Karlsruher Institut für Technologie

Institut für Technik der
Informationsverarbeitung (ITIV)

Bachelorarbeit Nr. ID-3226

von Herrn cand. el. Jan Niklas **Lettner**

Investigating Combining Quantitative And Textual Causal Knowledge In Learning Causal Structure; Untersuchung der Kombination von quantitativer und textueller Daten für die Erstellung kausaler Struktur

Beginn:	15.12.2022
Abgabe:	15.06.2023
Betreuer:	Ployplearn Ravivanpong KIT TECO
Korreferent:	Prof. Dr.-Ing. Michael Beigl Lehrstuhl für Pervasive Computing Systems (PCS)

Prof. Dr.rer.nat. Wilhelm Stork

Abstract

The study of causes and effects in large systems such as meteorology, biochemistry, finance, and sociology plays a critical role in predicting future developments and possible interventions. In the last decades, several new techniques and algorithms have been developed to discover causal structures in multivariate quantitative datasets. Yet, solely determining causal structure from observations is challenging and often yields ambiguous results. Additional knowledge from other sources is likely to be beneficial.

Recently emerging large-scale language models are showing impressive results in the field of natural language processing (NLP). One task in the field of NLP is to extract causal relations from text. Combining these with causal discovery algorithms could be advantageous.

This bachelor thesis investigates the combination of causal structures from quantitative and qualitative sources. A feasibility study was conducted on two datasets; (1) a biochemistry flow cytometry dataset and (2) a self-collected financial dataset. During this process, a common framework was developed that enables the combination of both sources. Considerations and problems were monitored and improvements suggested. A focus laid upon visualizing the evidences with different Python and R libraries.

In principle, it is possible to combine both domains. However, it was found, that a lack of training data for causal relation extraction exists. Knowledge graphs with an underlying ontology need to be leveraged to account for lexically different terms of the same entity. To improve the results from the qualitative data, it would be advantageous to extract events rather than causal relations.

This thesis makes a valuable contribution to the study of integrating quantitative and qualitative causal knowledge by applying various methods to two distinct datasets from different domains. Furthermore, it addresses a research gap, as there is limited existing literature in this specific area to the best of my knowledge.

Karlsruhe Institute of Technology

I confirm that the submitted thesis is original work and was written by me without further assistance. Appropriate credit has been given where reference has been made to the work of others. The thesis was not examined before, nor has it been published. The submitted electronic version of the thesis matches the printed version.

Karlsruhe, 15.06.2023 ______________________________

Acknowledgement

I would like to thank Prof. Stork and Prof. Beigl for reviewing my work. Furthermore, I would like to thank my supervisor Ms. MSc Ravivanpong for the very helpful discussions and suggestions.

A special thanks goes to my mother, father and brother who have always supported me during my studies.

Jan

Contents

List of Figures

List of Tables

1. Introduction

The introduction is split up into the sections Motivation 1.1, Related Work 1.2, Goal of this Thesis 1.3 and subsequently Structure of this Thesis 1.4.

1.1 Motivation

Whether it is biochemistry, the global warming, stock markets or (mis-)information flow, all can be modelled as systems of variables that interact with each other. In the field of biochemistry, it might be proteins, for global warming it is wind speed, temperature and precipitation. Stock markets can be modelled by observing stock prices and revenue, while the spread of information or misinformation can be monitored via social networks. However, for a deeper understanding, the question arises of "What causes what?" It is the question of cause and effect, that researchers in the field of Causality try to answer. The knowledge can be used to make predictions about the future or hypothetical interventions. Yet, it is difficult to infer causal knowledge. In particular, when it is only possible to observe the system, and not conducting various interventions for testing purposes. This leads to limitations of algorithms that try to infer such causal knowledge solely by observations.

The use of additional knowledge could be helpful to overcome these limitations. One source that has hardly been used by machines so far is natural language in the form of text. This is a challenging task, as language is often vague and ambiguous. Given for example the sentence: "The cat crosses the street. On the other side, it has some time to rest." What is meant with "it"? Is it the cat or the street? For us humans, the answer is obvious. However, a machine might have some problems with it. Therefore, trying to automatically extract meaningful knowledge is difficult.

The field of research is named Natural Language Processing (NLP). In the past, developers tried to find a solution with different methods, that were either rule-based or based on statistical machine-learning methods. These worked decent in the specific areas they were made for. However, the development was time-consuming and domain knowledge was necessary to annotate the rules or prepare the features.

In recent years, transformer-based language models were introduced. ChatGPT is probably the most prominent one. These models are pretrained on extensive amounts of unlabeled data to get an understanding of language. Afterwards, they can be fine-tuned on specific tasks. The extraction of causal relations from text is one application, they can be used for.

Combining these extracted causal relations with the causal structures derived from the quantitative algorithms could provide new meaningful insights. This will be investigated in this bachelor thesis. Related work with a similar notion will be presented in the next section.

1.2 Related Work

World Modelers [29] is a program developed by the Defense Advanced Research Projects Agency (DARPA). Its goal is to unify qualitative causal analysis with quantitative models to provide a better understanding of current crises to analysts. They mention as an example and use case the necessity to better understand the influences and dynamics that affect food insecurity crises. To accomplish that, they built a pipeline consisting of several subsystems that will be explained in the following.

The main part of the system is called CauseMos. It offers a human-machine interface with several workflows to make use of qualitative and quantitative data. The extraction of causal relations, events and statements from text is done by several subsystems. They extract found information in a rule or pattern based way, ground it to an ontology and assemble it together. Another subsystem leverages quantitative data. Analysts can register their own model and have the possibility of adding parameters and metadata to it.

CauseMos then merges all the information and creates a semi-quantitative computational model, which can be used to analyze scenarios and interventions. This is done in the last group of subsystems; the inference engines. They can model interactions, study the dynamics of the system and estimate uncertainties or trends.

The system provides a thorough approach that includes the acquisition, assembly, analysis, and inference of knowledge.

Weaknesses are, that the offered ontology is only useful for food insecurity crises. That makes it very specific. Due to the rule and pattern based readers, the portability is limited and there is a risk, that causal relations of very domain specific topics are overlooked. [29]

The work of Kang et al. [34] focused on explaining the cause of an event in time series data from text. The target time series data used in their paper was stock and polling data. They utilized the day-to-day popularity and sentiments of N-grams of tweets, blogs and news articles to calculate time series data. The Granger causality score between time series data generated from N-grams and the target time series was calculated to determine the best k textual features.

To build a chain of causation to explain the event further, they created a knowledge base graph, called CGraph. The causal graph was augmented with the help of the external knowledge base Freebase. To find a useful path between the target and source node, an algorithm was used that searches backwards from the effect to possible causes.

Additionally, to help with the lexical strictness of language, they trained a neural network, which they called neural reasoner. It consists of two LSTM layers and a multi-layer perceptron in between. The neural reasoner was trained by traversing CGraph, after it was initialized by word embeddings from word2vec.

The model was tested with several tasks. One of it was to forecast stock and polling data. It is evident, that the root-mean-squared error (RMSE) is lower, when textual features are considered in addition to the target's past time series.

It is a comprehensive work, which they accomplished. That is, because of the knowledge base they built, the neural network they trained and the attempt to even explain the events with a causal path. However, it would have been interesting to see a different algorithm next to Granger Causality. Additionally, they did not incorporate any other information contained in tweets, as for example the time they were published [34].

[8] tried to unify information of biochemical reactions in cells. This information comprises causal paths of biochemical reactions. The classic approach to retrieve knowledge about these processes is to design low-throughput controlled experiments. This pathway knowledge is then collected in interaction databases. However, there is a newer data-driven approach that directly infers graphical models from measurements. Both approaches have advantages and disadvantages.

They created an application "CausalPath" that maps the data-driven information to curated human pathways. It can detect potential causal links with a graphical pattern search. Links to literature is provided as well.

To leverage the knowledge from the databases, they manually curated 12 graphical patterns and extracted with these 28517 prior relations. A "causal conjecture" was defined to pair prior knowledge with the data-driven measurements in a logic-based way. Additionally, statistical measurements were implemented to increase the interpretability of the results. They provided an example:

1. Through the curated patterns, they detected from the prior knowledge that GAB1 can help to cause MAPK3.

2. Both were correlated on the data-driven side. CausalPath chose then the result as a possible explanation and provided a subgraph with the dependencies.

They summarized, that the data-driven side provides context-specific correlation, while the literature provides the causality. These will then be combined with the causal conjecture and the two statistical measurements.

[8] tested this method on three studies.

CausalPath leverages a lot of knowledge that is stored in these databases. Therefore, it is useful for building mechanistic models, grounded on already existing knowledge.

A downside is, that CausalPath cannot infer new causal connections. Because of different contexts, cell types and disease models, they stated that it is very challenging, which of the described relations are applicable.

1.3 Goal of this Thesis

The goal of this thesis is to investigate the combination of quantitative and qualitative causal knowledge. This is understood as feasibility study, where different approaches and methods will be tried.

Overall, it will be focused on developing a common framework for this task that can be applied to different domains. Occurring problems and considerations will be monitored during this process. Notably, there will be an emphasis on the visualization of causal evidence and structures. Additionally, a comparative analysis between qualitative and quantitative results will be conducted to further enhance the investigation.

The structure of this thesis will be explained in the succeeding section.

1.4 Structure of this Thesis

This feasibility study will be conducted on two datasets. Both of them contain qualitative and quantitative data. The first one is a flow cytometry biochemistry dataset, while the second one is a self-collected financial dataset.

However, first, an introduction into the fields of Causality and Relation Extraction is provided in chapter 2.

Afterwards, both datasets the investigations conducted will be presented in chapter 3 and chapter 4. Each dataset has its own methodology, implementation, result and discussion section. Each of these sections is split up into a part concerning the quantitative data, the qualitative data and the combination of both. This thesis ends with chapter 5 Conclusion & Outlook.

2. Background

In the following chapter necessary background knowledge is explained. The background is separated into three sections. Basics of Causality 2.1, where basic assumptions about causality and causal discovery algorithms will be elaborated. Consecutively, in section 2.2 utilized time-independent and time-dependent algorithms are presented. Lastly, in section 2.3 used relation extraction and deep learning models are explained. This includes word embeddings, the concept of attention, transformers and recent language models, like BERT variations and GPT models.

2.1 Basics of Causality

In this section, an introduction into the field of **Causality** is provided.

Causality can be described as the impact that an event has upon another event. When both are causal related, the influencing event is called *Cause*, whereas the influenced event is named *Effect* [39]. For example, given two events A and B, such that A causes B. In order for B to occur, it is necessary that A took place. However, it is not necessary that B arose to let A occur. This makes it inherently different to variables that are correlated to each other.

The field of causality can be split into two main tasks. The first is **Causal Discovery**. It tries to infer causal structure from observational data. The second is **Causal Inference**, which tries to quantify the impact of variables onto each other [39]. This thesis focuses on causal discovery.

Notations

In the following, several definitions are made that are necessary to better understand needed assumptions and subsequently the causal discovery algorithms used in this thesis.

The structure of causal relations can be visualized with a graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$. $\mathbf{V}$ represents a set of nodes and vertices, while $\mathbf{E}$ is a set of edges. Each edge can either be directed, undirected or bi-directed. For example $A \rightarrow B$, $A - B$ or $A \leftrightarrow B$.

A and B are said to be *adjacent* [39]. The terms directed edge and arc are used interchangeably in this thesis

Given two nodes $U, V \in \mathbf{V}$, U is called parent of V, if $U \to V$. V is the child of U. The set of parent and child nodes of a node U is denoted as $Pa(U)$ and $Ch(U)$ [39].

A (directed) path between two nodes in $\mathcal{G}$ is a distinct, non-repeating sequence of $V_1, \dots V_n$ with pairwise consecutive (directed) edges between them. V_1 is called an ancestor of V_n, while V_n is a descendant of V_1. The set of ancestors and descendants of a node V is denoted as $An(V)$ and $De(V)$, whereas $V \in An(V)$ and $V \in De(V)$.

A *collider* is a variable, in which adjacent edges point into. For example $A \to B \leftarrow C$, where B is a collider. The difference is a *non-collider*, that can either be a mediator $(\dots \to B \to \dots)$ or a common cause $(\dots \to B \leftarrow \dots)$. However, it depends on the path, what role a node takes [18].

A direct graph is *acyclic*, if there is no directed path from V to U and U to V. It is then named *Direct Acyclic Graph (DAG)* [39].

The connectivity of a graph can be expressed in terms of *d-separation* and *d-connection* [18]. A path p d-connects U and V given a conditioning set $\mathbf{C} \subseteq \mathbf{V} \setminus \{A, B\}$ if and only if all colliders on p are in $\mathbf{C}$ or have a descendant in $\mathbf{C}$ and no non-colliders of p are in $\mathbf{C}$. An explanation to this is, that when conditioning onto a collider, an association is created. This bias is called *Berksonś fallacy* [55]. U and V are d-separated, if there are no d-connecting paths between them, which can be denoted as $U \perp V$.

A set of random variables $\mathbf{X} = (X_1, \dots, X_p)$ and a DAG form a **Bayesian Network**, if

$$P(\mathbf{X}) = \prod_{X \in \mathbf{X}} (P(X|Pa(X))) \tag{2.1}$$

$P(X)$ represents the joint distribution. This network depicts the probabilistic relations between the variables $\mathbf{X}$, where vertices represent the variables, while edges show the conditional dependency between them.

However, there is no *a priori* reason for $P(\mathbf{V})$ to constrain the possible graphs [18]. Though, there are the following two principles that are used to connect probabilistic relations with causal relations. Together, they ensure that d-separation is equivalent to (conditional) probabilistic independence.

(1) The *Causal Markov Condition (CMS)* declares that every vertex $X \in \mathbf{X}$ is "[...] probabilistically independent of its non-descendents given its parents [...]" [18, p. 83]:

$$X \perp\!\!\!\perp NonDe(X)|Pa(X) \tag{2.2}$$

Unmeasured common confounders or a selection bias are examples that would violate this assumption.

(2) The *Causal Faithfulness Condition (CFS)* states that, if a variable U is independent of V conditioned on a set $\mathbf{C}$, that $U \perp V$ given $\mathbf{C}$. This is rather a simplification measure and can be easier violated, as it just have to exist causal connections that do not exhibit a dependence [18].

Another assumption often applied is *Causal Sufficiency*, which states that there are no unmeasured common causes that effect vertices $\mathbf{X}$.

With this knowledge, it is now possible to explain the different causal discovery algorithms that were used in this thesis.

2.2 Causal Discovery Algorithms

Causal discovery algorithms aim to learn causal structure. These can be classified in *constraint-based* and *score-based* algorithms [39]. Former use independence tests to find a set of edge constraints before distinguishing the direction of the edges by applying some rules. Latter utilize a relevance score to value different graphs and optimizing them.

Another way of classifying causal discovery algorithms is by the data, they are applied to. Cross-sectional data is the first type. This is data that is collected through simultaneous observations. Thus, there is no time, that can be used as an additional constraint, such that A has to be occurred before B. The algorithms used on this type are time-independent. The second type of data is time-series data, which does not have the disadvantage of the first type. For example, if there is an undirected edge between A and B, it is safe to state that past $\rightarrow$ future [39].

2.2.1 Time-Independent Algorithms

In the following section are three time-independent algorithms described, that were used in this thesis. PC and FCI are both constrained-based algorithms, while Tabu-Search is score based.

2.2.1.1 PC Algorithm

The PC (named after the developers Peter Spirtes and Clark Glymour [33]) algorithms relies on the faithfulness assumption. It works in two steps. In the first step searches the algorithm for the skeleton of the graph. The skeleton of a graph comprises all edges without a direction. In the second step, these directions are oriented.

To find the skeleton, the algorithm starts with a fully connected undirected graph. For each pair of adjacent nodes U and V, the algorithm tests the independence conditioned on an adjacent subset $\mathbf{C} \subseteq \mathbf{X}$, such that $U \perp\!\!\!\perp V|\mathbf{C}$. If a subset exists that causes independence, the connection between U and V is removed [39].

To orientate the edges, three rules are applied:

1. Given that there are three nodes U, V, W ordered such that $U - V - W$. Since U and W are not adjacent, the following can be stated: If $U \perp\!\!\!\perp W|\mathbf{C}$ and $V \notin \mathbf{C}$, a collider is constructed $U \rightarrow V \leftarrow W$. This structure is called a *v-structure*. The complete assumption works, because of Berkson's fallacy. If $V \in \mathbf{C}$ and it

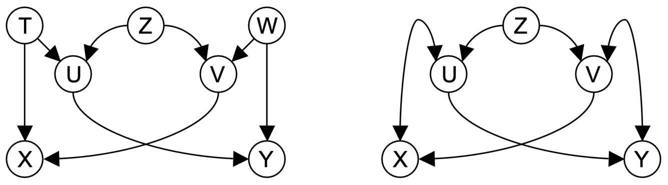


Figure 2.1: Two graphs, left graph meets the causal sufficiency assumption, whereas the right not [14]. Both variables U and V are colliders ($X \rightarrow U \leftarrow Z$ and $Y \rightarrow V \leftarrow Z$), if conditioned solely on them, this would infer associations $X - Z$ and $Y - Z$, resulting in a new dependency between X and Y.

is conditioned upon $\mathbf{C}$, given that the triple is a collider, this would result in a dependency between U and W. In this case, all three other variants are possible, that is, $U \rightarrow V \rightarrow W$, $U \leftarrow V \leftarrow W$ and $U \leftarrow V \rightarrow W$. This is, because all three belong to the same Markov equivalence class [18].

2. If there is a directed edge between U and V, while $V - W$ holds but not $U - W$, then $V - W$ is oriented into $V \rightarrow W$. This is, because a directed edge from $V \leftarrow W$ would deduce a new v-structure, although they should have been already found in the first step [33].

3. If there is a direct path between U and V and an undirected edge between them, it is oriented as $U \rightarrow V$. This is to preserve the acyclic character of the graph.

This algorithm works well as a benchmark. However, it is limited, since it assumes causal sufficiency. That means that it does not allow latent variables [39].

When applied to the left graph in figure 2.1 to test whether X and Y are independent, only the adjancies of both variables have to be tested, resulting in $X \perp\!\!\!\perp Y|V, T$. If causal sufficiency can not be assumed, the node Z has to be tested as well, which results in $X \perp\!\!\!\perp Y|U, V, Z$. However, Z is not in a subset of adjancies of X and Y. PC will miss that [14].

The following algorithm is going to solve this problem.

2.2.1.2 FCI+ Algorithm

The FCI (Fact Causal Inference) algorithm is a generalization of the PC algorithm, as it allows arbitrary many latent variables [33]. Therefore, the assumption of causal sufficiency is not necessary. Yet, faithfulness is still required. It belongs to constraint-based algorithms.

It has the same two steps, as the PC algorithm has. The first step is equal to the PC algorithm, that is, a skeleton is inferred, and v-structures are found with the colliders. However, the second step is different. Given $U, V \in \mathbf{X}$, the method tests **possible** d-separations such that $U \perp\!\!\!\perp V|\mathbf{Z}$. If there is a $W \in \mathbf{Z}$ that d-separates U and V, the edge is removed.

The FCI algorithm introduces new edge representations, to account for the unmeasured confounders [39, p. 8]. For two variables $X, Y \in \mathbf{X}$

- $X \rightarrow Y$ represents, that X causes Y;

- $X \leftrightarrow Y$ represents that there are unmeasured confounders from both variables;
- $X \circ \!\rightarrow Y$ represents that either X causes Y or that there are unmeasured confounders from both variables;
- $X \circ\!-\!\circ Y$ can represent, that either X causes Y, Y causes X, unmeasured confounders between both variables or X causes Y and unmeasured confounders and vice versa.

The FCI+ variations is an implementation with a different method to retrieve the set of variables that d-separates both variables U and V. This results in a (possibly) improved running time [14, 33].

2.2.1.3 Tabu Search Algorithm

The *Tabu-Search* algorithm belongs to score-based approaches. They can be separated into two parts: (1) A scoring metric that evaluates potential Bayesian networks and (2) a search procedure to carefully investigate the space of possible networks [10].

The goal is to optimize the following likelihood function

$$\mathcal{LL}(\mathcal{G}; D) = \prod_{d \in D} P(d|\mathcal{G}) \tag{2.3}$$

where $\mathcal{G}$ represents a graph, while D is the underlying dataset. To prevent overfitting, too complex graphs are penalized at the same time with a regularization term.

Tabu Search is an extension of the Hill Climbing algorithm. Hill Climbing tries to find the best solution of a search space S, by randomly generating one possible solution and searching in a defined neighborhood for an improved solution. This does not guarantee an optimal solution, since the algorithm can get stuck at a local maximum [10].

Tabu Search tries to improve that problem with an adaptive memory, that saves recently evaluated solutions. This should prevent cycling and potentially leads the algorithms to solutions that are not as "close" to the current one. The downside is, that it can cause the search process to stagnate and might prohibit attractive moves [10].

2.2.2 Time-Dependent Algorithms

In this section, utilized causal discovery algorithms for time-series data are introduced. The first one is the Granger causality [23], while the second one is PC - Momentary Conditional Independence (PCMCI) [48].

Though, first a few additional notations:

A time-series graph is defined as $\mathcal{G} = (\mathbf{V} \times \mathbb{Z}, \mathbf{E})$ for a multivariate process $\mathbf{X}$. Each node in the graph is viewed at each time $t \in \mathbb{Z}$. This graph is then defined to a lag of τ_{max}.

A connection between variables looks like this

$$X^i_{t-\tau} \rightarrow X^j_t \tag{2.4}$$

with $X^i_{t-\tau}, X^j_t \in \mathcal{G}$.

2.2.2.1 Granger Causality

The overall idea of Granger causality is, that X granger causes Y, if predicting Y based on bast values of X and Y performs better, as solely by using Y [39]. The logic behind this is, that something unique in the past of X exists, that the past of Y does not contain [48].

This results in the following auto-regression:

$$Y_t = \sum_{j=1}^{\tau_{max}} a_\tau Y_{t-\tau} + \sum_{\tau=1}^{\tau_{max}} b_\tau X_{t-\tau} + \varepsilon_t \tag{2.5}$$

m represents the maximum number of lags (e.g. days) to be used, while a_τ and b_τ are the contributions to the prediction and ε represents a white Gaussian noise. b_τ should be statistically significant for any lag between $\tau = 1 \ldots m$, to let X granger causes Y.

Disadvantages are that the algorithm assumes that the system is linear stationary and when pairwise applied, it might yield ambiguous results in terms of direct and mediated causal relations [6].

2.2.2.2 PCMCI Algorithm

The PCMCI is based upon the PC algorithm designed to deal with nonlinear and linear dependencies, but not with latent variables [39]. It is based upon the PC algorithm for time-independent data.

In a first phase, the algorithm searches for a skeleton in an iterative way. It conditions on all subsets S of $\mathcal{P}(X^j_t)$ of X^j_t and tests

$$X^i_{t-\tau} \perp\!\!\!\perp X^j_t | S \tag{2.6}$$

If both variables are not independent, the variable $X_{t-\tau}$ is removed from the set [48]. In the PCMCI algorithm, this was used for the first step with a marginal adjustment, that is that only the condition subset S with the largest association is tested.

In a second phase, the MCI test is applied. It conditions additionally on the parents of $X^i_{t-\tau}$ such that

$$X^i_{t-\tau} \perp\!\!\!\perp X^j_t | \mathcal{P}(X^j_t \backslash, \mathcal{P}(X^i_{t-\tau})) \tag{2.7}$$

This method further determines causal relationships between different timestamps. However, this accounts to autocorrelation and a well-controlled false positive rate [48, 39].

2.3 Causal Relation Extraction

An import role in the domain of natural language processing (NLP) plays **information extraction (IE)**. Its goal is to retrieve structured information from unstructured text in form of semantically defined entities and relationships [24].

An entity is the most basic unit of information [42]. It often refers to a real-world object. Usually, entities are tagged or labeled, to categorize them into groups. For example, Siemens (*ORGANIZATION*), Olaf Scholz (*PERSON*), the Brandenburg Gate (*LOCATION*) or France (*GEO-POLITICAL-ENTITY*). The whole process of extracting these entities is named **Named Entity Recognition (NER)**.

These entities often occur in relationships. For example in the sentence:
"[Olaf Scholz]$_{\text{PERSON}}$ is chancellor of [Germany]$_{\text{GPE}}$." The relation in this case would be *HEAD_OF_GOVERNMENT*. This information is often stored as tripe (h, r, t), whereas h and t depict the entities, while r is the respective relation [7]. This would result in (Olaf Scholz, *HEAD_OF_GOVERNMENT*, Germany). The process of retrieving the entities including the respective relations is called **Relation Extraction (RE)**.

These triples can be saved in a **Knowledge Base (KB)** and later be used to create a knowledge graph. An example is illustrated in 2.2. The edges show relations, while entities are presented as nodes.

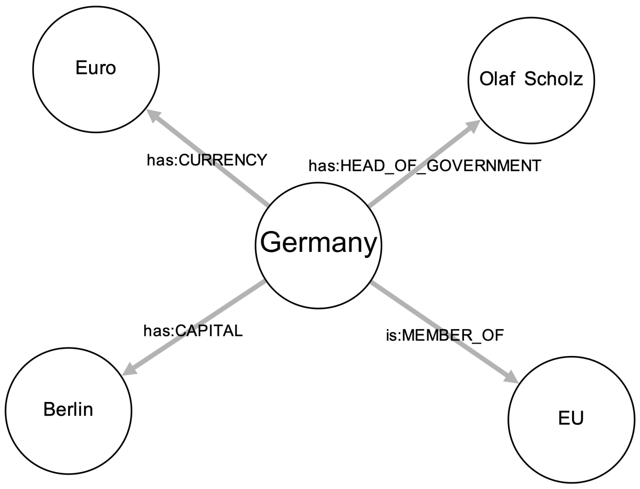


Figure 2.2: Depicted is a simple knowledge graph.

Though, before creating such KBs, it can be useful, to construct an underlying **ontology** [43]. There are various definitions to what an ontology is [26]. [25, p. 1] defines an ontology as "[...] explicit specification of a conceptualization", whereas a conceptualization is an "[...] abstract, simplified view of the world [...]". This definition will be used in the following, since the paper explicitly mentions as application the sharing of knowledge among AI systems.

A possible ontology for the example in figure 2.2 is presented in figure 2.3. Though, there are different possible ontologies with different grades of refinement. For instance, the entity ”Germany” could be classified as ”Geo-Political Entity”, as ”Country” or as ”Federal Republic”. With the help of an ontology, entities can be grounded and structure can be inferred. For example, since Germany is a country and therefore has a capital, it is very likely that France has a capital as well. Another advantage of a KB is, that it can be queried, by following the edges [43]. An example of an existing KB is DBpedia [37].

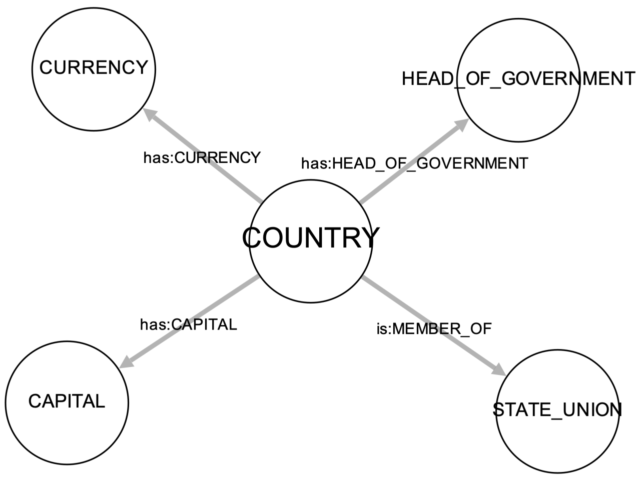


Figure 2.3: A possible ontology.

A more specific form of RE is **Causal Relation Extraction**. In this case, the focus lays upon finding cause-effect relations including underlying entities e_1 and e_2, whereas the occurrence of e_1 leads to the occurrence of e_2. Another difference is, that causal relation extraction is usually a binary classification task. However, general RE is a multi-classification problem [65]. The term RE will be used in the following to refer to causal relation extraction.

The resulting triples (e_1, r, e_2) can be saved in a **Causal Knowledge Base**, from which a causal knowledge graph can be constructed. The difference to a general knowledge graph is, that edges depict causal dependencies between entities, e.g. $e_1 \xrightarrow{causes} e_2$ [28]. In the following, when referred to a KB, a causal knowledge base or a causal knowledge graph is meant. Both terms, knowledge base and knowledge graph, are often used interchangeably [43].

Causality in text occurs in varies forms: (1) Explicit and (2) implicit causality, as well as (A) intra-sentential and (B) inter-sentential causality [65].

(1) Explicit causality is depicted by explicit causal connectives, e.g. *hence, because of, as, since*, but also causal verbs, e.g. *break, kill, influence*, resultive phrases and *if... else...* constructions [65].

(2) Implicit causality is different, as it either has an ambiguous causal connective, like *after, before, later* or none, e.g. *Eating a balanced diet contributes to overall good health.* Usually, background knowledge is necessary to validate whether a causal statement exist [65].

(A) In inter-sentential causality, the cause-effect pair occurs in one sentence, whereas in the case of (B) intra-sentential, the causality continues over several sentences [65].

It exists different approaches to RE. These can be categorized into (1) knowledge-based approaches, (2) statistical machine-learning (ML) based approaches and (3) deep-learning (DL) based approaches (see fig. 2.4)[65].

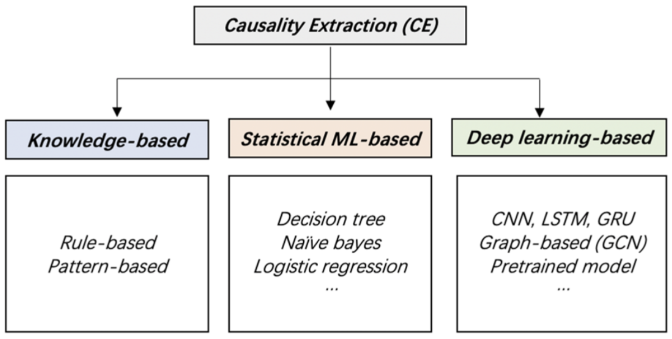


Figure 2.4: An overview of different RE approaches [65].

(1) Knowledge-based systems are dividable into pattern-based and rule-based systems. Former are systems that define a graphical pattern or orientate around keywords. Sentence structure analysis can be undertaken as well, to extract causality. Latter applies some set of algorithms on the structure of the sentence. These kinds of systems are viewed as straightforward and relatively simple. However, they need a lot of preparing by experts and when the data is versatile, perform poorly [65].

(2) Statistical ML-based approaches do not need as much preparation. Usually, a set of features is generated from the textual data, before applying an ML algorithm onto it, e.g. Support Vector Machines, Conditional Random Fields or logistic regression. They normally perform better than knowledge-based systems. Yet, the portability is limited, as well-prepared features are necessary [65].

(3) Deep learning-based systems overcome the need to generate features, as DL models map those directly into low-dimensional input vectors. For example Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory Neural Networks (LSTM) or Gated Recurrent Units (GRU) can be named. Lately, transformer-based pretrained models like Bidirectional Encoder Representations from Transformers (BERT) [16] appeared. DL approaches have the advantage, that less focus on feature preparation is necessary and the portability to other tasks or domains is easier done. On the downside, large corpora are needed for training and extensive computational resources are required [65].

In the following, a focus on transformer based systems will be taken, as they play a crucial role in this thesis.

2.3.1 Transformer

Transformers are DL models, that were first proposed by [61] in 2017. By the time then, the state-of-the-art models for sequence transduction were based on LSTMs or GRUs, which process inputs in a sequential order. This makes parallelization impossible. Transformers solve this issue, by simultaneously processing the input. This allows for parallel training and better understanding of contextual knowledge. Figure 2.5 shows the overall architecture of a transformer. However, before explaining in-depth the functionality of transformers, it is necessary to better understand the concepts of **Tokenization**, **Word Embeddings** and **Attention**.

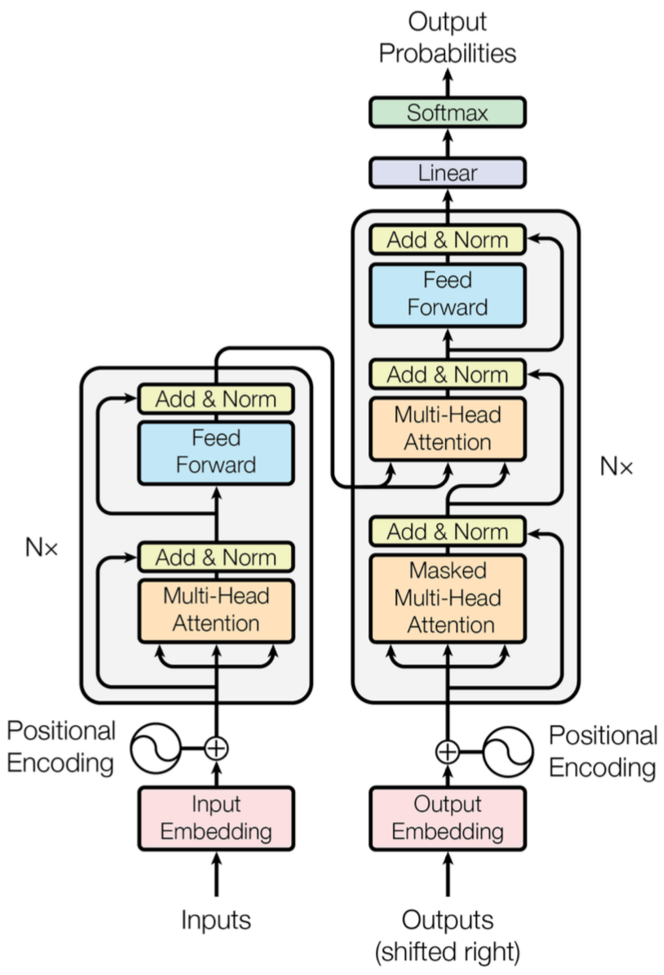


Figure 2.5: Transformer Architecture [61].

Tokenization

A first preparation step comprises the tokenization process. It splits up a sequence of characters into meaningful units, named **tokens**. For example the word *aren't* can be tokenized into *aren't*, *aren|t*, *arent* or *are|n't*. It depends on the model, how tokens are generated. Word2Vector uses tokens on word level, whereas BERT uses WordPiece [64] and GPT models Byte Pair Encoding (BPE) [60, 59]. However, in the following, the terms words and tokens are used interchangeably, as for example in [5].

Word Embeddings

In order to let AI models utilize text, it is necessary to convert words into a machine-readable format. One way of doing that, is by converting words into word embeddings. These are fixed-length, dense and distributed vector representations of words

[5]. This enables mathematical operations, such as addition, subtraction, distance measures or similarity calculations. Additionally, these vectors can be used as input to AI models. For example Word2Vec [38] or GloVe [44] were trained to generate word embeddings. Consecutively, resulting vectors can be used to calculate new relationships, such as a city and the country it belongs to, for example France is to Paris as Germany is to Berlin [38] or $king - queen = man - woman$ [44].

However, each of these models only allows one vector per word. This causes issues, when lexically similar words are used in a semantically different way, such as in the sentences: ”The man was accused of robbing a bank” and “The man went fishing by the bank of the river”. This lead to models, that take the contextual information into consideration, such as ELMo (Embeddings from Language Models) [45]. Each word is assigned a vector, that is the result of the entire input sentence.

Subsequent models are BERT and GPT, upon a special focus lays, as those models were used in this thesis.

Attention

Since transformers rely solely on attention mechanisms, it is important to understand this concept first, before subsequently explain transformers [61].

It was first proposed by [9] in 2016. The task was to build a machine translation model out of an encoder-decoder architecture. An encoder takes an input sentence and converts it into a feature vector. This feature vector is then utilized by a decoder to generate a new sentence. This comes with an issue, as the encoder has to compress all necessary information into that feature vector. [9] introduce an extension to the model, by implementing an attention functionality into the model. This is done in the following way:

(1) The encoder, which is a bidirectional RNN (biRNN) generates a hidden state h_j for each word x in the input sentence $(x_1, \ldots, x_{Tx})$. These words are usually combined into a vector c such that

$$\mathbf{c} = q([h_1, ..., h_{Tx}]) \tag{2.8}$$

whereas q is a non-linear function.

(2) Normally, the decoder would take this hidden state to predict the next word y_t given the context vector **c** and all previously predicted words $(y_1, \ldots, y_{t-1})$. However, with the attention mechanism it exists one $\mathbf{c}_i$ context vector for every target word y_i. The context vector is computed as

$$\mathbf{c}_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.9}$$

α_{ij} is the indirect result of a feedforward neural network called *alignment model*, which is jointly trained with the other components of the system [9]. α_{ji} depicts the importance of the hidden state h_j to the previous hidden states s_{i-1}, the next hidden state s_i and the generation of y_i. With this attention mechanism learns the decoder which parts of an input sentence are important, in generating a new word correctly.

The attention mechanism of transformers is build upon the one here explained. However, more background knowledge is necessary. Therefore, in the following transformers will be further explained, before talking about attention again.

Transformer

Transformers consist of several encoder and decoder layers. Each of these layers contain a *self-attention* mechanism (fig. 2.5). This is marginally different to the former explained attention concept, because it relates in this case to different positions of the same input sequence [61] that is currently under procession.

Each layer in the encoder has two sub-layers, whereas the first is a multi-head self-attention layer, while the second is a fully connected feedforward neural network. They produce an output dimension of $d_{\text{model}} = 512$.

The decoder contains additionally a third sublayer, which is used to perform a multi-head attention over the output of the encoder. The first sublayer in the decoder is masked to ensure, that it cannot self-attend to subsequent positions.

Transformers use a generalized attention method. It consists of a query and a set of key-value pairs. There are different methods to calculate these values together. In case of the original transformer, it was undertaken with the *Scaled Dot-Product Attention* [61].

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{QK}^T}{\sqrt{d_k}})\mathbf{V} \tag{2.10}$$

d_k is the dimension of queries and keys, while d_v depicts the dimension of values. Though, both dimensions are often equal. The *softmax* function maps the result of the dot product $z_0 \dots z_{d_{\text{model}-1}}$ divided by the constant to values between 0 and 1, such that $\sum z_i = 1$. Consecutively, these values will multiplied with the value vector $\mathbf{V}$ The *ith* position of the resulting vector represents the attention that the model pays to the *ith* input position.

[61] introduced the concept of multi-head attention. In this case, queries, values and keys receive an additional dimension h (fig. 2.6). This means, that each input word has h query and key-value pairs. These values are subsequently concatenated and multiplied with another weight matrix, to receive the final result. The advantage is, that the model has the possibility to attend to information from different representation subspaces. Just using one head would result in the average of these different representations. For each element, query, key and value exists a weight matrix, that is going to be trained.

This attention method is implemented in three different ways. (1) In the encoder-decoder attention layer, the queries are received from the previous decoder layer, while the encoder generates the key-value pairs. This functionality enables the decoder to attend to every input sentence, similarly to the method introduced by [9]. (2) The encoder applies self-attention. Queries and key-value pairs come from the previous layer in the encoder. (3) The decoder is similar to the encoder. However, it is ensured that the decoder only attends to recent or the current position. Greater positions are masked by setting the values to $-\infty$.

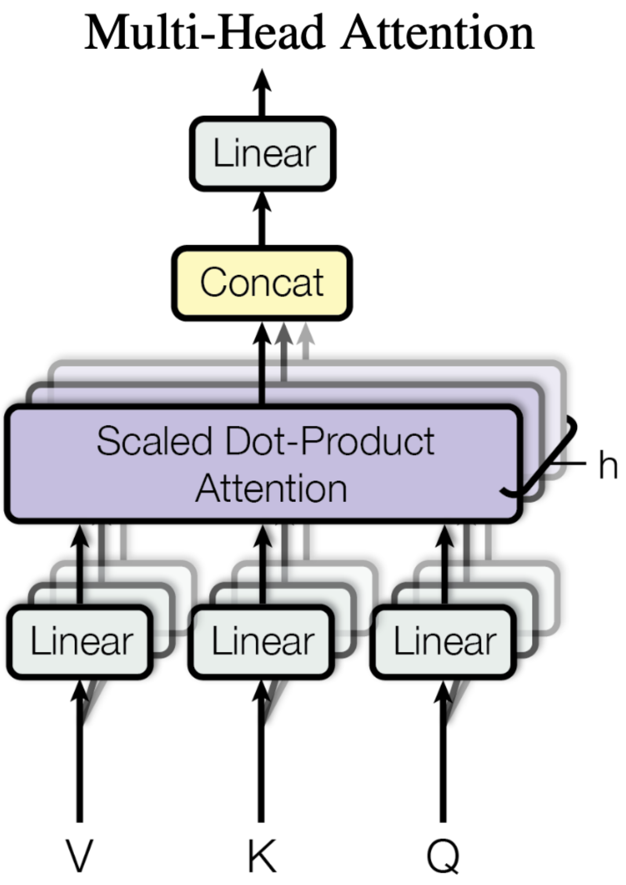


Figure 2.6: Multi-head attention [61].

[61] used as well learned word embeddings. Though, they do not specify, which method they leveraged. Additionally, they injected positional information into the model, as it lost its positional knowledge, that for example RNNs or LSTMs have by nature.

2.3.2 Language Models

Language Models are AI models that try to predict future tokens or missing tokens in text [70]. In the recent years, it was noticed that the capability of these systems increases, when more parameters were introduced. The development of transformers led to the first small-scale language models like BERT with 340 million parameters [16]. Large language models are systems that have approximately more than ten billion parameters, e.g. GPT-3 or LLaMA among others.

Subsequently, a basis to utilized language models is provided.

2.3.2.1 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a language representation model proposed by [16]. They utilized only the encoder blocks of a transformer. The model was pretrained and can be fine-tuned on down-stream tasks, such as text classification, NER or RE among others. In fact [16] constructed two models, BERT-base and BERT-large. The former model contains 12 encoder blocks, a hidden size of 768 and 12 attention heads, while the latter contains 24 encoder blocks, a hidden size of 1024 and 16 attention heads. This results in 110 million parameters for the base version, while the large version comprises 340 million parameters.

WordPiece [64] was utilized as tokenization method. It contains a vocabulary of 30000 tokens. If a word is unknown, it gets broken down into subtokens.

The model was trained on unlabeled data with two methods. (1) Masked language modeling (MLM), where 15% of the tokens were masked, and the model had to predict the missing tokens. (2) Next sentence prediction (NSP), where two sentences are presented to the model and it had to decide, whether the second sentence is a successor of the first one.

With both methods, BERT learned a contextualized representation of words [19].

The training corpus comprised the BooksCorpus (800 million words) and English Wikipedia (2500 Million words).

BERT achieved state-of-the-art results on several benchmark datasets. However, many other models were later trained that, that specialize in different tasks. In the following, three BERT derived models will be briefly introduced, that are later utilized in this thesis.

BioBERT

BioBERT [35] is a BERT model that was specifically trained on a biomedical text corpus, comprising 18 billion additional words. The training process did not differ from the original construction.

It achieves increased F1 scores on biomedical NER and RE.

SpanBERT

[32] pretrained SpanBERT. This model focuses on a better representation and prediction of text spans. The training differed from the one of BERT in three ways. (1) They utilized span masking. Instead of randomly masking single tokens, a complete span of tokens is masked with a mean of 3.8 words. (2) Span Boundary Objective (SBO), where a masked span has to be predicted by the next surrounding start and end boundary tokens. (3) They removed the NSP with two sentences, and replaced it with single sequences of up to 512 tokens.

The model was pretrained on the same corpus as BERT.

They yield mostly improved results on several benchmark datasets over the original BERT model.

RoBERTa

RoBERTa (Robustly Optimized BERT Approach) was introduced by [36]. They investigated the effects of different hyperparameters to the performance of BERT and optimized them.

They amount of training data was increased from originally 16 GB to 160 GB. Furthermore, they also changed the MLM procedure, as they created the masking dynamically during the training and included 10 different maskings per training sample. Additionally, [36] removed the NSP objective from training, since results showed that this training step does not improve the performance of the model. Last, the training batch size and the overall training steps were increased. This was all combined to create a new language model, RoBERTa, that contains 355 million parameters and achieved improved results over BERT on several benchmark datasets.

2.3.2.2 GPT

This thesis utilized GPT-3.5 text-davinci-003 by utilizing the OpenAI API. Yet, for the better understanding, a brief introduction into GPT-1 is provided, before describing GPT-3 and subsequently introducing GPT-3.5.

GPT (Generative Pretrained Transformer) was first introduced by [47] in 2018. The first model was named GPT-1, which is based on transformers. However, they utilized the decoder part of it. 12 layers were stacked upon each other with a hidden state size of 768 and 12 attention heads. 512 tokens were chosen as maximal input length. These tokens were tokenized with a bytepair encoding vocabulary with a size of 40000.

During the pretraining, the model tries to optimize the following likelihood function

$$L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \ldots, u_{i-1}; \Theta) \tag{2.11}$$

where $\mathcal{U} = (u_1, \ldots, u_n)$ is a corpus of tokens, Θ represents the parameters of a neural network and k the size of the context window. This means, that these neural network parameters were searched, such that the next token u_i is predicted correctly, depending on the previously k tokens. In contrast to BERT, only the left context is considered. This model is of autoregressive nature, what makes it especially good at generating text.

Subsequently, the model can be fine-tuned by maximizing the function

$$L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y | x^1, \ldots, x^m) \tag{2.12}$$

$\mathcal{C}$ represents a labeled dataset with m input tokens and a label y.

It was trained on the BooksCorpus dataset, which amounts to approximately 4.5 GB of data, while comprising 117 million parameters.

GPT-3

GPT-3 [12] is the second successor of GPT-1. It was trained on 570 GB of text and contains up to 175 trillion parameters. The architecture is just slightly different to the one of GPT-1. Differences are, that the initialization process is modified, pre-normalization is included and reversible tokenization was implemented. The model shows significant improvements in *few-shot settings*. These are tasks, where the model is not fine-tuned, but is only provided a few examples up until zero examples in case of *zero-shot settings*. In some tasks, it reaches almost the state-of-the-art results of fine-tuned models.

GPT-3.5 text-davinci-003

This model is a successor of InstructGPT [3], which is based upon GPT-3. InstructGPT is trained to follow an instruction in a prompt and provide a suitable response. This behavior was implemented by leveraging a training method called *Reinforcement Learning from Human Feedback (RLHF)*. Human labelers rank responses to provide feedback to the model. Regrettably, OpenAI did not publish a paper about text-davinci-003, but state that is an improved version over InstructGPT [40].

2.3.2.3 XLNet

The developers of XLNet [67] tried to mitigate disadvantages of models like BERT that use bidirectional encoders and autoregressive models like GPT. They reasoned, that the MLM training method in BERT results in a discrepancy between pretraining and fine-tuning, as masked tokens do not appear in real applications. Additionally, the model assumes that these tokens are independent, given the unmasked tokens. Having said, that autoregressive models have the disadvantage that they are only trained unidirectional. However, many bidirectional context information is often required in down-stream tasks.

XLNet tries to solve these issues, in particular through a new pretraining method called *Permutation Language Modeling*. For a sequence of length T, there are $T!$ possible permutations. In the usual autoregressive models, the next word x_t would depend upon the previously $t-1$ tokens. The likelihood function is then factorized in a given order. This order is, in the case of XLNet permutated. It is not the sequence order that gets ”shuffled”. By incorporating this, the model learns the context better, but is still of autoregressive nature.

Their large model has the same amount of parameters as BERT-large with about 340 million. The model was trained on 130 GB of training data. They achieve state-of-the-art results on several datasets, with better scores than RoBERTa. XLNet yields also better results than BERT, when trained on the same corpus that BERT was trained on.

3. Example 1: Flow Cytometry Dataset

In short: The dataset presented in the following chapter deals with molecules. It is useful to investigate the combination of causal structure with it, because a true-graph exists. The overall idea is, to leverage a language model, that is able to extract causal relations and later combine these with the quantitative received causal structure.

In the following, a brief introduction to the background of this first dataset is given and why it was chosen to be further investigated in the context of combining quantitative and qualitative causal structures. It is avoided to provide a too deep biological integration in the overall context, as there is a lack of in-domain knowledge. Additionally, the data itself and what it represents will be described. This all is located in section 3.1 Dataset.

Annotation was necessary, because the dataset contains only quantitative data. Stated sources of Sachs et al. [51] are examined to gather qualitative causal data. This process is explained in section 3.2 Annotation.

The various approaches undertaken are then presented in 3.4 Implementation before outlining the results and discussion in each respective section 3.5 and 3.6.

3.1 Dataset

The dataset was collected by Sachs et al. [51] who investigated signaling pathways on cellular level. These pathways work by modified molecules, which trigger other molecules in a cascading way. Subsequently, a reaction occurs in cells.

By then, knowledge of these pathways were an aggregation of the results of individual studies, which mainly considered and investigated only single signals but not complete paths [51]. However, this does not thoroughly account for the complexity of those processes, especially in terms of inter-pathway cross-talk. The systems have to be looked at as multivariate networks. To accomplish that, flow cytometry was measured to quantify multiple molecules in a cell. Causal dependencies were determined by interventions to inhibit or stimulate the modification of molecules in these signaling pathways. Sachs et al. [51] utilized score-based Bayesian networks for modelling the resulting networks, as they are able to account for the probabilistic and nonlinear dependencies. The nodes represent molecules, while the edges indicate causality.

The advantage of this dataset is, that a true-graph exists. After application of the Bayesian network, they compared the resulting edges with those described in literature. Thus, the true-graph can be used as a guide, while research is conducted how and under which circumstances the comparison with literature can be automized and whether it is possible to leverage the textual causal knowledge to enhance the true-graph.

The dataset consists of two sub-datasets. One, that contains general observations (853 samples), while the other one contains interventional data (5400 samples). 11 phosphorylated proteins (Raf, Mek, PLCγ, Erk, Akt, PKA, PKC, P38, Jnk) and phospholipids (PIP2, PIP3) where measured.

A brief excerpt of the dataset is seen in table 3.1 and 3.2. In the former, each sample represents the quantitative amount of the individual molecule simultaneously measured in cells by flow cytometry, while in the latter, this is discretized. Here, the additional column INT (interventional) shows, which of the molecules was either inhibited or stimulated.

This data was analyzed with the Bayesian network structure inference algorithm resulting in the graph visible in figure 3.1. Dashed edges were not found by the algorithm. According to Sachs et al. [51] this is due to the constraint, that Bayesian networks have to be acyclic. The remaining edges are reported at least once in different papers, with one exception between PLCγ and PIP3. The direction of this edge was wrongly inferred, though they did not explain why this happened. The algorithm found indirect connections. For example the causal path PKC $\rightarrow$ Raf $\rightarrow$ Mek is well known in literature. The algorithm successfully found a connection between Raf and Mek, although Raf was not perturbed by [51]. In certain instances, successful inference of indirect connections was achieved, even though they were mediated by unmeasured molecules.

Raf	**Mek**	**PLCγ**	**PIP2**	**PIP3**	**Erk**	**Akt**	**PKA**	**PKC**	**P38**	**Jnk**
26.4	13.2	8.82	18.3	58.8	6.61	17	414	17	44.9	40
35.9	16.5	12.3	16.8	8.13	18.6	32.5	352	3.37	16.5	61.5
59.4	44.1	14.6	10.2	13	14.9	32.5	403	11.4	31.9	19.5

Table 3.1: First three samples of the observational cytometry sub-dataset [51].

Raf	**Mek**	**PLCγ**	**PIP2**	**PIP3**	**Erk**	**Akt**	**PKA**	**PKC**	**P38**	**Jnk**	**INT**
1	1	1	2	3	2	1	3	1	2	1	8
1	1	1	1	3	3	2	3	1	2	1	8
1	1	2	2	3	2	1	3	2	1	1	8

Table 3.2: First three samples of the interventional cytometry sub-dataset [51].

Since this thesis focuses only on retrieving causal structure from observational data and does not investigate interventions of systems, only the observational dataset

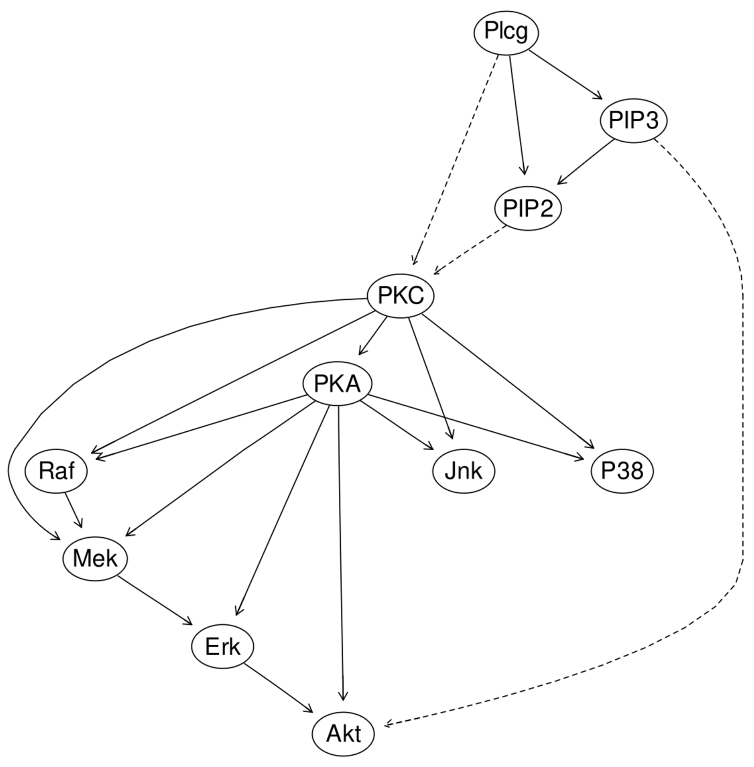


Figure 3.1: True-graph constructed by the Bayesian network structure inference algorithm with confidence > 0.85*, created by [52]. Missed edges are represented as dashed lines. The edge between Plcg and PIP3 should be reversed.*

will be further used. However, by doing that, it will not be possible to reproduce the true-graph solely by observations. For readers interested in the interventional dataset, detailed information and analysis can be found in the appendix.

3.2 Annotation

Above described dataset only comprises quantitative data. Thus, it was necessary to extend that with annotated qualitative data. Therefore, abstracts of cited works were reviewed to find reported edges and directions of the network. This was undertaken manually, as there were reasonable 23 papers mentioned by Sachs et al. [51]. One was not publicly available (PIP3 $\rightarrow$ PIP2). Doc id, title, abstract, relevant span, entity-1, entity-2, direction, cause, effect, search connection, entity-1 (alternative), entity-2 (alternative), source and availability were collected into an Excel table, see table 3.4 as example. Alternative names of the entities were included, because there are sometimes different abbreviations used for the same term.

Not every mentioned causal connection could be found in the stated literature. This might either be due to little experience in reading papers of this domain or because the connection was not referenced in the abstract. In the latter case, that would mean that the connection can be found in the body of the source. However, using a manually chosen paragraph would disturb the comparability, as in an abstract the information is usually denser, than in some text passage.

See table 3.3 for more information.

Connection	**Influence Path**	**Annotated**
PKC → Raf	PKC → Ras → Raf	not found
PKC → Mek	PKC → Raf → Mek	found
PKC → Jnk	PKC → MKKs → Jnk	found
PKC → P38	PKC → MKKs → P38	not found
PKC → PKA	PKC → cAMP → PKA	not found
PKA → Raf	PKA → Raf	found
PKA → Mek	PKA → Raf → Mek	found
PKA → Erk	Unknown	found
PKA → Jnk	PKA → MKKs → Jnk	found
PKA → P38	PKA → MKKs → P38	found
Raf → Mek	direct	found
PKA → Akt	PKA → CaMKK → Akt	found
Mek → Erk	direct	found
Plcγ → PIP2	direct	found
Plcγ → PIP3	direct (reversed edge)	found
PIP3 → PIP2	precursor-product	unavailable
Erk → Akt	direct or indirect	found

Table 3.3: Search connection and influence paths of the dataset. Information about where the phosphorylation happened was removed. Plcγ → shows the reversed inferred edge by the algorithm.

3.3 Methodology

Two approaches were attempted to extract causal relations from text. In the following, the overall ideas and top-level mechanics will be explained. For better clarity, the content is divided in subsequent subchapters: Qualitative Data 3.3.1, Quantitative Data 3.3.2 and Combination 3.3.3. The resulting framework is presented in the Combination section.

3.3.1 Qualitative Data

For this task, the idea was to leverage a language model, that could extract causal relations from above-mentioned abstracts. In a subsequent step, the extracted entities should be grounded to an ontology, to handle inaccuracies or synonyms. Because of the little domain-specific expertise, knowledge-based and machine-learning based methods were dismissed. Complicating matters was the difficulty in finding a dataset for training purposes, that includes labeled entities and relations of a similar type. According to the decision-tree of [65], a deep-learning approach was pursued. With that and because of the recent upcoming of language models like BERT, it was believed that the constraint imposed by the scarcity of data was mitigated to a sufficient degree. Two of these models should be fine-tuned, so that the first model extracts protein entities, while the second one classifies the relation.

This is called the traditional "pipeline" approach, which is admittedly not state-of-the-art, as it contains several drawbacks: possible errors from the NER model propagate to the RE model, resulting in an increase of misclassified relations, an imbalance of classes, as the class "no relation" will dominate, an increase in complexity

docid	27
title	Inhibitory cross-talk by cAMP kinase on the calmodulin-dependent protein kinase cascade
abstract	The calmodulin-dependent kinase (CaM-K) cascade, a Ca2+-triggered...
relevant_span	PKA gave rapid phosphorylation in vitro and in cells of recombinant CaM-KK, resulting in 50- 75% inhibition of CaM-KK activity, part of which was due to suppression of CaM-binding by phosphorylation of Ser458 in the CaM-binding domain.
entity1	PKA
entity2	CAMKK
direction	1
cause	PKA
effect	CAMKK
search_connection	PKA to AKT
entity1_alternative	-
entity2_alternative	PKB, Rac
source	https://ohsu.pure.elsevier.com/en/publications/inhibitory-cross-talk-by-camp-kinase-on-the-calmodulin-dependent–2
available	yes

Table 3.4: Sample of self collected qualitative data from Wayman, Tokumitsu, and Soderling [63] cited by Sachs et al. [51]. The complete causal path contains PKA → CAMKK → AKT. The connection between CAMKK → AKT was described in a consecutive sample. CAMKK is an unmeasured molecule, which acted as intermediary.

due to many possible entity pairs and a confused classifier, when entity pairs belong to several relations simultaneously [69].

But, since this thesis' goal is not to build a state-of-the-art model and the RE model has to be a binary classifier distinguishing between *cause* and *no_cause* mitigating at least the last concern, a pipeline approach was more appealing and straightforward.

With the GENETAG dataset [58], the named entity recognition task should be accomplished. It comprises 15000 annotated sentences with proteins and genes, sourced from MEDLINE database. The self-annotated textual samples serve for learning the relation extraction task, although the sample size is too little. A different biomedical RE dataset could not be found.

For the grounding of the extracted entities, the decision was made to use the ontology included in the TRIPS parser [4]. It was the most appropriate one in terms of simplicity and coverage.

3.3.1.1 BioBERT Approach

The decision was made in favor of BioBERT [35], as this is a BERT model specifically pre-trained on large-scale biomedical corpora, with increased F1 scores in named entity recognition (NER) and relation extraction (RE) in the biomedical field compared to standard BERT.

BioBERT has been fine-tuned on the GENETAG dataset. To increase the training speed, GoogleColab was used [22]. The results were not promising. Because of that, the RE model was dismissed.

3.3.1.2 GPT-3.5 Approach

The second approach conducted was to utilize GPT-3.5 text-davinci-003 by using the API of OpenAI. This model is based on GPT-3. It has 175 billion parameters [41], when BioBERT, which is based on BERT-base has 110 million [16]. While training, GPT-3 has seen more data than BERT, why it might deliver more accurate results in the area of biomedicine. According to Ye et al. [68], text-davinci-003 has promising skills in RE under zero-shot conditions, which is good, when having a limited amount of data.

A prompt text was created to establish a causal pathway between two specified protein molecules. The goal was to provide the identified path in a concise format, where the causing protein and affected protein are separated by a ">" sign. Subsequently, the entities were anchored using the TRIPS system, yielding satisfactory results. However, caution is warranted due to the inherent tendency of GPT models to occasionally generate fictitious information.

3.3.2 Quantitative Data

[51] showed in their work a graph solely constructed from observational data, to demonstrate the importance of interventional data. The approach is to recreate this as closely as possible without expecting to yield better results.

To facilitate this replication process, the R library bnlearn was utilized. This library offers a range of helpful functions specifically designed for this task. Additionally, the documentation provided by the library includes an example demonstrating how the graph was constructed [52], which served as a reference and was closely followed during the replication process. As a baseline, FCI is executed on the observational dataset, utilized from the R library pcalg. This algorithm was chosen, because it accounts for unconditioned confounders.

3.3.3 Combination

After constructing the quantitative and qualitative derived graphs separately, the approach was to finally compare them, by utilizing the R libraries iGraph and bnlearn. The previous steps and the combination process result in the framework illustrated in figure 3.2.

Several graphs were compared with respect to its nodes and arcs. Additionally, true-positive, false-positive and false-negative edges were calculated and plots of the resulting graphs were constructed.

3.4 Implementation

Subsequently, the implementation process will be described with focus on the most challenging or creatively solved tasks. Again, divided into the sections Qualitative 3.4.1, Quantitative 3.4.2 and Combination 3.4.3.

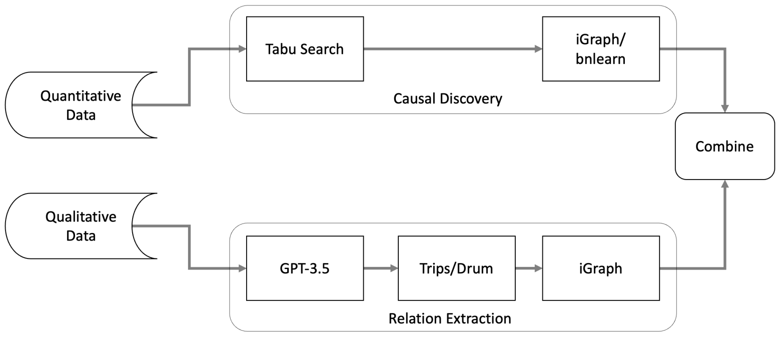


Figure 3.2: Constructed framework for the combination of causal structures derived from qualitative and quantitative data in the field of biochemistry.

3.4.1 Qualitative Data

This section is split up into the approach conducted with BioBERT (section 3.4.1.1) and consecutively the one with GPT-3.5 in section 3.4.1.2.

3.4.1.1 BioBERT Approach

Via the Hugging Face API, BioBERT v1.1 [17] and GENETAG [11] data was downloaded into the GoogleColab environment. The data was shortly analyzed before partitioned into train, validate and test sets. The sentences are tagged by the Part-of-Speech (POS) scheme. Proteins and genes have the tag *NEWGENE*. This was utilized to create a BIO-scheme. Since BERT uses the word-piece tokenizer, some words are broken up into sub-words. A function was programmed, to align the BIO-scheme with these tokens.

Furthermore, a `Datasequence` and `BertModel` class were constructed, followed by the creation of a training loop function. This function was executed for 5 epochs, employing a learning rate of $5 \cdot 10^{-3}$ and a batch size of 2. The evaluation function produced satisfactory results. However, it became apparent that the model was infeasible for practical use, due to inaccurate results.

3.4.1.2 GPT-3.5 Approach

In a subsequent attempt, the utilization of GPT-3.5 text-davinci-003 via the OpenAI API was explored. To initiate this process, a prompt, as shown in Listing 3.1, was formulated. However, due to the inaccurate results of the NER model, the objective was adjusted slightly. The goal became to validate the quantitative reasoning of the network using qualitative data. Consequently, text-davinci-003 was tasked with identifying the causal path between two pre-defined molecule entities.

```
f"What is the causal pathway between {row.From} and {row.
   To} in the following paragraph? Link the found chain
   by '>' signs:"
```

Listing 3.1: Prompt, that was send to GPT-3.5. `row.From` and `row.To` belong to the iteration over a `pandas.DataFrame`, where both denote cells of the respective protein molecules.

```
<ekb>
    <input type="text">
        <paragraphs>
            <paragraph>
                AKT phosphorylation
            </paragraph>
        </paragraphs>
    </input>
    <TERM>
        <type>ONT::GENE-PROTEIN</type>
        <name>AKT</name>
    </TERM>
</ekb>
```

Listing 3.2: Simplified TRIPS response to the request "AKT phosphorylation", saved as XML.

The temperature parameter was set to 0.0 to minimize the randomness of the replies. The results were split up and saved in a `pandas.DataFrame`. In a second step, this now existing extracted entities should be grounded to the ontology of the TRIPS system.

Replies of TRIPS were saved into an XML file. Simplified structure is visible in listing 3.2.

The content of the `<name>` tag was used for further processing. After testing the original molecules, measured by Sachs et al. [51], the permitted types were restricted to those listed in succeeding enumeration.

- ONT::GENE-PROTEIN
- ONT::PROTEIN-FAMILY
- ONT::GENE
- ONT::PROTEIN
- ONT::GENE-FAMILY
- ONT::CHEMICAL

Next, the causal paths were simplified by merging consecutive occurrences of the same entities, resulting in the consolidation of these sequential instances into a single entity.

Last but not least, two functions were programmed to make the graph iGraph and ggdag compatible by generating two literals, that the respective libraries are able to process. Both were saved into a .txt file to import them later into R.

3.4.2 Quantitative Data

In the following, the observational data was used and later compared with the true-graph.

The data was loaded into R. As a baseline, a simple FCI+ algorithm was run on the observational data with $\alpha = 0.01$ and `gaussCItest()` to test for independence.

Next, the observational dataset was briefly analyzed. Every molecule concentration is skewed, mostly close to zero. In a scatter plot, it is visible that the molecule relationships comprise linear and non-linear dependencies, visible in 3.3. Thus, an approach with Bayesian networks was attempted [52].

To begin, the observational dataset underwent a discretization process utilizing an iterative method [27], already incorporated within the bnlearn package. This approach yielded superior preservation of pairwise information compared to quantile or interval discretization techniques.

5000 random networks were created. This was undertaken with the function `random.graph()`, seen in listing 3.3. Next, a `tabu()` search was conducted, before the strength of the probabilistic relationships with `custom.strength()` was measured. Last, the network was averaged, containing only the significant edges. This was done with the function `averaged.network()`, while leaving the significance `threshold` parameter blank. By default, bnlearn offers a statistical way of determining a suitable value. The resulting threshold was determined to be 0.358.

3.4.3 Combination

The network derived from the qualitative data was imported into R and a comparison between the graphs were performed. Comparing the qualitative graph with some quantitative graph was done by `graph.intersection()`.

When evaluating differences and similarities between quantitative derived graphs, the function `compare()` from bnlearn was used. This offered a superior view, because it is directly visible, how many true-positive, false-positive and false-negative arcs exists. This was not possible with the qualitative network, as the set of vertices is different to the quantitative graphs.

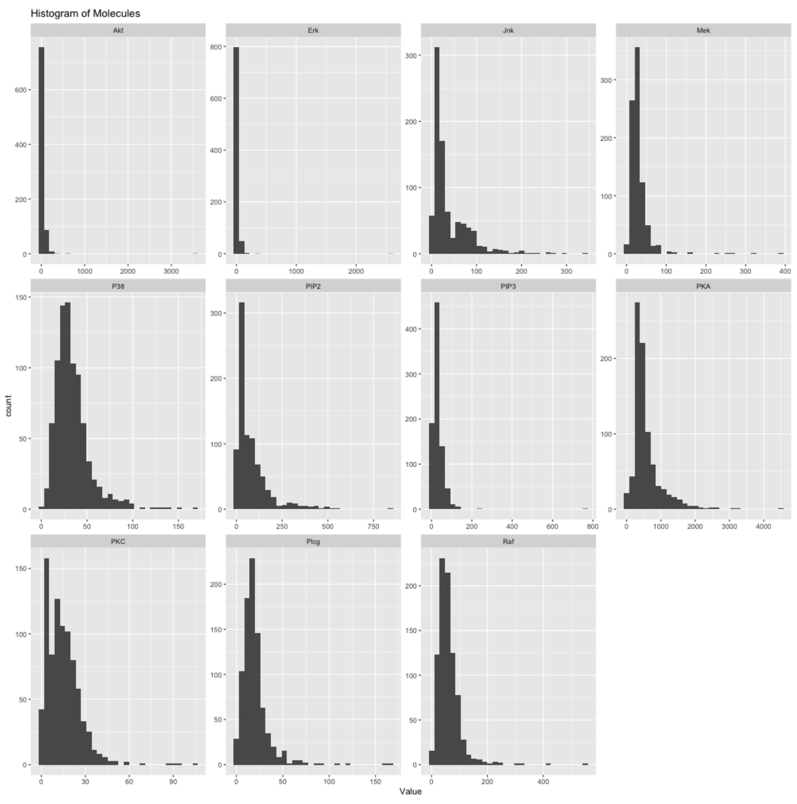
To address the inconsistencies in the ontology results, a manually constructed simplified graph was created. This graph was then compared to a modified version of the true-graph, in which molecule names were adjusted by capitalizing letters or making other minor changes, such as converting PIP-2 to PIP2 or Akt to AKT.

3.5 Results

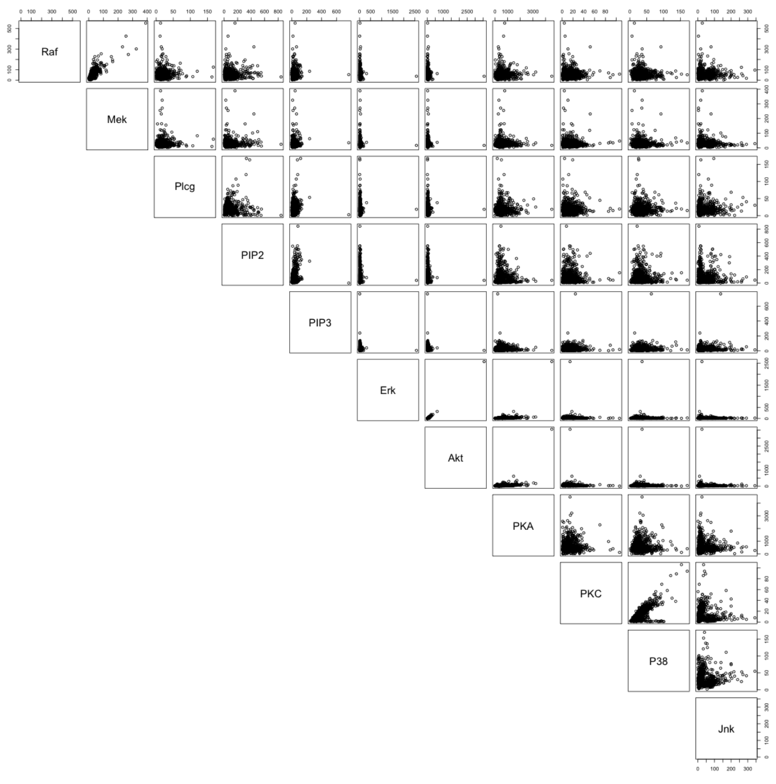
In this section, the results will be presented. It is structured in the previous order. First, the qualitative derived results will be shown in section 3.5.1. Second, a view on the quantitative results in section 3.5.2 is provided, before the chapter ends with the results obtained from the combination process in section 3.5.3.

3.5.1 Qualitative Data

The following section is separated by the conducted approach. Results obtained from the BioBERT approach are depicted in subsection 3.5.1.1, while results from the GPT-3.5 model are presented in subsection 3.5.1.2



(a) The histogram of each molecule.



(b) Scatter plot of the molecules.

Figure 3.3: Distribution and scatter plot of the molecules.

```
nodes <- names(dsachs)
start <- random.graph(
    nodes = nodes,
    method = "ic-dag",
    num = 5000,
    every = 100
    )
netlist <- lapply(
    start,
    function(net){
        tabu(
            dsachs,
            score = "bde",
            iss = 10,
            start = net,
            tabu = 200
            )
        }
    )
cus.strength <- custom.strength(netlist, nodes = nodes)
avg.observational <- averaged.network(cus.strength)
```

*Listing 3.3: 5000 random networks were randomly generated. The network space was investigated by Tabu search. IC-DAG refers to a certain method to randomly generate Bayesian networks. Only one random generated DAG every 100 iterations was kept, to ensure that they are different. BDE is short for Bayesian Dirichlet equivalent, ISS to imaginary sample size and tabu (argument) to the length of the list used in the **tabu()** function. Afterwards, the strength of the relationships was calculated, before averaging the network to its signifcant edges.*

3.5.1.1 BioBERT Approach

On the qualitative side, the first approach conducted was to train a NER BioBERT model. It achieved an accuracy of 80.1% on test data. However, the model was not usable. An anecdotal evidence is provided in table 3.5. Various other samples were examined, demonstrating similar outcomes.

Sample	Multiple single-stranded cis elements are associated with activated chromatin of the human c-myc gene in vivo.
Tokens	Multiple, single, -, stranded, c, ##is, elements, are, associated, with, activated, ch, ##roma, ##tin, of, the, human, c, -, my, ##c, gene, in, v, ##ivo, .
Labels	O, O, O, O, O, O, O, O, O, O, O, O, O, O, B, I, I, O, O, O, O, O, O, O, O, O
Extracted Entities	##tin, of, the
True Entities	human c-myc gene

Table 3.5: Sample no. 13 of the GENETAG test set. Generated tokens are visible in the second row. Labels are in the BIO-format. Last, the (falsely) extracted entities compared with the actual entities.

3.5.1.2 GPT-3.5 Approach

The next approach was to utilize GPT-3.5 text-davinci-003. This yielded the graph in figure 3.4a. Next to it is a simplified network, that was created from hand with similar named nodes merged. This was done to demonstrate the influence, the ontology can have on the output. For example RAF-1, RAF-B and RAF are merged into one node. The original graph contains 36 nodes and 53 edges. The manual simplification reduced these to 29 and 42.

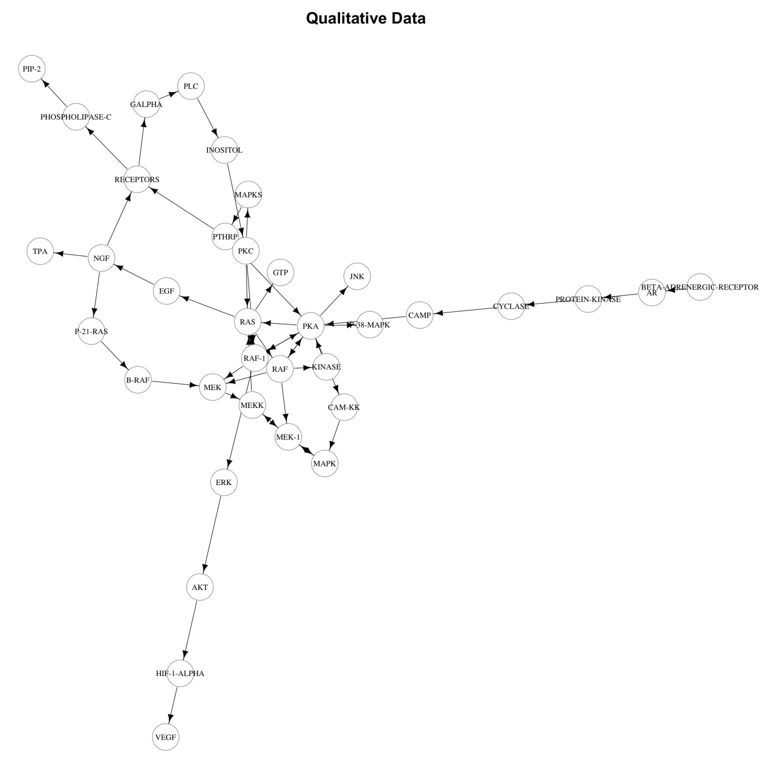
In figure 3.5, similarities between the true-graph, the qualitative graph and the simplified graph are visible. That means arcs that are contained in both graphs. Differences between the true-graph and qualitative graphs are visible in the appendix section A.2.2. That are arcs from the true-graph, that were not found by GPT-3.5.

Out of the 17 arcs present in the true-graph, the following 4 arcs were identified in the non-simplified comparison:

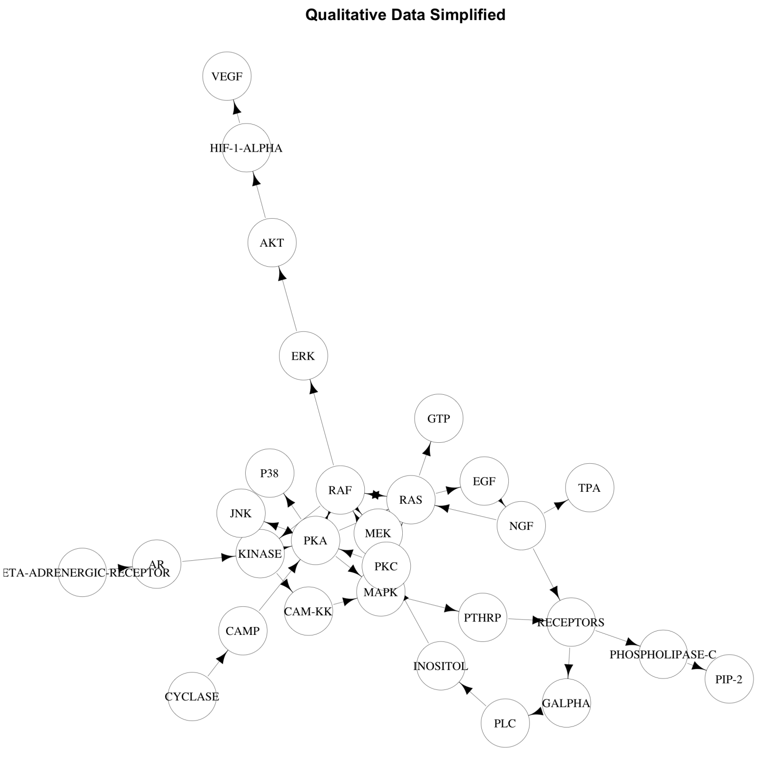
- PKA $\rightarrow$ JNK
- PKA $\rightarrow$ RAF
- RAF $\rightarrow$ MEK
- ERK $\rightarrow$ Akt

Comparing the simplified graph, this extends to the following 7 arcs out of 17:

- PKA $\rightarrow$ JNK
- PKA $\rightarrow$ RAF



(a) Graph extracted from GPT-3.5 text-davinci-003. It contains 36 nodes and 53 edges.



(b) Simplified graph by hand to demonstrate the influence of the ontology. It contains 29 nodes and 42 edges.

Figure 3.4: Qualitative Graphs

- RAF $\rightarrow$ MEK
- ERK $\rightarrow$ AKT
- PKA $\rightarrow$ P38
- PKC $\rightarrow$ PKA
- PKC $\rightarrow$ RAF

3.5.2 Quantitative Data

From a quantitative standpoint, scatter plots and histograms are visible in figure 3.3a and 3.3b. The distribution of the molecules is non-Gaussian, exhibiting a left-skewed pattern with the majority of concentrations around zero. The relationships among the molecules encompass a mixture of linear and non-linear dependencies, which are visible in cases such as Erk and Akt or Erk and Jnk for linear associations, and PIP2 and Jnk or PKC and P38 for non-linear connections.

The results from the FCI+ algorithm are visible in figure 3.6. Its skeleton comprises 8 true-positive (TP), 0 false-positive (FP) and 9 false-negative (FN) edges. When considering directions, the similarities reduce to 0 TP, 8 FP and 17 FN.

In figure 3.7b, the network resulting from 5000 random graphs, searched by `tabu()` on the observational data is shown. It has 8 TP, 1 FP and 9 FN edges, when comparing the skeleton with the true-graph. When the directions are taken into account, the similarity decreases to 4 TP, 5 FP and 13 FN edges.

3.5.3 Combination

Tables 3.6 and 3.7 present the comparative analysis of the networks. In their paper, [51] constructed a graph solely based on observational data. The results from this graph in comparison to the true-graph are visible in the tables as well. Additionally, it is visualized in figure 3.7a.

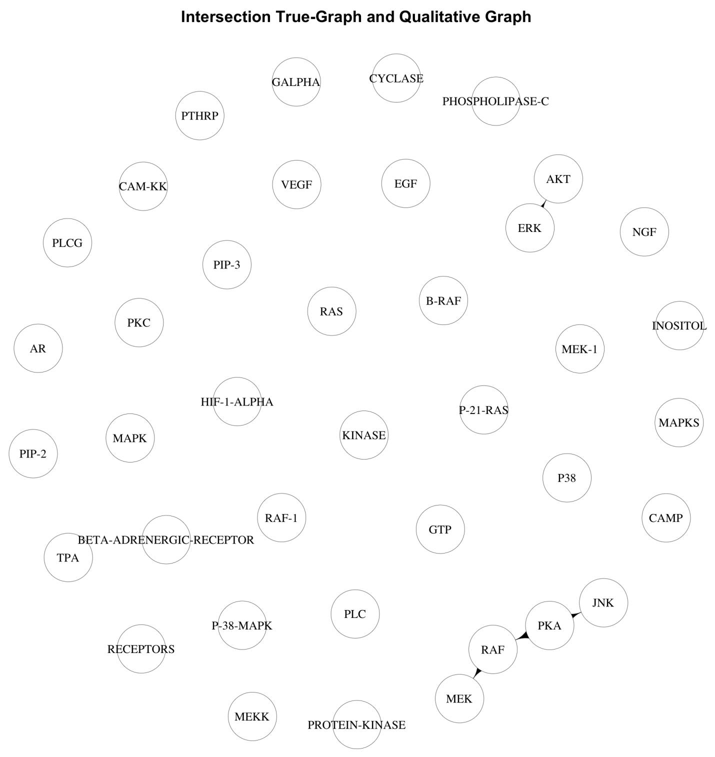
A comparison with the qualitative graph was not possible this detailed, because the set of nodes is different to the qualitative graph.

	True-Graph	**Sachs Observational**	**FCI+**	**Observational**
TP	17	9	8	8
FP	0	1	0	1
FN	0	8	9	9

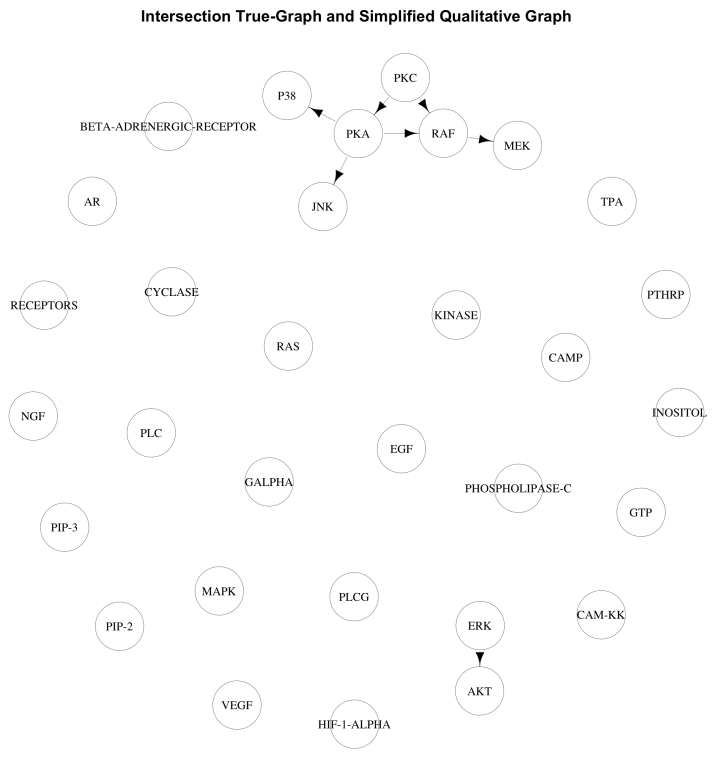
Table 3.6: Comparison of the skeletons.

	True-Graph	**Sachs Observational**	**FCI+**	**Observational**
TP	17	0	0	4
FP	0	10	8	5
FN	0	17	17	13

Table 3.7: Comparison of edges and its directions.



(a) qualitative graph.



(b) Manually simplified qualitative graph.

Figure 3.5: Intersections of the true-graph with the qualitative and qualitative simplified graphs.

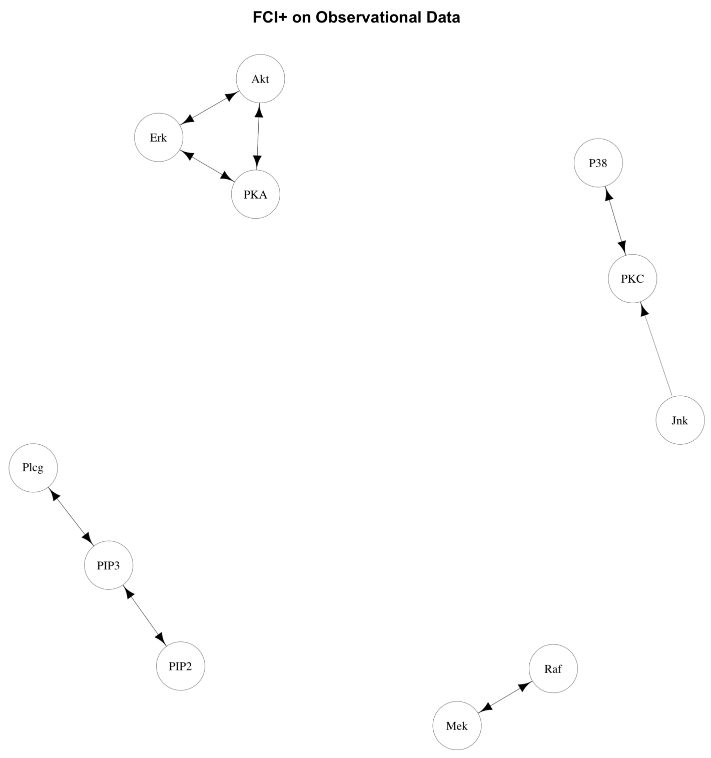


Figure 3.6: Graph yielded from the FCI+ algorithm, executed on the observational data.

3.6 Discussion

As a brief reminder, the goal of this thesis was to investigate, whether a common framework can be developed that enables the combination of qualitative and quantitative sourced causal structures. On this way, problems and considerations are monitored.

The following section is divided into the Qualitative Data 3.6.1 and the Quantitative Data 3.6.2. The combination process with the true-graph is included in the former section.

3.6.1 Qualitative Data

This section is split according to the two approaches conducted: BioBERT in subsection 3.6.1.1 and GPT-3.5 in subsection 3.6.1.2.

3.6.1.1 BioBERT Approach

At first glance, the NER BioBERT model looks promising, as it yielded an accuracy of 80.1% on test data, that was never seen before. However, upon personal review, the results were not usable.

One reason for the high accuracy might be, that the data is highly biased, with an overwhelming amount of tokens tagged as *OUTSIDE* in BIO-scheme. Thus, the model might have learned to predict very few times, rather randomly, a *BEGIN* or *IN* tag, but most of the time an *OUTSIDE* tag.

Adding complexity to the situation, the annotators [58] have chosen a broad definition for gene and protein entities, encompassing genes, proteins, RNA, domains, complexes, sequences, and fusion proteins. They argue, that NER tasks are inherently challenging due to the ambiguous nature of genetic nomenclature. Consequently, the dataset may not have been suitable as it covers a wide scope. Nevertheless, a different dataset could not be found, especially with annotated relations. This poses a bottleneck.

3.6.1.2 GPT-3.5 Approach

The results yielded from GPT-3.5 text-davinci-003 only confirm 4 arcs. When manually simplified, 7 arcs match out of 17 from the true-graph. One has to consider, that 4 connections were not annotated and searched by the model. That comprises the following influence paths:

- PKC → Ras → Raf
- PKC → MKKs → P38
- PKC → cAMP → PKA
- PIP3 → PIP2

When comparing these connections with the graph, visible in figure 3.4b, it is likely, that PKC → P38 and PIP3 → PIP2 are not found because of that. This would increase the found edges to 9 out of 17.

However, overall the result is not satisfying. There are four possible reasons for that:

(1) This setting was a zero-shot setting. The model did not have any fine-tuning on the dataset. This is still challenging. Especially in such a specific area of knowledge.

(2) The text-davinci-003 model was constructed primarily for text generation purposes. Consequently, one explanation for the density of the graph could be that when faced with uncertainty, the model tends to err on the side of providing more rather than fewer extracted entities. Furthermore, it is acknowledged that the model occasionally fabricates information to fulfill user expectations, which poses a risk. This went so far, that the model replied implicitly derived causes or effects. When testing these effects, it became eminent, that it happened mostly when searching without constraints for causality in text. Thus, text-davinci-003 was tasked with finding the causal connection between the two given molecules, rather than searching for it freely. Though, this causes the limitation, that an overall idea of possible connections has to be present, that the model tries to confirm.

(3) The true-graph is a rather shallow, generalized example to investigate the potential of Bayesian networks, by [51]. It is likely, that there are more molecules playing a role in signaling pathways, that [51] did not account for. However, these connections were investigated more closely in the referred papers, resulting in a different refinement. One example for this is the influence path PKC → Ras → Raf. Ras has not been measured by [51], therefore it does not exist in the true-graph. Though, text-davinci-003 discovered the connection correctly (figure 3.4b). In this case, Ras blocks the connection and misleads the comparison. That the arc exists anyway is just caused by the path PKC → Raf → Mek.

(4) It is evident, that the used TRIPS parser was not reviewing and simplifying enough. For example, the node *RECEPTORS* is not a specific protein. Still, TRIPS categorizes it as *ONT::PROTEIN*. Regrettably, the system of proteins and lipids is not easy understandable. Therefore, it was not possible to leverage a different ontology.

3.6.2 Quantitative Data

When comparing the pairwise quantitative sourced graphs, it is evident that there is a higher degree of similarity when only considering the skeleton. This outcome was expected. The graph skeleton constructed from observational data is slightly worse, compared to the FCI+ baseline. This is, because the connection P38 → Jnk is falsely identified as significant.

Though, when considering the directions, Tabu search does find more TP and less FP/FN edges, compared to the FCI+ algorithm. This is also an improvement, when compared to the results yielded from [51] on the observational data. Tabu search seems to be better in this case, compared to the annealing search algorithm used by [51].

However, it is still far away from the true-graph. This underlines the importance of interventions and interventional datasets.

Limitations

There are two limiting factors:

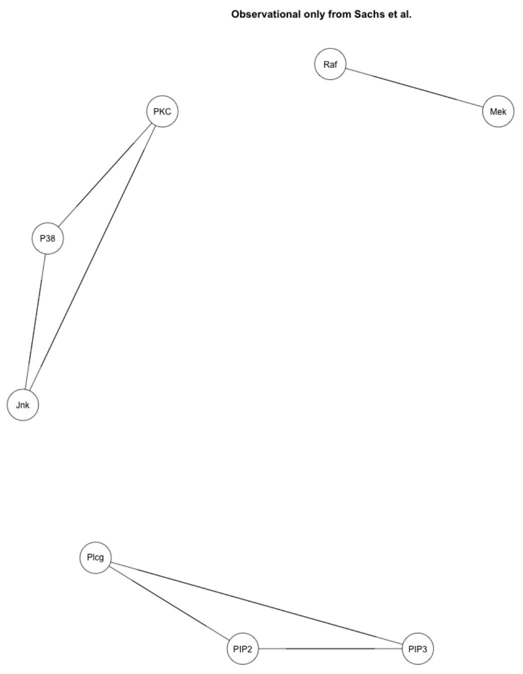
(1) It was already explained, that one has to provide the model the entities, between which a causal connection should be found. This is to avoid a too large searching scope and to minimize the extraction of possible fabricated or implicit deducted entities.

(2) The numerous and ambiguous abbreviations and different terms, that account for the same or very similar entity. This problem occurs for example in PIP2, PIP-2, PIP_2 or Phosphatidylinositol-4,5-bisphosphate and several others [1]. This makes it very difficult to consent on the same term.

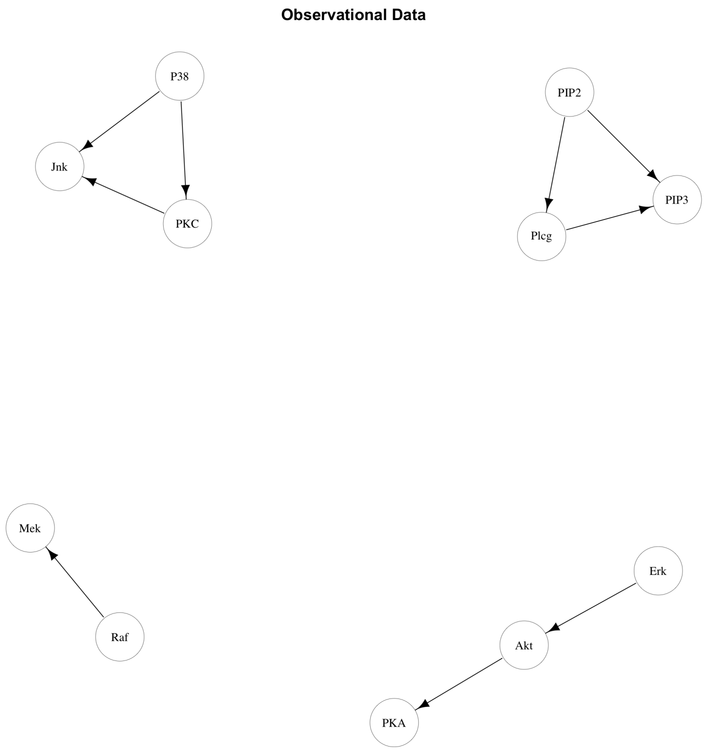
Improvements

A first step for improvement would be to leverage a specific and more powerful ontology, than the one included in the TRIPS system. Another approach to improve the results is to change the underlying model. For example by using GPT-4 or finding a suitable dataset and fine-tuning another language model on this task. By doing so, the noise might be reduced furthermore.

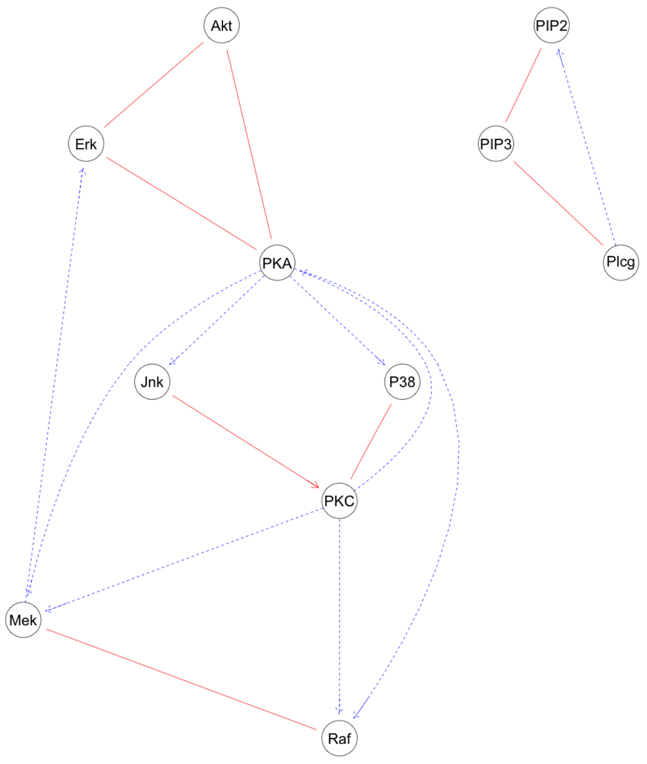
On the quantitative side, it would be necessary to consider interventional data to improve the results. This is shown in the appendix A.2.1.



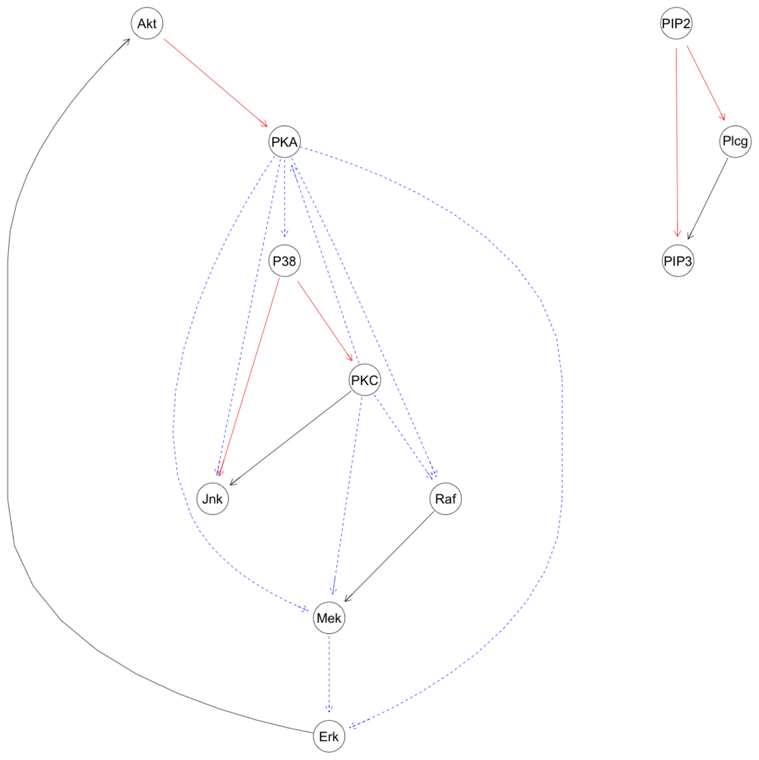
(a) Found graph from [51] on observational data only.



(b) Graph resulting from Tabu search, run on the observational data.



(a) Comparison between the true-graph and the graph generated from FCI+.



(b) Comparison between true-graph and Tabu search on observational data.

Figure 3.8: Visualized comparison between the true-graph and the graphs generated by FCI+ and Tabu search. Black arcs represent TP, blue-dashed arcs FN and red FP.

4. Example 2: Financial Dataset

In short: The upcoming utilized dataset is self-collected and comprises again qualitative and quantitative data. It will be investigated, how qualitative and quantitative received causal structure can be combined. For the qualitative part, four language models were fine-tuned. These language models were consecutively included in a self-constructed application, that analyzes the financial qualitative data for causality. It was named CausalExtraction_Linker (CE_Linker). Subsequently, these found connections will be combined with the causal structure received from the quantitative data.

In section 4.1, an introduction to the dataset will be presented. The overall concept and the methods employed will be further explained in Methodology 4.2, which will be divided into subsections: Qualitative Analysis, Quantitative Analysis, and the Combination of both methods. The Implementation 4.3 will follow the same structure as in Methodology but will provide more detailed information on challenging tasks encountered during the implementation process. The obtained results will be presented in the Results 4.4, and subsequently elaborated in the Discussion 4.5.

4.1 Dataset

The dataset used here consists of both qualitative and quantitative data, which has been self-collected from a total of 30 companies. These companies are categorized into three countries: Germany, the United States, and China. Specifically, there are 12 companies from Germany, 12 companies from the United States, and 6 companies from China. The selection of these companies took into account various factors, including the representation of different industries, the public prominence of the companies, and a balanced distribution across the three countries.

Nevertheless, an imbalance was inevitable, primarily due to challenges encountered in identifying suitable Chinese companies. These companies needed to meet certain criteria, such as being publicly traded in the United States and having sufficient public attention to generate public analyst reports. However, factors such as the ongoing trade conflict between the United States and China, as well as Chinese protectionism, posed challenges in this regard and had to be considered during the selection process.

Determining the country affiliation of a company was based on its place of domicile or previous companies that have been merged into the current entity (e.g., AIRBUS SE). The complete list of corporations is visible in table 4.1.

Sector	**Company**	**Symbol**	**Country**	**#**
Technology	Baidu, Inc.	BIDU	China	21
Technology	Tencent Holdings Limited	TCTZF	China	8
Technology	Microsoft Corporation	MSFT	USA	161
Technology	Apple Inc.	AAPL	USA	229
Technology	SAP SE	SAP	Germany	8
Energy	Exxon Mobil Corporation	XOM	USA	96
Energy	PetroChina Company Limited	PCCYF	China	4
Energy	RWE Aktiengesellschaft	RWNFF	Germany	2
Financial Services	The Goldman Sachs Group, Inc.	GS	USA	20
Financial Services	Deutsche Bank Aktiengesellschaft	DB	Germany	13
Insurance	The Allstate Corporation	ALL	USA	4
Insurance	Allianz SE	ALIZF	Germany	5
Aviation	The Boeing Company	BA	USA	57
Aviation	American Airlines Group Inc	AAL	USA	13
Aviation	Deutsche Lufthansa AG	DLAKF	Germany	5
Aviation	Airbus SE	EADSF	Germany	11
Healthcare	Johnson & Johnson	JNJ	USA	25
Healthcare	Bayer Aktiengesellschaft	BAYZF	Germany	5
Healthcare	Fresenius Medical Care AG & Co. KGaA	FMS	Germany	2
Consumer Goods	The Procter & Gamble Company	PG	USA	20
Consumer Goods	JD.com, Inc.	JD	China	13
Consumer Goods	PDD Holdings Inc.	PDD	China	13
Consumer Goods	Henkel AG & Co. KGaA	HELKF	Germany	6
Automotive	General Motors Company	GM	USA	20
Automotive	Geely Automobile Holdings Limited	GELYF	China	4
Automotive	Volkswagen AG	VLKAF	Germany	10
Defence	Lockheed Martin Corporation	LMT	USA	22
Defence	Rheinmetall AG	RNMBF	Germany	4
Chemicals	Dow Inc.	DOW	USA	18
Chemicals	BASF SE	BFFAF	Germany	6

Table 4.1: Companies in this dataset. Downloaded analysis reports are denoted in column #.

Share prices from these companies were gathered between 01.09.2022 and 28.02.2023, resulting in 6 months of time-series data (see fig. 4.1). For foreign companies (non-U.S. companies), the F-shares were chosen over American Depositary Receipts (ADRs, Y-shares), as they are usually more liquid [2]. The time-series data was collected via the free API from Alpha Vantage Inc. [20].

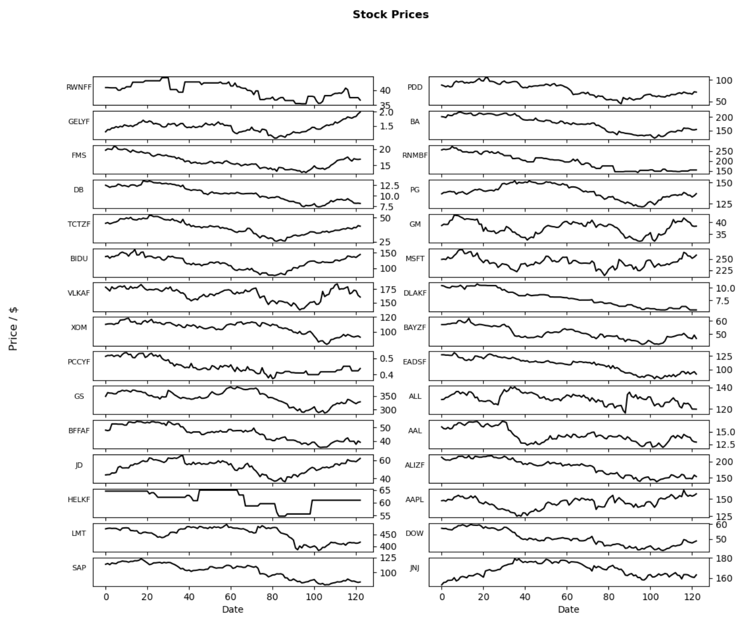


Figure 4.1: Overview over the stock prices.

824 analysis reports of these corporations, published during the same time period were retrieved by crowd-sourced financial markets' website Seeking Alpha [53]. A different, more professional source of textual knowledge could not be used, because of payment barriers.

Subsequently, this dataset was investigated on causality and later combined.

4.2 Methodology

In the upcoming section, the high-level ideas and conducted methods will be explained. The section Qualitative 4.2.1 contains the concept and utilized tools to fine-tune four language models and the basic ideas behind the developed program, that applies these models. Afterwards, the quantitative approach will be described in section 4.2.2 before ending this chapter with more information about the combination process in section 4.2.3.

4.2.1 Qualitative Data

Overall Idea

The overarching objective was to develop causal connections among analysis reports from diverse companies to later combine and compare them with connections established by the causal discovery algorithms, deployed on the quantitative data.

To analyze the qualitative data, a pipeline approach was employed. This decision was based on the factors previously discussed in section 3.3.1.

Subsequently, the following concept was chosen for the qualitative side:

1. A sentencizer splits up text into sentences.

2. A so-called text categorizer (TextCat) will act as a binary classifier, deciding whether a sample contains causality.
3. In positively labeled samples, a so-called span categorizer (SpanCat) will search for spans that can be classified as either *REASON* or *RESULT*.
4. Next, each span will be investigated for entities, that can be labeled as *PRODUCT*, *ORGANIZATION*, *LAW* or other relevant categories. This will be achieved with a standard NER model.
5. Last, a causal connection is going to be established between e.g. Company A and Company B, if the following criterion is met: Entities extracted from a *REASON* labeled span from Company A are found in a *RESULT* labeled span from Company B.

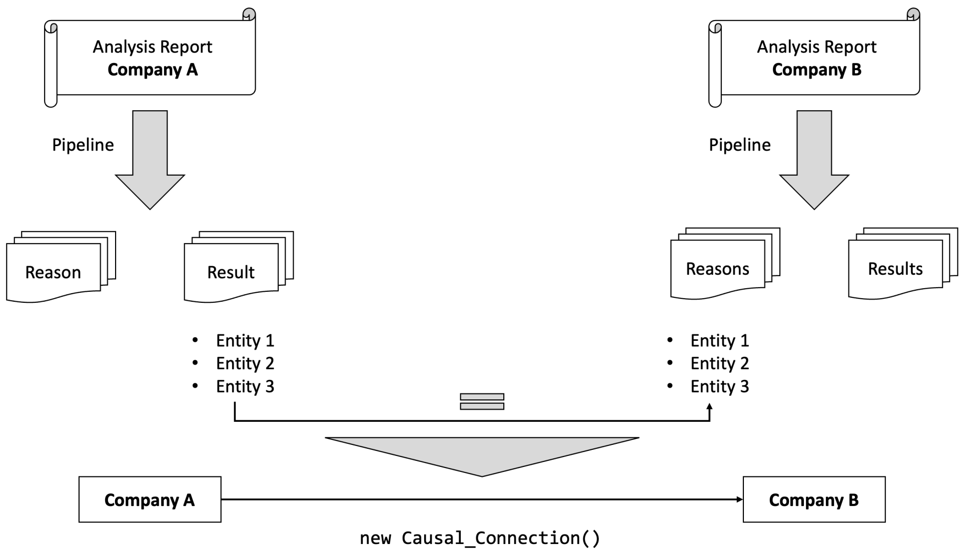


Figure 4.2: Basic idea, how two companies should establish a connection.

Though, this approach imposes some limitations:

It is important to emphasize that this process does not guarantee a causal connection between Company A and Company B. For instance, if the sample includes a **reason** that is unrelated to Company A, whose report is being analyzed, a causal connection cannot be established. In such cases, if the **result** matches with Company B, it would inaccurately imply a causal connection. However, because the analysis is written about Company A, it is safe to say, that there has to be at least some connection.

Additionally, the date published will not be taken into account, because it is unclear, when the extracted spans are located timewise. It might be possible, that the analyst talks about some past event or about an event that might happen in the future. Distinguishing, when a certain event happens seemed too challenging. However, the publishing date will be saved already, to be able to later account for that.

Training of the Pipeline

A well-suited dataset was identified that can be used to train the pipeline models: the fine-grained causal reasoning dataset developed by [66]. It consists of annotated

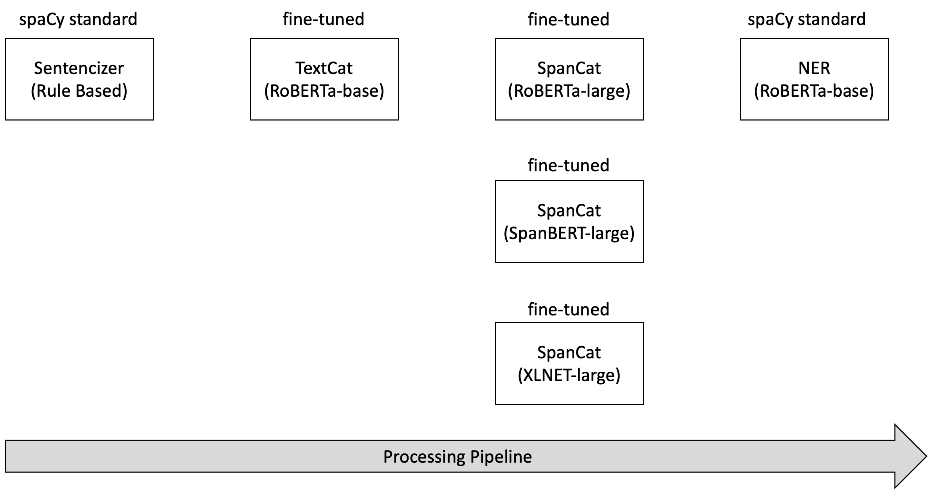


Figure 4.3: Diagram of the pipeline. Three SpanCat models exist, because it was tested which would deliver the best results. All three performed almost equally well. So they are interchangeable.

samples, originated from 6786 financial article reports, published between December 2020 and July 2021. Though, included were only U.S. listed companies.

This dataset can be further divided into sub-datasets for causality sentence classification, cause-effect event pairs and causal question-answering pairs. Only the first two were utilized.

The first one comprises 21046 positive and 29979 negative samples. Positive samples are those, that include at least one cause-effect pair. Out of these, 846 contain more than one sentence. The second dataset consists of 18457 uni-causal and 3017 multi-causal text spans. Labels distinguish between *CAUSE*, *ENABLE* and *PREVENT*. First can be understood as necessary and sufficient, while second means sufficient but not necessary and the last one, that both cannot exist at the same time. However, this will not be part of this task and each label was treated as *CAUSE*.

Both incorporate several industries, including communication services, consumer cyclical, financial services and some more.

Training the models for the TextCat and SpanCat task was undertaken with Google Colab Pro [22] and Python library spaCy [56]. This library offers a comprehensive environment for the development of natural language processing pipelines and to fine-tune custom models. The main idea is, that several (custom) models, called ”components” can be added to one pipeline object. When processing text, these components then label tokens in regard to their task. A diagram of the pipeline is seen in figure 4.3.

Surrounding Program

These pipeline was embedded in a python program, containing several classes to thoroughly analyze the qualitative dataset. It was named **CausalExtraction_Linker (CE_Linker)**. In section 4.3.1, an in-dept explanation is provided.

4.2.2 Quantitative Data

The approach involved analyzing the share prices previously collected. The adjusted closing price was utilized, which incorporates factors such as dividends, stock splits, or rights offerings that could impact the nominal share price. This ensured comparability.

No data preparation process was required since all the stocks used were officially tradable in the U.S.

Next, the time-series data was analyzed. Initially, a baseline Granger Causality test [23] was employed, followed by the utilization of a more advanced algorithm called PC Momentary Conditional Independence (PCMCI) [50]. As independence test, partial correlation was chosen. This actually assumes linear dependencies and Gaussian noise. However, these simplifying assumptions have to be made in the context of stock prices, because the independence test that accounts for nonlinear dependencies is too computational extensive.

To carry out these analyses, the Python library TIGRAMITE [49] was applied, which specializes in causal discovery within a time-series context.

4.2.3 Combination

Subsequently, the combination task, which is similar to the one already elaborated in section 3.3.3. Though, this time only iGraph was used. The respective framework is illustrated in figure 4.4.

The time dependency of the quantitative graph was resolved by merging the connections found over the complete lag. Directed edges had the highest order meaning that those were kept, when previously found undirected edges were dismissed.

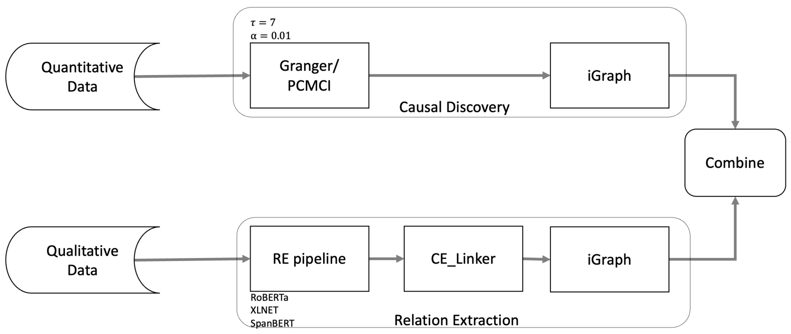


Figure 4.4: Constructed framework for the combination of financial causal structures derived from qualitative and quantitative data.

4.3 Implementation

In the succeeding section, the implementation process will be described. Section 4.3.1 will provide an in-depth look into the fine-tuning process of several language models. Additionally, the program developed to leverage the resulting language models and construct causal connections between the analyst reports is going to be explained. The analysis of the time-series stock prices will be elaborated in section 4.3.2, before ending the implementation section with a presentation of the combination process in section 4.3.3.

4.3.1 Qualitative Data

Upcoming content is quite extensive. That is why it has been divided into separate sections. In 4.3.1.1 will be illustrated how the four language models were trained. Hereafter in section 4.3.1.2, a detailed explanation to each constructed class for the causal extraction from financial reports is provided.

4.3.1.1 The Extraction Pipeline

Sentencizer

A rule-based sentencizer model from spaCy was leveraged, which is fed by a spaCy dependency parser. The text was decomposed into its sentences with this model in a first step.

TextCat

A TextCat model classifies text, according to previously learned features.

Initially, the dataset from [66] was prepared for the training and testing process. Since it already had a predefined train, validate, and test split, the data was merged, shuffled, and subsequently divided into 75% training, 15% validation, and 10% for testing purposes.

The data contains two columns. The first comprises one or multiple sentences, while the second contains labels, whether a causal relation exists in these sentences. With this structure, no data transformation was necessary to be able to convert it into a spaCy readable data format. This was accomplished by the function 4.1.

First, an empty spaCy pipeline object is created with `spaCy.blank("en")`. The variable `db` represents a `DocBin` object, that stores the annotated samples. The function iterates over the samples, creating spaCy `Doc` objects with the adjusted `doc.cats` property. Subsequently, the `DocBin` object is binary serialized and saved to a file.

The files created in that way can than later be easily used by spaCy to train and validate or test a model.

The generated data was used, to train a RoBERTa-base model on text classification. The training process was undertaken with spaCy's command line functionality and an according config file. By doing it that way, spaCy elegantly coordinates the training process in the background. The training was executed and yielded promising results. It did not seem necessary to leverage a RoBERTa-large model. The following hyperparameters were used:

- optimizer: Adam
- batch size = 2000 words
- max epoch = ∞ (early stopping)

```
nlp = spaCy.blank("en")

def convert(data, outfile, nlp):
    db = spaCy.tokens.DocBin()

    for doc, label in nlp.pipe(data, as_tuples=True):
        doc.cats["POS"] = label == "positive"
        doc.cats["NEG"] = label == "negative"

        db.add(doc)
    db.to_disk(outfile)
```

Listing 4.1: Function to convert the data from a .json file into a serialized spaCy readable file.

SpanCat

A SpanCat model labels spans of text. In case of spaCy, this component consists of two parts: A suggester function that proposes possible spans and a model, that labels the spans [56].

Multi-causal text spans were not utilized, as it was unclear how to label the spaCy `Doc` objects with more than one span that belong to each other. They make up for approximately 14% of the samples.

The training procedure for this task was similar to the training of the TextCat model. However, it was noticed that the data had some faults, when using it. SpaCy seems to have problems, when annotated spans start or end in a token. That is probably, because in this case it is unclear, whether the token belongs to the span or not. Another cause of errors recognized is, when leading or trailing white spaces exist. In the first case, samples were filtered out, while in the second case, the leading and trailing white spaces were adjusted.

When this was done, spaCy `Doc` objects were generated again and the respective `doc.char_span` adjusted.

Subsequently, three models were trained with this data. It was tested, which one would deliver the best results. That were: RoBERTa-large, SpanBERT-large-cased and XLNET-large-cased. RoBERTa models are by default cased. Cased models were preferred to ensure comparability to RoBERTa models. All three displayed satisfying results. The following hyperparameters were used:

- optimizer: Adam
- batch size = 2000 words
- max epoch = ∞ (early stopping)
- ngram suggester size = $[1 \ldots 40]$

SpaCy NER

To extract entities from the labeled *REASON* and *RESULT* spans, a standard spaCy pipeline was utilized, called `en_core_web_lg`. An existing RoBERTa-base model for NER can then be easily added to this pipeline as a component.

4.3.1.2 CE_Linker

The succeeding content describes the development of CE_Linker, that applies the training pipeline to find and extract causal connections, before connecting companies with each other. The class diagram is visible in figure 4.5. The description will start at the smallest component `Article_Info` and proceed step by step up to `Collection_Storage`. The last subparagraphs briefly describes the main run.

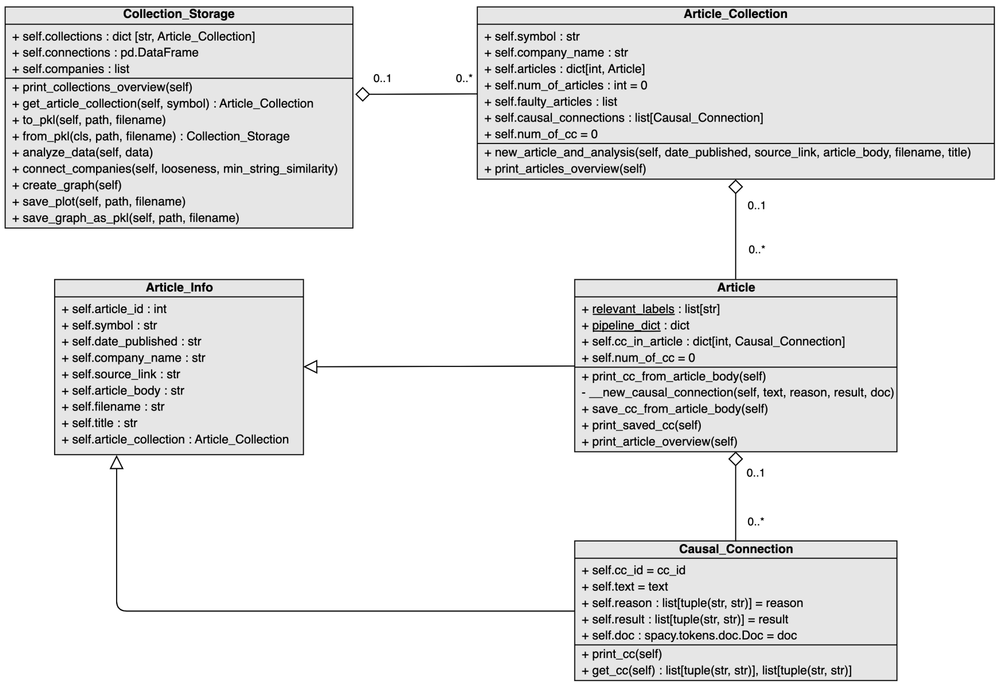

Collection_Storage
+ self.collections : dict [str, Article_Collection]
+ self.connections : pd.DataFrame
+ self.companies : list
+ print_collections_overview(self)
+ get_article_collection(self, symbol) : Article_Collection
+ to_pkl(self, path, filename)
+ from_pkl(cls, path, filename) : Collection_Storage
+ analyze_data(self, data)
+ connect_companies(self, looseness, min_string_similarity)
+ create_graph(self)
+ save_plot(self, path, filename)
+ save_graph_as_pkl(self, path, filename)
0..1
0..*
Article_Collection
+ self.symbol : str
+ self.company_name : str
+ self.articles : dict[int, Article]
+ self.num_of_articles : int = 0
+ self.faulty_articles : list
+ self.causal_connections : list[Causal_Connection]
+ self.num_of_cc = 0
+ new_article_and_analysis(self, date_published, source_link, article_body, filename, title)
+ print_articles_overview(self)
0..1
0..*
Article_Info
+ self.article_id : int
+ self.symbol : str
+ self.date_published : str
+ self.company_name : str
+ self.source_link : str
+ self.article_body : str
+ self.filename : str
+ self.title : str
+ self.article_collection : Article_Collection
Article
+ relevant_labels : list[str]
+ pipeline_dict : dict
+ self.cc_in_article : dict[int, Causal_Connection]
+ self.num_of_cc = 0
+ print_cc_from_article_body(self)
- __new_causal_connection(self, text, reason, result, doc)
+ save_cc_from_article_body(self)
+ print_saved_cc(self)
+ print_article_overview(self)
0..1
0..*
Causal_Connection
+ self.cc_id = cc_id
+ self.text = text
+ self.reason : list[tuple(str, str)] = reason
+ self.result : list[tuple(str, str)] = result
+ self.doc : spacy.tokens.doc.Doc = doc
+ print_cc(self)
+ get_cc(self) : list[tuple(str, str)], list[tuple(str, str)]

Figure 4.5: Class diagram of the constructed qualitative knowledge analyzing program.

Article_Info

This class exists to inherit several attributes containing basic information about analyst reports to the classes `Article` and `Causal_Connection`.

Causal_Connection

A `Causal_Connection` object will be instantiated, if the pipeline finds a **reason** and a **result** in a sentence. It inherits from the class `Article_Info` basic attributes that describe to which article it belongs. It also contains the attribute `self.doc`. This holds the same sentence, but in a format that spaCy returned, when the sentence was processed. The advantage is, in case that the program will be extended in the future, it is going to be faster, because these sentence were already processed.

The abbreviation ”cc” for example in `print_cc(self)` and `get_cc(self)` is short for **causal connection**.

Article

The upcoming class represents an analyst report. It is either part of an `Article_Collection` object or stays for itself. Two class variables exist: `relevant_labels` and `pipeline_dict`. Former is a list that determines, which labels are of interest. SpaCy's NER model contains multiple labels. But not all of them are useful to justify a causal connection. The labels *CARDINAL*, *MONEY* and *PERCENT* were removed. It was expected, that these entities are just too arbitrary. On the other hand, the following entities were accepted:

1. *DATE*
2. *EVENT*
3. *FAC* (Faculty)
4. *GPE* (Geopolitical entity)
5. *LANGUAGE*
6. *LAW*
7. *LOC* (Location)
8. *NORP* (Nationalities or Religious or Political Groups)
9. *ORG* (Organization)
10. *PERSON*
11. *PRODUCT*
12. *WORK_OF_ART*
13. *TIME*

The other class variable called `pipeline_dict` contains the different models used. It was mainly generated to pass the different pipeline models simpler.

The main method is `save_cc_from_article_body()`. It applies the pipeline models to the text body. This happens by calling the function `process_text()`, visible in listing 4.2. Hereafter, the labeled entities in the respective **reason** and **result** spans are extracted. If at least one entity in each span can be found, a new causal connection is created by calling `self.__new_causal_connection()`

Article_Collection

`Article` objects can be generated by an `Article_Collection`. This class holds several articles that belong to one company. Its main method is named `new_article_and_analysis()`. When running this, a new `Article` object is instantiated and the method `Article.save_cc_from_art` will be executed. Subsequently, the newly generated object will be saved under its ID as value in a dictionary called `articles`.

Possibly found causal connection in the article will be saved in the `Article_Collection` object as well, because it later is of less relevance, from which analyst report the causal connection was extracted, to connect companies with each other. Thus, in a subsequent step, each `Article_Collection` can be used to establish connections based on the causal connections between companies.

```
processed_docs = []
docs = senter(text)

for doc in docs.sents:

    doc_cat = text_cat_model(doc.text)
    if doc_cat.cats is not None:
        if doc_cat.cats["POS"] > 0.5:
            doc_span = span_cat_model(doc_cat)
            if len(doc_span.spans["sc"])==2:
                doc_ner = ner_model(doc_span)

                processed_docs.append(doc_span)
```

Listing 4.2: Main part of the ***`process_text()`*** *function, that applies the pipeline components on some text. Because the models are not necessarily the same, the* ***`nlp.Vocab`*** *might differ. Thus, it was not possible to create a single pipeline with the models as components. First, the* ***`senter()`*** *decomposes the text into its sentences. Next, the TextCat model runs over the sentences. In case that the likelihood to have a causality in the sentence is greater than* 0.5*, the SpanCat model searches for possible* ***reasons*** *and* ***results****. If two spans are found, the NER model searches for entities. The complete annotated sentence will than be appended to the list* ***`processed_docs`*** *and later returned.*

Collection_Storage

This is the top-level class. It holds several `Article_Collection` objects. The main mechanics are implemented in the methods `analyze_data()` and `connect_companies()`.

The first method takes a `pandas.DataFrame`. In this dataframe, all analyst reports and the respective metadata is saved row-wise. It will be iterated over it. If a new ticker symbol is found, a new `Article_Collection` object is instantiated. This consecutively executes its method `new_article_and_analysis()` to analyze the current analyst report.

Once that is done, the method `connect_companies()` can be run. This searches for connections between companies. Important to mention are two parameters. The first one is named `looseness`. It distinguishes, under which conditions a new connection is established. By now, it has three steps: 0, 1, 2. If 0, the found entities of Company A's **result** have to match exactly with the found entities of Company B's **reason**. If 1, at least one entity has to match in both spans. If 2, the entity's names are compared string-wise in order. In case that the names match with a certain percentage, a connection is generated. The necessary threshold score is the formerly named second important parameter, called `min_string_similarity`. It was mainly introduced, because the NER model sometimes extract entities that are actually the same, but are named slightly different, e.g. because of a genitive ”s”.

The found connection are added up and written into a `pandas.DataFrame` that acts as weighted adjacency matrix.

```
results = pcmci.run_pcmci(
    tau_max=7,
    pc_alpha=None,
    alpha_level=0.01,
    tau_min=0
    )
```

Listing 4.3: Execution of the PCMCI algorithm. A significane level of 0.01 was chosen. `tau_max` *refers to the maximum time lag. Accordingly does* `tau_min`. `Pc_alpha=None` *let the algorithm decides what the best significance level during the condition-selection step is.*

This class contains also a method to create and plot a graph. Additionally, the complete `Collection_Storage` can be saved to a file as pickle and load again from a file. This was implemented, because a complete analysis of the analyst reports took approximately a full hour.

Main Run

The several classes were tested manually. Logging and loading bars were included to make it more user-friendly and easier to debug. Where it seemed necessary, try/except blocks were used. Several complete runs were conducted with different `looseness` values. A resulting graph will be presented in section 4.4.

4.3.2 Quantitative Data

In the quantitative analysis, the R library lmtest [31] was used to perform Granger's causality test [23] as a baseline. For better results, the PCMCI algorithm was utilized. It is included in the Python library TIGRAMITE [49].

The Granger causality test was performed pair-wise, by iterating twice, in each direction once, over the different companies. $\alpha = 0.01$ was chosen. If the test showed significance, the causal relationship was noted in an adjacency matrix.

For analyzing the data with the PCMCI algorithm, all stock prices were concatenated as columns into a `pandas.DataFrame`. As independence test, partial correlation was chosen. Subsequently, the causal discovery algorithm was executed, visible in listing 4.3.

To eliminate the time-dependency in the resulting network, a method was implemented involving the iteration over different lags. In this method, undirected edges were transformed into two arcs, one from Company A to Company B and another from Company B to Company A. However, if a directed edge was present, such as from Company A to Company B, only a single arc from Company A to Company B was created.

The process of creating two directed edges when encountering an undirected edge can be considered inaccurate. However, this step was necessary in order to facilitate a comparison of the resulting networks using iGraph. iGraph only supports either directed or undirected networks, and does not allow for a combination of both. Therefore, by converting the undirected edges into two directed arcs, it was possible to ensure compatibility with iGraph and conduct the desired network analysis.

4.3.3 Combination

The combination of the retrieved networks was undertaken with iGraph. Though, this time Python was used. To account for the different industries the companies belong to, a Python dictionary was created, that holds different RGB values. Vertices of companies of the same industry were painted in a similar color. Additionally, a different node shape was introduced, to differentiate between the different countries of origin.

The similarity between the networks were calculated and subsequently plotted.

4.4 Results

In subsection Pipeline Models 4.4.1, the training results will be presented. After that in subsection 4.4.2, the qualitative derived graph will be illustrated. The quantitative networks will be shown in subsection 4.4.3. The chapter will end with the results obtained from the combination process in subsection 4.4.4

4.4.1 Pipeline Models

The results from the trained models are visible in table 4.2. The outcome corresponds mostly with the published results from [66]. Table 4.3 shows some examples extracted by the pipeline. Additionally, for the sample in the last row, three entities could be found. For the reason span, **apple** (*ORG*) was found, while for the result span, **iphone** *PRODUCT*) and **eve jobs** (*PERSON*) were returned. It is noteworthy, that these are just a few examples. Others include either none extracted span or just one. As already mentioned, those were not further processed. A wrongly inferred tuple of spans is the following:

- Original sample: JD.com's relentless focus on user experience, cost and efficiency has allowed us to continuously expand our user base while delivering profitable growth.
- Extracted reason: JD.com's relentless focus on user experience
- Extracted result: cost and efficiency
- True reason: JD.com's relentless focus on user experience, cost and efficiency
- True result: continuously expand our user base while delivering profitable growth

4.4.2 Qualitative Data

The text corpus was once analyzed with a `looseness = 0` and `looseness = 2` parameter. The qualitative graphs are presented in figure 4.6 and 4.7. The qualitative graph with `looseness = 0` contains 14 edges. The other one with `looseness = 02` comprises 58 edges. However, to reduce the different necessary comparisons, in the subsequent sections only the graph with `looseness = 2` will be discussed and further investigated. It will be referred to it as "qualitative graph".

The nodes with the most outgoing and incoming edges are visible in table 4.4.

Task	**Model**	**Recall**	**Precision**	**F1**
TextCat	RoBERTa-base	84.46	83.91	84.13
SpanCat	RoBERTa-large	54.00	63.80	54.00
	SpanBERT-large-cased	53.10	62.40	57.38
	XLNET-large-cased	54.85	62.00	**58.21**

Table 4.2: The results from the trained models. The SpanCat output was only considered right, when the spans exactly match.

Company	**Reason**	**Result**
Baidu	Baidu continues growing and expanding profitability	its stock price should increase significantly in the coming years
JD.com	the Chinese New Year approaches	the company will see increased demand
Microsoft	the economic outlook should improve	the Federal Reserve moderates its pace of hiking interest rates and corporations have more disposable income
Apple	Potential users will wait to buy an Apple product	the unaltered appearance of the iPhones, and because Eve Jobs posted a meme about it on her Instagram Stories

Table 4.3: Some examples extracted from the pipeline.

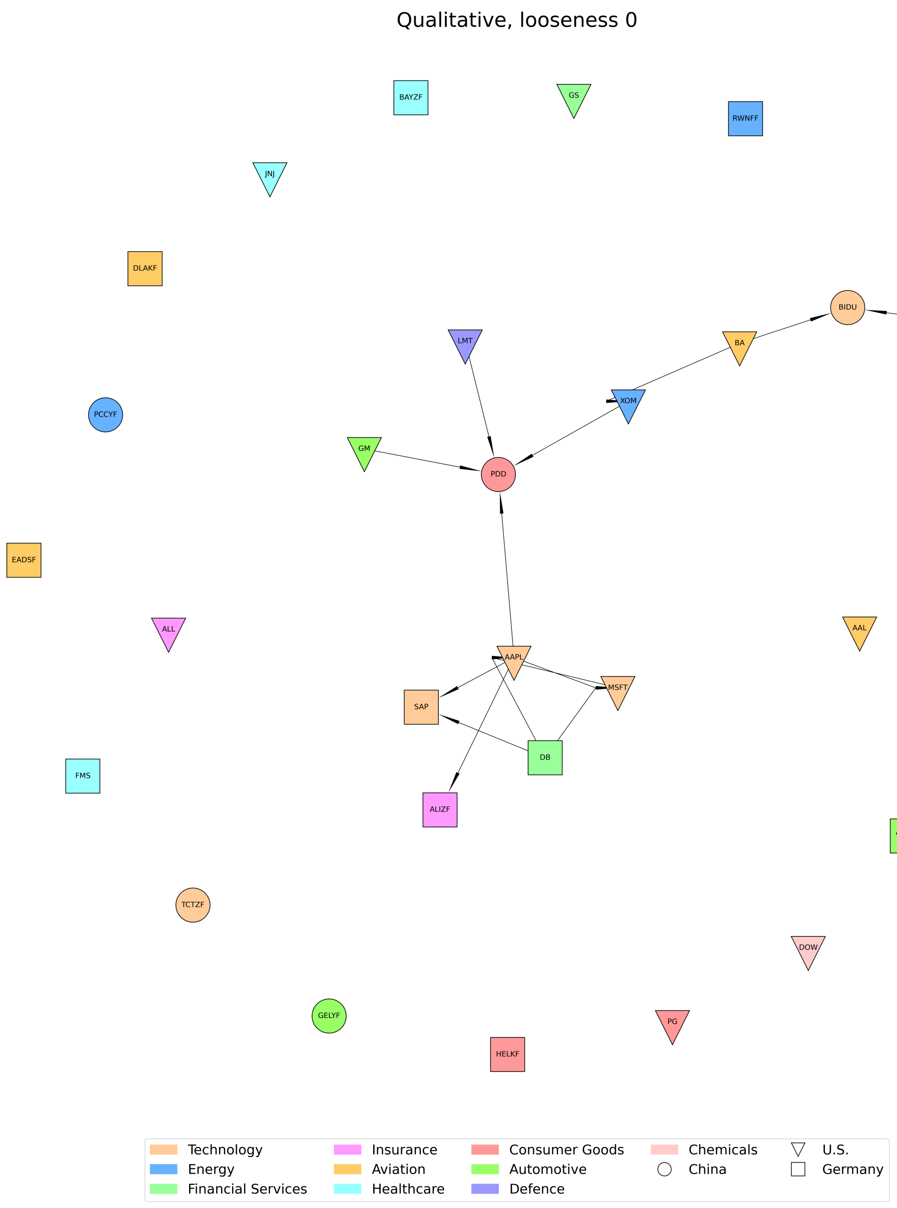


Figure 4.6: Found network from the causal relation extraction program. The `looseness` *parameter was set to 0.*

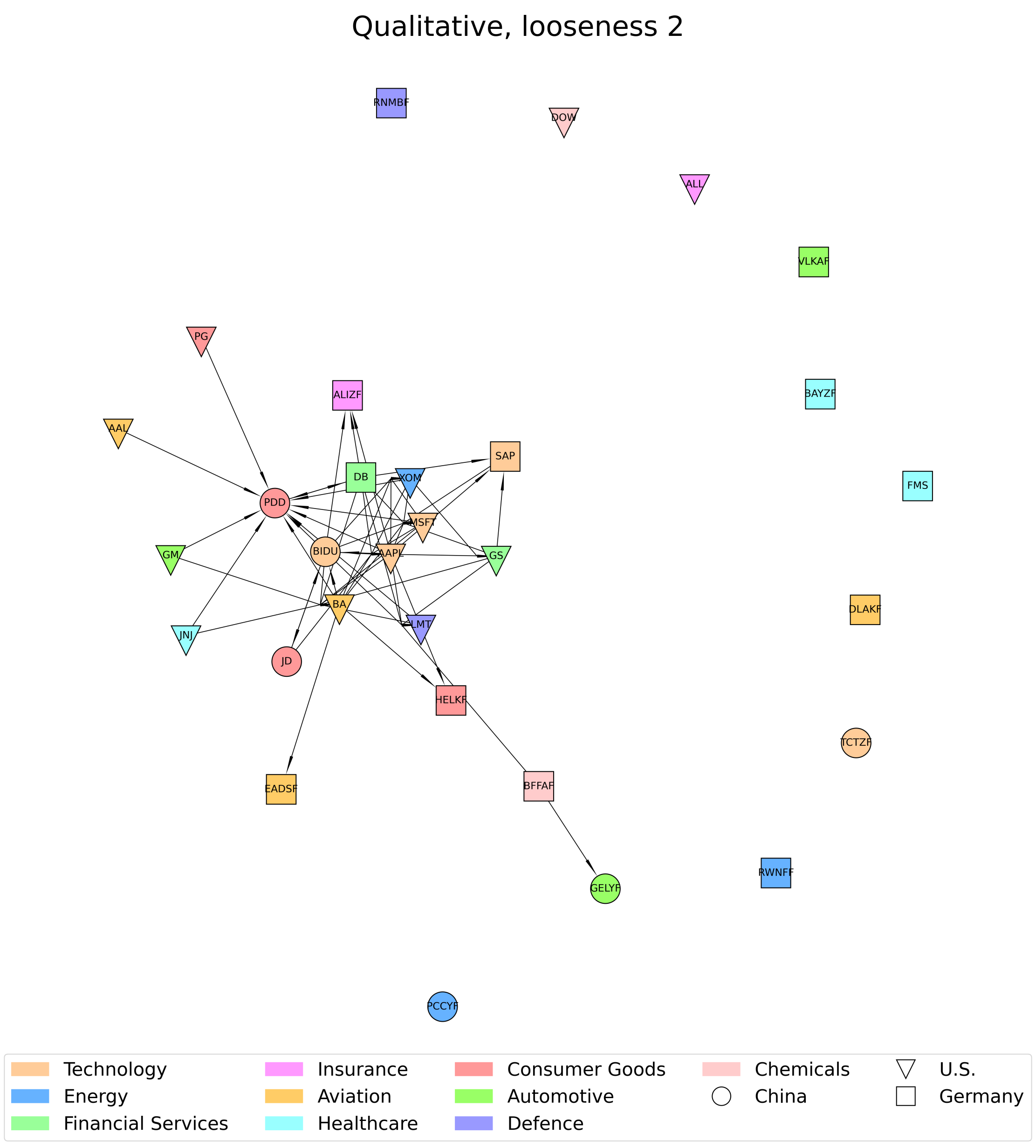


Figure 4.7: Found network from the causal relation extraction program. The `looseness` *parameter was set to 2*

Pos.	Top Outgoing	# Out	% Out	Top Incoming	# In	% In
1	AAPL	9	15.5	PDD	11	19.0
2	BIDU	8	13.8	BA	8	13.8
3	DB	7	12.1	MSF	7	12.1
4	BA	7	12.1	AAPL	7	12.1
5	GS	5	8.6	LMT	5	8.6

Table 4.4: Top outgoing and incoming nodes of the qualitative generated graph.

Pos.	Top Outgoing	# Out	% Out	Top Incoming	# In	% In
1	BAYZF	3	8.6	JNJ	6	17.1
2	BFFAF	3	8.6	BFFAF	3	8.6
3	DLAKF	3	8.6	DOW	3	8.6
4	HELKF	3	8.6	EADSF	3	8.6
5	JD	3	8.6	LMT	3	8.6

Table 4.5: Top outgoing and incoming nodes of the resulting graph from Granger causality test.

4.4.3 Quantitative Data

Granger

The baseline Granger causality test yielded the network visible in figure 4.8. A lag of $\tau = 7$ days and a significance of $\alpha = 0.01$ was investigated. The top outgoing and incoming nodes are listed in table 4.5.

PCMCI

In figure 4.9, the network yielded by the PCMCI algorithm is illustrated. The top outgoing and incoming nodes are presented in table 4.6. As with the Granger causality test, a lag of $\tau = 7$ and $\alpha = 0.01$ was chosen.

A comparison between Granger and PCMCI is found in the appendix.

4.4.4 Combination

The qualitative graph was compared with the graphs resulting from the Granger causality test and the PCMCI. The intersections are presented in the figures 4.10a

Pos.	Top Outgoing	# Out	% Out	Top Incoming	# In	% In
1	DOW	23	5.9	GS	21	5.4
2	GS	22	5.7	DOW	21	5.4
3	SAP	21	5.4	SAP	21	5.4
4	BA	19	4.9	MSFT	19	4.9
5	MSFT	19	4.9	AAL	19	4.9

Table 4.6: Top outgoing and incoming nodes of the PCMCI generated graph.

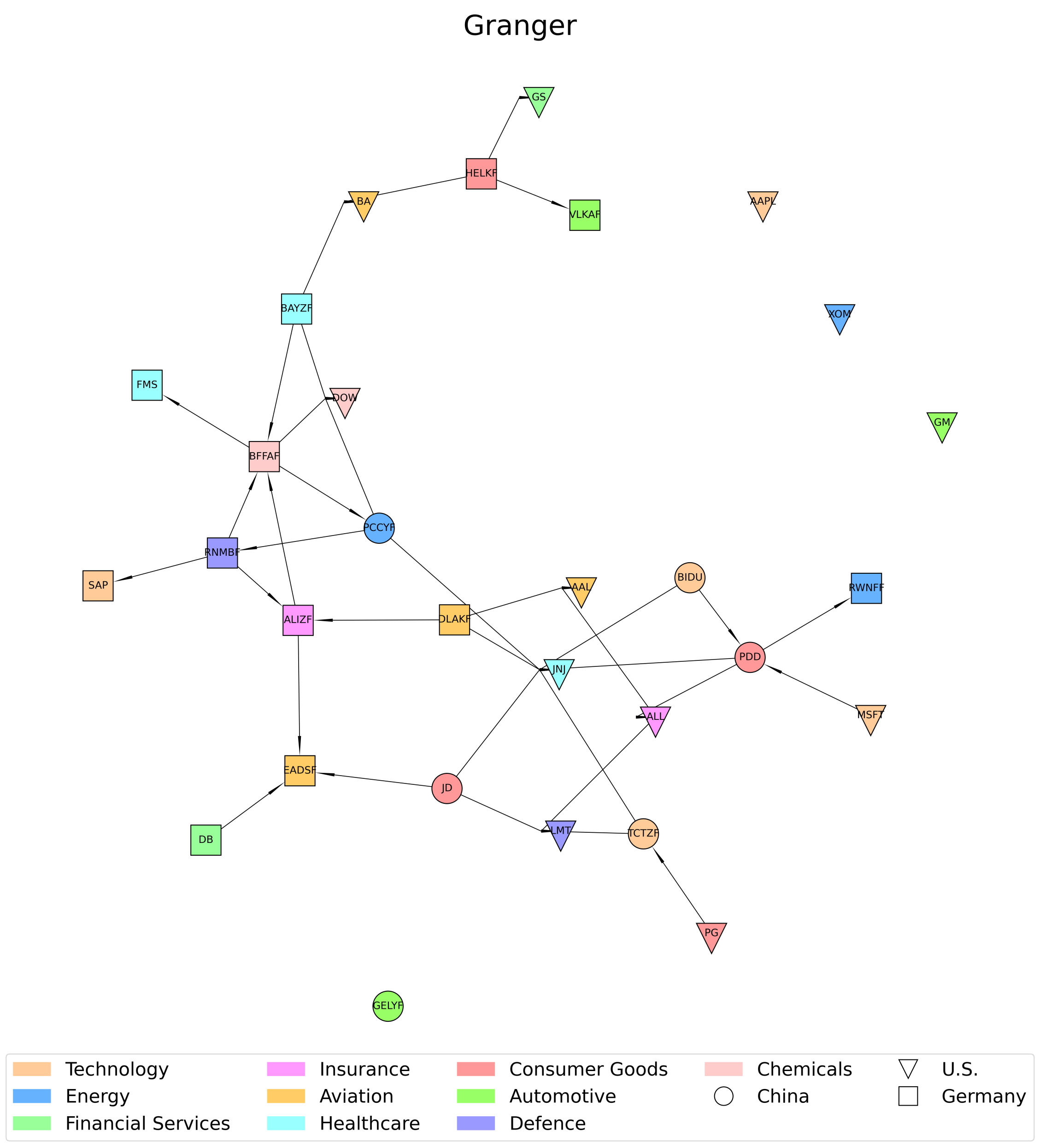


Figure 4.8: Network obtained by the Granger causality test with $\tau = 7$ and $\alpha = 0.01$. It comprises 35 edges.

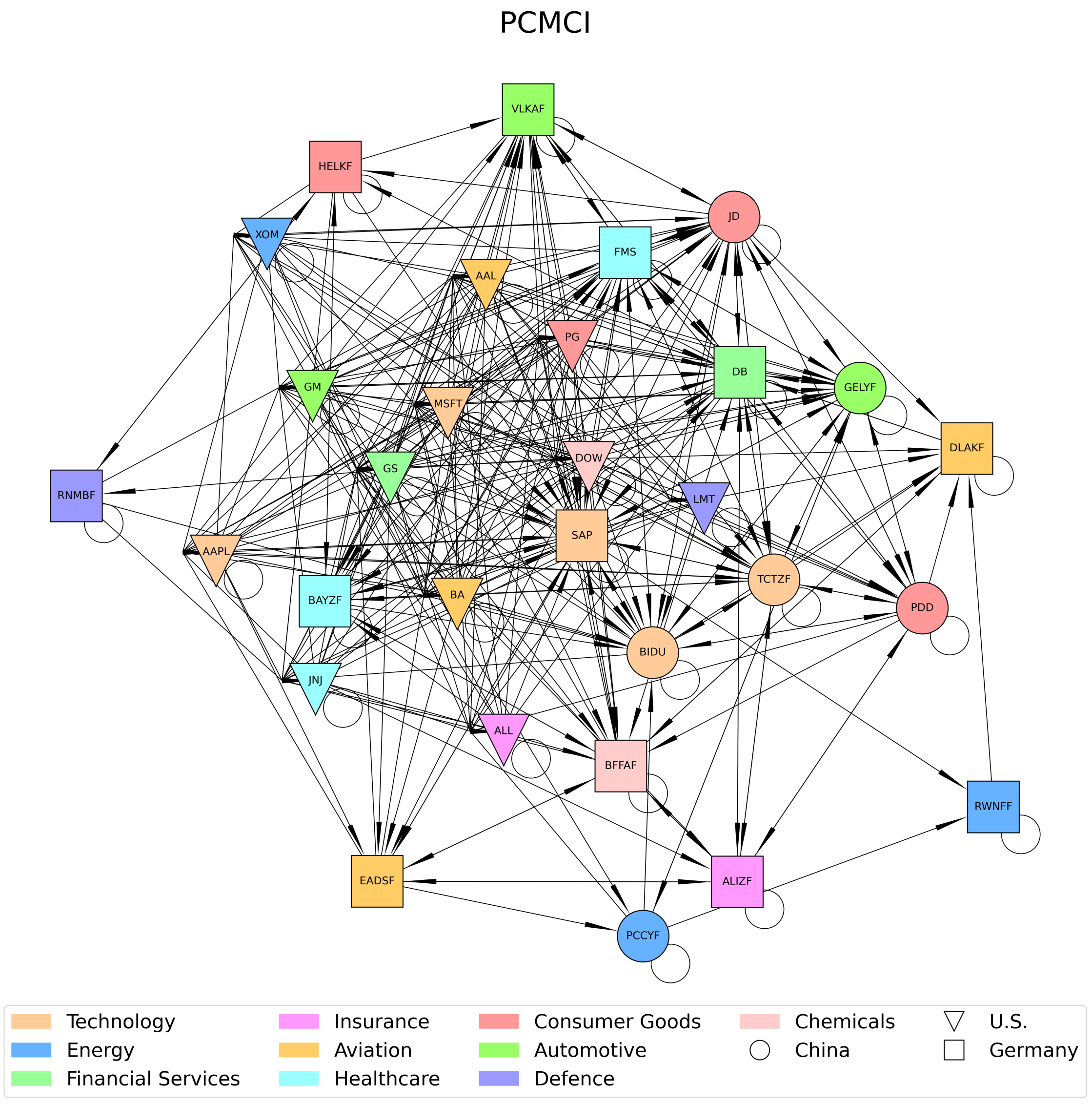


Figure 4.9: Network resulting from the PCMCI algorithm with $\tau = 7$ and $\alpha = 0.01$. It contains 389 edges.

and 4.10b. The graph shares 2 edges with the Granger caused graph, while it shares 34 edges with the PCMCI derived graph. In table 4.7 and 4.8, a comparison by country and industry sector is presented. The values are relative values, because countries and industries are unequally distributed.

∅ Edges Out and In per Country							
Country		**# (G)**	**% (G)**	**# (P)**	**% (P)**	**# (Q)**	**% (Q)**
USA	Out	.3	7.5	16.0	**41.2**	3.1	**54.4**
	In	1.5	36.6	15.9	**41.3**	2.7	**43.5**
Germany	Out	1.5	37.5	10.0	25.8	0.8	14.0
	In	1.9	**46.3**	10.4	27.0	0.8	12.9
China	Out	2.2	**55.0**	12.8	33.0	1.8	31.6
	In	.7	17.1	12.2	31.7	2.7	**43.5**

Table 4.7: Average out and in connections per country. Q: Granger, P: PCMCI, Q: Qualitative

4.5 Discussion

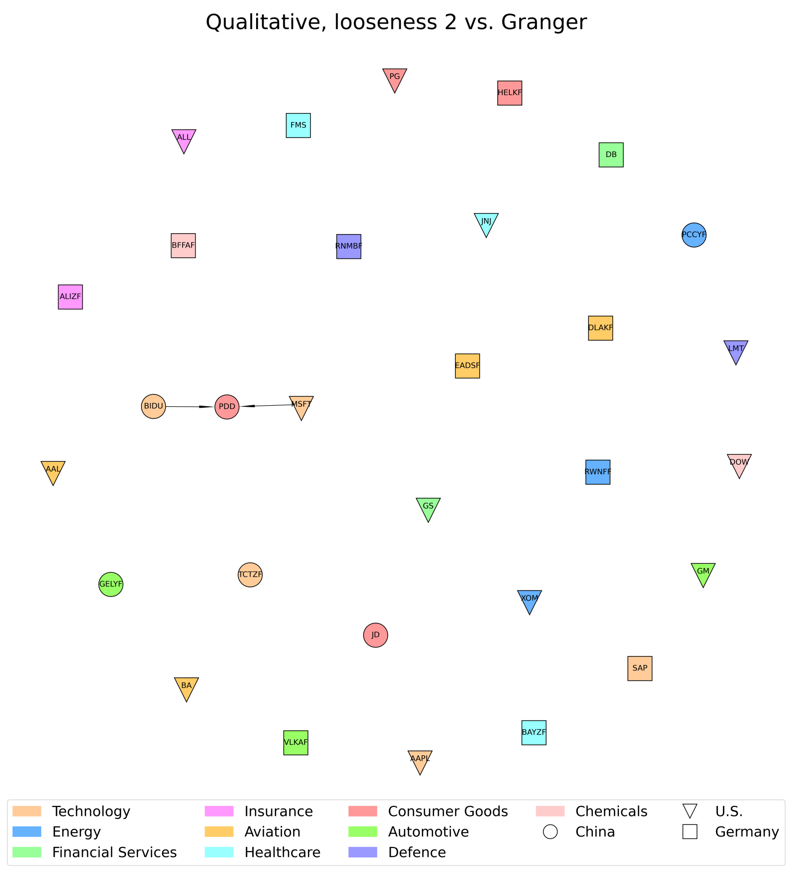
The discussion is presented in the following chapter. As a brief reminder, the goal of this thesis was to investigate, whether a common framework can be applied to combine causal structures derived from qualitative and quantitative data and what problems occur and which considerations are necessary. The chapter contains the section 4.5.1 Pipeline Models, where a discussion to the results of the fine-tuned models is presented. Afterwards, the qualitative and quantitative results are elaborated in sections 4.5.2 and 4.5.3 before the chapter ends with a discussion about the combination process and its result in section 4.5.4.

4.5.1 Pipeline Models

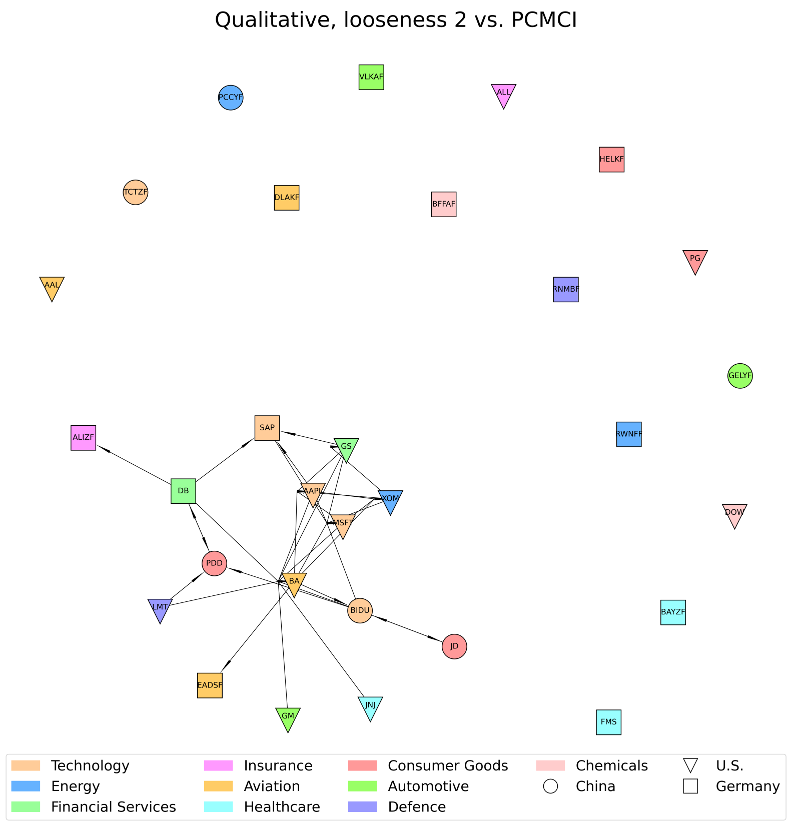
The results of the trained pipeline match with the ones yielded by [66]. Their TextCat RoBERTa-base model achieved 84.31 F1 on test data. Best results on TextCat delivered a RoBERTa-large model with 84.64 F1. The model fine-tuned in the context of this work, a RoBERTa-base model achieved 84.13 F1, which is just slightly worse.

The best SpanCat model from [66] was SpanBERT-large with 60.26 F1 on exact match. A possible reason for their better result might be, that they did not have to dismiss some samples because of faulty annotations. They did not use SpaCy, but downloaded the pretrained models directly from HuggingFace. So the problematic annotations might have stayed unnoticed. Their RoBERTa-large model achieved 56.77 F1. A comparison with the XLNET model was not possible, as they did not decide to fine-tune it. Overall are their results similar to the ones yielded by this work.

When looking at the brief excerpt of the labeled span, the reader can see reasonable causal relations. However, with except to the last one, none of them could be used, because no entities were recognized in a subsequent step.



(a) Intersections between the qualitative derived graph and the graph generated by the Granger causality test.



(b) Intersections between the qualitative derived graph and the graph generated by the PCMCI algorithm.

Figure 4.10: Intersections between the qualitative graph and the quantitative graphs.

∅ Edges Out and In per Industry												
Industry	**Granger**				**PCMCI**				**Qualitative, l2**			
	# O	# I	% O	% I	# O	# I	% O	% I	# O	# I	% O	% I
Technology	1.0	.4	8.5	3.1	16.2	16.6	12.6	13.0	4.4	4.0	18.0	**24.4**
Energy	1.0	.7	8.5	5.4	5.7	6.3	4.4	4.9	1.3	1.3	5.3	7.9
Financial Services	.5	.5	4.2	3.8	20.0	19.0	**15.6**	**14.9**	6.0	1.0	24.6	6.1
Insurance	2.0	1.5	16.9	11.5	8.0	9.5	6.2	7.4	.0	1.5	.0	9.1
Aviation	.8	1.8	6.8	13.8	11.8	13.3	9.2	10.4	2.0	2.3	8.2	14.0
Healthcare	1.0	2.3	8.5	17.7	13.3	13.7	10.4	10.7	7.0	.0	**28.7**	.0
Consumer Goods	2.5	.5	**21.2**	3.8	13.8	11.5	10.8	9.0	1.0	3.5	4.1	21.3
Automotive	.0	.3	.0	2.3	14.3	14.7	11.2	11.5	.7	.3	2.9	1.8
Defence	1.5	2.0	12.7	15.4	8.0	6.0	6.2	4.7	1.0	2.5	4.1	15.2
Chemicals	1.5	3.0	12.7	**23.1**	17.0	17.0	13.3	13.3	1.0	.0	4.1	.0

Table 4.8: Average out and in connections per industry. Q: Granger, P: PCMCI, Q: Qualitative

Limitations & Improvements

In general, a pipeline approach imposes problems, in particular because of error propagation. When the TextCat model decides in only 84% correctly and in a subsequent step, only 56% of the spans are labeled correctly, the overall accuracy drops significantly. Joint extraction approaches yield clearly better results [57].

When it comes to the used dataset, the reduced amount of training samples could be improved due to faulty annotations. Despite that, multi-causal samples should be included, since they make up for roughly 14% of the dataset of [66].

A way to improve results could be to change the underlying models and increase the parameter size. The largest models used in the context of this thesis, next to GPT-3.5, are XLNET-large with approximately 340 million parameters [46] and RoBERTa-large with 355 million parameters [46]. These are to my best knowledge, the largest existing encoder-only models. Exchanging these with another model, potentially a decoder-only autoregressive model like GPT-3, GPT-4 or PaLM could improve the results.

Causal extraction datasets are still sparse, as it is only a sub-topic in the relation extraction research. This problem could be overcome with data augmentation. [15] created a method called AugGPT on the basis of ChatGPT. It shows some problems with very specific topics, like in the medical context. Yet, for causal relation extraction, it might be possible to utilize that.

4.5.2 Qualitative Data

One third of the nodes stood unconnected. In table 4.9, the ticker symbols and the number of analyzed articles are presented. These reports were the only published in the respective time span. The sparse amount of these companies might be a reason, why there is no connection found. However, there are in fact companies with limited reports, like Allianz (ALIZF, 5), Allstate (ALL, 4) or SAP (8), where connections are found. So this is not necessarily a reason.

Unconnected Nodes	# Analysis Reports
RNMBF (Rheinmetall)	4
DOW	18
ALL (Allstate)	4
VLKAF (Volkswagen)	10
BAYZF (Bayer)	5
FMS (Fresenius)	2
DLKAF (Deutsche Lufthansa)	5
TCTZF (Tencent)	8
RWNFF (RWE)	2
PCCYF (PetroChina)	4

Table 4.9: Unconnected nodes of the qualitative graph and the number of analyzed reports.

When looking at the top outgoing and incoming nodes (table 4.4), Apple is represented in both categories. A possible reason for that is, that there are 229 reports about Apple in the dataset. However, PDD is strongly represented as well, though here only 13 reports were analyzed.

Limitations & Improvements

(1) Analyzing a more evenly distributed collection of analyst reports would have been beneficial. This comes hand in hand with utilizing a more professional source for analyst reports. It is likely, that the style and the structure that those reports are written in are more similar. This would improve the comparability between companies and the respective established connections between these. However, analyzing crowd-sourced reports does have its advantages as well. [13] investigated, whether a connection between Seeking Alpha articles and abnormal stock returns and earnings surprises exist. They did this, by measuring certain negative words in articles retrieved from Seeking Alpha and using these as coefficients in a multivariate regression. Published articles seem to strongly predict future stock returns and earnings surprises. Even when controlled for effects from more professional sources and news media.

(2) Improvements should be made, when it comes to establishing connections between companies. In this thesis, these were made, when entities from a **result** span of Company A match with entities from a **reason** span of Company B. Though, whether there is a real causal connection is questionable. The problem is, that an ontology is missing, that levels the extracted spans to some common ground. For example Wikipedia could be utilized to structure the extracted spans. Probably even better suiting would be DBpedia [30] because of a tighter structure, that the knowledge exists in, compared to Wikipedia. DBpedia offers currently 768 ontology classes, that could be leveraged. Alternatively, entities could be used to traverse the knowledge graph directly. By implementing that, possible connections could be established, by classifying extracted entities via DBpedia before grounding these to companies. For example **iPhone 8 (Device)** $\xleftarrow{\text{brand}}$ **Apple_Inc. (Public_Company)** could be traced back. It would probably improve the results, if only companies would be compared, after other entities were grounded to them, as it would reduce the noise.

(3) Another possibility would be to self-construct a causal knowledge base. It could be done in the following way: In a first step, the semantic similarity between spans could be measured, for example with a Bag-of-Words model [62] or by directly accessing the embedding vectors and measure the distance between these vectors. Hereafter, a knowledge base could be constructed by clustering the similar text samples. Subsequently, new spans could be grounded to the nodes of this knowledge base, by calculating the highest semantic similarity.

(4) By now, the pipeline is only able to find inter-sentential causality with one cause-effect pair per sentence. This is a bottleneck, as it does not account for the variety of different causal relations.

(5) The amount of found connections should be utilized as well. The current state is, that each causal connection is added up in a matrix. This could be used to estimate the strength between companies. One could even go further and include the probability of the pipeline models in the equation. The strength of the connections could be expressed by different colors or different sizes of edges in the plots.

4.5.3 Quantitative Data

Both networks, Granger and PCMCI, were created with a lag of $\tau = 7$ and $\alpha = 0.01$.

Explaining connections that appear or do not appear from the Granger causality is difficult. There are only 35 edges present. A pattern cannot be identified. Noticeable is, that three U.S. companies are not connected and only one Chinese and no German.

The graph inferred by PCMCI contains 389 edges. Many technology companies are in the midst of the graph well-connected. However, the top outgoing node is Dow Inc., which comes unexpected, that a chemical company has so much influence. The same goes for Boeing. However, Goldman Sachs was expected, as financial institutes play an important role in an economy. This matches with the average in and out connections per industry in table 4.8. Companies from the defence, insurance and energy sector influence and are influenced the least.

RWE, Rheinmetall and Henkel do not play a central role according to PCMCI. In fact, most of the outer nodes are German companies (Rheinmetall, Airbus, Allianz, RWE, Lufthansa, Volkswagen, Henkel), despite Exxon Mobile from the U.S. and PetroChina from China, while many companies in the inner circle are U.S. companies. This is noticeable from the table 4.7 as well. U.S. based companies seem to influence and are influenced the most. However, a multiple regression would be necessary to determine, whether the in- and outdegree of the node depends more on the sector or the origin of the company.

Limitations & Improvements

(1) The graph created by Granger looks much sparser, compared to the PCMCI algorithm. A reason for might be, that Granger actually assumes stationary linear systems. This is not given, when applied to financial data. Therefore, nonlinear dependencies are not detected. Another reason for so only 35 edges is, that the algorithm is implemented pairwise and does not account for possible confounders or indirect links [54, 50].

(2) PCMCI found 389 edges. The same necessary assumptions as for the Granger Causality were applied here, because the partial correlation independence test was used, which only accounts for linear dependencies. [48] states, that through the MCI test, a limited amount of false positive connections are inferred. However, this does not explain the vast difference to the Granger causality. It is rather, because undirected edges were converted into two directed ones, e.g. $A - B \Rightarrow A \leftarrow B \wedge A \rightarrow B$. This is of course inaccurate. Yet, it was the only solution to compare both graphs with iGraph, as iGraph cannot handle mixed graphs. Originally, 312 undirected connections were detected and 101 directed. Some of them were merged together, because they belong to the same pair of companies, but account for different lags τ.

(3) PCMCI could be improved by using Latent-PCMCI (LPCMCI) [21]. However, this is computational extensive. Another option to account for unmeasured confounder would be to change the investigated companies. One could for example choose all companies from the SDAX index. These are rather small in terms of revenue and market capitalization and might not be too dependent from other international companies. This could reduce the amount of confounders.

(4) It would be useful to include the strength of the connections, when the graph is constructed. The different strength of connections could be expressed by adjusting the width or color of the edges.

4.5.4 Combination

Granger causality combined with the qualitative graph show only two connections. Both are not explainable.

PCMCI and the qualitative graph share 34 intersections. The arc from Boeing to Airbus seems logic. Both financial service companies, Deutsche Bank and Goldman Sachs, are represented. Understandable are the connections between SAP, Apple, Microsoft and Baidu, since all belong to the technology sector. When considering the countries of origin, the triplet Allianz, Deutsche Bank and SAP seem explainable. JD.com, Baidu and PDD share connections as well. This is understandable. The same goes for the cluster Goldman Sachs, Apple, Microsoft, Exxon Mobile, Boeing, General Motors, Johnson & Johnson and Lockheed Martin. Noticeable is, that Allstate, Dow, P&G and American Airlines do not belong to them. For Allstate and Dow were no qualitative connections found. P&G and American Airlines are qualitatively connected to PDD. These connections could not be confirmed with the PCMCI algorithm.

However, The connection between U.S. defence corporation Lockheed Martin and Chinese consumer goods company PDD appears unusual.

The companies Lufthansa, Tencent, Volkswagen, Allstate, Rheinmetall, RWE, Bayer Dow and Fresenius were already unconnected in the qualitative graph. Henkel, P&G, Geely, BASF and American Airlines were not. However, the PCMCI algorithm does not share any common edge with the qualitative graph.

Overall, 15 of 30 nodes have a degree greater zero.

Limitations & Improvements

(1) The main reason for companies that have a degree of zero is, that the qualitative data is not equally distributed.

(2) Including a measure of strength between the companies would be useful. For the qualitative data, a combination of the amount of connection and the likelihood from the pipeline models could be used. For the qualitative data, the strengths calculated from the PCMCI algorithm could be utilized. They both could be standardized, compared and merged.

(3) By now, the qualitative data does not utilize the date the article was published. Yet, this could be some valuable information to estimate whether a condition is possible or not. Implementing this would not be a challenge. The question is rather how reliable this is, as there are analyst reports that talk about the past or the future. It would be necessary to determine this. A similar question is, how to distinguish between hypothetical scenarios presented in these articles and facts. Event extraction is necessary to account for these concerns.

(4) A knowledge base and an ontology should be implemented. By grounding the extracted entities and classify them by an ontology, noise would be reduced. A knowledge base could be used to infer more connections between different types of extracted entities. As already stated in the qualitative subsection, DBpedia could be interesting.

(5) Using a different graph construction library would be beneficial, as iGraph only allows for direct or either undirected edges. NetworkX is restricted in this regard as well. Both are sometimes difficult to adjust, so that the nodes do not overlap, and the labels are well readable. bnlearn does allow for mixed graphs. Yet, it is required that the networks have the same node set. A method like `intersection` from the iGraph package does not exist in the bnlearn library. Gephi was used as well, to create graphs for the presentation. It allows mixed graphs and interactively adjusting the position of the nodes was helpful. iGraph offers a direct connection to Gephi. How well Gephi can be automized is unclear. Both was not tested.

5. Conclusion & Outlook

5.1 Conclusion

In the context of this thesis, several approaches could be investigated, on how to combine quantitative and qualitative causal knowledge.

The research conducted in the context of the flow cytometry dataset shows, that training data for relation extraction is still scarce. In particular for domain-specific tasks such as biochemistry. The few-shot capabilities from GPT-3.5 resolved this issue. However, the model extracted a substantial amount of noise at the same time. A powerful ontology is necessary to ground these entities and make them comparable. An adjustable refinement grade is necessary, to match with the refinement grade of the true-graph. When manually annotated to overcome the issues with the ontology, 7 out of 17 arcs could be correctly inferred. This could increase to 9 out of 17 arcs, once all connections were found and annotated in the abstracts. On the quantitative side, the utilized heuristic Tabu search algorithm proved to find more true-positive arcs, than the originally used heuristic Annealing search algorithm and the applied causal discovery algorithm FCI+ baseline. Yet, it is evident that interventions are necessary to fully reconstruct a causal graph purely from data, as Tabu search found 5 false-positive and misses 13 false-negative arcs. The leveraged graph libraries iGraph and bnlearn have the weakness, that former does not enable to merge directed and undirected graphs. Latter does not have this weakness, however, the comparison of graphs with a different node set is not possible.

In terms of the self-collected financial dataset, a pipeline with four fine-tuned language models was developed. To apply this pipeline, a program named CausalExtraction_Linker was constructed, that analyzes a text corpus and consecutively infers causal connections between companies. Yet, it is questionable whether these connections can be understood as causal. The detection and extraction of events would be necessary to meet these shortcomings. Additionally, the application of a knowledge base like DBPedia would have been advantageous. Extracted entities could have been grounded, and the graph could have been utilized to draw more meaningful connections compared to the current string comparison. A reason for unconnected nodes in the qualitative graph is, that the dataset text corpus is unequally distributed.

Quantitatively, the PCMCI algorithm was employed to detect causal connections from stock prices. It found significantly more connections than the baseline pairwise Granger Causality test. However, nonlinear dependencies and latent variables were not considered. Both results combined show a graph that is partially well-explainable. Yet, unusual connections were inferred as well. Regrettably, due to time-constraints it was not possible to utilize the inferred strengths connections and compare them.

Similar issues as in the first dataset occurred, in terms of visualizing the evidences. Gephi could be used. However, it is unclear how well it can be automized.

To summarize: A common framework, that is applicable to different fields could be developed in this thesis. Yet, components of this framework have to be improved. In particular, a knowledge base with an underlying ontology should be utilized. Additionally, it is necessary to extract events, rather than relations. Leveraging a more powerful causal discovery algorithm that accounts for latent variables would be beneficial as well. Subsequently, using a joint extraction model than a pipeline could improve results furthermore.

5.2 Outlook

With the recent emergence of ChatGPT, AI, and natural language processing in particular, has become the focus of people around the world. This results in the faster development of language models with new capabilities. It is questionable, whether as much data annotation will be necessary in the future, because the development of data augmentation proceeds with these new language models. For tasks where results do not have to be as accurate, few-shot capabilities might deliver satisfying results, making the fine-tuning process completely obsolete. This development favors the causal extraction task and is beneficial for the more challenging event extraction task.

The field of Causality invented several new techniques in the last decades. It is likely that this development proceeds similarly. It is crucial to understand causes and effects in large scaled system, as in meteorology, finance, biochemistry or sociology. In situations where interventions were unethical or infeasible, deriving causal connections from purely observational data is important.

Combining both fields could improve results furthermore. To my best knowledge, it exists only few works that try to utilize both sources yet. There is a lot of potential in this area.

A. Appendix

In section A.1, a more detailed description of related work is provided. Afterwards, an analysis of interventional flow cytometry data is shown and explained in section A.2. Additionally, plots that show not similarities, but differences of the quantitative findings compared with the true-graph are provided. In context of the finance dataset, a comparison between the Granger causality and PCMCI algorithm is shown and discussed in section A.3. Additionally, intersections of graphs inferred from the qualitative data, but with difference `looseness` parameters are presented.

A.1 Related Work in more Detail

A.1.1 World Modelers

World Modelers [29] is a program developed by the Defense Advanced Research Projects Agency (DARPA). Its goal is to unify qualitative causal analysis with quantitative models to provide a better understanding of current crises to analysts. To accomplish that, they built a pipeline consisting of several subsystems that will be explained in the following.

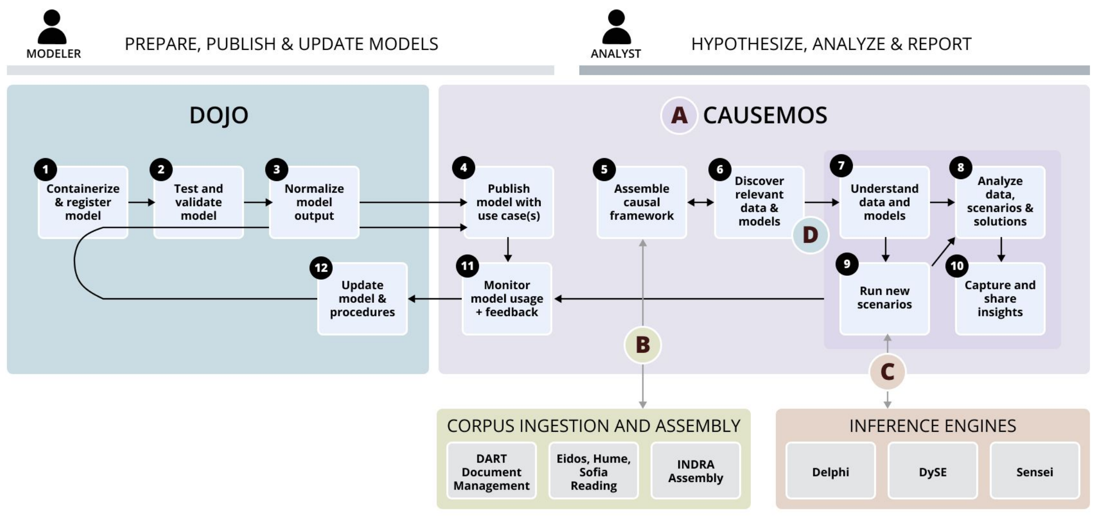

MODELER
PREPARE, PUBLISH & UPDATE MODELS
ANALYST
HYPOTHESIZE, ANALYZE & REPORT
DOJO
A CAUSEMOS
1 Containerize & register model
2 Test and validate model
3 Normalize model output
4 Publish model with use case(s)
5 Assemble causal framework
6 Discover relevant data & models
7 Understand data and models
8 Analyze data, scenarios & solutions
9 Run new scenarios
10 Capture and share insights
11 Monitor model usage + feedback
12 Update model & procedures
B
C
D
CORPUS INGESTION AND ASSEMBLY
DART Document Management
Eidos, Hume, Sofia Reading
INDRA Assembly
INFERENCE ENGINES
Delphi
DySE
Sensei

Figure A.1: World Modeler with its subsystems

CauseMos is the main part of the system. It offers a human-machine interface (HMI) with several workflows to make use of qualitative and quantitative data. It is possible for the user to build causal graphs which can be augmented by uploaded literature, that is scanned for further evidence. One is also able to composite an index based on the data and plan and simulate possible intervention.

The extraction of causal relations, events and statements from text is done by several subsystems. The documents are first passed to the Data Analytics and Reasoning Toolkit (DART). Here, metadata and text is extracted and passed to the readers EIDOS, HUME and SOFIA.

EIDOS extracts entities, arguments and events before grounding them to an ontology. It does this by a rule-based system called Odin information extraction framework. HUME is similar. It can also extract causal relations and events from text. Grounding extractions to an ontology is possible as well. Unfortunately, it is not further described on what base the system works. SOFIA does basically the same even though there is an important difference. SOFIA has an internal ontology which is not completely coherent with the ontology of World Modeler. As a reason, the authors mention that the development of SOFIA stopped, before the World Modeler finished. Thus, the use of SOFIA in the integrated system must be made carefully.

The assembly of the extracted information is undertaken by INDRA World. It can find relationships between statements and grounds them to an ontology. Equal statements are merged, and the evidence is combined. Statements of the same kind, but with a different generalization or specification degree are structured in a so-called refinement graph. INDRA World is also able to assign a "belief" score between 0 and 1, depending on the found support for a given statement. This all works incrementally. Adding more documents to the system lets INDRA World calculate an "assembly delta", which shows the differences generated in terms of the statements, connections and belief scores. It has to be enhanced, that the resulting graph is based on qualitative data.

Using quantitative data is done by the subsystem Dojo. Analysts can register their own model and have the possibility of adding parameters and metadata to it. Worthwhile mentioning is, that models should have a geospatial or temporal dimension.

After registration, CauseMos utilizes these models by automatically mapping available data to concepts of a causal analysis graph. The result is a semi-quantitative computational model, which can be used to analyze scenarios and interventions.

This is done in the last of the subsystems; the inference engines. The first one is called Delphi. It can model interactions between concepts linked by causal relations. As a base, it needs a causal analysis graph. When the strength of each node in the network is known, Delphi is able to use this as a starting point for the training of a probabilistic forecasting model. It is also possible for the analyst to specify default, minimum, maximum edge values or even freeze some.

Second, there is DySE, which stands for Dynamic Systems Explanation. It is a framework for the study of dynamics in a system. The DySE simulator is capable of visualizing the influence a concept has in its graph and how the change propagates through the network. DySE's sensitivity and path analysis is designed to quantify the influence of concepts on each other.

Third, World Modeler supports the use of Sensei. With this modeling engine, one can estimate uncertainty in multivariate time series data. Once again, the base is a causal analysis graph. Each node is modeled as a univariate time series. As a result, the engine has the power to estimate trend and seasonality of the time series data. This is undertaken in a step-wise manner. Mathematically, parameters and uncertainty is assessed by using Maximum A Posterior (MAP), whilst the foundation of the univariate time series is the Damped Local Trend (DLT) model. [29]

A.1.2 Explaining Causes from Text

The notion driving the work of Kang et al. [34] is to explain the cause of an event in time series data from text. The target time series data used in their paper is stock data and polling data from the 2012 US presidential election. They utilize the day-to-day popularity and sentiments of N-grams of tweets, blogs and news articles between 2008 and 2013 to calculate time series data. The Granger causality score between time series data generated from N-grams and the target time series is calculated to determine the best k textual features.

To build a chain of causation to explain the event further, they create a knowledge base graph, called CGraph, where edges have a direction and show the causality between entities. Six years of tweets and New York Times articles from 1989 to 2007 are used as basis. Due to low recall, the causal graph is augmented with the help of the external knowledge base Freebase. To find a useful path between the target and source node, an algorithm is used that searches backwards from the effect to possible causes. This is done while the Granger causality score with the target node is higher than a threshold.

Additionally, to help with the lexical strictness of language, they train a neural network, which they call neural reasoner. It consists of two LSTM layers and a multi-layer perceptron in between. The first layer acts as encoder and takes a causal phrase, while the decoder (second layer) takes the corresponding effectual phrase. The idea of the multi-layer perceptron is to attend the cause to the effect by taking different types of relations from FrameNet. The neural reasoner is trained by traversing CGraph, after it were initialized by word embeddings from word2vec.

The model is tested with several tasks. One of them is forecasting stock and polling data. It is evident, that the root-mean-squared error (RMSE) is lower, when textual features are considered in addition to the target's past time series. Another test conducted is, how well the neural reasoner works. This is done by comparing the predicted causes or effects with the CGraph using the BLEU metric. The neural reasoner with its relation attention outperforms a basic sequence-two-sequence (Se2Seq) model trained on the CGraph and a Seq2Seq with word embedding as baselines. The generation of explanations is evaluated as well. As there is no quantitative evaluation measure for this task, humans had to annotate these explanations. As baseline, a symbolic graph traverse algorithm is used. Here, the neural reasoner demonstrates superior output compared to the baseline. [34]

A.2 Flow Cytometry Dataset

[51] conducted interventions to retrieve the causal structure of the molecules. This section shows the application of the Tabu search algorithm on this dataset, to retrieve a DAG from it.

A.2.1 Analysis of the Interventional Flow Cytometry Dataset

The interventional dataset is already discretized to three levels: 1 for LOW, 2 MEDIUM and 3 for HIGH. It comprises 5400 samples. The INT (interventional) column shows, which of the proteins was either inhibited or stimulated (see table 3.2).

For the interventional dataset, the data was transformed into a list, where each sample is grouped based on the molecule that was either inhibited or activated. This list was passed again to a `tabu()` search function, similar to the one in listing 3.3. As before, with `custom.strength()` the strength of the relationships was measured and subsequently only the significant edges were chosen with `averaged.network()`.

In comparison to the true-graph, the resulting network exhibits 17 TP edges, 8 FP edges, and 0 FN edges in the skeleton. There is no change, when considering the directions.

The graph, reasoned from interventional data yields the best results. According to [52], the 8 FP arcs were later dismissed by [51], due to a comparable low significance. This difference is probably caused by the fact, that [51] used a heuristic simulated annealing search, in contrast to the here used Tabu search algorithm. This finding strengthens the requirement to undertake interventions for the investigation of causal structures.

A.2.2 Differences Qualitative Graph and True-Graph

Differences mean, that connections were included, that exist in the true-graph, but not in the qualitative/ qualitative simplified graph. Thus, directed edges which were not found by text-davinci-003.

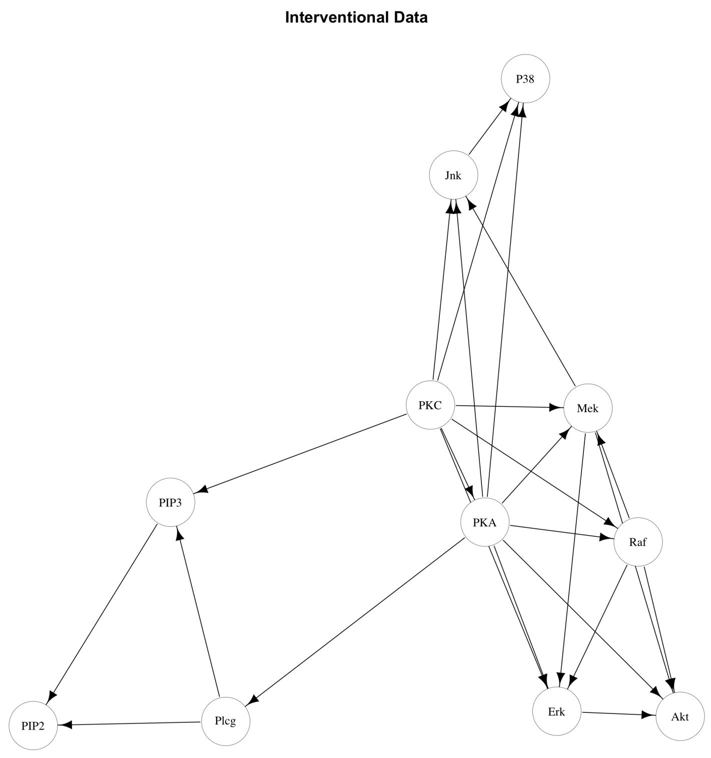


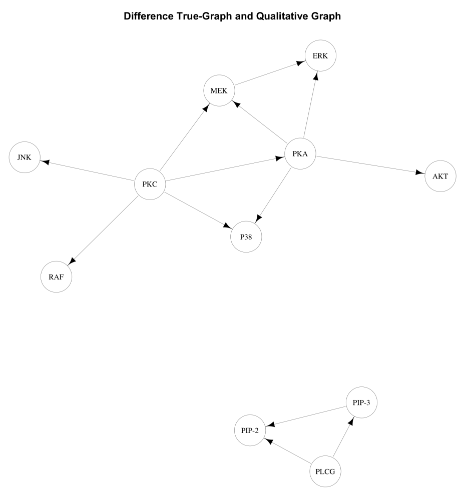
Figure A.2: Graph resulting from Tabu search, run on the interventional data.

TP/FP/FN	**FCI+**	**Observational**	**Interventional**
True-Graph	8/0/9	9/2/8	17/8/0
FCI+		8/3/0	8/17/0
Observational			10/15/1

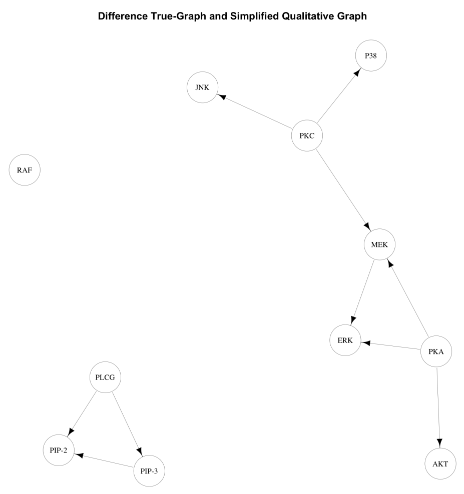
Table A.1: Comparison of the skeletons. Networks row-wise are taken as true, while networks column-wise are than compared, resulting in true-positive, false-positive and false-negative edges.

A.3 Financial Dataset

A.4 Additional intersections between graphs



(a) Differences of true-graph and qualitative graph.



(b) Differences of the true-graph with the manually simplified qualitative graph.

Figure A.3: Comparison of differences between the true-graph and the qualitative and qualitative simplified graphs.

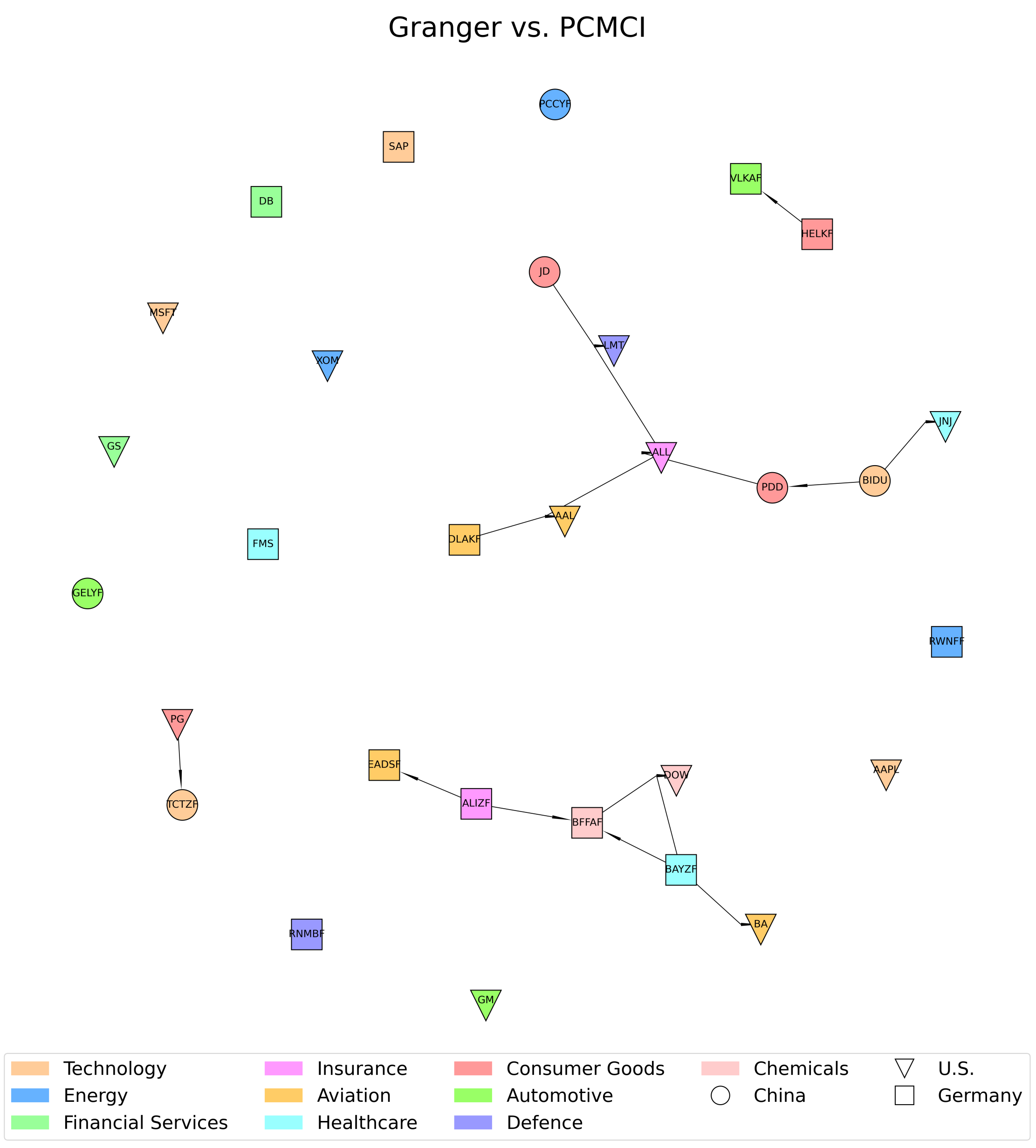


Figure A.4: Intersections between the Granger causality test constructed graph and the PCMCI graph. 15 edges are present.

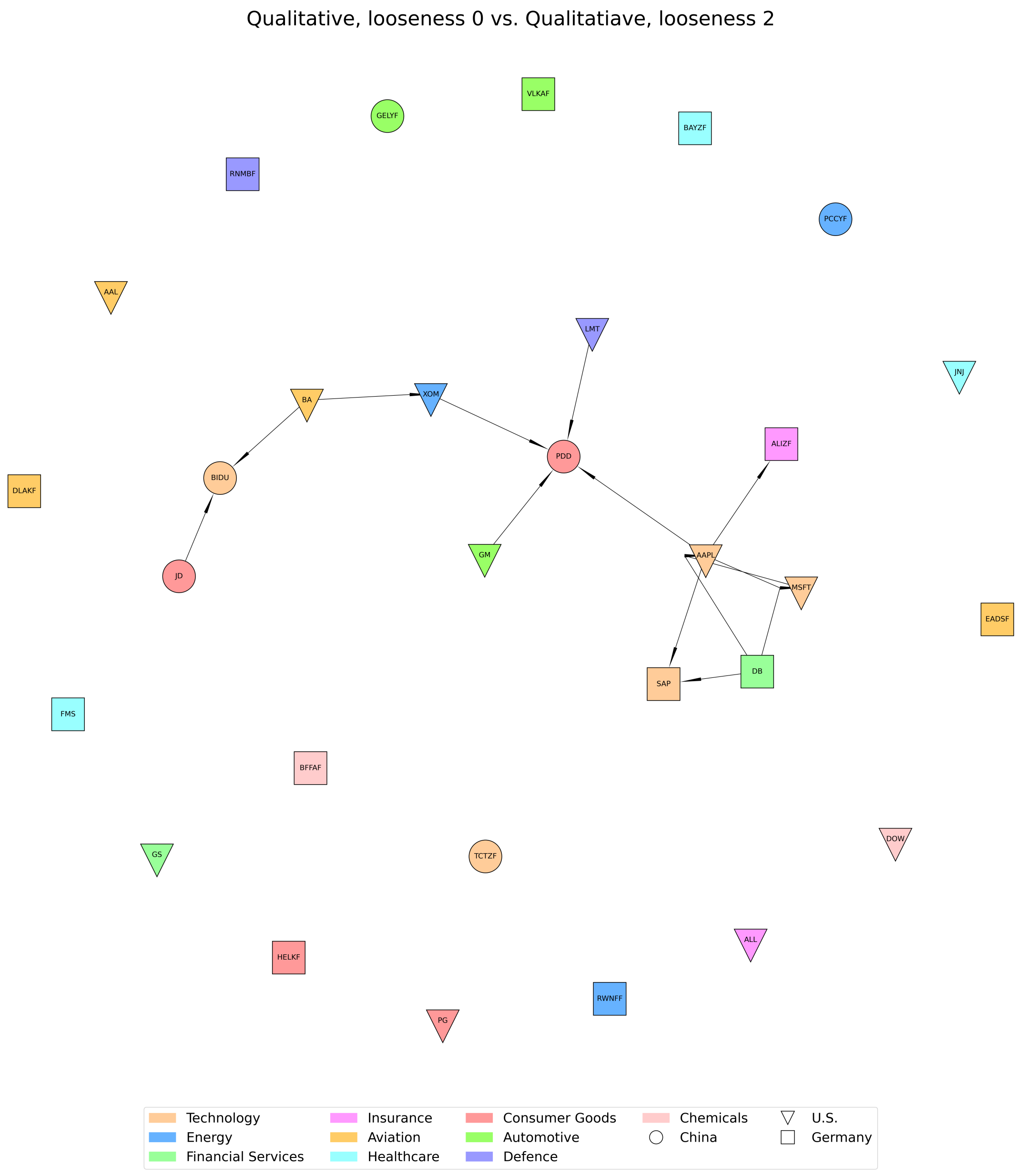


Figure A.5: Intersections between the qualitative derived graphs with `looseness = 0` *and* `looseness = 2`*.*

TP/FP/FN	**FCI+**	**Observational**	**Interventional**
True-Graph	0/8/17	4/7/13	17/8/0
FCI+		0/11/8	0/25/8
Observational			4/21/7

Table A.2: Comparison of the arcs. Networks row-wise are taken as true, while networks column-wise are than compared, resulting in true-positive, false-positive and false-negative arcs.

Bibliography

[1] *1-Phosphatidyl-1D-myo-inositol 3,4-Bisphosphate (CHEBI:16152).* URL: https://www.ebi.ac.uk/chebi/searchId.do?chebiId=CHEBI%3A16152 (visited on 05/29/2023).

[2] *ADRs, Foreign Ordinaries, & Canadian Stocks.* Schwab Brokerage. URL: https://www.schwab.com/stocks/understand-stocks/adrs-foreign-ordinaries-canadian-stocks (visited on 05/29/2023).

[3] *Aligning Language Models to Follow Instructions.* URL: https://openai.com/research/instruction-following (visited on 06/11/2023).

[4] James F Allen and Choh Man Teng. “Broad Coverage, Domain-Generic Deep Semantic Parsing”. In: ().

[5] Felipe Almeida and Geraldo Xexéo. *Word Embeddings: A Survey.* May 1, 2023. arXiv: 1901.09069 `[cs, stat]`. URL: http://arxiv.org/abs/1901.09069 (visited on 06/10/2023). preprint.

[6] Charles K. Assaad, Emilie Devijver, and Eric Gaussier. “Survey and Evaluation of Causal Discovery Methods for Time Series”. In: *Journal of Artificial Intelligence Research* 73 (Feb. 28, 2022), pp. 767–819. ISSN: 1076-9757.

[7] Mehmet Aydar, Ozge Bozal, and Furkan Ozbay. *Neural Relation Extraction: A Survey.* June 23, 2020. arXiv: 2007.04247 `[cs]`. URL: http://arxiv.org/abs/2007.04247 (visited on 01/31/2023). preprint.

[8] Özgün Babur et al. “Causal Interactions from Proteomic Profiles: Molecular Data Meet Pathway Knowledge”. In: *Patterns (New York, N.Y.)* 2.6 (June 11, 2021), p. 100257. ISSN: 2666-3899. pmid: 34179843.

[9] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate.* May 19, 2016. arXiv: 1409.0473 `[cs, stat]`. URL: http://arxiv.org/abs/1409.0473 (visited on 06/10/2023). preprint.

[10] Stefano Beretta et al. *Learning the Structure of Bayesian Networks: A Quantitative Assessment of the Effect of Different Algorithmic Schemes.* Aug. 3, 2018. arXiv: 1704.08676 `[cs, stat]`. URL: http://arxiv.org/abs/1704.08676 (visited on 06/13/2023). preprint.

[11] *Bigbio/Genetag · Datasets at Hugging Face.* URL: https://huggingface.co/datasets/bigbio/genetag (visited on 05/21/2023).

[12] Tom B. Brown et al. *Language Models Are Few-Shot Learners.* July 22, 2020. arXiv: 2005.14165 `[cs]`. URL: http://arxiv.org/abs/2005.14165 (visited on 05/21/2023). preprint.

[13] Hailiang Chen et al. “Wisdom of Crowds: The Value of Stock Opinions Transmitted Through Social Media”. In: *The Review of Financial Studies* 27.5 (May 1, 2014), pp. 1367–1403. ISSN: 0893-9454.

[14] Tom Claassen, Joris Mooij, and Tom Heskes. *Learning Sparse Causal Models Is Not NP-hard.* Sept. 26, 2013. arXiv: 1309.6824 [cs]. URL: http://arxiv.org/abs/1309.6824 (visited on 06/13/2023). preprint.

[15] Haixing Dai et al. *AugGPT: Leveraging ChatGPT for Text Data Augmentation.* Mar. 20, 2023. arXiv: 2302.13007 [cs]. URL: http://arxiv.org/abs/2302.13007 (visited on 06/06/2023). preprint.

[16] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers).* NAACL-HLT 2019. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186.

[17] *Dmis-Lab/Biobert-v1.1 · Hugging Face.* URL: https://huggingface.co/dmis-lab/biobert-v1.1 (visited on 05/21/2023).

[18] Frederick Eberhardt. “Introduction to the Foundations of Causal Discovery”. In: *International Journal of Data Science and Analytics* 3.2 (Mar. 1, 2017), pp. 81–91. ISSN: 2364-4168.

[19] Kawin Ethayarajh. *BERT, ELMo, & GPT-2: How Contextual Are Contextualized Word Representations?* SAIL Blog. URL: http://ai.stanford.edu/blog/contextual/ (visited on 06/11/2023).

[20] *Free Stock APIs in JSON & Excel | Alpha Vantage.* URL: https://www.alphavantage.co/ (visited on 05/29/2023).

[21] Andreas Gerhardus and Jakob Runge. “High-Recall Causal Discovery for Autocorrelated Time Series with Latent Confounders”. In: *Advances in Neural Information Processing Systems.* Vol. 33. Curran Associates, Inc., 2020, pp. 12615–12625.

[22] *Google Colaboratory.* URL: https://colab.research.google.com/ (visited on 05/21/2023).

[23] C. W. J. Granger. “Investigating Causal Relations by Econometric Models and Cross-spectral Methods”. In: *The Econometric Society* Vol. 37 (Aug. 1969), pp. 424–438. JSTOR: 1912791.

[24] Ralph Grishman. “Information Extraction”. In: *IEEE Intelligent Systems* 30.5 (Sept. 2015), pp. 8–15. ISSN: 1941-1294.

[25] Thomas R. Gruber. “A Translation Approach to Portable Ontology Specifications”. In: *Knowledge Acquisition* 5.2 (June 1, 1993), pp. 199–220. ISSN: 1042-8143.

[26] Nicola Guarino and Pierdaniele Giaretta. “Ontologies and Knowledge Bases”. In: ().

[27] Alexander John Hartemink. “Principled Computational Methods for the Validation and Discovery of Genetic Regulatory Networks”. In: ().

[28] Stefan Heindorf et al. “CauseNet: Towards a Causality Graph Extracted from the Web”. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. CIKM ’20: The 29th ACM International Conference on Information and Knowledge Management. Virtual Event Ireland: ACM, Oct. 19, 2020, pp. 3023–3030. ISBN: 978-1-4503-6859-9.

[29] *Home*. World Modelers. URL: https://worldmodelers.com/ (visited on 02/06/2023).

[30] *Home*. DBpedia Association. May 4, 2023. URL: https://www.dbpedia.org/ (visited on 06/06/2023).

[31] Torsten Hothorn et al. *Lmtest: Testing Linear Regression Models*. Version 0.9-40. Mar. 21, 2022.

[32] Mandar Joshi et al. “SpanBERT: Improving Pre-training by Representing and Predicting Spans”. In: *Transactions of the Association for Computational Linguistics* 8 (Jan. 1, 2020), pp. 64–77. ISSN: 2307-387X.

[33] Markus Kalisch. *CRAN - Package Pcalg*. URL: https://cran.r-project.org/web/packages/pcalg/index.html (visited on 06/12/2023).

[34] Dongyeop Kang et al. “Detecting and Explaining Causes From Text For a Time Series Event”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2017. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2758–2767.

[35] Jinhyuk Lee et al. “BioBERT: A Pre-Trained Biomedical Language Representation Model for Biomedical Text Mining”. In: *Bioinformatics* 36.4 (Feb. 15, 2020), pp. 1234–1240. ISSN: 1367-4803, 1367-4811. arXiv: 1901.08746 [cs].

[36] Yinhan Liu et al. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. July 26, 2019. arXiv: 1907.11692 [cs]. URL: http://arxiv.org/abs/1907.11692 (visited on 06/02/2023). preprint.

[37] Pablo N Mendes, Max Jakob, and Christian Bizer. “DBpedia: A Multilingual Cross-Domain Knowledge Base”. In: ().

[38] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. Sept. 6, 2013. arXiv: 1301.3781 [cs]. URL: http://arxiv.org/abs/1301.3781 (visited on 06/10/2023). preprint.

[39] Ana Rita Nogueira et al. “Methods and Tools for Causal Discovery and Causal Inference”. In: *WIREs Data Mining and Knowledge Discovery* 12.2 (2022), e1449. ISSN: 1942-4795.

[40] *OpenAI API*. URL: https://platform.openai.com (visited on 05/21/2023).

[41] *OpenAI API*. URL: https://platform.openai.com/docs/model-index-for-researchers (visited on 05/21/2023).

[42] Sachin Pawar, Girish K. Palshikar, and Pushpak Bhattacharyya. *Relation Extraction : A Survey*. Dec. 14, 2017. arXiv: 1712.05191 [cs]. URL: http://arxiv.org/abs/1712.05191 (visited on 01/30/2023). preprint.

[43] Ciyuan Peng et al. “Knowledge Graphs: Opportunities and Challenges”. In: *Artificial Intelligence Review* (Apr. 3, 2023). ISSN: 1573-7462.

[44] 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)*. EMNLP 2014. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543.

[45] Matthew E. Peters et al. *Deep Contextualized Word Representations*. Mar. 22, 2018. arXiv: 1802.05365 `[cs]`. URL: http://arxiv.org/abs/1802.05365 (visited on 06/10/2023). preprint.

[46] *Pretrained Models — Transformers 2.4.0 Documentation*. URL: https://huggingface.co/transformers/v2.4.0/pretrained_models.html (visited on 06/06/2023).

[47] Alec Radford et al. “Improving Language Understanding by Generative Pre-Training”. In: ().

[48] J. Runge. “Causal Network Reconstruction from Time Series: From Theoretical Assumptions to Practical Estimation”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.7 (July 23, 2018), p. 075310. ISSN: 1054-1500.

[49] Jakob Runge. *Tigramite – Causal Inference for Time Series Datasets*.

[50] Jakob Runge et al. “Detecting and Quantifying Causal Associations in Large Nonlinear Time Series Datasets”. In: *Science Advances* 5.11 (Nov. 2019), eaau4996. ISSN: 2375-2548.

[51] Karen Sachs et al. “Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data”. In: *Science* 308 (Apr. 22, 2005). ISSN: 1095-9203.

[52] Marco Scutari. *Reproducing the Causal Signalling Network in Sachs et al., Science (2005)*. bnlearn - an R package for Bayesian network learning and inference. May 17, 2023. URL: https://www.bnlearn.com/research/sachs05/.

[53] *Seeking Alpha | Stock Market Analysis & Tools for Investors*. URL: https://seekingalpha.com (visited on 05/28/2023).

[54] Ali Shojaie and Emily B. Fox. “Granger Causality: A Review and Recent Advances”. In: *Annual Review of Statistics and Its Application* 9.1 (Mar. 7, 2022), pp. 289–319. ISSN: 2326-8298, 2326-831X.

[55] Jaapjan D Snoep et al. “Commentary: A Structural Approach to Berkson’s Fallacy and a Guide to a History of Opinions about It”. In: *International Journal of Epidemiology* 43.2 (Apr. 1, 2014), pp. 515–521. ISSN: 0300-5771.

[56] *spaCy · Industrial-strength Natural Language Processing in Python*. URL: https://spacy.io/ (visited on 05/30/2023).

[57] Dianbo Sui et al. “Joint Entity and Relation Extraction With Set Prediction Networks”. In: *IEEE Transactions on Neural Networks and Learning Systems* (2023), pp. 1–12. ISSN: 2162-2388.

[58] Lorraine Tanabe et al. “GENETAG: A Tagged Corpus for Gene/Protein Named Entity Recognition”. In: *BMC Bioinformatics* 6.1 (May 24, 2005), S3. ISSN: 1471-2105.

[59] *Tokenization*. URL: https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html (visited on 06/10/2023).

[60] Cagri Toraman et al. “Impact of Tokenization on Language Models: An Analysis for Turkish”. In: *ACM Transactions on Asian and Low-Resource Language Information Processing* 22.4 (Apr. 30, 2023), pp. 1–21. ISSN: 2375-4699, 2375-4702. arXiv: 2204.08832 [cs].

[61] Ashish Vaswani et al. *Attention Is All You Need.* Dec. 5, 2017. arXiv: 1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 05/21/2023). preprint.

[62] Jiapeng Wang and Yihong Dong. “Measurement of Text Similarity: A Survey”. In: *Information* 11.9 (9 Sept. 2020), p. 421. ISSN: 2078-2489.

[63] Gary A. Wayman, Hiroshi Tokumitsu, and Thomas R. Soderling. “Inhibitory Cross-Talk by cAMP Kinase on the Calmodulin-Dependent Protein Kinase Cascade”. In: *Journal of Biological Chemistry* 272.26 (June 27, 1997), pp. 16073–16076. ISSN: 0021-9258.

[64] Yonghui Wu et al. *Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.* Oct. 8, 2016. arXiv: 1609.08144 [cs]. URL: http://arxiv.org/abs/1609.08144 (visited on 06/10/2023). preprint.

[65] Jie Yang, Soyeon Caren Han, and Josiah Poon. “A Survey on Extraction of Causal Relations from Natural Language Text”. In: *Knowledge and Information Systems* 64.5 (May 1, 2022), pp. 1161–1186. ISSN: 0219-3116.

[66] Linyi Yang et al. *Towards Fine-grained Causal Reasoning and QA.* Apr. 15, 2022. arXiv: 2204.07408 [cs]. URL: http://arxiv.org/abs/2204.07408 (visited on 05/29/2023). preprint.

[67] Zhilin Yang et al. *XLNet: Generalized Autoregressive Pretraining for Language Understanding.* Jan. 2, 2020. arXiv: 1906.08237 [cs]. URL: http://arxiv.org/abs/1906.08237 (visited on 06/08/2023). preprint.

[68] Junjie Ye et al. *A Comprehensive Capability Analysis of GPT-3 and GPT-3.5 Series Models.* Mar. 18, 2023. arXiv: 2303.10420 [cs]. URL: http://arxiv.org/abs/2303.10420 (visited on 05/21/2023). preprint.

[69] Mina Esmail Zadeh Nojoo Kambar, Armin Esmaeilzadeh, and Maryam Heidari. “A Survey on Deep Learning Techniques for Joint Named Entities and Relation Extraction”. In: *2022 IEEE World AI IoT Congress (AIIoT).* 2022 IEEE World AI IoT Congress (AIIoT). June 2022, pp. 218–224.

[70] Wayne Xin Zhao et al. *A Survey of Large Language Models.* May 7, 2023. arXiv: 2303.18223 [cs]. URL: http://arxiv.org/abs/2303.18223 (visited on 05/21/2023). preprint.