or not. If you know the statement to rate is actually correct but not equal to the shown correct statement you should still rate the distance and not treat it as correct.

You should also answer whether you think the statement you are rating is a mistake made by a human, a mistake made artificially or a completely correct statement. Use your best human judgement for this answer and explain the reason in short.

The code can be built by using the provided numbers (eg. x-y-z) and writing the x-th letter of the first entity, the y-th letter of the connection and the z-th letter of the second entity of the statement that you have to grade (not the correct one). Small and capital letters can't be interchanged for this and spaces need to be counted as well and replaced with "_" if you have to write them in the code. The numbers will always stay below 6, sou you don't have to worry about counting too much. In the example below the respective letters for the code are written ***bold and italic***, make sure you understand this process before answering as this (while not the only one) is a required criteria for acceptance.

Example A

Numbers: 1-4-5

Correct:

Avatar (2009 film) —— director ——➤ James Cameron

To rate:

Avatar (2009 film) —— director ——➤ James Cameron

Code: Aes

This example would be considered completely correct and rated with a rating of 5. Note that it doesn't matter if there are also other directors besides him or if he also had other relations to this movie. This statements can be understand as "The movie Avatar has the director James Cameron".

Example B

Numbers: 3-1-2

Correct:

Titanic (1997 film) —— producer ——➤ James Cameron

To rate:

Titanic (1997 film) —— musicComposer ——➤ James Cameron

Code: tma

This example would be considered partially wrong because James Cameron was not the music composer of the movie. Your rating should reflect how similar the job of producer is to the job of music composer.

Figure 6.1: AMT instructions

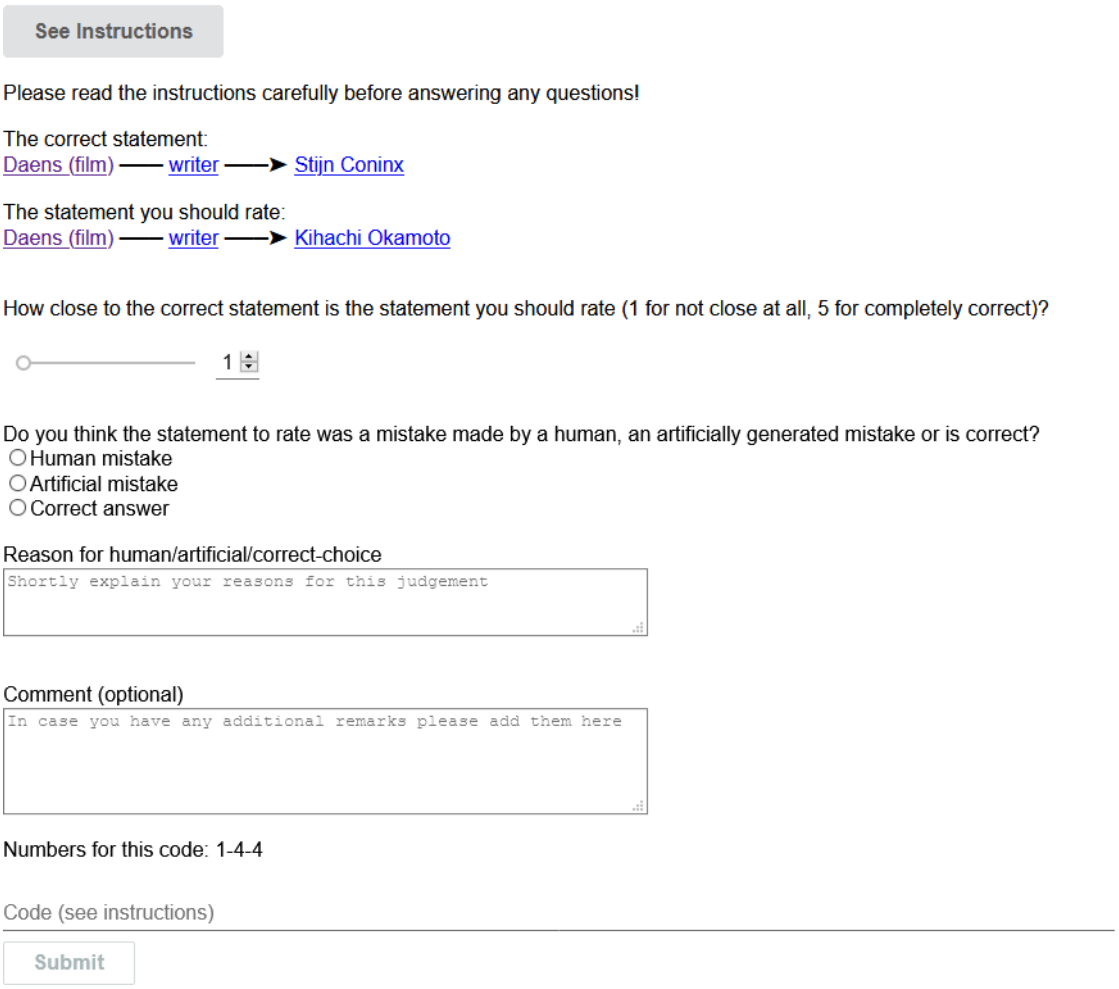


Figure 6.2: AMT questionnaire

6.5 Procedure

As with the environment AMT already mostly defines the procedure. Workers log in to the site, select which questionnaires they want to answer and then work their way through the tasks.

In our task they are first instructed to read the instructions carefully. To ensure that users actually do this and don't just click on random answers we introduced a changing code that has to be answered in addition to every question. While this can't guarantee the correctness of answers it will quickly discourage workers that don't plan on answering questions earnestly.

Workers are then shown two statements, one is the original data from the graph and the

other one is the ambiguified version. The entities are linked to their DBpedia pages to provide further information to the workers should they require it.

Workers are then queried about their views on how close the two statements are to each other, whether the difference was caused by a human, artificially or (in case there is no difference) if it was actually correct. They also are asked to describe the reason for their decision on the source in a few words, but not about their decision regarding the rating. In case something is unclear or not working correctly we also allowed them to enter a comment.

6.6 Analysis

As stated before (see section 6.1) our main goals are to establish whether our ambiguities are human-like and whether their perceived severity follows our configured severity.

To check if answers are actually perceived as human-like by workers we usually would compare their answers to a human-generated baseline and analyse whether we are able to achieve similar values. Since we don't have access to any baseline, this question can't be answered definitely at this point in time. Instead we will investigate our hypothesis that we are able to generate at least some ambiguities that are perceived to be human-like. To check our hypothesis that a higher severity in the configuration also causes a higher perceived severity for workers we can compute the correlation of these two values.

To ensure our data is not subject to any faults we also investigate some other values and only use data for our analysis that came from accepted answers. We set three criteria to decide if an answer was accepted or rejected.

- **Code requirement** The first requirement was for workers to have entered the correct code. Four users with a combined total of 12 answers seemingly didn't read the instructions and only entered wrong codes, but some users only made small mistakes in the code and would probably have provided a valid answer. Since recognising these near-misses isn't trivial we also filtered them out.
- **Correctness requirement** The second criteria was whether the answer was obviously wrong. For this we only checked if a completely correct question was answered with a rating of 5 and the source had the value "correct" selected, otherwise the answer was rejected. All questions that were not completely correct in the first place were not rejected due to this requirement.
- **Worktime requirement** The third criteria was an irregular worktime. To decide if any given worktime was irregular we calculated the inter-quantile-ratio between the 25%- and the 75%-quantile and permitted worktimes as long as they didn't exceed trice the ratio. This also causes the first answer of every user to be filtered if they don't read the instructions before accepting.

We also added two metrics to be able to investigate the distances between the vectors as used in the configuration. These values were also calculated for elements generated using other methods to compare them properly.

- **Severity** The first metric is called severity and is equivalent to the configured distance of the distance-method (see section 4.3.1). The original element has a distance of 0, the next closest element has a distance of 1 and so on.
- **Closeness** The second metric is called closeness and is almost equivalent to the configured closeness as used by the closeness-method (see section 4.3.2), but with a slight variation for the sake of data analysis. The original element has a closeness of 1, all other elements have a closeness depending on the absolute cosine similarity between their vector and the original elements vector. The inverse of the original element would have the furthest possible distance with a value of 0 if it existed in the knowledge graph.

In further analysis (starting with section 6.6.2) only the data filtered by the criteria as described will be used.

6.6.1 Analysis of entire collected data

We will briefly investigate the results of the entire, unfiltered data in Table 6.2 and Table 6.3, but since this data includes obviously wrong answers we will only do the in-depth-analysis on the filtered data.

In the unfiltered data shown in Table 6.2 we can observe that some answers are perceived as human-like, but in general answers are considered to be artificially generated at a higher rate.

method	mistakeSource	count
dist	artificial	89
dist	correct	51
dist	human	35
random	artificial	28
random	correct	2
random	human	8

Table 6.2: Unfiltered count per method and source

In Table 6.3 we can observe the correlation between different variables. We can observe the correlation of rating and the worktime to be slightly negative (meaning the higher the rating the faster users were able to answer it). The correlation between rating and severity is more negative (meaning that higher severities caused lower ratings) than in

the case of the filtered data, but not as much as the filtered data without the randomly generated elements. The correlation between rating and closeness is positive (meaning a higher closeness tends to cause a higher rating) and close to the filtered data.

	rating	WorkTimeInSeconds	severity	closeness
rating	1.000000	-0.091471	-0.174340	0.450930
WorkTimeInSeconds	-0.091471	1.000000	-0.029612	0.000701
severity	-0.174340	-0.029612	1.000000	-0.501202
closeness	0.450930	0.000701	-0.501202	1.000000

Table 6.3: Overview of correlation of unfiltered data

6.6.2 General analysis

We will briefly investigate some general characteristics of the data to get a better understanding of the kind of answers we received and to detect possible irregularities.

Figure 6.3 depicts the amount of answers per rating of individual workers (named after their WorkerId on AMT) and the mean value. The total amount of answers per rating is also shown in Table 6.4. There is a noticeably higher amount of answers with a rating of 1, 3 and 5 which seem to happen because workers don't use the full scale of the rating but instead tend to answer mostly in absolutes (ie. correct, incorrect or in the middle). This behaviour was more extreme for some workers, especially those that also failed to answer the codes correctly.

rating	count
1.0	96
1.5	5
2.0	3
2.5	3
3.0	13
3.5	3
4.0	4
4.5	4
5.0	31

Table 6.4: Total answers per rating

In Figure 6.4 a scatterplot of the severities per rating is shown. The four combinations of method (dist and random) and type (node and relation) are colour-coded and the random method is additionally plotted as 'x' instead of 'o' like the distance method.

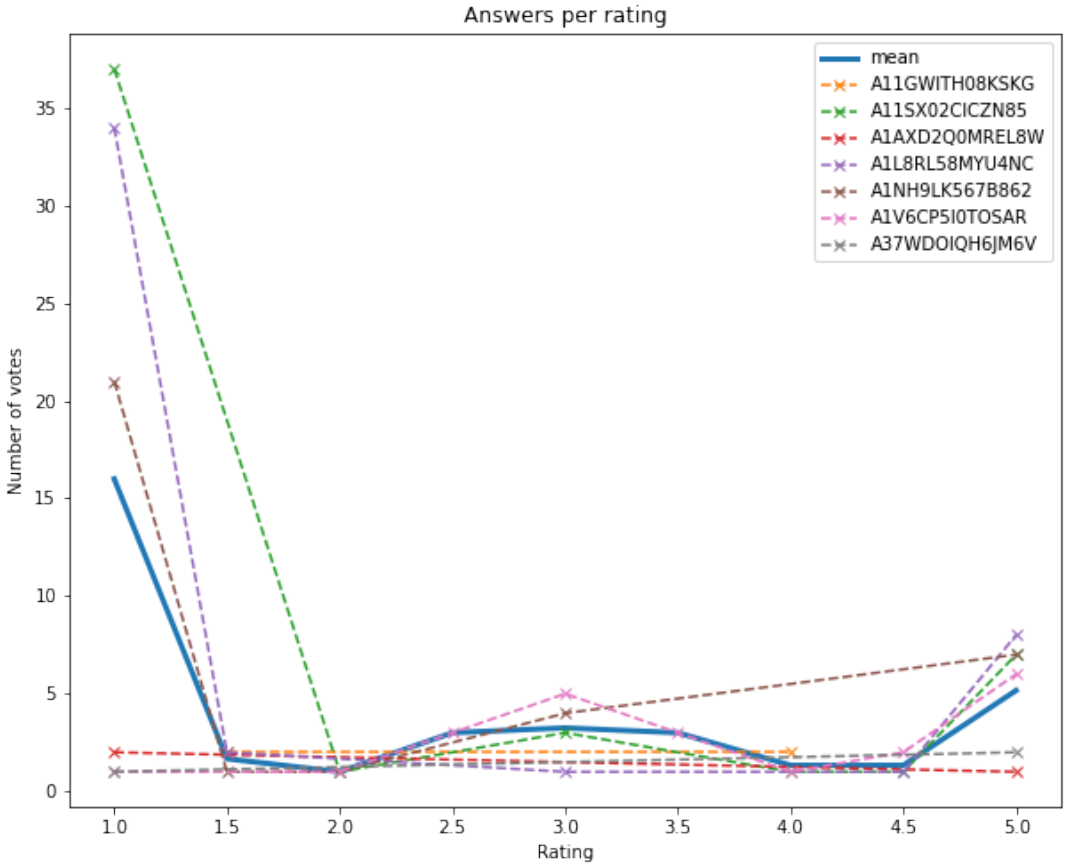


Figure 6.3: Answers per rating

The random relations can be easily identified in the scatterplot by their higher severity, but the rating doesn't reflect this as clearly, especially when compared with the nodes. This may be caused by the relatively smaller number of possible relation types and their relatively higher similarity when compared to the nodes. It simply is not possible to select a relation with a severity as high as the random nodes that, as a consequence, were also easily identified by most workers and received a low rating.

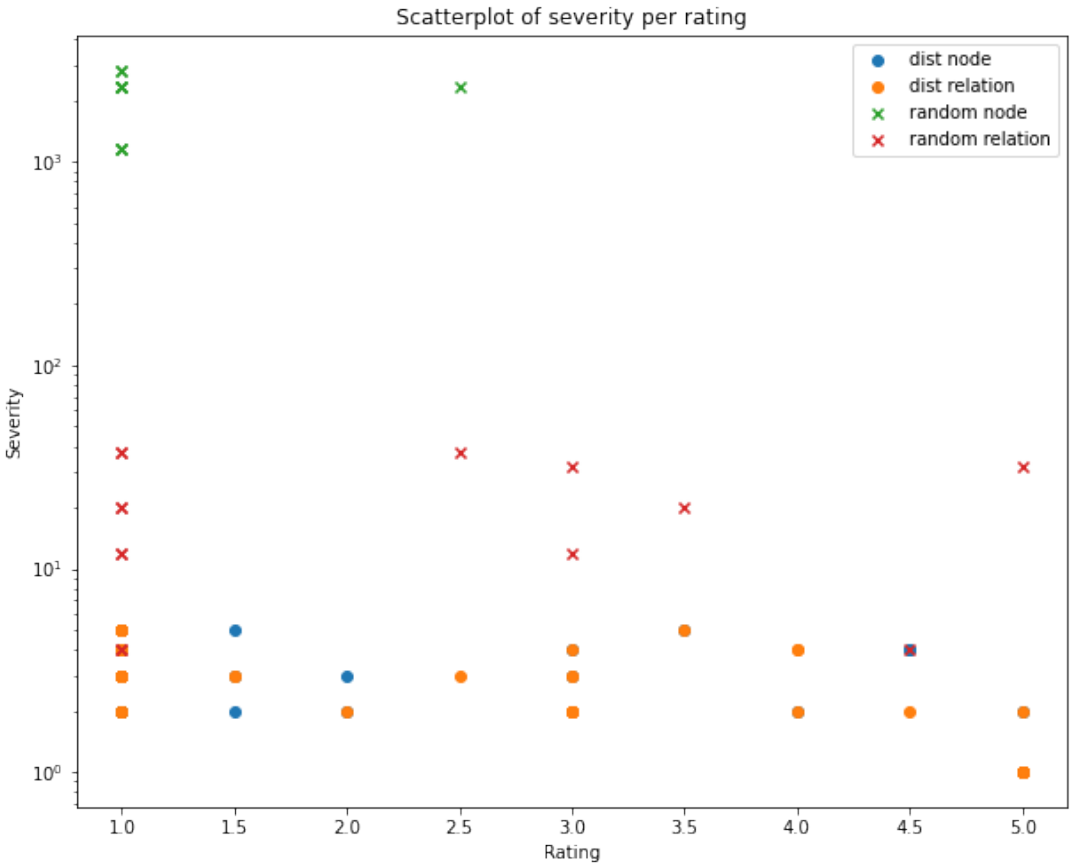


Figure 6.4: Scatterplot of severity per rating

6.6.3 Human-likeness

Table 6.5 shows the number of answers per method and source. While we can't compare the numbers against a human baseline we can still compare it against the randomly generated elements to some extent. When comparing the distribution we can observe that there is only a marginal difference between the distance method (23.08% human-like, 76.92% artificial) and the random method (21.17% human-like, 78.26% artificial) before investigating further.

method	mistakeSource	count
dist	artificial	80
dist	correct	34
dist	human	24
random	artificial	18
random	correct	1
random	human	5

Table 6.5: Count per method and source

Doing further analysis of the distributions per rating in Table 6.6 and Figure 6.5 we can observe that users generally consider statements more human-like the higher the rating was and also that a vast majority of the 'artificial'-answers was given to the lowest rating.

rating	mistakeSource	count
1.0	artificial	85
1.0	human	11
1.5	artificial	5
2.0	human	3
2.5	artificial	3
3.0	artificial	3
3.0	human	10
3.5	artificial	1
3.5	human	2
4.0	artificial	1
4.0	correct	2
4.0	human	1
4.5	correct	2
4.5	human	2
5.0	correct	31

Table 6.6: Count per rating and source

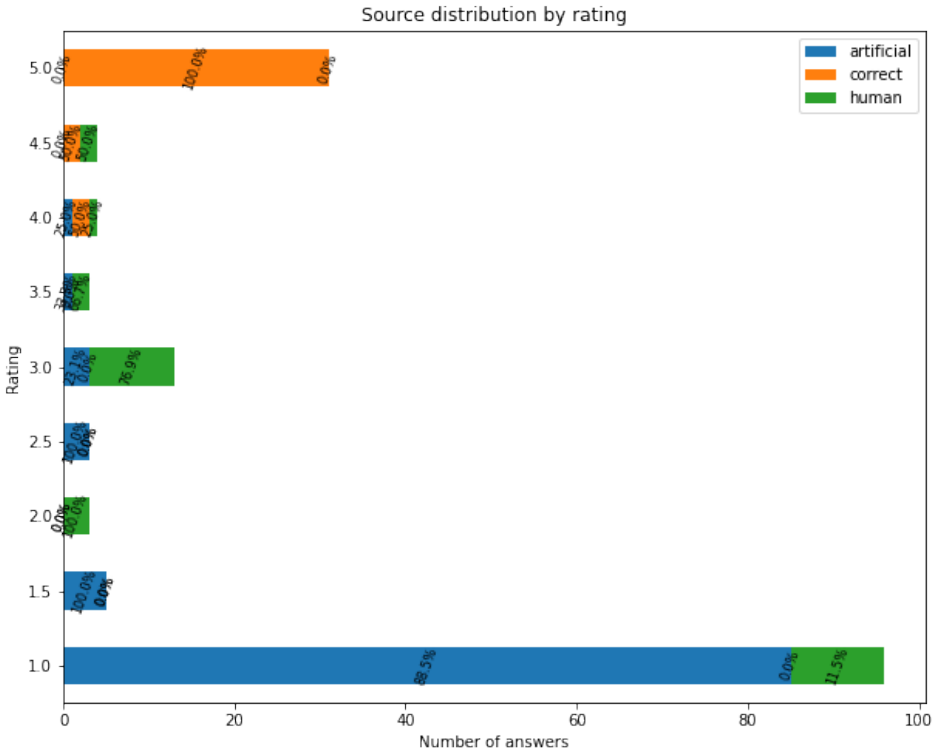


Figure 6.5: Source distribution by rating

In Table 6.7 we investigate the the number of answers each source got, split by severity. Here the distribution is more complex, even small severities tend to be considered artificial whereas higher severities are more evenly distributed.

We can therefore conclude, that while the human-likeness of the distance-method didn't result in much better results than randomly generated elements, it was able to trick 23.08% of workers. This means we can reject our hypothesis H_0 in favour of our alternative hypothesis H_1 saying that we are able to generate some ambiguities that are perceived to be human-like.

severity	mistakeSource	count
0	correct	28
1	artificial	16
1	correct	3
1	human	12
2	artificial	22
2	human	5
3	artificial	20
3	correct	3
3	human	5
4	artificial	24
4	human	3
11	artificial	3
19	artificial	1
19	human	2
31	correct	1
31	human	1
36	artificial	3
1165	artificial	2
1165	human	1
2361	artificial	5
2808	artificial	2

Table 6.7: Count per severity and source

6.6.4 Correlation of configured and perceived severity

In Figure 6.6 we depict a boxplot of severity and rating. We can see that the lower half of the ratings has a lot more spread than the higher half and features most of the high severity answers. This means that workers agreed more on higher ratings with low severity (as generated by the distance method) than they did on lower ratings with high severity (as generated by the random method) or low severity (as generated by our approach). This relation is further investigated using the correlation.

Table 6.8 shows the overview of correlation without any regard to method or whether nodes or relations were viewed. The relevant p-values are rounded to three digits after the comma and displayed in addition to the correlation. As intended there is a negative correlation between rating and severity (meaning a lower severity will cause a higher rating) and rating and worktime (meaning a lower worktime tends to result in a higher rating) and a positive correlation between rating and closeness (meaning that replacements closer to the correct element tend to get a higher rating). This means that our configuration causes the rating to be influenced correctly by our approach, although the presence of the randomly generated elements with high severity included in this

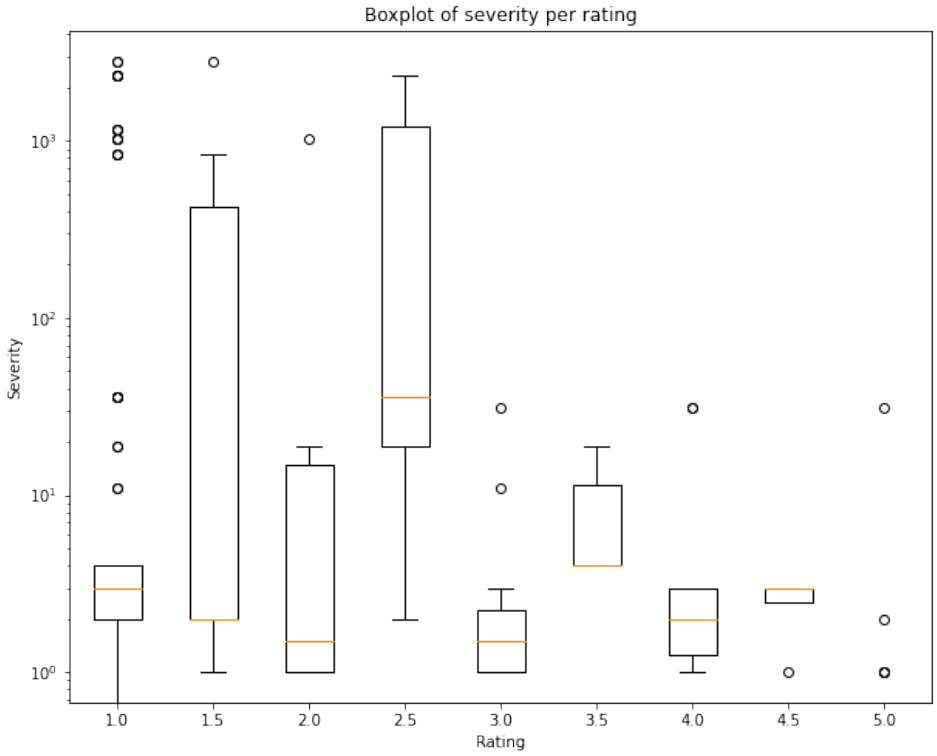


Figure 6.6: Boxplot of severity per rating

correlation influences the values negatively. This effect is investigated more closely in the next paragraphs.

	rating	WorkTimeInSeconds	severity	closeness
rating	1.000000	(-0.224183, 0.004)	(-0.155466, 0.049)	(0.471142, 0)
WorkTimeInSeconds	-0.224183	1.000000	0.010070	-0.098724
severity	-0.155466	0.010070	1.000000	-0.521641
closeness	0.471142	-0.098724	-0.521641	1.000000

Table 6.8: Correlation overview

In Table 6.9 we do an in-depth analysis of the correlation matrix and further split it by method and element type. The relevant p-values are rounded to three digits after the comma and displayed in addition to the correlation.

For nodes ambiguified with the distance-method this shows more significant correlation

of rating to closeness and severity, but a slight decrease of the correlation between rating and worktime.

For relations ambiguified with the distance-method the correlation between rating and worktime is almost the same, also for rating and closeness with only a minimal decrease. The correlation between rating and severity is even more significant than before and is actually the most significant correlation the rating has.

The nodes ambiguified with the random-method show a slightly increased negative correlation between rating and worktime (meaning a higher worktime causes a lower rating) but a low, positive correlation between rating and severity (meaning a higher severity tended to get higher ratings) and the correlation between rating and closeness now being negative (meaning a higher closeness tended to get a lower rating) but close to zero.

The relations ambiguified with the random method show the highest negative correlation between rating and worktime (meaning a higher worktime tended to result in a lower rating), but also shows a low, positive correlation between rating and severity (meaning a higher severity tended to get higher ratings). The correlation between rating and closeness is slightly negative (meaning a lower closeness tended to get higher ratings).

Due to the low p-values calculated for the distance method we can conclude that the hypothesis H_0 can be rejected in favour of the alternative hypothesis H_1, meaning that we are indeed able to linearly influence the perceived severity with our configured severity.

method	type		rating	WorkTimeInSeconds	severity	closeness
dist	node	rating	1.000000	(-0.202658, 0.114)	(-0.580155, 0)	(0.788691, 0)
		WorkTimeInSeconds	-0.202658	1.000000	0.146614	-0.047184
		severity	-0.580155	0.146614	1.000000	-0.457776
		closeness	0.788691	-0.047184	-0.457776	1.000000
	relation	rating	1.000000	(-0.221788, 0.054)	(-0.728948, 0)	(0.444616, 0)
		WorkTimeInSeconds	-0.221788	1.000000	0.123902	-0.252233
		severity	-0.728948	0.123902	1.000000	-0.330346
		closeness	0.444616	-0.252233	-0.330346	1.000000
random	node	rating	1.000000	(-0.383738, 0.274)	(0.142610, 0.693)	(-0.057825, 0.874)
		WorkTimeInSeconds	-0.383738	1.000000	0.008433	-0.028107
		severity	0.142610	0.008433	1.000000	0.815937
		closeness	-0.057825	-0.028107	0.815937	1.000000
	relation	rating	1.000000	(-0.523381, 0.055)	(0.091828, 0.755)	(-0.155185, 0.596)
		WorkTimeInSeconds	-0.523381	1.000000	0.125210	-0.061037
		severity	0.091828	0.125210	1.000000	-0.930268
		closeness	-0.155185	-0.061037	-0.930268	1.000000

Table 6.9: Correlation and relevant p-values per method and per type

6.7 Conclusion of the evaluation

The first purpose of our evaluation was to evaluate whether generated ambiguities match human created ones close enough for real-world applications. While we weren't able to show how human-like the generated ambiguities are due to a lack of a suitable baseline we were able to show that workers tend to consider higher rated ambiguities to be more human-like. We were also able to show that this is not the case for low severities, where we suspect that a mismatch of classes (eg. a movie being replaced with a person) could be the cause for this. Even under these circumstances, 23.08% of workers thought artificial ambiguities to be human made, allowing us to reject our hypothesis H_0 in favour of our alternative hypothesis H_1.

The second purpose of the evaluation was to evaluate how well the configuration used to generate ambiguities influences the perception of the workers when rating the ambiguities. Regarding the impact of our configuration on the perceived severity we were able to show that our configuration does indeed work well for small severities. Whether this also holds for higher severities wasn't clear from the small amount of randomly generated ambiguities in this range, as ratings generally get more inconsistent the higher the severity becomes. We suspect this may be caused by the lack of reference data in this area, causing workers to only see a few high-severity-questions over the course of the questionnaire with no option to go back once more experienced and reevaluate their earlier answers after they completed them. The low p-values calculated for our correlation allow us to reject our hypothesis H_0 in favour of our alternative hypothesis H_1.

Seeing that we were able to reject both hypotheses H_0 in favour of the alternative hypotheses H_1 we can conclude that our third research question is as sufficiently answered as possible.

CHAPTER 7

Conclusion and further work

This chapter concludes the outcome of all of our research questions and their results (see section 7.1). We also expand on the limitations our work is experiencing (see section 7.2) and give insight in further work that could improve our workflow in various areas (see section 7.3).

7.1 Conclusion

In this section we summarise and conclude the outcome of our research structured by our research questions.

- **RQ1: Ambiguities and sources** We defined what we consider to be an ambiguity in the context of crowdsourcing in contrast to similar concepts, and based on that knowledge we identified 8 sources of ambiguities, namely "Subjective ambiguity", "Ambiguity of specialisation", "Inherited ambiguity", "No knowledge", "Most common answers of other users", "Temporal changes", "Failed attempts to correct data" and "Malicious behaviour". We also grouped the output of the 8 sources into 3 uniform characteristics that can be artificially generated, namely "Similar answers", "Opposite answers", and "Different viewpoints".
- **RQ2: Creation of ambiguities** We developed the "AmbiVec" workflow and a reference implementation in python that allow for robust and performant generation of ambiguities in arbitrary knowledge graphs as well as the detection and prevention of low quality early in the process. This generation is done by our methods, which leverage vector embeddings to calculate the similarity between the elements of the graph. These similarity values can then be used to rank all appropriate candidates for specific ambiguities and select the mot fitting one according to the given configuration.

- **RQ3: Evaluation of ambiguities** We designed and ran an experimental questionnaire using crowdsourcing and analysed the results. Our evaluation shows that while the configuration of severity in our approach translates to predictable perception of severity, we can't draw a definitive conclusion on the human-likeness of generated ambiguities due to the lack of a baseline for comparison, but we were able to show that, in principle, workers consider the generated ambiguities to be human-made.

We can therefore conclude that our research question "How can one introduce a well defined, structured and measurable amount of adequately human-like ambiguities as defined by our research into a pre-existing knowledge graph?" was mostly achieved by our "AmbiVec" approach, although a final conclusion on the human-likeness of generated ambiguities, while promising, will require further research once suitable datasets become available.

7.2 Limitations

Our work is subject to some limitations that were caused by external factors, the restricted scope of a master thesis or monetary restrictions. While we unfortunately can't alleviate these limitations we will clarify where the boundary of our work lies in as much detail as necessary.

7.2.1 Input data restrictions

To use our workflow a knowledge graph and a dictionary are required. Both of these underlie some restrictions to ensure results that do not degrade in quality. For the knowledge graph it is required that a high enough number of distinct node and relation types is present. The exact number depends on the structure of the graph and the required output quality. The dictionary has to be able to resolve enough node names to prevent a degrading of output quality, but as before the exact number depends on the structure of the knowledge graph. The detection of low quality output is described in section 4.4.

7.2.2 Internal validity

Since we were unable to retrieve any datasets containing authentic, annotated human mistakes (see section 2.3) we are not able to use them in our evaluation. This means that while we can evaluate our approach intrinsically, we are unable to compare the results with a human baseline. For intrinsic evaluation we investigated the correlation between our configured and worker-perceived severity and the possibility of workers perceiving generated ambiguities to be human-made, but we are not able to investigate the correlation of the closeness and severity of human made mistakes with their worker-perception or to compare the human-likeness of generated ambiguities to human-made ambiguities. Therefore, we also are unable to answer our third research question completely.

7.2.3 External validity

Due to time and budgetary restrictions we were only able run a single questionnaire with a scope of 300 individual answers set in a single domain. We therefore can't yet generalise that our approach will achieve comparable output quality in other domains, with datasets containing other characteristics or for severities outside of the area we investigated. As our planned scope of 300 answers would correspond to a minimum of 5 distinct workers answering the whole questionnaire a larger experiment, both in content and number of participants would help to ensure the feasibility of our approach further.

7.2.4 Conclusion validity

A downside of using AMT for our questionnaire was the inability to control participants surroundings while answering. While there were incentives for workers to follow the instructions and answer truthfully in the form of the monetary compensation and the rating we gave them, we can't guarantee their intent or a proper context for the questionnaire. We tried to alleviate this by using a content-dependent code and by filtering out obviously inconsistent answers (see section **??**).

7.3 Further work

During the creation of this thesis we found some areas that could profit from improvements and new ideas. They are structured by the area they improve the most, but might also be beneficial for other applications.

7.3.1 Improvements to runtime-performance

While performance was satisfying for our proof-of-concept other projects especially with larger graphs might profit from additional improvements to reduce runtime requirements.

One such improvement would be to not compare every vector to ever other vector in the graph when generating ambiguities. One way to implement this is the Annoy-library[1] as it allows us to first find the approximate k-nearest-neighbours and only do exact calculations on this neighbours. While this will cause a small decrease in quality, at some point the reduction in computation time will outweight this downside.

Most calculations done can also be adapted to make better use of parallel computation and for example run on a cluster instead of a single machine. While this would not reduce the computational requirements, it could decrease the waiting time significantly due to parallelisation. In addition to that all cores of a CPU as well as graphic cards could be used which are also generally more optimised for this kind of calculations.

[1] Website: `https://github.com/spotify/annoy`

7.3.2 Improvements to quality

For graphs where the quality of generated ambiguities starts to decrease some additional adaptions might improve the output. While graphs that don't have quality problems will also benefit from those changes in most cases they will not require these improvements.

We currently only use one dictionary for vectorisation, but since a dictionary holds the values and worldviews of its source multiple dictionaries that are used for different runs of the calculations could simulate different individual persons or groups having alternative views of the content of the knowledge graph. Since complete dictionaries can be dynamically and easily created out of some of the big knowledge graphs even a crowd of people could potentially be simulated to be able to generate datasets for crowd decision algorithms. It would also be possible to mix or compare general dictionaries to expert dictionaries for experiments.

If users mix general dictionaries and expert knowledge graphs or the other way around it will will cause low quality results due to unresolvable elements and inaccurate vectors, ideally the dictionary should be trained specifically for the knowledge graph and cover the same area and words. While detection of this problem via the metrics (see section 4.5) is already possible the caused decrease in quality can't be corrected. By implementing dynamic dictionary generation there should be improvement with this problem in those situations.

Multi-word nodes currently use a adequate, but not perfect technique to resolve them (see section 5.1). More sophisticated approaches can be implemented for higher quality results. While easier solutions involve interchanging median and mean or using the estimated positions more complex approaches like generating a dictionary also containing those specific names should provide better results.

The negative-method (see section 4.3.4) currently does not provide what humans would describe as a logical opposite. Word2vec-dictionaries in general can't provide bidirectional antonyms (eg. black and white, good and bad, ...) due to their structure. Implementing inaccurate estimates might be satisfying for some graphs, but a robust, general solution will require additional approaches like the use of WordNet [Fel98].

The current approach also ignores all structural properties of knowledge graphs except for the label. While this was an intentional design decision for the sake of generalisability, future development could also make use of these properties. Especially the class could be used to improve the output quality by restricting replacement selection to elements with similar classes or by using distances between classes. Some nodes might have similar classes and be easily mistaken like 'danube' and 'danube delta'. Both are bodies of water and the delta is arguably just another part of the river, but some might be hard to swap but still have close vectors like 'french' and 'france'. Similar constraints could improve vectors too.

7.3.3 Improvements to analysis

Improving the analysis functionality used to interpret dataframes during the process and the generated outputs afterwards could yield more insights to increase quality, prevent unnecessary computation and understand the generated data better.

More research into human mistakes and sources using a real dataset accompanied by a gold standard to compare against could lead to improved or novel generation methods that can be included into the workflow. Using datasets as reference the configuration can more closely follow human distributions, maybe even individual sources of ambiguity if the datasets provide these insights.

A quality metric like already used for the relations could be implemented for the nodes too. In addition to the percentage of usable elements it could also include the distance of the individual element to its types mean. The overall ambiguity of the graph could use this percentage to provide more accurate estimates.

If the calculated vectors are of low quality the output quality will also degrade. In some cases it might be preferable to only use vectors with a high enough quality and in exchange ignore parts of the graph for this goal. This would also cause the output quality to be more predictable than with the currently used random selection. The quality of the vectors can already be estimated by their abbreviation from its types mean, but the configuration and selection processes would need to be added.

The current distance to the mean are only available as minimum, maximum and average off all elements of a type, but this data could also be presented as a boxplot. The benefit would not only lie in presentation but also in the added outlier detection.

The node- & relation-vectors could also be compared visually in a 2D-graph with their mean, min and max deviated elements and the estimated position of the element. This would provide quick insights for single elements and could enable manual selection approaches instead of random or quality-based selection.

Additional evaluation approaches for complete graphs might provide interesting insights. For this purpose one or more full graphs could be modified and compared against their original versions using multiple detection or correction algorithms, performance in question answering or other related systems. The already implemented measurement of the graph ambiguity could be compared to the amount of detected ambiguities.

List of Figures

List of Tables

Bibliography

[10.17] *K-CAP 2017: Proceedings of the Knowledge Capture Conference*, New York, NY, USA, 2017. Association for Computing Machinery.

[10.18] *CSCW '18: Companion of the 2018 ACM Conference on Computer Supported Cooperative Work and Social Computing*, New York, NY, USA, 2018. Association for Computing Machinery.

[AD17] BENJAMIN TIMMERMANS LORA AROYO ANCA DUMITRACHE, OANA INEL. Crowdsourcing ambiguity-aware ground truth. Collective Intelligence 2017, 2017.

[Alt98] Gerry T.M Altmann. Ambiguity in sentence processing. *Journal: Trends in Cognitive Sciences*, 2(4):146 – 152, 1998.

[CBD20] Alessandro Checco, Jo Bates, and Gianluca Demartini. Adversarial attacks on crowdsourcing quality control. *J. Artif. Intell. Res.*, 67:375–408, 2020.

[CFG+18] Silvana Castano, Alfio Ferrara, Enrico Gallinucci, Matteo Golfarelli, Stefano Montanelli, Lorenzo Mosca, Stefano Rizzi, and Cristian Vaccari. Sabine: A multi-purpose dataset of semantically-annotated social content. In Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors, *The Semantic Web – ISWC 2018*, pages 70–85, Cham, 2018. Springer International Publishing.

[CHI19] *CHI '19: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA, 2019. Association for Computing Machinery.

[CK18] Yiling Chen and Gabriella Kazai, editors. *Proceedings of the Sixth AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2018, Zürich, Switzerland, July 5-8, 2018*. AAAI Press, 2018.

[d'A09] Mathieu d'Aquin. Formally measuring agreement and disagreement in ontologies. In Yolanda Gil and Natasha Fridman Noy, editors, *Proceedings of the 5th International Conference on Knowledge Capture (K-CAP 2009),*

September 1-4, 2009, Redondo Beach, California, USA, pages 145–152. ACM, 2009.

[DAW18] Anca Dumitrache, Lora Aroyo, and Chris Welty. Capturing ambiguity in crowdsourcing frame disambiguation. in publication at the sixth AAAI Conference on Human Computation and Crowdsourcing (HCOMP) 2018, 2018.

[Dem19] Gianluca Demartini. Implicit bias in crowdsourced knowledge graphs. In *Companion Proceedings of The 2019 World Wide Web Conference*, WWW '19, page 624–630, New York, NY, USA, 2019. Association for Computing Machinery.

[DHRB+11] Andreas Dengel, J. Hees, Thomas Roth-Berghofer, R. Biedert, and Benjamin Adrian. Betterrelations: Detailed evaluation of a game to rate linked data triples. In: Bach J., Edelkamp S. (eds) KI 2011: Advances in Artificial Intelligence. KI 2011. Lecture Notes in Computer Science, vol 7006. Springer, Berlin, Heidelberg, 10 2011.

[DMA+18] Markus De Jong, Panagiotis Mavridis, Lora Aroyo, Alessandro Bozzon, Jesse De Vos, Johan Oomen, Antoaneta Dimitrova, and Alec Badenoch. Capturebias: Supporting media scholars with ambiguity-aware bias representation for news videos. In Lora Aroyo and Anca Dumitrache , editors, *Joint Proceedings SAD 2018 and CrowdBias 2018*, CEUR Workshop Proceedings, pages 32–40. CEUR-WS, 2018.

[EW16] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. Conference: Joint Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS2016 and 1st International Workshop on Semantic Change Evolving Semantics (SuCCESS16)At: Leipzig, GermanyVolume: 1695, 09 2016.

[FBMR17] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web*, 9:1–53, 03 2017.

[Fel98] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA, 1998.

[FPCM20] Joseph Fisher, Dave Palfrey, Christos Christodoulopoulos, and Arpit Mittal. Measuring social bias in knowledge graph embeddings. Amazon @ AKBC Workshop on Bias in Automatic Knowledge Graph Construction 2020, 2020.

[GAS16] M. Galkin, S. Auer, and S. Scerri. Enterprise knowledge graphs: A backbone of linked enterprise data. pages 497–502. 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2016.

[HBC+20] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d'Amato, Gerard de Melo, Claudio Gutierrez, José Emilio Labra Gayo, Sabrina Kirrane, Sebastian Neumaier, Axel Polleres, Roberto Navigli, Axel-Cyrille Ngonga Ngomo, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. Knowledge graphs. ACM-CSUR-crc20210122 - Accepted Manuscript, 2020.

[HFJ+19] Pascal Hitzler, Miriam Fernández, Krzysztof Janowicz, Amrapali Zaveri, Alasdair J.G. Gray, Vanessa Lopez, Armin Haller, and Karl Hammar, editors. *The Semantic Web*. Springer International Publishing, 2019.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Q.*, 28(1):75–105, March 2004.

[IHvH+17] Al Koudous Idrissou, Rinke Hoekstra, Frank van Harmelen, Ali Khalili, and Peter van den Besselaar. Is my:sameas the same as your:sameas? lenticular lenses for context-specific identity. In *Proceedings of the Knowledge Capture Conference*, K-CAP 2017, New York, NY, USA, 2017. Association for Computing Machinery.

[JPJJ19] Manuela Jeyaraj, Srinath Perera, Malith Jayasinghe, and Nadheesh Jihan. Probabilistic error detection model for knowledge graph refinement. Conference: 20th International Conference on Computational Linguistics and Intelligent Text Processing (CiCLing 2019), 04 2019.

[Kle01] Michel Klein. Combining and relating ontologies: An analysis of problems and solutions. *Ontologies and Information Sharing*, 47, 05 2001.

[LA18] Matthew Lease and Omar Alonso. *Crowdsourcing and Human Computation: Introduction*, pages 499–510. Springer New York, New York, NY, 2018.

[LC18] Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1771–1776, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[MMJ+14] Jonathan Mortensen, Evan Minty, Michael Januszyk, Timothy Sweeney, Alan Rector, Natalya Noy, and Mark Musen. Using the wisdom of the crowds to find critical errors in biomedical ontologies: a study of snomed ct. *Journal of the American Medical Informatics Association : JAMIA*, 22, 10 2014.

[MV10] Thomas Markotschi and Johanna Völker. Guesswhat?! human intelligence for mining linked data. In Valentina Presutti, editor, *KIELD 2010 : Proceedings of the 1st Workshop on Knowledge Injection into and Extraction from Linked Data; Lisbon, Portugal, October 15, 2010*, volume 631, pages 28–39, Aachen, 2010. RWTH.

[NFG+20] Eirini Ntoutsi, Pavlos Fafalios, Ujwal Gadiraju, Vasileios Iosifidis, Wolfgang Nejdl, Maria-Esther Vidal, Salvatore Ruggieri, Franco Turini, Symeon Papadopoulos, Emmanouil Krasanakis, Ioannis Kompatsiaris, Katharina Kinder-Kurlanda, Claudia Wagner, Fariba Karimi, Miriam Fernandez, Harith Alani, Bettina Berendt, Tina Kruegel, Christian Heinze, Klaus Broelemann, Gjergji Kasneci, Thanassis Tiropanis, and Steffen Staab. Bias in data-driven artificial intelligence systems—an introductory survey. *WIREs Data Mining and Knowledge Discovery*, 10(3):e1356, 2020.

[OdV18] Agnieszka Onuchowska and Gert-Jan de Vreede. Disruption and deception in crowdsourcing: Towards a crowdsourcing risk framework. Conference: Hawaii International Conference on System Sciences, 01 2018.

[Pau17] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017.

[PHg14] Barbara Plank, Dirk Hovy, and Anders gaard. Linguistically debatable or just plain wrong? *52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014 - Proceedings of the Conference*, 2:507–511, 06 2014.

[PILL18] Catia Pesquita, Valentina Ivanova, Steffen Lohmann, and Patrick Lambrix. A framework to conduct and report on empirical user studies in semantic web contexts. In Catherine Faron Zucker, Chiara Ghidini, Amedeo Napoli, and Yannick Toussaint, editors, *Knowledge Engineering and Knowledge Management*, pages 567–583, Cham, 2018. Springer International Publishing.

[PM04] Robert Porzel and Rainer Malaka. A task-based approach for ontology evaluation. *ECAI Workshop on Ontology Learning and Population, Valencia, Spain*, 01 2004.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[Ris18] Petar Ristoski. Exploiting semantic web knowledge graphs in data mining. Mannheim, 2018. Published in Studies on the Semantic Web 2019.

[RRN+19] Petar Ristoski, Jessica Rosati, Tommaso Di Noia, Renato De Leone, and Heiko Paulheim. Rdf2vec: Rdf graph embeddings and their applications. *Semantic Web*, 10(4):721–752, 2019.

[SHHQ16] V. Sharmanska, D. Hernández-Lobato, J. M. Hernández-Lobato, and N. Quadrianto. Ambiguity helps: Classification with disagreements in

crowdsourced annotations. pages 2194–2202. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[SLA+13] Elena Simperl, Jens Lehmann, Maribel Acosta, Amrapali Zaveri, Dimitris Kontokostas, and Sören Auer. Crowdsourcing linked data quality assessment. In *The Semantic Web ? ISWC 2013*, pages 260–276. International Semantic Web Conference, Springer, Oktober 2013.

[SSSF16] Carl Salk, Tobias Sturn, Linda See, and Steffen Fritz. Limitations of majority agreement in crowdsourced image interpretation. *Transactions in GIS*, 21:n/a–n/a, 03 2016.

[VBSF+18a] Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors. *The Semantic Web – ISWC 2018 - Part I*. Springer International Publishing, 2018.

[VBSF+18b] Denny Vrandečić, Kalina Bontcheva, Mari Carmen Suárez-Figueroa, Valentina Presutti, Irene Celino, Marta Sabou, Lucie-Aimée Kaffee, and Elena Simperl, editors. *The Semantic Web – ISWC 2018 - Part II*. Springer International Publishing, 2018.

[WDK17] Michael White, Manjuan Duan, and David L. King. A simple method for clarifying sentences with coordination ambiguities. In *Proceedings of the 1st Workshop on Explainable Computational Intelligence (XCI 2017)*, Dundee, United Kingdom, September 2017. Association for Computational Linguistics.

[Zel18] T. Zeller. Detecting ambiguity in statutory texts. Universität Stuttgart, Abschlussarbeit (Bachelor), 2018.

[ZGEBI16] Maayan Zhitomirsky-Geffet, Eden Erez, and Judit Bar-Ilan. Toward multiviewpoint ontology construction by collaboration of non-experts and crowdsourcing: The case of the effect of diet on health. *Journal of the Association for Information Science and Technology*, 68, 04 2016.

[ZGNT18] Catherine Faron Zucker, Chiara Ghidini, Amedeo Napoli, and Yannick Toussaint, editors. *Knowledge Engineering and Knowledge Management*. Springer International Publishing, 2018.

[ZKS+13] Amrapali Zaveri, Dimitris Kontokostas, Mohamed A. Sherif, Lorenz Bühmann, Mohamed Morsey, Sören Auer, and Jens Lehmann. User-driven quality evaluation of dbpedia. In *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, page 97–104, New York, NY, USA, 2013. Association for Computing Machinery.

[ZLL+17] Yudian Zheng, Guoliang Li, Yuanbing Li, Caihua Shan, and Reynold Cheng. Truth inference in crowdsourcing: Is the problem solved? *Proc. VLDB Endow.*, 10(5):541–552, January 2017.

[ZRM+15] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann, and Sören Auer. Quality assessment for linked data: A survey. *Semantic Web*, 7:63–93, 03 2015.