Disc*o*vering, expliciting, and handling semantical relati*o*nships using deep learning *o*n vari*o*us, large databases *o*f varying trustw*o*rthiness

Jacques Everwyn

Normandie Université

THÈSE

Pour obtenir le diplôme de doctorat

Spécialité INFORMATIQUE

Préparée au sein de l'Université de Caen Normandie

Découverte, explicitation et gestion de relations sémantiques par apprentissage profond sur base d'informations variée, volumineuse et de véracité variable - Application au renseignement

Présentée et soutenue par
Jacques EVERWYN

Thèse soutenue le 14/06/2021 devant le jury composé de		
M. ALDO NAPOLI	Directeur de recherche, ECOLE DES MINES DE PARIS	Rapporteur du jury
Mme CELINE ROUVEIROL	Professeur des universités, Université Paris 1 Panthéon-Sorbonne	Rapporteur du jury
M. SÉBASTIEN FERRÉ	Professeur des universités, Université Rennes 1	Membre du jury
M. SYLVAIN GATEPAILLE	Ingénieur de recherche, Airbus	Membre du jury
M. ENGELBERT MEPHU NGIFO	Professeur des universités, Université Clermont Ferrand 1 Auvergne	Membre du jury
M. Bruno ZANUTTINI	Professeur des universités, Université Caen Normandie	Membre du jury
Mme FATIHA SAIS	Professeur des universités, Université Paris-Saclay	Président du jury
M. Abdel-Illah MOUADDIB	Professeur des universités, Université Caen Normandie	Directeur de thèse

Thèse dirigée par Abdel-Illah MOUADDIB, Groupe de recherche en informatique, image, automatique et instrumentation

UNIVERSITÉ CAEN NORMANDIE

MIIS

GREYC

Notice

Remerciements

Cette thèse CIFRE n'aurait pu exister sans le soutien de l'ANRT qui a permis cette collaboration entre Airbus Defence and Space et le GREYC, que je remercie tout autant pour leur confiance.

Un grand merci à Abdel-Illah Mouaddib et Bruno Zanuttini pour m'avoir dirigé et encadré pendant ces trois ans. Nos sessions de travail resteront un très bon souvenir et vos précieux conseils m'ont permis d'arriver jusqu'au bout. Sur le côté industriel, merci à Sylvain Gatepaille et Stephan Brunessaux qui ont su me guider dans mon parcours en entreprise m'aider lorsque j'en avais besoin.

Merci également aux membres du GREYC et tout particulièrement à l'équipe MAD dont j'ai fait partie. Nous ne nous nous sommes pas vus autant que je l'aurais souhaité - la faute à la distance et à 2020 - mais je n'oublierai pas les GT, les journées vertes, les pauses de midi et les parties de cartes endiablées dont vous avez le secret. Merci à Virginie et Corinne pour leur inestimable soutien sur la partie administrative. Merci également à Fatiha et François d'avoir accepté d'être mon comité de suivi, et à Aldo, Céline, Sébastien, Fatiha et Engelbert d'avoir accepté de faire partie de mon jury.

J'ai été accueilli à Airbus par l'équipe de Sylvie en traitement avancé de l'information et je l'en remercie, ainsi que Franck qui lui a succédé. Au cours de ces quatre ans à Elancourt, la petite équipe à forte tendance normande s'est bien agrandie et les stocks de croissants n'ont jamais cessé d'augmenter. Pour toutes ces discussions, réunions, débats plus ou moins sérieux autour d'un café ou d'une pinte, merci à Florentin, Howard, Guillaume, Paul, Kilian, Jonathan, Valérian, Antoine, les Vincent, les Kevin, et toute la famille du B3 qui fait de cet étage un endroit incroyable. Merci aussi à Edouard, Berylia, Nicolas, Allison et Emilie, avec qui j'ai passé de (probablement trop) nombreux midis à refaire le monde avec nos petites mains.

Merci également à tous mes collègues et amis thésards avec qui j'ai partagé les joies et souffrances du doctorat: Sébastien, Josselin, Romain, Valentin, Quentin, Raphael, Guillaume, Louis, Florent, Syrielle, et tous ceux que j'ai pu croiser en conférence ou sur les internets.

Enfin, je ne peux pas terminer ces remerciements sans mentionner mes amis, ma famille, mes parents et ma chérie, Safiya, qui m'ont apporté le soutien nécessaire pour arriver au bout de cette aventure.

Abstract

Maritime surveillance operators need to have the best view on the situation they monitor, a view known as Maritime Situational Awareness (MSA). With the rapid development of sensors and satellite data, these operators face a large flow of data coming from multiple sources in the maritime ecosystem. Hence, they require a decision aid to navigate through all the available information.

We choose dynamic attributed knowledge graphs to represent a maritime situation. They are graph structures that represent the interactions between entities and the evolution of their attributes. Such structures can be used to represent real-life situations involving moving and interacting objects, unlocking visualization and processing capabilities with a semantic layer. By being able to predict new links between entities of a situation, we could raise early alerts to operators who will thus know where to look in the data to assess what is happening. However, most state-of-the-art link prediction models tackle only static (attributed) graphs or dynamic non-attributed graphs, which makes them unusable in this setting.

In this thesis, we propose Joint Evolution, an end-to-end encoder-decoder system composed of two neural networks. Joint Evolution uses both attributes and relations of a knowledge graph in a dynamic setting to predict new relations between entities. The main hypothesis tested in this work is whether the inclusion of attributes during the relational link prediction can improve the results thanks to the additional information it provides. We test our method on two different datasets: datAcron, a maritime dataset that report events near the Bay of Biscay in France for 364 vessels, and DBLP, a citation network with thousands of authors that are co-authors or cite each other.

Contents

CHAPTER 1

Introduction

How to efficiently monitor maritime activities and anticipate abnormal behaviors from vessels that travel the world and the seven seas? Well, everybody is looking for something, and that is the question I want to answer. This quest leads to a very interesting research field that combines operational data, time-series, knowledge graphs and machine learning. This manuscript is made of these, and presents how we use the data transmitted by vessels to predict interactions between them.

To start with a bit of context, this PhD thesis was done in collaboration with Airbus Defence and Space (Elancourt, France) and the GREYC laboratory located in the University of Caen Normandy (Caen, France) in the scope of the CIFRE program (Industrial Research Training Agreements). Led by the French National Agency for Research and Technology (ANRT), it supports French companies that are willing to hire a PhD student and create a research collaboration with a public laboratory, thus facilitating the exchange of knowledge and technologies between the industrial and the academic world.

The core topic of this work is the representation of evolving situations to extract new information. The monitoring of such situations is of particular interest for intelligence or surveillance operators, because they need to know as much as possible about an area or person of interest. This knowledge of situation is also called *Situational Awareness*. Situational Awareness is critical in many cases such as the protection of human life and property, law enforcement, aviation, air traffic control, ship navigation, health care, emergency response, military command and control operations. It needs three pillars to be effective: perception of the elements in the environment, understanding of the situation, and projection of future status.

The first one focuses on the gathering of heterogeneous data about the area and entities of interest. This data can be hard data, i.e. coming from physical sensors (quantitative, Figure A.1), or soft data, i.e. linguistic data produced by humans (qualitative). The second pillar makes use of this data to extract knowledge about the situation and get all the ins and outs from it. It usually comes as events or interactions between entities of the situation. At last, the third one uses what we grasped from the current situation to predict what may happen in the future. Such inference from the current knowledge provides foresight to the operators,who can better anticipate problems or threats.

Thanks to Situational Awareness, the intelligence operator can perform an analysis to have a better situation understanding. During this analysis, she will establish links between entities of the situation to represent their interaction (for instance, links of affiliation, commandment, subordination, conflict) and create new entities. As of today, this analysis has a very low level of automation and covers a small part of the available data. With the rising number of entities to monitor (persons, vehicles, devices...) and of deployed sensors that massively produce data, it has become nearly impossible for an operator to analyze and assess a situation by hand. There is a need for automation in the monitoring process, so that the operators and experts may look at events of interests proposed by an automated system rather than browsing the large flow of data by hand.

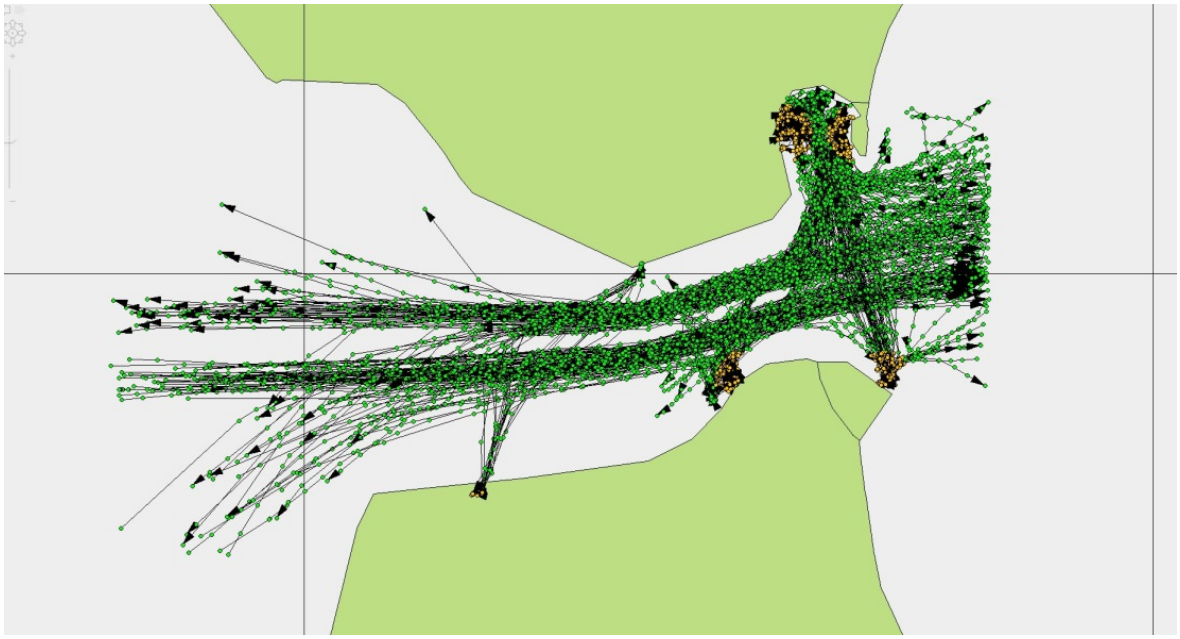


Figure 1.1: One day of vessel positional data in the Strait of Gibraltar. The traffic is dense and difficult to analyze.

To this end, the representation of the knowledge is of uttermost importance and faces several challenges. The data structure must be dynamic to follow the evolution of the situation, contain the attributes and characteristics of each entity and model their interactions with their environment. Since there is always a human in the loop, the information should be easy to visualize, understand and browse. Moreover, any decision taken by an automated system must be interpretable or explainable to win the trust of the human operator., i.e. the operator using it needs to know why the system has taken a decision. In a context of defense and surveillance, it is not possible for decision-makers to accept and use the output of such a system if it is not reliable and justified. And in addition to the uncertainty of the projections, the quality of the data must also be taken into account: an intelligence report may have a typo, sensors have a range and an accuracy (e.g. +/- 500 meters), or data loss may happen when satellites receive signals. At last, the monitoring of a situation must be continuous and the forecasts feasible at all times: the perception of the situation (i.e. the addition of new information on the environment) must ideally be made in real-time and its processing for projections made in a reasonable time so that the appropriate reaction can be deployed as early as possible.

Knowledge graphs are increasingly used to represent fields of knowledge: some pairs of nodes in the graph are linked by an edge, thus forming a relation between the two nodes in the form of a triple (subject, relation, object). This network of relations constitutes the knowledge graph, and models information such as (RMS Titanic, :builtBy, WhiteStarLine). Most often, this knowledge is static and does not evolve over time. It can also indicates the time when the events happened (e.g. (RMS Titanic, :builtBy, WhiteStarLine, 1909)). With dynamic graphs, new links can be inferred using temporal dependency learning. This helps to understand the behavior of entities over time.

Additionally, the nodes of dynamic graphs can be enriched with attributes, which characterize each entity. Usually, static attributes are attached to the nodes, such as (RMS Titanic, :size, 269.1), and give more information about the entities. Dynamicity also applies to attributes, especially when dealing with moving entities such as planes, boats or people, e.g., (RMS Titanic, :latitude, 40.963, 23:40 1912/04/14).

From such graphs, relational learning aims at reasoning and inferring new or unseen links between entities, leading to applications such as knowledge graph completion and recommender systems. Generating new links in a graph is now a common AI application, but many state-of-the art techniques mainly focus on static links and attributes, or on dynamic links, but rarely on dynamic links *and* attributes. When there are only a few relations between entities, the evolution of their attributes can greatly help to characterize them.

With this in mind, vessels, ports, countries, and maritime companies have various interactions, and the status of vessels evolves rapidly. Various data, such as positions, speed, course or destination can be gathered. This data can give more precise knowledge on the ongoing events: a vessel is entering a port, two boats are meeting, or the pattern of a vessel is indicating a fishing activity. Knowing this, a surveillance operator may want to know when and where two vessels will meet, or when a vessel will come back to port given the surrounding context. This also applies to piracy, smuggling or illegal fishing. Such maritime situations can be modelled by dynamic attributed knowledge graphs. The objective is to use such a graph to infer new relations between maritime entities, and hence to improve the maritime situational awareness of the operator by detecting new facts earlier.

The contributions of this work address this task: we propose a dynamic attributed knowledge graph structure to represent evolving situations, and apply a deep learning-based framework made of two temporal neural networks that learn representations of the entities of the situation to predict their future behavior based on past information. Our main usecase is the maritime context, since maritime situations involve moving entities, environmental changes, interactions, sensors, uncertainty and human operators. This work is also fitted to social network analysis, and we will test it on a well-known citation network, DBLP, to see if it generalizes well.

The first part of this manuscript is dedicated to the state of the art made on this topic and is divided in three chapters describing respectively the current methods used to improve Maritime Situational Awareness, the knowledge graph structures, and how to create representation for these graphs and perform link prediction on them. The second part presents our contributions. The first chapter formalizes the link prediction problem on dynamic attributed knowledge graphs. The second one introduces our framework, JointEvolution, and how we designed it. The third part is dedicated to experiments on both academic and operational datasets, and comments on the results. The last part sums up the work done and concludes

with some perspectives.

Part I

State of the art

CHAPTER 2

Maritime Situational Awareness

The field of maritime surveillance is strongly linked to Situational Awareness, produces large amounts of heterogeneous data and is in need of automated processing. It will thus be the main application domain for our work.

2.1 Introduction to Maritime Situational Awareness

2.1.1 Maritime Situational Awareness

Maritime transportation provides the most energy efficient means of transporting large quantities of goods over large distances [Kaluza et al., 2010]. In fact, approximately 90% of all international trade is carried out by sea, which represents more than 50.000 vessels each day for 7.5 billion tons of goods transported across the oceans each year (Figure 2.1). However, the maritime environment might be hazardous: shipwrecks, collision or other life-threatening dangers are to happen with such a dense traffic. Apart from these events, more unlawful ones such as terrorism, piracy, drug/arms smuggling and illegal immigration dampen traffic security and efficiency. To illustrate this, in Africa, Asia, Latin America and the Caribbean, 1,455 piracy or robbery incidents occurred and 25,091 seafarers were impacted between 2015 and 2019. 350 of them were kidnapped or held hostages in 2019 [Joubert, 2020].

Maritime Situational Awareness (MSA) is defined by the International Maritime Organization as "the effective understanding of anything associated with the global maritime environment that could impact the security, safety, economy or environment". The purpose is to establish a picture of the maritime situation at a given moment by perceiving the different elements acting in the considered environment and understanding their significance in order to perform efficient decision making and to forecast what is next.

To be effective, MSA needs an accurate, fresh and constant data stream, trained operators with expert knowledge and explainable decisions. Thus, for efficient MSA, the following challenges must be overcome:

- *Fusion of heterogeneous data*: to have the best view of a situation, data from various sources must be aggregated into a common data structure. These sources can include

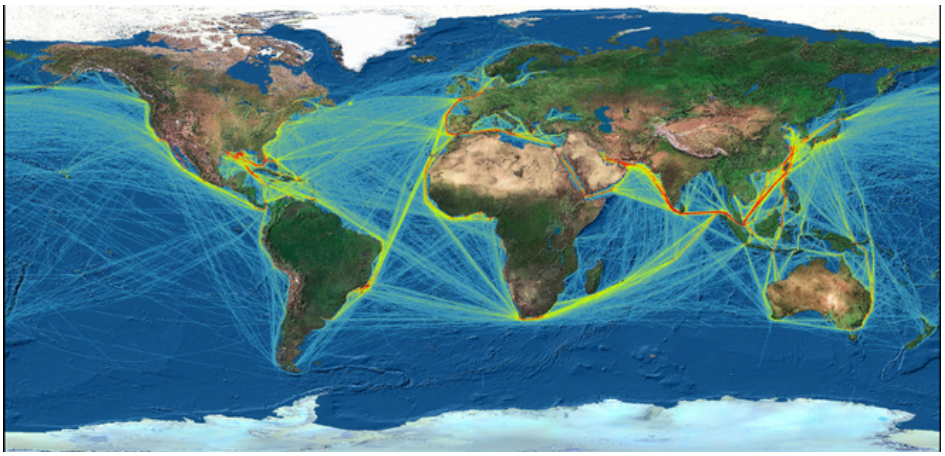


Figure 2.1: Maritime traffic density (source: Luxspace 2010)

navigational data, sensors data such as temperature, weather, currents or acoustic, images from satellites or Synthetic Aperture Radar (SAR), reports from port authorities... Trajectories, images, texts and sounds must be assembled to create the most complete picture for the surveillance operator.

- *Evolution of entities*: a maritime situation is a fast evolving world with very little time between two events. In MSA, a good evolutionary model is needed for change detection and the granularity depends on the task. For a change in the position/course/speed of a vessel (dynamic attributes), the information must be given within minutes (e.g. rapid response needed in case of piracy). But to detect a change in a vessel particulars (identifier, name...), the granularity needed can be in hours or days.

- *Event and threat detection*: an event can be represented by a new relation between two entities or an above-mentioned attribute evolution. If event mining is for fact detection, threat detection is a task highly related to its context and definition. A nation will not consider a transshipping between two fishing vessels as a threat since they are more likely to exchange fish than warheads, but an Non-Governmental Organization (NGO) for ocean conservation can suspect illicit fishing of an endangered species. Performing this task requires either expert knowledge or labeled events.

- *Streaming*: MSA requires a constant monitoring of maritime areas, meaning that a surveillance model must deal with a continuous flow of data.

- *Uncertainty*: maritime data often results from hard (sensors) and soft (websites, intelligence) data fusion. However, this data is not always 100% certain: an intelligence report may have a typo, sensors have a range and precision (e.g. +/- 500 meters), or collisions may happen when satellites receive signals. This uncertainty must be taken into account when making decisions.

- *Explainability*: AI-based models are often black boxes when it comes to the origin of an alert. However, a surveillance operator needs to know why a prediction was made in order to understand it and justify any upcoming response to an event. Because operators still do not trust AI-based systems to make decisions, explainability is needed to use automated models for MSA [Adadi and Berrada, 2018], but also in general.[1]

[1] `https://ec.europa.eu/futurium/en/ai-alliance-consultation`

2.1.2 AIS/S-AIS data

Maritime navigation relies on AIS (Automatic Identification System), a short range ship-to-ship and ship-to-shore navigational data exchange system (Figures 2.2 and 2.3). Created as an anti-collision and traffic management system, it is now the main source of data to monitor the maritime traffic. There are two kinds of AIS messages: those received by coastal stations and those received by satellites.

2.1.2.1 AIS messages

AIS is currently the main source of information available in support of maritime surveillance (Figure 2.3). It uses two VHF frequencies, Channel 87 – 131.975 MHz and Channel 88 – 162.025 MHz that are dedicated to this system worldwide. AIS messages are collaborative and depends on how the captain of the vessel configured the AIS system that emits the messages. They provide the following information about ships [BigOceanData, 2016,Berger, 2018].

Static data is set up during the AIS system installation and only needs to change when the vessel changes its name or is converted to another vessel type:

- The Maritime Mobile Service Identity (MMSI) of the vessel;
- Call sign and name of vessel;
- The International Maritime Organization (IMO) number of the vessel;
- Length and beam;
- Type of ship;
- Location of position fixing antenna.

Dynamic data (or position report) is sent every 2 to 10 seconds depending on the speed of the vessel when moving, and 3 minutes while at anchor. It is automatically updated from the ship's sensors connected to the AIS and includes:

- The MMSI of the vessel;
- The IMO number of the vessel;
- The longitude: in 1/10 000 min (±180°, East = positive, West = negative. 181 = not available);
- The latitude: in 1/10 000 min (±90°, North = positive, South = negative. 91 = not available);
- The course over ground (COG): in 1/10° (0-3599). 3600 = not available);
- The speed over ground (SOG): in 1/10 knot steps (0-102.2 knots. 1 023 = not available);
- The heading (0-360°);
- The type of ship (and the type of cargo).

Voyage related data is broadcast every 6 minutes and needs to be manually entered and updated:

- Ship's draught;
- Hazardous cargo (type) (e.g. DG (Dangerous goods), HS (Harmful substances) or MP (Marine pollutants);
- Destination and ETA.

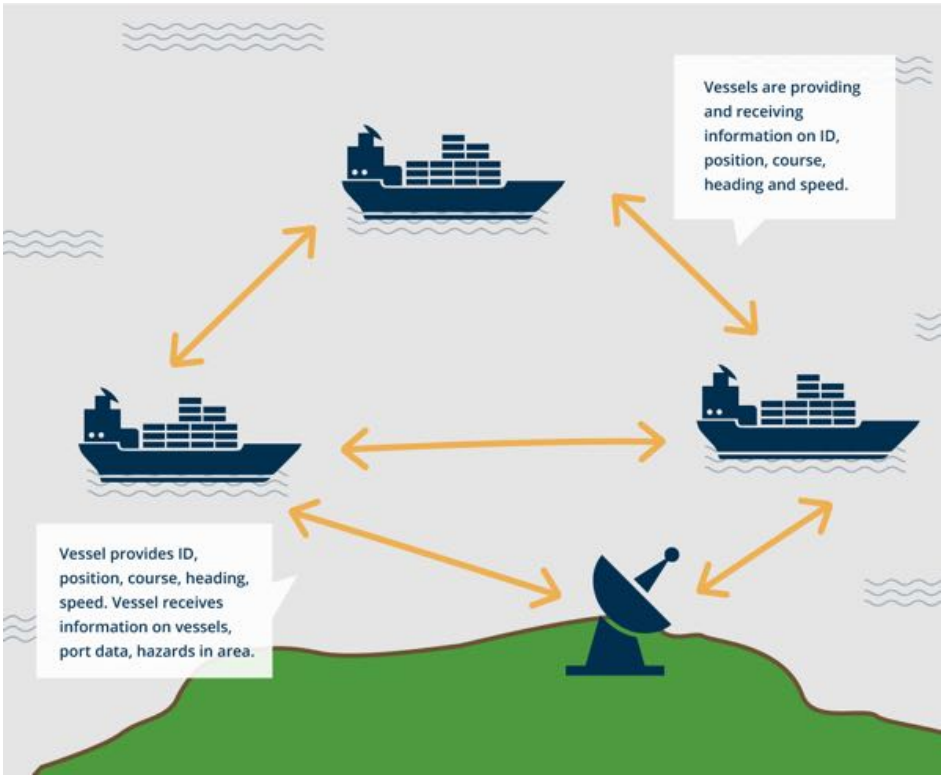


Figure 2.2: AIS communication [BigOceanData, 2016]

There are two classes of AIS equipment, class A and class B. According to the International Maritime Organisation (IMO), AIS class A equipment is compulsory for:

- 300 or more gross tonnage engaged on international voyages,
- All cargo ships of 500 or more gross tonnage not engaged on international voyages,
- All passenger ships and tankers regardless of their size.

AIS class A currently represents about 60.000 ships. But since the introduction of AIS class B in 2007, hundreds of thousands of vessels now have AIS equipment. Class B messages have a range limited to 10 nautical miles (1 NM = 1852m) to avoid overloading the bandwidth. They also withhold less information than class A, being limited to:

- MMSI (Maritime Mobile Service Identity);
- Call sign and name of vessel ;

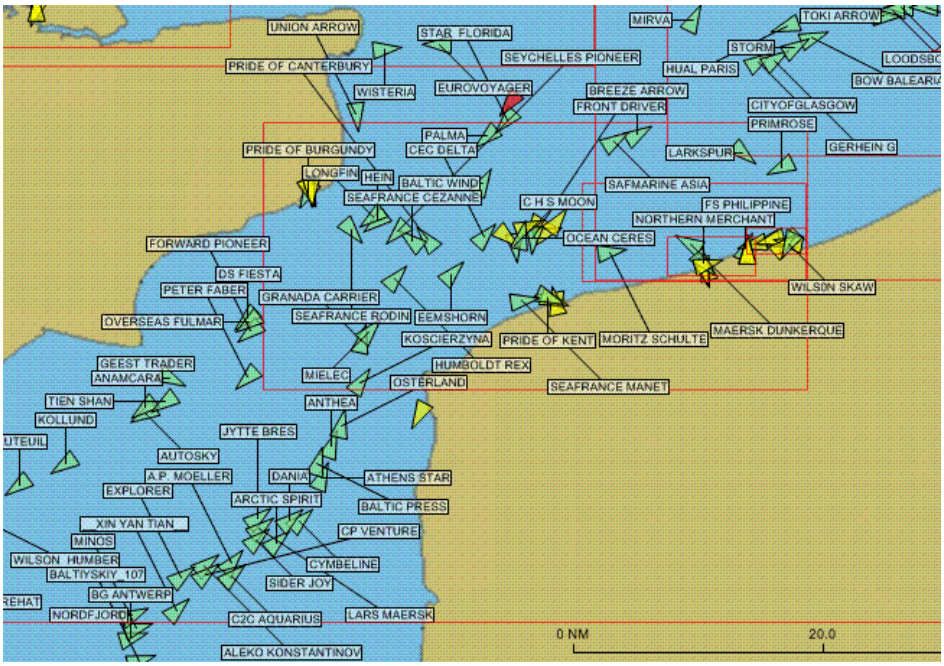


Figure 2.3: AIS used to display position and name of vessels - Source: AIS/wikipedia.org

- Length and beam;
- Type of vessel.

In Figure 2.4 is an example of a raw AIS position report encoded in the NMEA format - which is the data specification for communication between maritime electronic devices - and in Figure 2.5 the same message once decoded.

$$!AIVDM,1,1,,B,1INS<8@P001cnWFEdSmh00bT0000,0*38$$

Figure 2.4: An example of AIS message [Le Guillarme and Lerouvreur, 2013]

Parameter	*Value*	*Description*
Message ID	1	
Repeat indicator	1	Repeat once
User ID (MMSI)	636013601	
Navigational status	0	Under way using engine
Rate of turn ROTAIS	-128.0	
SOG	0.0	
Position accuracy	0	Low (> 10 m) (default)
Longitude	23*33.6555′E	
Latitude	37*55.0167′N	
COG	0.0	
True heading	21	
Time stamp	18	
Special manoeuvre indicator	0	
Spare	0	
RAIM-flag	0	RAIM not in use (default)

Figure 2.5: AIS information after NMEA sentence decoding [Le Guillarme and Lerouvreur, 2013]

2.1.2.2 S-AIS messages

Despite a near ten-meters precision, AIS has a very limited range (37-74 km): this is the reason why AIS is often coupled with S-AIS, which is the same but with satellites instead of coastal stations. The precision is not as good as AIS in dense areas but we have a huge gain in range: sensors from an AIS satellite can reach 5000 km at 650/850km altitude [Le Guillarme and Lerouvreur, 2013].

The problem with S-AIS data is the message collisions due to the uncoordinated emission of messages from ships belonging to different areas but in the scope of the same satellite: these interferences are the main limitation of this detection equipment. That is why both AIS and S-AIS data are needed to achieve complete maritime domain awareness, as well as having several AIS data providers.

2.1.2.3 Usecases

With AIS, maritime authorities have access to navigation information to monitor their Regions of Interest (ROI). It also facilitate voice communication and the exchange of vital information such as speed, position and heading. Now that it is well developed and widespread, it is used to assess activities and detect anomalies thanks to a better access to information. A deviant trajectory can be spotted without having a visual on the vessel, dangerous cargoes in an unusual maneuver can be spotted and ports can manage the traffic better in their vicinity.

Since it is easy to get unnoticed in the immensity of the ocean, the maritime domain is the theater of many illicit activities such as illegal immigration, smuggling, piracy or terrorism, but also environmental threats like oil spilling or illicit craft discharge [Serry and Lévêque, 2015]. AIS provides information to detect such behaviors thanks to anomaly detection systems, rule-based engines or learned models but being a collaborative source, it also has its limitations and other sources are needed to achieve MSA.

2.1.2.4 Limitations

The collaborative nature of AIS is its main weakness: the reliability of the information mainly depends on the users of the vessel. The captain can turn it off or forget to update an information (destination), installation errors may provide faulty values (MMSI) or not precise enough vessel's type categories [Harati-Mokhtari et al., 2008]. Technical difficulties are also to be considered. The range of traditional AIS is rather limited (20-40 nautical miles) and S-AIS can operate further from the coastline but at the cost of precision due to the uncoordinated emission of messages from ships belonging to different coverage regions [Le Guillarme and Lerouvreur, 2013]. The latency induced by the transmissions also degrades the quality of the data and the theoretical 2 to 10 sec when underway is rarely found. Indeed, the observed gaps between two observations of the same vessel are often of several minutes or hours, meaning that we may have no information on the vessel's whereabouts for a large period of time. Moreover, small vessels cannot be followed since AIS is not mandatory for them and hence other techniques are required, such as detection from spaceborne images [Kanjir et al., 2018]. At last, AIS can also be tampered with to falsify information and deceive the authorities [Iphar et al., 2016]: a smuggler may change its identity, or the presence of vessels on a coastline can be simulated with false transmitted positions. There exist ways to detect such forgery, such as AIS/radar comparison [Katsilieris et al., 2013], but this often requires other data sources. The Long Range Identification and Tracking system (LRIT) is a complementary system that also provides positional data, but it is meant for long-

range tracking and is only accessible to concerned parties. With only one emission every 6 hours,[2] it is not suitable to replace AIS in MSA, at least not for swift responses.

In a nutshell:

- MSA is required to monitor the maritime traffic;
- A streaming flow of heterogeneous data is needed for this;
- AIS is the main source of information but is not sufficient;
- Anomaly/event detection can be performed on AIS.

2.2 Anomaly and event detection

Maritime Situational Awareness often focuses on anomaly detection [Riveiro et al., 2018]. Anomalies or unexpected behaviors (e.g. unusual fishing pattern) are the main center of interest for surveillance operators, along with the search for specific events such as an encounter between two vessels.

2.2.1 What is an anomaly?

The concept of anomaly is ill-defined and can vary depending on the domain in which it is used. A common definition of anomaly (or outlier) is the one quoted in [Hodge and Austin, 2004]:

> *An outlying observation, or outlier, is one that appears to deviate markedly from other members of the sample in which it occurs.*

A more behavior-driven definition is given by [Khatkhate et al., 2007]:

> *An anomaly is defined as deviation from the nominal behavior of a dynamical system and is often associated with parametric and non-parametric changes that may gradually evolve in time.*

In the case of maritime awareness, *normalcy* and *context* must both be taken into account to define an anomaly. With AIS data, it is relatively easy to extract the main maritime routes and to spot any vessel that deviates from these "normal" routes. However, the context is important to explain this deviation. One may think that this vessel is engaging in illicit activities, when the real reason for the deviation is strong winds and currents, or a damage that put the vessel out of control. In either case, the reported position is still an anomaly compared to the majority of observations and should not be discarded. If the weather should not raise an alert for danger on the deviation, the ship may require assistance if it is damaged. For this reason, each anomaly must be put into context before reaching a conclusion on its nature.

The characteristics of an anomaly are inherent to the system in which it occurs. In maritime situation, we can distinguish two kinds of anomalies [Roy, 2008]:

[2] `http://blog.exactearth.com/blog/bid/275363/comparing-satellite-ais-to-lrit`

- Static anomalies which are related to static information about a given vessel: its name, its MMSI, the type of the ship... A static anomaly can be a change or a lack in information that prevents vessel identification.

- Dynamic anomalies, split up in two subcategories:

  - Kinematic anomalies related to information such as location, speed and course,

  - Non-kinematic anomalies related to information such as passengers, cargo and crew.

The dynamic anomalies are more detailed in Figures 2.6 and 2.7. The most common detected anomalies are kinetic anomalies thanks to AIS messages. Non-kinematic anomalies are much more difficult to detect because the data is not easily accessible. But lists exist for crew, cargo and passengers and can be wrong, on purpose or not. Such data can reinforce other anomaly detection, for instance when a vessel deviated from its course and a convicted felon is on board. At last, static anomalies, also visible with AIS, can reflect illegal behavior through identity fraud or incompatibility between current activity and vessel type. In this work, we will focus on ship positions, proximity and speed to detect events.

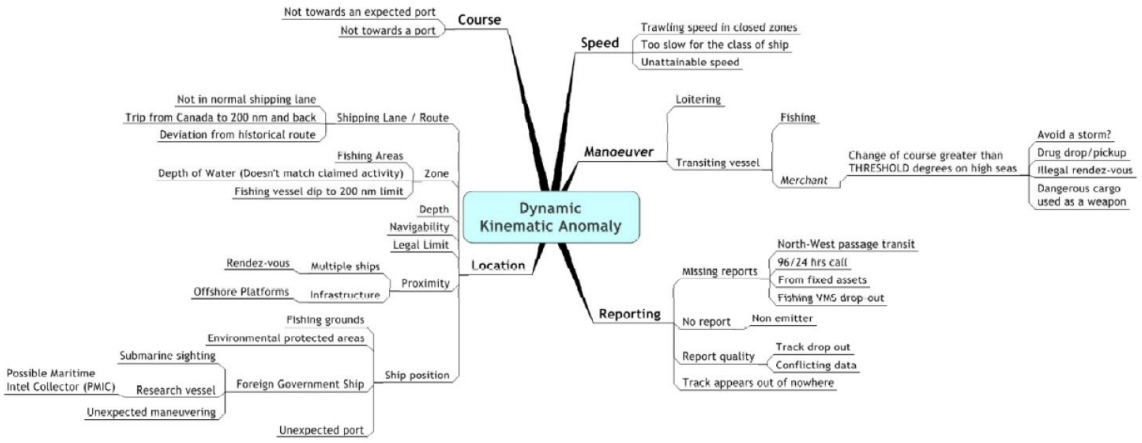


Figure 2.6: Taxonomy of dynamic, kinematic anomalies [Roy, 2008]

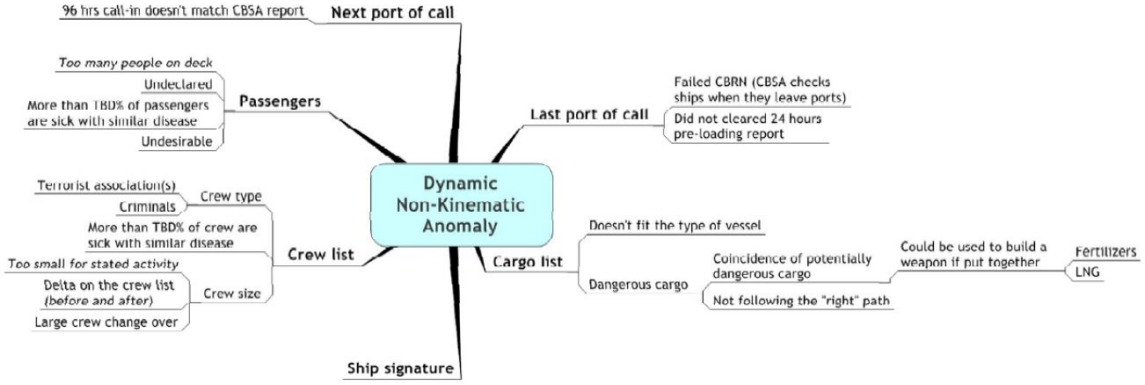


Figure 2.7: Taxonomy of dynamic, non-kinematic anomalies [Roy, 2008]

2.2.2 An overview of anomaly detection techniques

To spot the outliers, anomaly detection techniques must be devised. In most cases, normal data is used to generate a normalcy model to discriminate the samples that diverge too much from it, as stated in [Portnoy et al., 2001]:

> *Anomaly detection approaches build models of normal data and then attempt to detect deviations from the normal model in observed data.*

Commonly, there are three anomaly detection techniques:

- *Supervised techniques*, using both normalcy and abnormality models to create a learning base. The two classes must be represented equally and encompass a wide range of examples in order to achieve good classification performance.
- *Semi-supervised techniques*, using only a normalcy model. As they require only the normal instances, they are more widely applicable than supervised techniques.
- *Unsupervised techniques*, which can detect outliers without any model assuming the majority of the studied instances is normal. With the massive presence of unlabeled data in the maritime ecosystem, the choice to generate a model based on the assumption that most observations are normal is the most frequent [Riveiro et al., 2018].

The choice of a technique mainly depends on the availability of labeled data. For an exhaustive review of anomaly detection techniques see [Chandola et al., 2009, Mao et al., 2016].

2.2.3 Maritime events

When we look globally at the maritime traffic, the dangerous or illicit events are considered to be rare and unusual and thus fit the definition of an anomaly. Detecting them is an arduous task because we have lots of data from which we have to extract significant elements, hence the need for an automated detection tool for anomalous events.

Those events have root causes that can be identified with the available positional data. Each fishing method has its own pattern [Souza et al., 2016], a transshipment between two vessels means that they are very close and moving slowly or stopped, and a go-fast for smuggling can be spotted with an unusual speed (although they rarely have an AIS system). These causes are known by experts, who can use their knowledge to create rule-based systems to detect these events or look for known patterns. Without expert knowledge, data-driven approaches can learn these causes by correlating them with the consequences during the learning of a sequence of events.

Here is a non-exhaustive list of maritime events that can be detected from positional data and context:

- Ship damaged or immobilized;
- Loss of cargo;
- Marine pollution;
- Storms or any form of weather that can impede the voyage;

- Collision;
- Transshipment/transboarding;
- Unusual speed/course over ground;
- Fishing in a restricted area;
- ...

In a nutshell:

- An anomaly is an observation whose behavior is different from the others;
- Anomaly detection can be supervised, semi-supervised or unsupervised;
- Maritime events can be extracted from position data or contextual information.

2.3 Previous works

Maritime Situational Awareness has already been the subject of extended research works, and many systems have been devised to improved it. They tackle anomaly detection, but also route estimation and event detection. We present here some of the works that exist pertaining to these tasks, but this is not an exhaustive list given the large number of works.

2.3.1 Anomaly detection

Anomaly detection has been subject to a lot of research, even in the maritime domain. In [Mao et al., 2016] and [Riveiro et al., 2018], the authors made a survey of existing techniques. We present below some of these techniques that use AIS data:

2.3.1.1 Positional and kinematic anomalies

Positional and kinematic anomalies can be directly extracted from AIS. They are related to the normalcy of the trajectories.

Trajectory Cluster Modelling (TCM) TCM [Riveiro et al., 2008] clusters the trajectories, represented by *(lat, lon, speed, velocity)* vectors, using a self-organizing map [Kohonen, 1998] and apply a Gaussian Mixture Model (GMM) to model the characteristics of the clusters. The probability for a new sample to be anomalous is obtained by applying Bayes' rule on GMM probabilities.

Another clustering approach is proposed by Le Guillarme et al. [Le Guillarme and Lerouvreur, 2013]. They first separate trajectories into two categories: *Move* and *Stop* trajectories. The trajectories are then cut into several parts using a sliding window algorithm that detects direction changes. These parts are then clustered using the OPTICS algorithm, and patterns are created from them. Using the distributions of the positions, speeds and courses of each pattern, it is possible to evaluate the normalcy of an observation (= an AIS) with a chi-squared test. At last, using the patterns as states, a Hidden Markov Model can represent the transition between the patterns and predict the most probable new pattern.

- Pros: handles static and kinematic anomalies, good performance.
- Cons: high computational cost, not easy to update with new samples.

Normalcy Box (NB) The Normalcy Box method [Rhodes et al., 2005] defines a series of nesting regions over a port area. After being presented with a labeled historical motion of a set of vessels, the system learns the acceptable maximum and minimum speed for vessels in each region. Once these normalcy models are learned, any observation that does not fit its box is considered as unusual.

- Pros: Online learning ability.
- Cons: Needs expert knowledge, cannot detect static anomalies.

Holst Model The Holst model [A.Holst and Ekman, 2005] defines a grid over a port area. It then attempts to model the velocity distribution in each grid cell by a GMM, whose parameters are estimated using Maximum Likelihood optimized with an Expectation Maximisation algorithm. Anomalous instances can be detected by thresholding over the probability distribution obtained.

- Pros: unsupervised learning, good performance.
- Cons: lot of normalcy training, high computational cost.

Adaptive mixture based Neural Network This method is presented in [Garagic et al., 2009]. It consists in neural networks that detect deviations from normalcy and the number of components and the parameters are estimated from the data. The network utilizes a recursive version of the Expectation Maximization algorithm to minimize the Kullback-Leibler information metric.

- Pros: unsupervised, online, robust to noise and operator error.
- Cons: no real performance test provided, high number of false positives.

Bayesian Networks To build a maritime Bayesian network [Mascaro et al., 2014, Johansson and Falkman, 2007], the variables (or states) must represent the maritime situation. States can represent the ship type (cargo, passenger, tanker...) or the speed in a discrete manner (0-10 knots, 10-20 knots...). The directed edges between the states are then defined using expert knowledge and the training data. Once the network is trained, the probability of a succession of states (or behavior) is computed. If this joint probability is below a defined threshold, then the behavior is anomalous. Mascaro et al. [Mascaro et al., 2014] extended this model with contextual information such as weather or the hour of the day to create a more complex (and precise) representation.

- Pros: easy to include expert knowledge, easily verified by non-experts, very powerful in modeling cause and effect.
- Cons: performance very sensitive to modeling assumptions, a lot of expert knowledge required.

2.3.1.2 Complex anomalies

Complex anomalies can be assimilated to illegal events, such as smuggling or spoofing. Kowalska and Peel [Kowalska and Peel, 2012] used Gaussian processes to detect anomalies by computing the distance between the predictions of the model and the actual positions. This was successfully applied on scenarii involving smuggling and terrorism, and can also be used to detect spoofing.

Spoofing can be used to conceal the entrance in a forbidden zone or an illegal transaction. Katsilieris et al. [Katsilieris et al., 2013] uses Gaussian probability density function to formulate the spoofing detection problem in the context of hypothesis testing (true AIS or spoofed). The hypothesis are then tested with a varying number of radars and spoofing distance. Mazarella et al. [Mazzarella et al., 2017] tackles the AIS on-off switching problem. To make the difference between not received AIS and intentional transmission termination, the authors created two normalcy models: one for the vessel tracks and one for the AIS Base Stations that receive the messages. A risk level is computed with both models and triggers an alert from a certain threshold. The drawbacks of these works is that anomalies have to be classified using expert knowledge or to be specifically looked for.

2.3.2 Route estimation

Route estimation is the task of creating a model that captures the kinematic behavior of vessel and can estimate their future position. Perera et al. [Perera and Soares, 2010] opted for an Extended Kalman filter: this filter uses a dynamic model (like the laws of physics) and applies it to a system. Knowing the control inputs on that system and with multiple sequential measurements, the next state of the system can be estimated.

Zissis et al. [Zissis et al., 2016] uses an Artificial Neural Network, while Forti et al. [Forti et al., 2020] applied a more complex neural network that works with sequences (recurrent network). These data-driven approaches are effective and require little to no expert knowledge, but their main drawback resides in its absence of explainable decisions for a surveillance operator.

2.3.3 Event detection

Events are consequences with causes: the proximity between two vessels is related to a similar position, fishing activities can be detected with a speed and course pattern, and the disappearance of AIS positions can be caused by human intervention or bad atmospheric conditions. To detect these events, Alvarez et al. [Alvarez et al., 2016] discretize the studied area into a grid of 0.25×0.25 nautical miles cells and set triggers for event detection, e.g. a proximity event when two vessels are in the same cell or the entrance in an area of interest. [Artikis et al., 2015] chose a symbolic approach with predicates made to recognize patterns. The rule-based systems are very effective at finding events when their causes are known, but they require extensive expert knowledge, and the systems can become very complex with hundreds of rules.

In a nutshell:

- A lot of AIS-based anomaly detection techniques are available;
- Complex anomalies require contextual information such as weather or AIS base stations behavior;
- Data-driven models exist for route estimation and are effective;
- However, event detection is more complex and still requires hand-crafted rules.

2.4 Conclusion

The widespread availability of AIS data in large quantities made data analysis and mining for Maritime Situational Awareness possible. Automated anomaly detection became available and anomalous observations can now be spotted without mandatory human intervention. Still, the collaborative nature of AIS makes it prone to error or manipulation, and corrective techniques are required to ensure the completeness and trustworthiness of the data. From the correctly preprocessed AIS and contextual information, it is possible to extract complex events such as transshipping or smuggling. However, this requires either a discretization of the situation states or the hand-crafting of expert rules. At last, automated models often focus on one type of events.

Goal of our thesis

This work is focused on event detection and prediction: our goal is to create a model that learns the behavior of vessels using labeled events between entities of the situation (vessels, ports, companies...) and automatically detects the causes of these events to look for similar patterns elsewhere in the data. So instead of telling the model how to find the events, we feed it with events, AIS data and contextual information, so that it can learn the causes by itself.

CHAPTER 3

A formal framework and background for graphs

Representing a maritime situation requires some tools to characterize the entities, so we naturally turned to *knowledge graphs* for their effectiveness to describe objects and interactions. This chapter introduces the concept of a knowledge graph, and the notation framework and definitions that we will use through the manuscript.

3.1 Introduction to knowledge graphs

3.1.1 Resource Description Format

RDF stands for Resource Description Format. According to [Yu, 2007], it is:

- the basic building block for supporting the Semantic Web,
- to the Semantic Web what HTML has been to the Web,
- capable of describing any fact (resource) independent of any domain,
- structured i.e. machine-understandable. Machines can do useful operations with the knowledge expressed in RDF.

RDF works with triplets (Figure 3.1) of the form (Resource, Property, Object). A *resource* can be anything, described by an RDF expression. Whether it is a web page, an animal, an object, a person, it can be identified using a Uniform Resource Identifier (URI) that can be used to name the resource. This name is unique to avoid ambiguity between two RDF resources. A *property* is also a resource and can be described as such, but its purpose is to describe another resource: its attributes, its relation to other resources... At last, the *object* is the value of the property given to the resource. It can be the value of a characteristic (e.g. "blond" for the property "hair color") or the second member of a relation (e.g. a family link). The property value can be a string literal or a resource. Therefore, in general, an RDF triplet indicates that a resource (the subject) is linked to another resource (the object) via an arc labeled by a relation (the predicate) [Yu, 2007]. It can be interpreted as follow:

$$< subject > \textit{ has a property } < predicate >\textit{, whose value is } < object >.$$

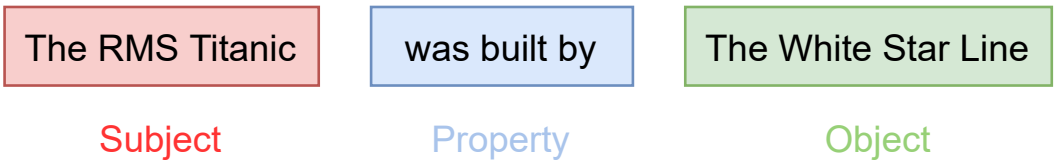


Figure 3.1: A RDF statement

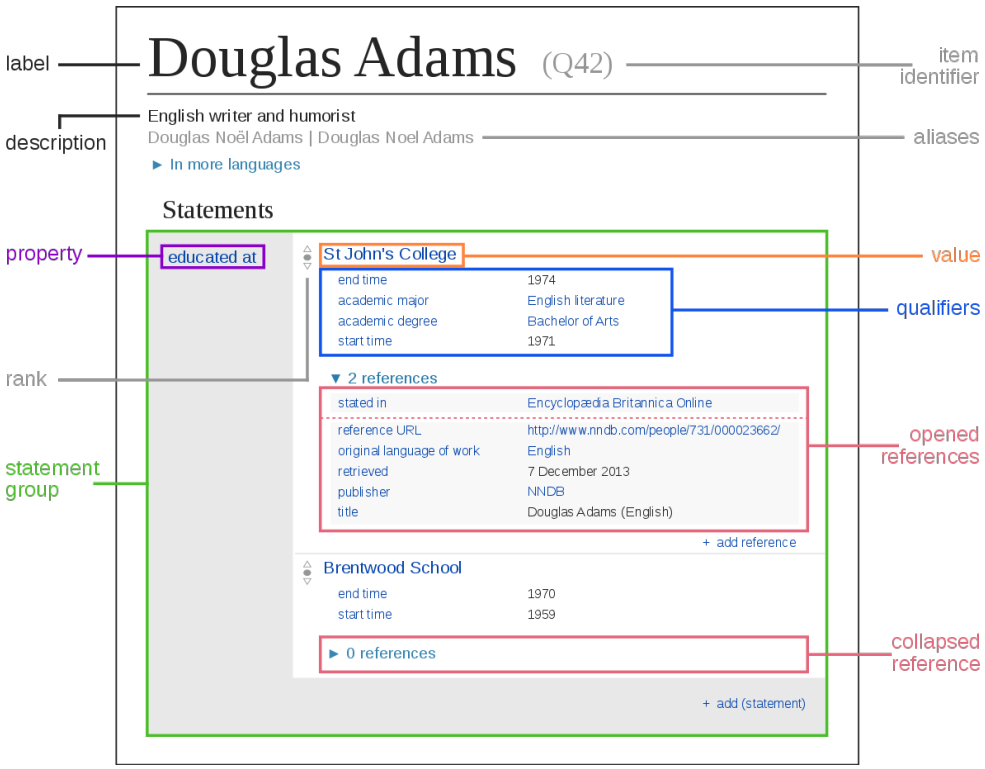


Figure 3.2: The ontology description of "Douglas Adams" (Wikidata)

3.1.2 Ontologies

An ontology is the descriptive schema of a field of knowledge. It defines the properties of each resource, whether they describe an attribute of the resource or how it can interact with others.

Figure 3.2 shows the schema for the writer and humorist Douglas Adams. The shown property is *educated at*, which describes his education. The object (or value) of this property is St John's College. If we follow the link to St John's College, we will also find a schema for this resource. Note that Douglas Adams is an instance of the class *human* and *person*, and that the property *educated at* is relevant to these classes and not only to Douglas Adams. We can also note that the property *educated at* has several qualifiers since it is also a resource. We thus learn that Mr Adams' education took place from 1971 to 1974, which introduces the notion of duration for an event (or at least, a starting date and an ending date). Many other properties are not shown here, such as his place and date of birth, his relatives, etc.

Figure 3.3 brings us back to the maritime application. It describes a part of the dat-ACRON ontology [Vouros et al., 2019] that specifies semantic trajectories. For instance, a Trajectory is composed of Trajectory Parts that have spatial and temporal features, and events such as Move, Stop, Fishing or Loitering occur during a specific part of a trajectory. These trajectories apply to a moving object, generally a vessel in our case (but it could be a

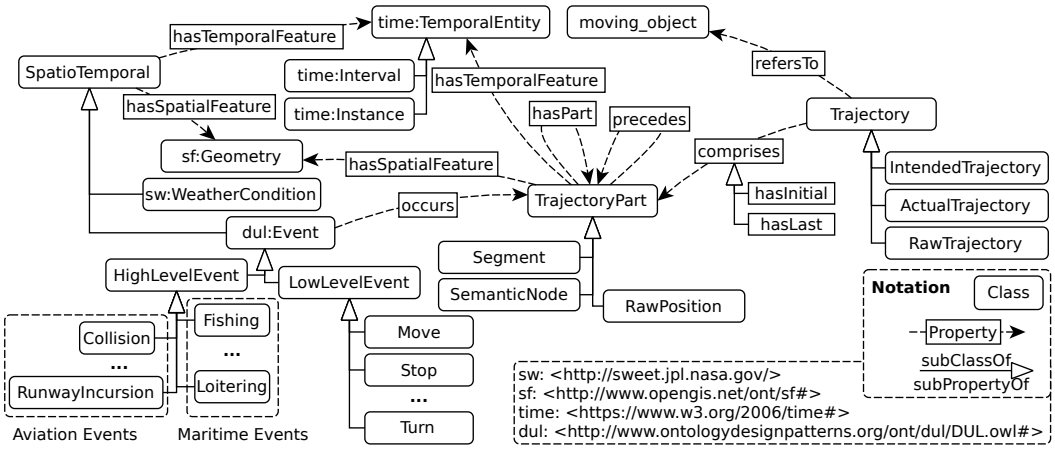


Figure 3.3: The maritime ontology from the datACRON project

plane). Such an ontology is a great help to define a maritime situation.

3.1.3 Knowledge graph

A knowledge graph (KG) represents the interactions between entities using the RDF standard. The nodes of the graph are entities and its edges are the facts that the relationships represent. [Nickel et al., 2016] call an event of a KG a Subject-Predicate-Object (SPO) triplet. This type of graph is oriented to know which node is the Subject and which one is the Object. Table 3.1 shows several examples of SPO triplets. It defines several predicates for the actor Leonard Nimoy, the fictional character Spock and the movie Star Trek. In Figure 3.4, this stack of SPO triplets is put into a graph and this makes clear that these three resources are linked. Similarly to RDF, the edges/predicates have labels that show the nature of the link.

subject	predicate	object
(LeonardNimoy,	profession,	Actor)
(LeonardNimoy,	starredIn,	starTrek)
(LeonardNimoy,	played,	Spock)
(Spock,	characterIn,	StarTrek)
(StarTrek,	genre,	ScienceFiction)

Table 3.1: SPO triplets examples from [Nickel et al., 2016]

With this structure, it becomes easier to get knowledge from the set of RDF triplets thanks to classical graph analysis tools. We can use the neighborhood of a node (e.g. find all Science Fiction movies), find a path between two entities to see if they are linked (is Alec Guiness related to Science Fiction?) or more generally perform question answering.

3.1.4 Closed and open world assumptions

Two assumptions can be made about the world we are trying to describe: the closed and open world assumptions [Nickel et al., 2016]. The *closed world assumption* (CWA) tells that if a triplet does not exist in the KG, then it is a false relationship, i.e., what happened is exclusively restricted to the available triplets. This assumption is convenient because it allows us to generate false events without a risk for them to be actually true. It is often used

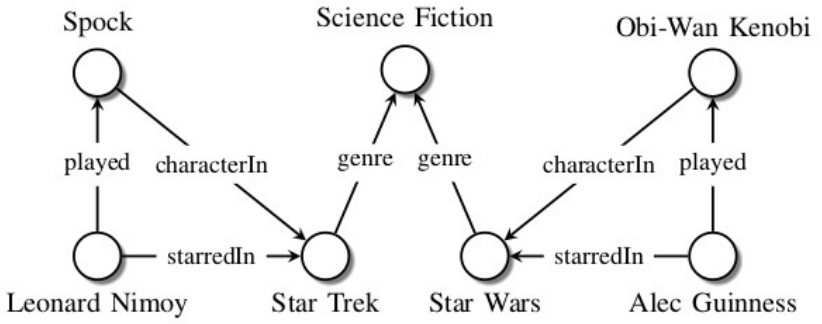


Figure 3.4: Sample knowledge graph [Nickel et al., 2016]. Nodes represent entities, edge labels represent types of relations, and edges represent existing relationships

as a reduction of real-world problems where we consider that the observer is omniscient. On the contrary, the *open world assumption* (OWA) considers the existing triplets to be true, but assumes nothing about the missing ones. Thus, a missing triplet could be a missed observation as well as a false fact.

The RDF standard makes the OWA [Nickel et al., 2016]. Since most KGs are incomplete, this is the safer option. It is also the best option for MSA: as stated in chapter 2, AIS observations can be missed and vessels can go under the radar. Missing observations will however be a difficult problem to solve, and one of our pain point in Chapter 6 when dealing with positive and negative events.

3.1.5 Schema-based and schema-free knowledge base

A knowledge base can have a strict way of describing entities or give more latitude to name them and their properties. We can thus define two main types of KGs [Nickel et al., 2016]:

- Schema-based KGs: each entity is referred to using a unique ID. For instance, the fact that Barack Obama is born in Honolulu is represented in Freebase by the triplet (/m/02mjmr, /people/person/born-in, /m/03gh4). The ID */m/02mjmr* is the unique way to refer to Barack Obama, and */people/person/born-in* is the unique property to tell a person's place of birth. Such a schema applies hierarchy and constraints on the KG, visible on the previous property: Barack Obama belongs to the class "person" which is a subclass of "people" (and then "human" and so on). At the same time, this property *"born-in"* can only apply to a person, and the object must be a *place*. In the MSA case, a schema-based KG would mean to rely exclusively on the definitions presented in the datACRON ontology.

- Schema-free KGs: in a schema-free KG, the same fact about Barack Obama can be described as ("Obama", "born in", "Hawaii"), as well as ("Barack Obama", "place of birth", "Honolulu"). The entities, properties and objects are not disambiguated and may have different identifiers (although here, "Honolulu" is more precise than "Hawaii"). It is easier to add information to a schema-free KG but also more difficult to keep it consistent and avoid information loss: if the same vessel is identified in two different ways, it could be considered as two distinct entities resulting in an incomplete analysis of the vessel. There are ways to identify a duplicated vessel (attributes and tracks comparison, name, captain...), but entity discovery is a field of research on its own between Natural Language Processing (NLP) and Data Fusion, and is not in the scope of this work.

Some KGs can be hybrid and use both systems: a schema defines the core structure of the KG, but any information can be added to it. This is the case for graph and document oriented databases such as OrientDB[1] that are specialized in representing KGs and semantic information.

3.1.6 Homogeneous and heterogeneous graphs

We have seen that the edges of a KG are the properties of the SPO triplets, and there are multiple types of properties. In the same way, nodes can be of different natures: such a KG is called a *heterogeneous* graph. Examples of such networks are those in the medical domain that describe patients, a bibliographic network that deals with authors, publications and organizations (like DBLP)[2] or event-based networks (like GDELT [Leetaru and Schrodt, 2013]).

However, some simpler networks have a single type of nodes and edges. Those are *homogeneous* networks and are largely represented by social networks (an edge = a friendship link) and the World Wide Web. According to [Getoor and Diehl, 2005], learning on homogeneous graphs is "consistent with the classical statistical inference problem of trying to identify a model given an independent, identically distributed sample". But applying such techniques to heterogeneous graphs can give misleading conclusions on the data since the different types of relations and nodes are not taken into account. The semantic information around the network must be used to improve the accuracy of the learned models. A KG build for MSA is obviously a heterogeneous network given the complexity and the variety of the possible interactions.

3.1.7 An introduction to dynamicity and attributes

All the KGs presented until now are *static* graphs. Each SPO triplet is considered to be valid, but there is no indication as to when it happened. Of course, properties can introduce dates, for instance ("The RMS Titanic", "built_in", "1912"). But what to do with the triplet ("Vessel1", "enter_in", "Port2")? A surveillance operator will indeed know that a specific vessel has entered a designated port, but when? The quadruplet ("Vessel1", "enter_in", "Port2", "24 November 2020, 16:34:22") is much more useful to follow the whereabouts of the vessels. A set of such quadruplets forms a dynamic (or temporal) knowledge graph.

Moreover, the current SPO format does not distinguish the relational triplets from the descriptive ones. The ("Vessel1", "enter_in", "Port2") triplet refers to an event, but ("Vessel1", "has_size", "125m") defines how Vessel1 *is*. Structurally, it is an SPO triplet like another, but the type of information it gives is different. From now on, we call the first type of triplets *relational* and the second *attributional*. To give a more formal definition, the Object of an attributional triplet is not considered as an entity (e.g. "125m", which is a value). However, the line between the two can sometimes be blurry. If we consider the triplet ("Vessel1", "has_flag", "France"), what is "France"? It can either be the entity France with its own relations to other entities, or a part of the set of discrete possible values for the attribute "flag". It depends of the context of the situation, but generally, if the Object is used as an entity elsewhere, then it is a relation. These choices must be reflected by the schema used to build the KG. Our choices will be detailed for each dataset under study in Chapter 7.

[1] `https://www.orientdb.org/`
[2] `https://dblp.org/`

3.1.8 Embeddings

Embeddings are continuous vector representations that have been popularized in NLP tasks by *word2vec* [Mikolov et al., 2013], and in knowledge graph processing by *TransE* [Bordes et al., 2013]. An embedding represents an object by projecting it to a latent vector space, so as to retain its features and make it comparable to the others.

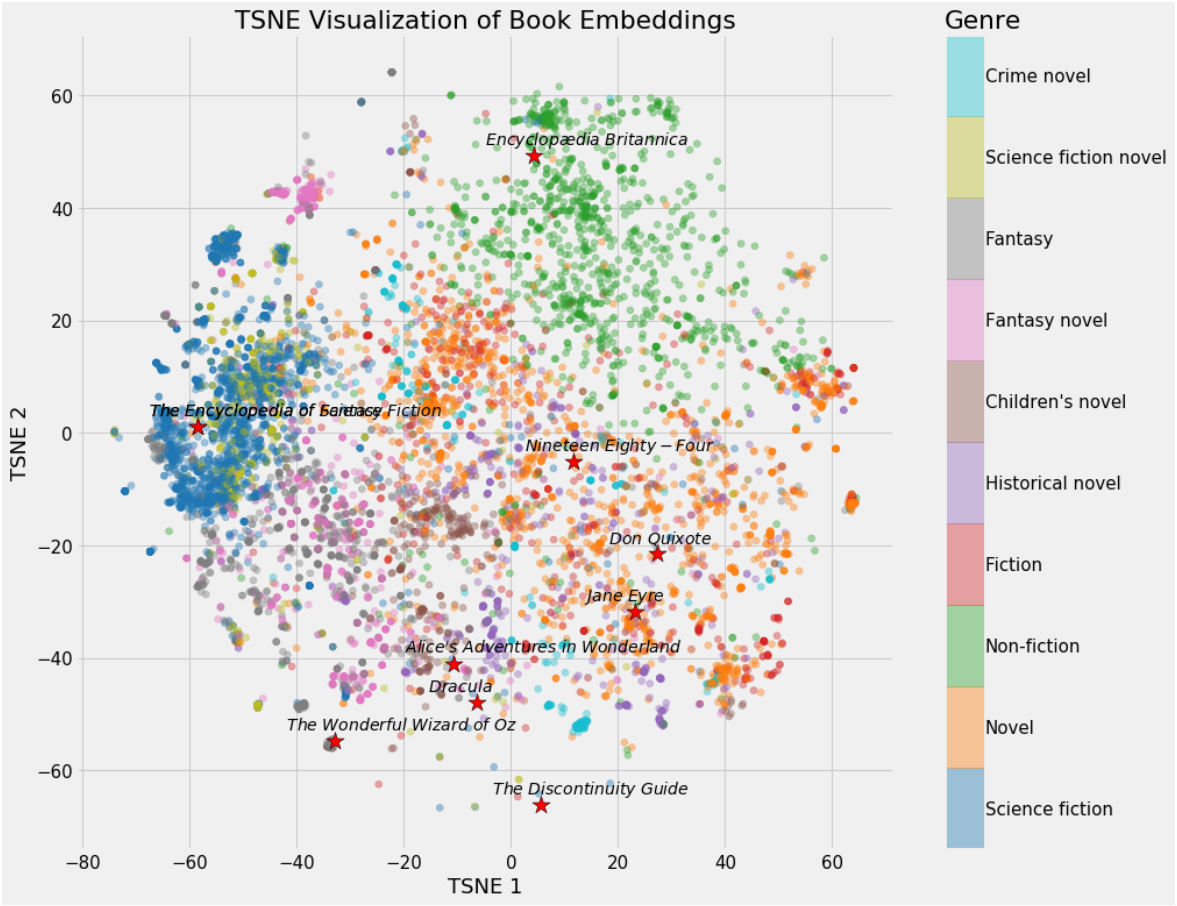


Figure 3.5: Embeddings of book for a recommendation system (TowardsDataScience)

Figure 3.5 is an example of how embeddings can be used to classify entities. Books are represented by their embedding, and the t-Distributed Stochastic Neighbor Embedding (T-SNE) visualization shows which books have similar embeddings. To recommend a book based on one the user read, one just has to look for other books close to this one in the visualization. The figure also shows that books are grouped by genre, meaning that the embeddings retain the features of the books well. These embeddings can be used to run nearest neighbors algorithms (e.g. for recommendations like above), to compute similarity measures or as input for downstream machine learning tasks. The same process can be applied to entities of a maritime situation (Table 3.2: features and movements of vessels can be projected into an embedding and be used to learn how two of them interact). We will use embeddings as feature vectors that can be regularly updated, and we will use one-hot vectors to access them from the matrix they are stored in.

Entity	*Dim0*	*Dim1*	...	*Dim48*	*Dim50*
Vessel1	0.38263	0.71627	...	-0.32834	-0.01238
Port1	0.83623	-0.09254	...	0.87362	0.47263
Vessel3	-0.71823	0.51023	...	0.18364	0.12943

Table 3.2: Embedding vectors for entities

In a nutshell:

- The RDF standard describes facts as (Resource, Property, Object) triplets;
- An ontology is a schema that defines how the Resources and Properties work and generally represents a field of study;
- A knowledge graph (KG) is a set of RDF triplets that represents interactions between entities of a situation;
- The close world assumption considers all absent triplets to be false while the open world assumption tells nothing about them;
- A schema (or ontology) is not mandatory for a KG, but it is advised;
- KGs with node and edge types are more difficult to use but are more precise;
- KGs can include temporal information and attributes for nodes;
- Embeddings can hold the features of the entities for further analysis.

3.2 Framework

In this section, we present the notation we will use through this manuscript to describe knowledge graphs and their representations. A large part of the works in the literature does not use the same notation and our goal is to have a unified writing when presenting them. We consider three settings for KG: static, dynamic, and attributive. Note that a KG can be static and attributive, or dynamic and attributive.

3.2.1 Static Knowledge Graphs

A static knowledge graph refers to a graph with no temporal information, as described in Section 3.1.3. From now on, we will refer to the components of an SPO triplet as *entities* and *relations* that respectively define the nodes (or vertices) and edges of the KG.

Definition 1 (static relational KG)**.** *Let* E *and* R *be two finite sets of* entities *and* relations*, respectively. A* static relational knowledge graph KG^{R} *is a finite subset of* $\mathrm{E} \times \mathrm{R} \times \mathrm{E}$*. For a triplet* $p^{\mathrm{R}} = (e^s, r, e^o) \in \mathrm{KG}^{\mathrm{R}}$*,* e^s *is called the* subject *of* p^{R}*,* r *its* relation*,* e^o *its* object*.*

In some settings, a static KG can be composed of a relational KG *and* an attributive KG. This KG describes the attributes given to each entity using a network just like the relational one.

Definition 2 (static attributive KG)**.** *Let* A *be a finite set of* attributes*, and* D *be a function assigning a* domain $\mathrm{D}_a \subseteq \mathbb{R}$ *to each attribute* $a \in \mathrm{A}$*. An attributive graph* KG^{A} *is a finite subset of* $\mathrm{E} \times \mathrm{A} \times \mathbb{R}$ *such that for all triplets* $p^{\mathrm{A}} = (e, a, v) \in \mathrm{KG}^{\mathrm{A}}$*,* $v \in \mathrm{D}_a$ *holds. For a triplet* p*,* e *is called its* entity*,* a *its* attribute type *and* v *its* attribute value*.*

Note that the function D of Definition 2 is for continuous values, but some domains can also be discrete. In this case, we have $\mathrm{D}_a \subseteq \mathrm{V} \subseteq \mathbb{R}$, with V a set of discrete values.

3.2.2 Dynamic Knowledge Graphs

When time is taken into account, the static KG becomes a dynamic (or temporal) KG, written *DKG*. Time-aware quadruplets are defined as follows.

Definition 3 (relational *DKG*)**.** *Let* E *and* R *be two finite sets of* entities *and* relations*, respectively, and let* T *be a set of time points. A dynamic* relational knowledge graph DKG^{R} *is a finite subset of* $\mathrm{E} \times \mathrm{R} \times \mathrm{E} \times \mathrm{T}$*. For a quadruplet* $q^{\mathrm{R}} = (e^s, r, e^o, t) \in DKG^{\mathrm{R}}$*,* e^s *is called the* subject *of* q^{R}*,* r *its* relation*,* e^o *its* object *and* t *its* timestamp*.*

Definition 4 (attributive *DKG*)**.** *Let* A *be a finite set of* attributes*, and* D *be a function assigning a* domain $\mathrm{D}_a \subseteq \mathbb{R}$ *to each attribute* $a \in \mathrm{A}$*. An dynamic attributive knowledge graph* DKG^{A} *is a finite subset of* $\mathrm{E} \times \mathrm{A} \times \mathbb{R} \times \mathrm{T}$ *such that for all quadruplets* $q^{\mathrm{A}} = (e, a, v, t) \in DKG^{\mathrm{A}}$*,* $v \in \mathrm{D}_a$ *holds.*

We will refer to a pair $DKG = (DKG^{\mathrm{R}}, DKG^{\mathrm{A}})$ as a (full) knowledge graph, whose relational (resp. attributive) part will always be written DKG^{R} (resp. DKG^{A}). This is also valid for KG, KG^{R} and KG^{A}. For notational brevity, we will often simply write (s, r, o, t) for a relational quadruplet, and we will use q instead of $q^{\mathrm{R}}, q^{\mathrm{A}}$ when the type of quadruplet is clear from the context. Figure 3.6 gives an example *DKG*. The nodes $e_i \in \mathrm{E}$ are *entities* and the nodes $a_j \in \mathrm{A}$ are *attributes*. The *relations* $r_n \in \mathrm{R}$ are written on the edges between two

entities. The *values* v_k, written on the edges between an entity and an attribute, belong to the *domain* D_a of the attribute they are attached to, and notation t_ℓ denotes the *timestamp* of an edge. Attributes of entities can change their value over time (*e.g.*, e_3 and a_2), and two entities can share attributes (e_4, e_1 and a_1), but not necessarily with the same value. The blue nodes and edges together make DKG^{R}, and the red edges with all the nodes together make DKG^{A}. In a static context, the same notation applies with KG in place of *DKG*.

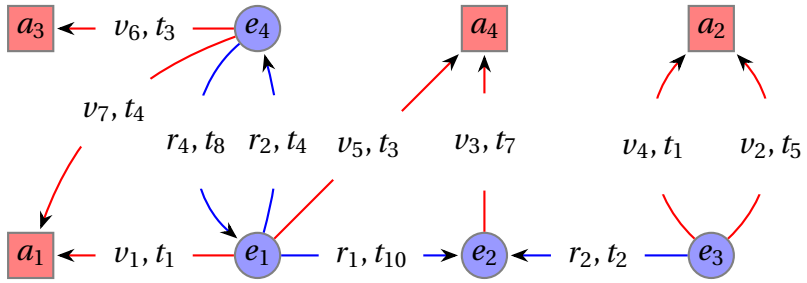


Figure 3.6: Example of a knowledge graph *DKG*.

Sliced DKG Definitions 3 and 4 refers to a *streamed DKG*, where the timestamps are not fixed in advance and where each timestamp is the timestamp of a quadruplet (or observation). In the case of a *sliced DKG*, the graph is a sequence of snapshots from the dynamic graph. A sliced *DKG* (S-*DKG*) is formally defined by S-$DKG = \{DKG^1, DKG^2, \ldots, DKG^\tau\}$, where DKG^t is the state of the graph at time $t \in \mathrm{T} = [1, \tau]$. For instance, slicing applied to MSA means that we create a snapshot of the situation every 5 minutes, while streaming means that we update the situation each time we receive a new observation. The use of snapshots may induce a loss of information but creating models on them is generally easier [Kazemi et al., 2019]. In this work, we use streaming *DKG* since dealing with streaming data is one of the requirements to achieve MSA.

3.2.3 Embeddings and history

As mentioned in Section 3.1.8, entities can be represented by embeddings. In a static setting, the embedding models all the known triplets of the KG. It can be defined as follows for attributes and relations.

Definition 5 (static embedding of attributive history)**.** *A* static attributive embedding function *is a function which maps an entity* e *and a set of timestamped attributive triplets* $\mathrm{KG}_e^{\mathrm{A}}$ *for* e *to a vector in* $\mathbb{R}^m$*, written* $\boldsymbol{\alpha}_e$ *and called the* attributive embedding of *e.*

Definition 6 (static embedding of relational history)**.** *A* static relational embedding function *is a function which maps an entity* e *and a set of timestamped relational triplets* $\mathrm{KG}_e^{\mathrm{R}}$ *for* e *to a vector in* $\mathbb{R}^n$*, written* $\boldsymbol{\rho}_e$ *and called the* relational embedding of *e.*

In a dynamic context, the entities of a *DKG* evolves over time and the embedding can change accordingly. An embedding set at time t takes into account the history of its entity until time t and is defined as follows:

Definition 7 (embedding of attributive history)**.** *Let* $DKG_e^{\mathrm{A},<t}$ *be the restriction of the DKG to the attributive quadruplets of entity* e *in* A *before time* t *and let* $\boldsymbol{\alpha}_e^{t^-}$ *be the embedding of*

the history of all the attributes of entity $e \in \mathrm{E}$ *before time* t. *The function* $f : DKG_e^{\mathrm{A},<t} \rightarrow \mathbb{R}^m$ *is the function that maps the history to an embedding of size* m.

Definition 8 (embedding of relational history)**.** *Let* $DKG_e^{\mathrm{R},<t}$ *be the restriction of the DKG to the relational quadruplets of entity* e *in* R *before time* t *and let* $\boldsymbol{\rho}_e^{t-} = f(DKG_e^{\mathrm{R},<t})$ *be the embedding of the history of all the relations/events involving entity* $e \in \mathrm{E}$ *before time* t. *The function* $f : DKG_e^{\mathrm{R},<t} \rightarrow \mathbb{R}^n$ *is the function that map the history to an embedding of size* n.

[Feng et al., 2016b] characterizes three types of graph context:

- The first is the neighbor context, or the set of triplets in KG;
- The second is path context: a path is a sequence of relations that connects two entities;
- The last and third is edge context, i.e. the set of relation types the entity has with others. For instance, Douglas Adams is likely to be included in relations such as :bornIn, :diedIn, :hasWife or :isAuthorOf, all indicating that he is a Person.

These three contexts are the main ones that are used in the literature and they are often used in embedding techniques to define what is taken into account in the embedding.

3.3 Conclusion

We laid out the theoretical context for knowledge graphs and entity embeddings. We define the prediction problem at hand as a link and attribute values prediction problem in a dynamic graph, both relational and attributive ($DKG^{\mathrm{R}}, DKG^{\mathrm{A}}$). Observed events will be formalized as relational and attributive quadruplets, and each entity will be given a relational embedding *and* an attributive embedding. Using the notation we just defined (summed up in Table 3.3), we will now review the literature for embedding and link prediction in static (attributed) and dynamic (attributed) graphs to find what is better suited to improve MSA.

Symbol(s) or abbreviation	Meaning
E, R, A, D_a, T	The sets of entities, relations, attribute types, domains for attributes, timestamps
e, r, a, t	An entity(or a node) from the graph, a relation type, an attribute type and a timestamp
e^s, e^o, s, o	Subject and object entities and their abbreviated notations
$p^{\mathrm{R}}, p^{\mathrm{A}}$	Relational and attributive triplets
$q^{\mathrm{R}}, q^{\mathrm{A}}$	Relational and attributive quadruplets
(D)KG^{R}, (D)KG^{A}	Sets of relational and attributive triplets (quadruplets) of a (dynamic) KG.
DKG, S-DKG	Streamed *DKG* and sliced DKG
τ	The number of snapshots in an S-*DKG*
$\boldsymbol{\alpha}_e, \boldsymbol{\rho}_e$	Relational and attributive embeddings of e in a KG
$\boldsymbol{\alpha}_e^t, \boldsymbol{\rho}_e^t$	Relational and attributive embeddings of e in a *DKG* at time t
m, n	The sizes of attributive and relational embeddings (resp.)
l	The size of the relation type embeddings

Table 3.3: Summary of the notation

CHAPTER 4

Embedding and link prediction in knowledge graphs

This chapter is dedicated to the state of the art of embedding and prediction techniques for (knowledge) graphs. We first describe how the knowledge graphs can be used to get new information, and then tackle the different embedding techniques by types of graphs: static, attributed, and dynamic. The last section is dedicated to the addition of semantic information in the processing of the graph. Everything in this section will be presented using the notation defined in Chapter 3.

4.1 Knowledge graphs applications

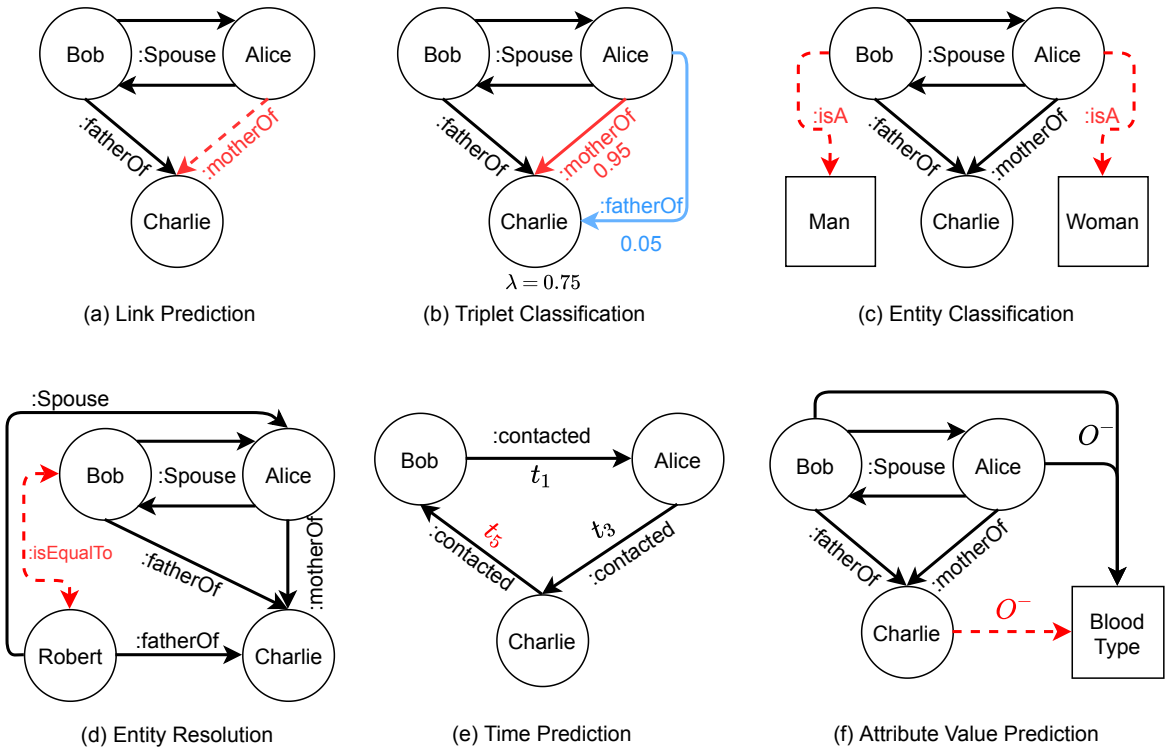
Knowledge discovery in graphs can take several shapes and work at different levels, such as edge level, entity, triplet/quadruplet or timestamp. Figure 4.1 sums up the different tasks. In this work, we will focus on the link prediction task and, to a lesser extent, on the attribute value prediction problem. Parameters of models are learnt on data with loss functions, which depend on the usecases. Each reasonable training technique can be used, and we will make precise which ones suit our model.

4.1.1 Link prediction

The link prediction task consists of finding an edge between two nodes that was missing from the graph. In the case of dynamic graphs, it can refer to a link in the future, which is a more difficult task since we generally have little information about the future in the graph, whereas predicting missing values can be performed using past timestamps and ground truth.

Definition 9 (Link prediction)**.** *Let* KG^{R} *be a relational knowledge graph and* $p^{\mathrm{R}} = (s, r, o) \notin \mathrm{KG}^{\mathrm{R}}$ *be a relational triplet, with* $s, r \in \mathrm{E}$ *and* $r \in \mathrm{R}$*. Given two out of the three components of* p^{R} *and* KG^{R}*, the link prediction task aims at finding the third component so that* p^{R} *is true. For instance, for subject prediction, it comes down to finding the function* $f : \mathrm{KG}^{\mathrm{R}}, r, o \mapsto (s, r, o)$.

An application could be to answer *"Which vessel is currently transshipping fish (r) with vessel o"?*, and the prediction would be *"Vessel s is currently transshipping fish (r) with o"*.



(a) Link Prediction (b) Triplet Classification (c) Entity Classification

(d) Entity Resolution (e) Time Prediction (f) Attribute Value Prediction

Figure 4.1: Summary of main knowledge graphs applications (edges in color are the one to predict)

Link prediction models are often specialized, either in subject and object prediction [Bordes et al., 2013], [Wang et al., 2014], [Lin et al., 2015b], [Nickel et al., 2015] or in relation prediction [Xie et al., 2016], [Lin et al., 2015a].

4.1.2 Attribute value prediction

Most KGs are incomplete and a missing value for an attribute is not unusual. A common task in KGs is to find these values using other attributes.

Definition 10 (Attribute value prediction)**.** *Let* KG^{A} *be an attributive knowledge graph and* $p^{\mathrm{A}} = (e, a, v) \notin \mathrm{KG}^{\mathrm{A}}$ *be an attributive triplet, with* $e \in \mathrm{E}$*,* $a \in \mathrm{A}$ *and* $v \in \mathrm{D(A)}$*. Given* e *and* a *of* p^{A} *and* KG^{A}*, the attribute value prediction task aims at finding* v *so that* p^{A} *is true. It comes down to finding the function* $f : \mathrm{KG}^{\mathrm{A}}, e, a \mapsto (e, a, v)$*.*

A straightforward example is the prediction of a vessel speed in a dynamic setting, or its length if it is missing in a static context.

4.1.3 Triplet classification

Triplet classification takes the triplet as a whole and determines if it is true or false. This is generally done by computing a score for the triplet with a thresholding step [Wang et al., 2014, Lin et al., 2015b].

Definition 11 (Triplet classification)**.** *The* triplet classification problem *takes as input a part of a relational graph* KG^{R} *and a relational triplet* $p^{\mathrm{R}} = (s, r, o)$*, and asks whether* (s, r, o) *is in* KG^{R}*.*

This is usually done by estimating a likelihood $f(\mathrm{KG}^{\mathrm{R}}, p^{\mathrm{R}})$, and answering $p^{\mathrm{R}} \in \mathrm{KG}^{\mathrm{R}}$ if and only if $f(\mathrm{KG}^{\mathrm{R}}, p^{\mathrm{R}}) \geq \lambda$ for a given threshold λ. This problem models questions such as *"Is boat s currently performing illegal fishing (r) in the Mediterranean Sea (o)?"*.

4.1.4 Other applications

Other applications exist, though less connected to our usecase. Entity classification is when the class of an entity e is not known, but can be deduced from the information in the graph. It boils down to solving the triplet $(e, :isA, x)$ where x is a type of entity, e.g. (Vessel1, :isA, fishingVessel) [Nickel et al., 2011, Nickel et al., 2012].

Resolving an entity, as mentioned in Section 3.1.5, is the task that de-duplicates nodes written in different ways [Nickel et al., 2011, Bordes et al., 2014].

Most of the tasks aim to find an entity or a relation type, but sometimes it is the time associated to the edge that is important. To find this timestamp, a triplet and the *DKG* are presented to the model that will output the most probable timestamp for this edge.

4.2 Static graphs

This section describes the state of the art on methods that deal with static graphs (KGs). This is not an exhaustive presentation of the static graph bibliography, and more details can be found in the surveys by [Wang et al., 2017] and [Cai et al., 2018].

4.2.1 Translational models

In a static setting, each node is represented by a single vector. This field is largely covered with a broad range of techniques. Translational models evaluate the likelihood of a fact (subject, relation, object) by measuring the distance between the two entities, typically using the relation during the translation. TransE [Bordes et al., 2013] is the most representative translational distance model and has been extended multiple times. A summary of all methods is presented in Figure 4.2.

TransE TransE represents both entities and relations as vectors in the same space R^n. Given a fact $p^{\mathrm{R}} = (s, r, o)$, the objective is to create a vector for r so that the embeddings of s and o are connected by r with low error. This comes down to minimizing the distance between $\rho_s + \rho_r$ and ρ_o. The scoring function is then defined as the (negative) distance between s + r and o, i.e.,

$$f_r(s, o) = -\|\boldsymbol{\rho_s} + \boldsymbol{\rho_r} - \boldsymbol{\rho_o}\|_{1/2}$$

The score is expected to be large if (s, r, o) holds, i.e. is an existing triplet in KG^{R}. But this method has problems dealing with 1-to-N, N-to-1 and N-to-N relations.

TransE improvements Several enhanced versions of TransE have been proposed by different authors. To overcome the problems of TransE, [Wang et al., 2014] propose a model which enables an entity to have distributed representations when involved in different relations: Translating on Hyperplanes (TransH). By introducing relation-specific hyperplanes, TransH can identify the interactions between entities better. TransR [Lin et al., 2015b] is very similar to TransH but it introduces relation-specific spaces rather than hyperplanes. Its

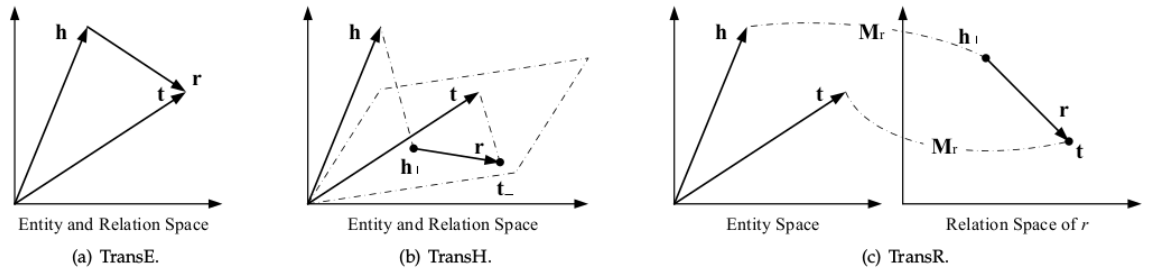


Figure 4.2: Simple illustrations of TransE, TransH, and TransR from [Wang et al., 2017]. The figures are adapted from [Wang et al., 2014], [Lin et al., 2015b]

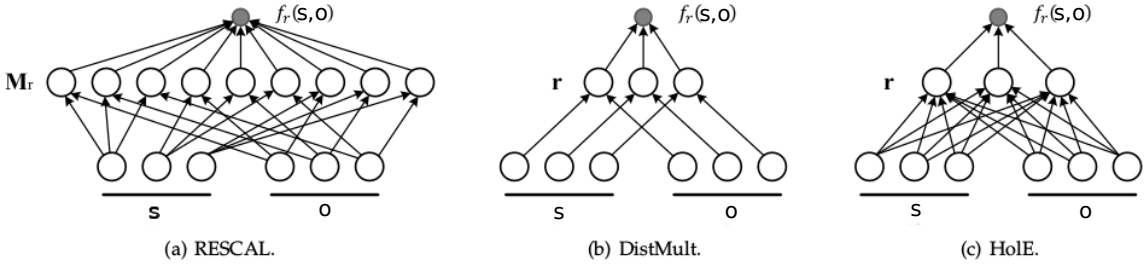


Figure 4.3: Simple illustrations of RESCAL, DistMult, and HolE adapted from [Wang et al., 2017].

main issue is the loss of simplicity and the higher number of parameters per relation due to the use of a matrix instead of a vector. TransD [Ji et al., 2015] simplifies TransR by replacing the projection matrix by a product of two vectors. TransSparse [Ji et al., 2016] also simplifies TransR by enforcing sparseness on the projection matrix. Some other extensions make improvements by relaxing the translational constraint, i.e. $\rho_s + \rho_r \approx \rho_o$ [Fan et al., 2014, Xiao et al., 2016, Feng et al., 2016a, Xiao et al., 2015]. Overall, the upgrades on TransE aim to deal with different types of relations while keeping the complexity to a minimum.

4.2.2 Semantic matching models

Semantic matching models use scores to compare the embeddings of entities and relation types, based on their similarity. An illustration of several methods is presented in Figure 4.3.

4.2.2.1 RESCAL and its extensions

RESCAL [Nickel et al., 2011] was the first semantic matching method and has been extended multiple times [Nickel et al., 2015, Liu et al., 2017].

RESCAL RESCAL [Nickel et al., 2011] assigns an embedding vector to each entity and a matrix to each relation type, which models the interaction between two entities. The score of a fact (s, r, o) is defined by a bilinear function

$$f_r(s,o) = \boldsymbol{\rho}_{\boldsymbol{s}}^{\mathsf{T}} \mathbf{M}_r \boldsymbol{\rho}_{\boldsymbol{o}} = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} [\mathbf{M}_r]_{ij} * [\boldsymbol{\rho}_{\boldsymbol{s}}]_i * [\boldsymbol{\rho}_{\boldsymbol{o}}]_j$$

where $\mathbf{M}_r \in \mathrm{R}^{n*n}$ is the matrix associated with the relation type. This score captures how the entities interact by multiplying the latent features of the entities and relations. It requires $\mathrm{O}(n^2)$ parameters per relation.

Extensions of RESCAL Like TransE, RESCAL has been extended multiple times. TATEC [García-Durán et al., 2014] models the three-way interactions like RESCAL (entity-relation-entity), but also the two-way interactions, i.e. each entity with the relation type. DistMult [Yang et al., 2014] simplifies RESCAL by reducing the number of parameters. HolE [Nickel et al., 2015] takes the best of RESCAL and DistMult to be simple *and* efficient. ComplEx [Trouillon et al., 2016] extends DistMult by creating the embeddings in the complex space $\mathbb{C}^d$. ANALOGY [Liu et al., 2017] creates representations that model analogical properties of entities and relations, for instance "Douglas Adams" is to "Hitchhiker's Guide to the Galaxy" what "Georges Lucas" is to "Star Wars". It uses the same bilinear score as RESCAL but requires the matrices M_r to be normal and mutually commutative.

4.2.2.2 Matching with neural networks

Neural Tensor Network NTN [Socher et al., 2013] is one of the best-known neural network architecture for graphs. The input layer of the network project the entities into their embedding space, that are then combined using a tensor $\mathbf{M}_r \in \mathrm{R}^{n \times n \times k}$ that is relation specific. It resembles TATEC insofar the subject and object are combined using the relation type representation, but it also has the two-ways interactions subject-relation and object-relation. These interactions are then given to a non-linear hidden layer and then to a relation specific layer that outputs the following score:

$$f_r(s,o) = \rho_r^\intercal \, tanh(\rho_s^\intercal \underline{\mathbf{M}}_r \rho_o + \mathbf{M}_r^1 \rho_s + \mathbf{M}_r^2 \rho_o + \boldsymbol{b}_r)$$

with $\underline{\mathbf{M}}_r \in \mathbb{R}^{n \times n \times k}$ and $\mathbf{M}_r^1, \mathbf{M}_r^2 \in \mathbb{R}^{k \times n}$ relation-specific weight matrices and $b_r \in \mathrm{R}^k$ a bias vector (also relation-specific). It is one of the most expressive models to date but it has two main drawbacks: it overfits on small and medium datasets which leads to not so good scores, and it requires $\mathrm{O}(n^2 k)$ parameters per relation, which makes it too computationally expensive for large-scale KGs.

Multi-layer Perceptron MLP [Dong et al., 2014] is simpler than NTN: it concatenates the embeddings of each member of the triplet in the input layer and then passes them through a non-linear hidden layer:

$$f_r(s,o) = \boldsymbol{w}^\intercal \, tanh(\mathbf{W}_s \boldsymbol{\rho_s} \oplus \mathbf{W}_r \boldsymbol{\rho_r} \oplus \mathbf{W}_o \boldsymbol{\rho_o})$$

with $\mathbf{W}_s, \mathbf{W}_r, \mathbf{W}_o \in \mathbb{R}^{n \times n}$ the weights of the first layer, $\boldsymbol{w} \in \mathbb{R}^{3n}$ the weights of the second layer and $\oplus$ the concatenation operator.

RDFDNN RDFDNN [Murata et al., 2017] is a neural-based system that predicts the relation r of a triplet p^{R}. It first processes the embeddings of the subject and the object through the same layer and concatenates them. It then goes through a second hidden layer and a softmax layer that gives the predicted relation type for the triplet.

$$h_r(s,o) = tanh(\mathrm{W}_1 \boldsymbol{\rho_s}) \oplus tanh(\mathrm{W}_1 \boldsymbol{\rho_o})$$

$$f_r(s,o) = softmax(\mathrm{W}_3 \, tanh(\mathrm{W}_2 h_r(s,o)))$$

with $\mathrm{W}_1 \in \mathbb{R}^{n \times n}, \mathrm{W}_2 \in \mathbb{R}^{\ell \times 2n}$ and $\mathrm{W}_3^{\ell \times |\mathrm{R}| \times \ell}$ the weight matrices of the network, with $|\mathrm{R}|$ the number of relation types. RDFDNN has better results than the methods it compares itself to (TransE, TransR), but it is only good at predicting r, not s and o.

Graph Convolutional Network Graph Convolutional Network (GCN) [Kipf and Welling, 2016] encodes both the local graph structure and the features of nodes into embeddings. The use of the adjacency matrix of the graph allows it to make use of each node neighborhood when building its representation. It performs well on node classification and its time complexity grows linearly in the size of the dataset, but it does not naturally support edge features and is limited to undirected graphs. A simple version of the convolutional model, a two-layer GCN for semi-supervised node classification, can be represented as follows:

$$f(\mathrm{X}, \mathrm{A}) = \mathrm{softmax}(\widehat{\mathrm{A}}\ \mathrm{ReLU}(\widehat{\mathrm{A}}\mathrm{X}\mathrm{W}^{(0)})\mathrm{W}^{(1)})$$

with $\mathrm{X} \in \mathbb{R}^{n \times |\mathrm{E}|}$ the matrix of node features, $\mathrm{W}^0 \in \mathbb{R}^{n \times c}$ and $\mathrm{W}_1 \in \mathbb{R}^{c \times d}$ the weight matrices of the network with c the number of features for the first layer and d the number of labels for node classes, A the adjacency matrix, D the degree matrix and $\widehat{\mathrm{A}} = \tilde{\mathrm{D}}^{\frac{-1}{2}}\tilde{\mathrm{A}}\tilde{\mathrm{D}}^{\frac{-1}{2}}$.

The GCN model is extended by [Schlichtkrull et al., 2017] to add relation types (R-GCN). The forward-pass update of an entity is expressed as:

$$h_i^{(\ell+1)} = \mathrm{ReLU}(\sum_{r \in \mathrm{R}} \sum_{j \in \mathrm{N}_i^r} \frac{1}{c_{i,r}} \mathrm{W}_r^{(\ell)} h_j^{(\ell)} + \mathrm{W}_0^{(\ell)} h_i^{(\ell)})$$

with N_r^i is the set of neighbor indices of node i under relation $r \in \mathrm{R}$, $c_{i,r}$ the problem-specific normalization constant and $h_j^{(\ell)}$ and $h_i^{(\ell)}$ the hidden representations of nodes j and i at layer ℓ. W_r is a relation-specific weight matrix and W_0 a transverse weight matrix.

4.2.2.3 Walking through a graph

The following methods uses random walks to generate node features. These methods are less node-centered and allow the exploration of a node neighborhood, i.e. its context.

node2vec node2vec [Grover and Leskovec, 2016] is a semi-supervised algorithm that learns embeddings for nodes that maximizes the likelihood of preserving network neighborhoods of nodes. The approach is flexible and explores diverse neighborhoods of a given node using 2^{nd} order random walks to generate the neighborhood. By interpolating between breadth-first and depth-first sampling, node2vec can learn local graph structures but also explore further with higher variance in the creation of the paths. Once the neighborhood extracted as a set of paths, a Skip-gram architecture [Mikolov et al., 2013] is used to learn the representation with nodes as the vocabulary and paths as sequences of words.

The 2^{nd} order random walk is defined as follows: for a walk that went from node e_t to node e_v, the transition probabilities π_{vx} between node e_v and nodes e_x with $x \in \mathrm{E}$ is set to $\pi_{vx} = \beta_{pq}(e_t, e_x) \cdot w_{vx}$, where

$$\beta_{pq}(e_t, e_x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

w_{vx} is the static edge weight and d_{tx} denotes the shortest path between nodes e_t and e_v. The parameter p controls the likelihood of immediately revisiting a node: a high value will mostly lead to an unseen node while a low value fosters backtracking. On the other hand, the parameter q sets the stance of the walk towards breadth-first (high value) or depth-first (low value) sampling. This sampling strategy is flexible and efficient in time and space complexity. It is used for multi-label node classification and link prediction, but only works on homogeneous graphs.

LINE LINE [Tang et al., 2015] minimizes the first-order and second-order proximities of node embeddings to learn the representations. The first-order proximity, or pairwise connection through an edge, is defined through the following joint probability:

$$p_1(e_i, e_j) = \frac{1}{1+exp(-\boldsymbol{\rho}_i^\intercal \cdot \boldsymbol{\rho}_j)}$$

The second-order proximity looks for nodes which have the same neighborhood or context. To do so, it computes the sets of first-order proximities between a node and all the other nodes and compares the obtained distributions. A new embedding $\boldsymbol{\rho}_e^{'}$, the representation vector of the entity when treated as a context, is introduced. For a directed edge (e_i, e_j), the probability of "context" e_j generated by vertex e_i is defined as:

$$p_2(e_j|e_i) = \frac{exp(\boldsymbol{\rho}_j^{'\intercal} \cdot \boldsymbol{\rho}_i)}{\sum_{k=1}^{|\mathrm{E}|} exp(\boldsymbol{\rho}_k^{'\intercal} \cdot \boldsymbol{\rho}_i)}$$

The first-order proximity works only on undirected graphs and the second-order one with both directed and undirected. It is possible to use both orders by training the models separately and concatenating the resulting vectors.

GraphSAGE GraphSAGE [Hamilton et al., 2017] is an inductive embedding method that samples and aggregates features form the local neighborhood of a node so that it can generalize to unseen nodes. The embedding generation unfolds as follows:

- Get node neighborhood;
- Aggregate embeddings of the neighborhood;
- Concatenate node embedding with neighborhood embedding;
- Process the concatenated vectors with a fully connected layer and a nonlinear activation function;
- Normalize;
- Repeat for K steps, with K the depth of the model.

The neighborhood is uniformly sampled from E with a fixed size to reduce the neighborhood space. The authors test three types of aggregators. The mean aggregator takes the elementwise mean of vectors in the neighborhood $\mathrm{N}(e_v)$. It is close to the convolutional propagation rule used in GCN [Kipf and Welling, 2016].

$$\boldsymbol{\rho}_{e_v}^{\boldsymbol{k}} \leftarrow \sigma(\mathrm{W} \cdot \mathrm{MEAN}(\{\boldsymbol{\rho}_{e_u}^{\boldsymbol{k-1}}, \forall e_u \in \mathrm{N}(e_v)\}))$$

The LSTM aggregator is the second tested type. It takes the input in a sequential manner, meaning that the neighborhood is processed sequentially in a random order. It takes advantage of the expressiveness of LSTMs. The last one is the pooling aggregator: each vector is processed by a fully-connected layer, then an element-wise max operator is applied (max pooling).

$$\mathrm{AGGREGATE}_k^{pool} = max(\{\sigma(\mathrm{W}_{pool}\boldsymbol{\rho}_{\boldsymbol{e_u}}^{\boldsymbol{k}} + \boldsymbol{b} \cup \{\rho_{e_u}^{k-1}\}, \forall e_u \in \mathrm{N}(e_v)\})$$

The LSTM and pooling are the best ones, with LSTM being slower than pooling. The initial features $\rho^0_{e_v}$ are taken from the node attributes. In the setting of citation networks, it can be the degrees of the nodes or the word vector of the abstract obtained by word2vec [Mikolov et al., 2013] or GloVe [Pennington et al., 2014], commonly used for word embedding in NLP tasks. GraphSAGE performs well on node classification but works only on homogeneous, undirected graphs.

4.2.3 Conclusion

There are two main categories of techniques to embed static graphs: translational methods, based on the distance between feature vectors, and semantic matching based on scores. The semantic matching models are divided between RESCAL and its by-products, neural networks and random walks.

4.3 Attributed graphs

A knowledge graph links entities via relations, and in many of them the word "relations" refers indifferently to relations *between two entities* or *between an entity and one of its attributes*. Most KG models regard all these relations equally and do not take into account the added value of these attributes. Moreover, the frontier between relation and attribute can sometimes be blurry: is the flag of a vessel one of its attribute or a direct relation to a country? On top of that, relations and attributes behave differently: relations connect entities from a large set with only a few other entities, while attributes have properties from a discrete set of entries (e.g. gender has two possible values) or a continuous set of values. Hence, entities and attributes should then not be represented the same way. In the MSA context, we need to use the attributes of our entities such as position or speed to infer new relations. Since this type of networks is more adequated to our needs, we will in this section describe fewer approaches but with more depth.

4.3.1 KR-EAR

Knowledge Representation Learning with Entities, Attributes and Relations (KR-EAR) [Lin et al., 2016] is one of the first models do explicitly distinguish the relations and attributes of a KG (Figure 4.4). Attributes are the main source of one-to-many and many-to-one relations. For instance, the property Male of the attribute Gender is related to millions of human entities. According to [Lin et al., 2016], TransE [Bordes et al., 2013] and its extensions like TransH [Wang et al., 2014] and TransR [Lin et al., 2015b] "cannot sufficiently build translation between entities and their attribute properties".

In KR-EAR, both entities and relations are represented by low-dimensional vectors $\boldsymbol{\rho_e}$ and $\boldsymbol{\alpha_e}$. The embeddings are learnt by building translations between entities using relations, like a classic translational method (e.g., TransE), and by inferring attribute properties based on the embedding of the entity.

4.3.1.1 Problem formulation

The relations are still represented in the shape of SPO triplets (s, r, o): the novelty comes from the attributional triplets p^{A}. The objective is to learn embeddings $\mathbf{X} = (\boldsymbol{\rho_e}, \boldsymbol{\rho_r}, \boldsymbol{\alpha_e})$ of entities, relations and attributes to predict relations between entities and attributes of entities.

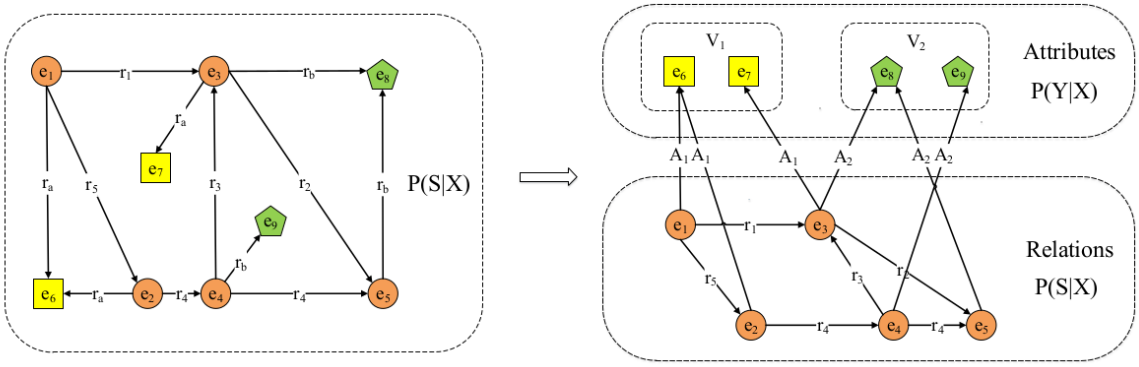


Figure 4.4: An illustration of the difference between a traditional KR model and KR-EAR [Lin et al., 2016]. Here, A_1 and A_2 are two attributes with respective sets of values V_1 and V_2. In the KR model, attributes are seen as relations r_a and r_b while KR-EAR considers their prediction as a classification problem.

The objective function is defined by maximizing the joint probability of relational and attributional triples given the embeddings $\mathbf{X}$, i.e. $P(p^R, p^A|\mathbf{X})$. Both types of triplets are assumed to be conditionally independent, i.e. the relations are predicted without the attributes and vice versa:

$$P(KG^R, KG^A|\mathbf{X}) = P(KG^R|\mathbf{X})P(KG^A|\mathbf{X}) = \prod_{(s,r,o)\in KG^R} P((s,r,o)|\mathbf{X}) \prod_{(e,a,v)\in KG^A} P((e,a,v)|\mathbf{X})$$

where $P((s,r,o)|\mathbf{X})$ denotes the conditional probability of relational triples and $P((e,a,v)|\mathbf{X})$ the conditional probability of the attributional ones. Hence, there are two components in KR-EAR: the Relational Triple Encoder and the Attributional Triple Encoder.

4.3.1.2 Relational Triplet Encoder

The goal of the Relational Triplet Encoder is to capture correlations between entities and relations. Usually, we optimize $P(s|r,o,\mathbf{X})$,$P(r|s,o,\mathbf{X})$ and $P(o|s,p,\mathbf{X})$ instead of $P(s,r,o|\mathbf{X})$. The encoding of a relational triplet is made using TransE and TransR.

$$P(s|r,o,\mathbf{X}) = \frac{exp(g(s,r,o))}{\sum_{\hat{s}\in E} exp(g(\hat{s},r,o))}$$

where g is the energy function indicating the correlation between the relation and the two entities. With **TransE**, $g(s,r,o)$ translates as:

$$g(s,r,o) = -\|\rho_s + \rho_r - \rho_o\|_{L1/L2} + b_1$$

where b_1 is a bias constant.

4.3.1.3 Attributional Triplet Encoder

Intuitively, a classification model should be used to capture correlations between entities and their attributes. Therefore, the conditional probability $P(v|e,a,\mathbf{X})$ for each triplet (e,a,v) is considered and described as follows:

$$P(v|e,a,\mathbf{X}) = \frac{exp(h(e,a,v))}{\sum_{\hat{v}\in D_a} exp(h(e,a,\hat{v}))}$$

where h is the scoring function that computes the semantic similarity between the embedding of the entity in the attribute space and the embedding of the attribute value:

$$h(e, a, v) = -\|f(\boldsymbol{\rho}_e \mathrm{W}_a + b_a) - \boldsymbol{\alpha}_{av}\|_{\mathrm{L1/L2}} + b_2$$

where $f()$ is a non-linear function such as **tanh**, α_{av} is the embedding of attribute value v and b_2 is a bias constant.

KR-EAR improves the results of TransE/H/R by using the attributes, but can only do so with discrete values. It can predict an entity, a relation type or an attribute value out of an incomplete triplet.

4.3.2 MT-KGNN

In Multi-Task Neural Network for Non-discrete Attribute Prediction in Knowledge Graphs (MT-KGNN) [Tay et al., 2017], the authors observe that many relational learning models do not consider attributes despite their efficiency to alleviate the sparsity of KGs. This is because dealing with non-discrete data types is difficult due to the binary nature of knowledge graphs. That model trains a neural network for triplet prediction along with a separate network for attribute value regression (Figure 4.5). The objective is to improve relational learning and enable prediction of non-discrete attributes.

4.3.2.1 Motivation

Non-discrete attributes such as the price of a product are often present in knowledge graphs. They:

- help in relational learning: missing data, such as *gender*, could be retrieved using, for instance, the *height* attribute. More, attributes can characterize entities: the products *phone* and *car* could be distinguished using their *height*.
- can be predicted and thus complete missing values in the KG,
- represent 33% of the triplets (in Easy Freebase),
- are difficult to handle because they are not binary truths, common to knowledge graphs. More, the feature vectors of attributes can be extremely sparse and casting attributes as entities causes the size of the KG to blow up.

4.3.2.2 Problem formulation

The main objective is to produce a score that denotes the probability or strength of a fact or triplet:

$$h(s, r, o) \in [0, 1]$$

This can be seen as a simple binary classification problem in which a sigmoid or 2-class softmax layer can be used. The attributes triplets are defined as e, a, v as before, but this time v is a normalized continuous value $\in [0, 1]$. The range of each attribute is deduced from the training data and any new value below or above min-max will be casted to 0 or 1. During training, the objective is to maximize the score of positive triplets.

4.3.2.3 Multi-task Knowledge Graph Neural Network

There are two networks in the model: the Relational Network (RelNet) and the Attribute Network (AttrNet).

Relational Network (RelNet) The inputs of the RelNet are $[\boldsymbol{\rho}_s, \boldsymbol{\rho}_r, \boldsymbol{\rho}_o, t]$ with:

- $\boldsymbol{\rho}_s, \boldsymbol{\rho}_r, \boldsymbol{\rho}_o \in \mathbb{R}^n$
- t the target of classification, either 0 or 1.

RelNet is a simple concatenation of the triplet through a nonlinear transform and finally a linear transform with sigmoid activation.

$$g_r(s, r, o) = \sigma(\boldsymbol{w}_d^\intercal f(\mathrm{W}_d^\intercal[\boldsymbol{\rho_s}; \boldsymbol{\rho_o}; \boldsymbol{\rho_r}]) + b_r)$$

where f is $tanh$, σ is the sigmoid function and b_r is a scalar bias. $\boldsymbol{w_d} \in \mathbb{R}^{h \times 1}$ and $\mathbf{W}_d \in \mathbb{R}^{3n \times h}$ are parameters of the network.

The function to minimize is a cross-entropy loss:

$$\mathrm{L}_{rel} = -\sum_{i=1}^{\mathrm{N}} t_i log(g_r(p_i^{\mathrm{R}})) + (1 - t_i) log(1 - g_{rel}(p_i^{\mathrm{R}}))$$

where $\boldsymbol{\rho}_{\boldsymbol{i}}^{\mathbf{R}}$ denotes triplet i in batch of size N and t_i a value of $0, 1$. Cross-entropy is common for non-continuous targets and commonly used in binary classification problems. At this point, RelNet is equivalent to the ER-MLP model [Dong et al., 2014] and performs triplet classification (i.e., tells if a given triplet is true or false).

Attribute Network (AttrNet) The inputs of the AttrNet are $[\boldsymbol{\alpha}_s, \upsilon_s, \boldsymbol{\alpha}_o, \upsilon_o]$ with:

- $\boldsymbol{\alpha}_s, \boldsymbol{\alpha}_o \in \mathbb{R}^n$
- $\upsilon_s, \upsilon_o \in [0, 1]$

A single layer network is trained by concatenating the embeddings of the attribute and entity to predict the continuous value ($\in [0, 1]$). The input includes the attributes of the subject *and* of the object entities. Moreover, the network takes into account the positon of an entity as subject or object. Hence, AttrNet will optimize a joint loss function for both regression tasks. The score function is thus defined as follows:

$$g_s(a_s) = \sigma(\boldsymbol{w_s}^\intercal f(\mathrm{B}^\intercal[\boldsymbol{\alpha_s}; \boldsymbol{\rho_s}]) + b_{z_1}) \tag{4.1}$$

$$g_o(a_o) = \sigma(\boldsymbol{w_o}^\intercal f(\mathrm{C}^\intercal[\boldsymbol{\alpha_o}; \boldsymbol{\rho e_o}]) + b_{z_2}) \tag{4.2}$$

where $w_s, w_o \in \mathbb{R}^{h_a}$ and $\mathrm{B}, \mathrm{C} \in \mathbb{R}^{2n \times h_a}$ are parameters of AttrNet. h_a is the size of the hidden layer and $b_{z_1}, b_{z_2} \in \mathbb{R}$ are scalar biases. The final output of each side of the AttrNet is a scalar value, as for RelNet. As this is a regression problem, the mean squared error (MSE) is used:

$$\mathrm{MSE}(x, x^*) = \tfrac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (x_i - x_i^*)^2$$

$$\mathrm{L}_{attr} = \mathrm{L}_{subject} + \mathrm{L}_{object}$$

$$\mathrm{L}_{head} = \mathrm{MSE}(g_s(a_s), (a_s)^*)$$

$$\mathrm{L}_{tail} = \mathrm{MSE}(g_o((a_o), (a_o)^*)$$

where $(a_s)^*, (a_o)^*$ are ground truth labels. AttrNet can thus infer missing attribute values.

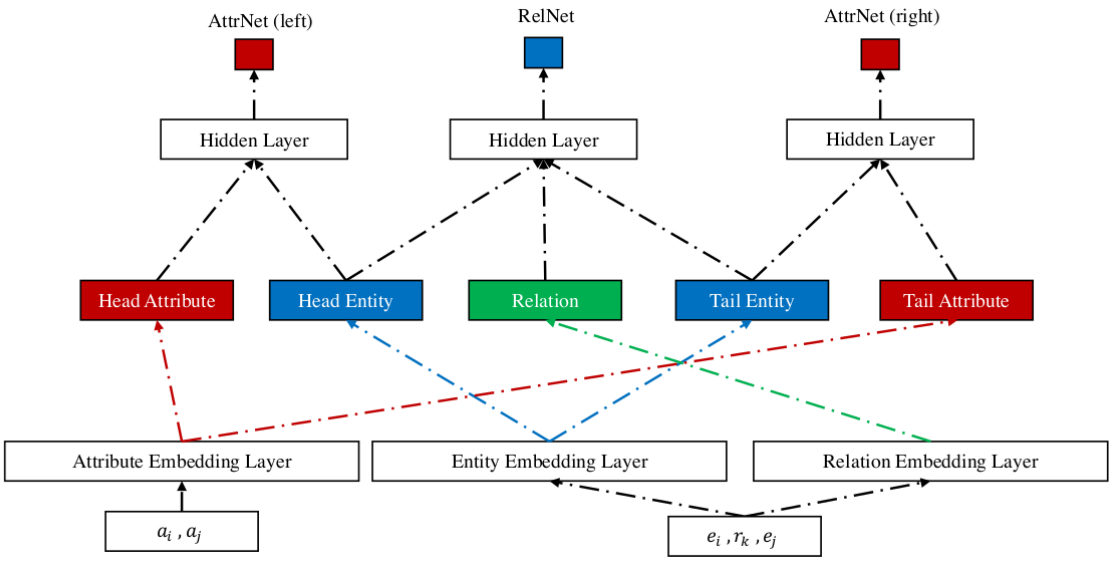


Figure 4.5: Illustration of the architecture of MT-KGNN [Tay et al., 2017]. Relational embeddings are used to predict attribute values.

4.3.3 Other techniques

attri2vec [Zhang et al., 2019] extends the DeepWalk model [Perozzi et al., 2014] (close to node2vec [Grover and Leskovec, 2016]) to capture both network structure and node content features. Co-embedding Attributed Networks (CAN) [Meng et al., 2019] embeds each node and attribute with the mean and variance of a Gaussian distribution using a variational autoencoder.

4.3.4 Conclusion

KR-EAR and MT-KGNN are two frameworks that extend existing relational techniques by dealing with attributes. The first one cannot handle non-discrete value, while the second can handle non-discrete data types and perform triplet classification instead of link prediction (but the two are close). But it is not clear that the latter can handle discrete data types. The common feature is the separation of the entity and attribute models.

4.4 Dynamic graphs

In a dynamic setting, each node is represented by a time series of vectors modelling its evolution. With the rise of tempral datasets (such as GDELT [Leetaru and Schrodt, 2013] and ICEWS [Boschee et al., 2015], this topic is emerging and has fewer contributions than static graphs, but advances have already been made. The models of KGs are now dynamic and become *DKG*s.

4.4.1 Predicting the Co-Evolution of Event and Knowledge Graphs

This temporal model is proposed by [Esteban et al., 2016] and deals with the graph at the event level, i.e. what happens to entities over time. The authors model the *DKG* with two tensors: the tensor representing the *DKG* itself and an event tensor. They assume that changes

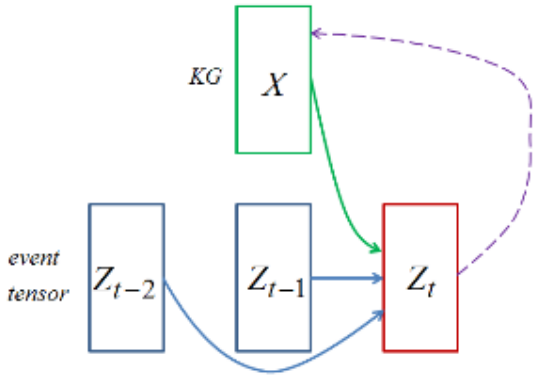


Figure 4.6: The figure shows an example where the event tensor is predicted from the representations of the events in the last two time steps and from the KG representation. The dotted line indicate the transfer of observed events into the *DKG* [Esteban et al., 2016]

.

in a *DKG* arrive in the form of events, and that for a given timestep, events can be described by a very sparse event triple graph. The events contain facts that change the *DKG*, e.g. from True to False and vice versa (Figure 4.6). The triples in the KG that do not appear in the event graph are assumed to remain unchanged. Then, a separate prediction model is trained using the latent representations of both the previous events and the involved entities in the *DKG*.

4.4.2 Know-Evolve: Deep Temporal Reasoning for Temporal Knowledge Graphs

This model was presented by Trivedi et al. [Trivedi et al., 2017] during the 34^{st} International Conference on Machine Learning. It will be our main inspiration for a streaming, dynamic relational model.

4.4.2.1 The concept

The authors use a *temporal point process* framework for temporal reasoning over dynamically evolving knowledge graphs; this process models the *occurrence of a fact*. They propose a novel deep learning architecture that evolves over time based on the availability of new facts. The dynamically evolving network ingests the incoming new facts, learn from them and *updates the embeddings of involved entities* based on their recent relationships and temporal behavior. Their model can predict the occurrence of a fact, but also the time when a fact may potentially occur. It supports the Open World Assumption and can predict over unseen entities. The main purpose is to predict new relations.

The evolutionary knowledge network has three components:

- A *temporal point process* that models the occurrence of a fact,
- A bilinear relationship score that captures relational interactions between entities and modulates the intensity function of the point process,

- A *deep recurrent network* that learns non-linearly and mutually evolving latent representations of entities based on their interactions with other entities in the multi-relational space over time.

4.4.2.2 Temporal Point Process

A temporal point process [Cox and Lewis, 1972] is a random process whose realization consists of a list of events localized in time, $\{t_i\}$, with $t_i \in \mathbb{R}^+$. It is characterized via the *conditional intensity function* $\lambda(t)$. Formally, $\lambda(t)dt$ is the conditional probability of observing an event in a small window $[t, t+dt)$ given the history $\tau(t) := \{t_k | t_k < t\}$.

In this paper, the functional form of the intensity $\lambda(t)$ is a *Rayleigh Process*, which is well adapted when event likelihood drops rapidly after rising to a peak. Its intensity function is $\lambda(t) = \alpha.(t)$ where $\alpha > 0$ is the weight parameter and the log survival function is $logS(t|\alpha) = \frac{-\alpha.(t)^2}{2}$. The log survival function gives the probability that the event of interest has not occurred by duration t.

4.4.2.3 Evolutionary Knowledge Network

In Know-Evolve, time is modeled as a random variable and the temporal point process models the occurrence of a fact. It uses a bilinear score function and dynamically evolves the embeddings of the entities, and then computes the conditional intensity function of the temporal point process. Figure 4.7 illustrates the network.

- Conditional intensity function: $\lambda_r^{s,o}(t|\bar{t}) = exp(g_r^{s,o}(\bar{t})) * (t - \bar{t})$ where $t > \bar{t}$ is the time of the current event and $\bar{t} = max(t^{e_s}-, t^{e_o}-)$ is the most recent time point where the subject or object entity was involved in an event before t,

- $g_r^{s,o} = \rho_s^{t-\intercal}.\ R_r\ .\rho_o^{t-}$ is the relational compatibility score between the involved entities in that specific relationship, with $R_r \in \mathbb{R}^{n \times n}$ the relationship weight matrix and $t-$ the time point just before time t.

The embeddings are updated using a recurrent neural network (RNN), with different parameters W_s and W_o depending on the status of the updated entity: subject or object.

Subject embeddings

$$\rho^s(t_a) = \sigma(\boldsymbol{w_t^s}(t_a - t_{a-1}) + W^{hh}.\ h^s(t_a-))$$

$$h^s(t_a-) = \sigma(W^h.\ [\boldsymbol{\rho}_s^{t_{a-1}} \oplus \boldsymbol{\rho}_o^{t_a-} \oplus \boldsymbol{r}_s^{a-1}])$$

Object embeddings

$$\rho^o(t_b) = \sigma(\boldsymbol{w_t^o}(t_b - t_{b-1}) + W^{hh}.\ h^{e_o}(t_b-))$$

$$h^o(t_b-) = \sigma(W^h.\ [\boldsymbol{\rho}_o^{t_{b-1}} \oplus \boldsymbol{\rho}_s^{t_{b-1}} \oplus \boldsymbol{r}_o^{t_{b-1}}])$$

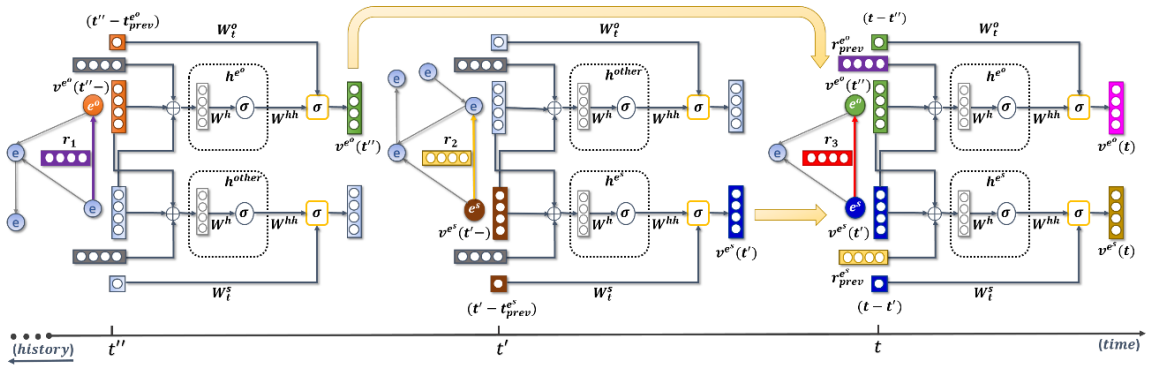


Figure 4.7: Realization of Evolutionary Knowledge Network Architecture over a timeline [Trivedi et al., 2017]. Here $t"$, t' and t may or may not be consecutive. $t_{a-1} = t'$ and $t_{b-1} = t"$. $t^{e_s}_{prev}, t^{e_o}_{prev}$ represent previous time point in history before t', $t"$. $r^{e_s}_{prev} = r_2$ and $r^{e_o}_{prev} = r_1$.

where $\boldsymbol{w}^{\boldsymbol{s}}_{\boldsymbol{t}}, \boldsymbol{w}^{\boldsymbol{o}}_{\boldsymbol{t}} \in \mathbb{R}^n$, t_a, t_b are scalar values for time, $\mathrm{W}^{hh} \in \mathbb{R}^{n \times n}$, $\mathrm{W}^{h} \in \mathbb{R}^{n \times 3n}$, $\oplus$ represent simple concatenation operator and σ a non-linear function, often tanh. The matrix R_r and vectors $\boldsymbol{r}^{a-1}_s, \boldsymbol{r}^{t_{b-1}}_o \in \mathbb{R}^{\ell}$ are unique for each relation type in the dataset and are learned during training. $h^s(t_a-) \in \mathbb{R}$ is the hidden layer. It reasons for an event by capturing the compatibility of the most recent object embeddings in the previous relationship of the subject entity. After training, the parameters are frozen and the embeddings are continuously updated with incoming events. The update of the embedding and the computation of the intensity are illustrated in Figures 4.7 and 4.8.

4.4.2.4 Training procedure

After applying the conditional intensity function on both scores, they maximize the probability of these events using the joint negative log likelihood of the intensity function [Daley and Vere-Jones, 2008], defined for a batch of events of size N as:

$$\mathrm{L} = -\sum_{i=1}^{\mathrm{N}} log(\lambda^{s_i,o_i}_{r_i}(t'|t)) + \mathrm{S} \tag{4.3}$$

This loss function maximizes the probability of happened events, but nothing prevents it to maximize all events. To counterbalance the positive examples, negatives ones must be included. The naive method is to compute the conditional probability of all non-happened events in a given time window around the happened event, but this would be computationally intractable because of the complexity in $\mathrm{O}(|\mathrm{E}|^2|\mathrm{R}|)$.

$$\mathrm{S} = \sum_{r=1}^{|\mathrm{R}|} \sum_{s=1}^{|\mathrm{E}|} \sum_{o=1}^{|\mathrm{E}|} \int_0^{\mathrm{T}} \lambda^{s,o}_r(\tau|\bar{\tau}) d\tau$$

To reduce the complexity, the authors only sample negative examples from the entities of the current batch. It is thus based on the Closed World Assumption, since all events outside of the current batch are considered to be false. Know-Evolve performs well on link and time prediction tasks.

4.4.3 CTDNE

Continuous-Time Dynamic Network Embeddings [Nguyen et al., 2018] is an embedding method on dynamic, homogeneous graphs using temporal random walks in continuous time.

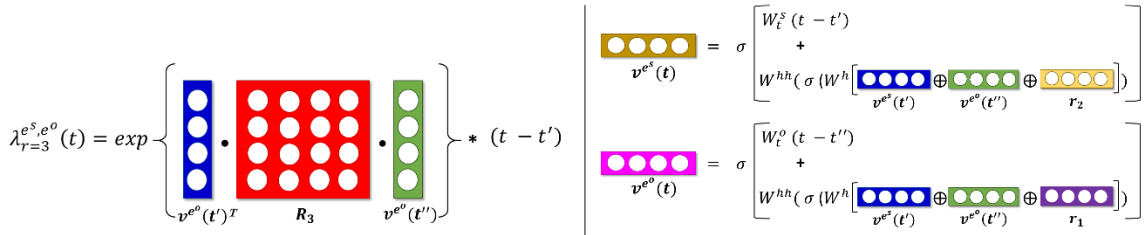


Figure 4.8: One step visualization of Know-Evolve computations done in Figure 4.7 after observing an event at time t [Trivedi et al., 2017]. (Best viewed in color)

The temporal random walk N_v^t of a node v at time t in the triplet $p^{\mathrm{R}} = (e_x, e_y, t)$ is defined as[1]:

$$\mathrm{N}_v^t = \{(e_z, t' | q^{\mathrm{R}} = (e_y, e_z, t') \in DKG^{\mathrm{R}} \wedge t' > t\}$$

Here in this case, DKG^{R} is the special case where $|\mathrm{R}| = 1$ (homogeneous graph). The selection of the neighbors can be unbiased (equal probabilities), or with a linear (closer in time) or exponential (decaying probability of consecutive contact) bias.

After the creation of a set of temporal random walks of a given length, the Skip-Gram architecture is used to generate embeddings as in [Grover and Leskovec, 2016] and is generalized to continuous-time dynamic networks. Given a temporal walk S_t, the optimization problem for the learning of the embeddings is:

$$\max_f \; log \; \mathrm{Pr}(\mathrm{W_T} = \{e_{i-\omega}, \ldots, _{i+\omega}\} | \rho_{e_i})$$

where ω is the context window size and $\mathrm{W_T} \subseteq \mathrm{S}_t$ is an arbitrary context window. Conditional independence is assumed between the nodes of $\mathrm{W_T}$, that is:

$$\mathrm{Pr}(\mathrm{W_T} | \rho_{e_i}) = \prod_{e_{i+k} \in \mathrm{W_T}} \mathrm{Pr}(e_{i+k} | \rho_{e_i})$$

Learning is done through stochastic gradient descent, and several hyperparameters need to be tuned: walk length, size and number of context windows, sampling strategy. The learned embeddings can then be used for downstream machine learning task, and improves the results of static random walks methods on link prediction in temporal datasets such as Enron [Klimt and Yang, 2004].

4.4.4 Other models

Although we described the main methods, other models have been tried to deal with dynamic graphs. DyRep [Trivedi et al., 2019] deals separately with the topological evolution of the graph and the activities between the nodes and then aggregates the two scales. It also define a temporal attention mechanism. Streaming Graph Neural Network [Ma et al., 2020] updates the node representations using LSTMs and propagates the new events to neighbor nodes on homogeneous graphs. DynamicTriad [Zhou et al., 2018] is based on triads which are groups of three vertices. It tries to close a triad by finding the third vertice to connect to a group of two in a weighted homogeneous sliced *DKG*. Sequence encoder [García-Durán et al., 2018] uses LSTMs to learn the sequences and represents time using *year, month, day* tokens. A more complete survey on relational representation learning on dynamic graphs can be found in [Kazemi et al., 2019] and [Divakaran and Mohan, 2019].

[1] We omit r since the graph is homogeneous

4.4.5 Conclusion

Although these models are time-aware, they do not include attribute information in the prediction task, and most of them use snapshots of graphs that are not suitable to process streaming sequences. They form a solid basis to learn on DKG, but to deal with the situational awareness constraints on maritime situations requires the dynamic learning of attributes. We thus aim at add a dynamic attribute model to the dynamic relational one.

4.5 Dynamic attributed graphs

So far we reviewed techniques that deal with static relational and attributed graphs as well as dynamic relational graphs. However, the dynamic attributed graphs seem the most suited techniques to use graphs for MSA. This field is relatively new and only a few models exist.

4.5.1 DANE

DANE [Li et al., 2017] is an embedding technique that uses both the network structure (relational) and node attributes. Their hypothesis is that the available information on attributes can mitigate the network sparsity and help find better embeddings, since sparse graphs provide less proximity information. DANE first performs spectral embedding separately on both the network structure and the node attributes (Figure 4.9).

To learn from the structure, the adjacency matrix $\mathrm{A}^{(t)} \in \mathrm{R}^{n \times n}$ is used at time t to optimize the following loss function:

$$\frac{1}{2} \sum_{i,j} \mathrm{A}^{(t)}(i,j) \| y_i - y_j \|_2^2$$

where $y_i, y_j \in \mathrm{R}^n$ are the embeddings of the nodes i and j connected by edge (i,j). It should be noted that DANE works with undirected, unlabeled edges (homogeneous graph). This objective function boils down to the following eigen-problem:

$$\mathrm{L}_{\mathrm{A}}^{(t)} a = \lambda \mathrm{D}_{\mathrm{A}}^{(t)} a$$

where $\mathrm{L}_{\mathrm{A}}^{(t)}$ is the Laplacian matrix of $\mathrm{A}^{(t)}$ and $\mathrm{D}_{\mathrm{A}}^{(t)}$ its diagonal matrix. Let $\rho_1, \rho_2, \ldots, \rho_n$ be the eigenvectors of the corresponding eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, then the k-dimensional embedding $\mathrm{Y}_{\mathrm{A}}^{(t)} \in \mathrm{R}^{n \times k}$ of the network structure is given by the top-k eigenvectors starting from ρ_2, i.e. $\mathrm{Y}_{\mathrm{A}}^{(t)} = [\rho_2, \ldots, _k, \rho_{k+1}]$.

The same thing is done with the feature matrix of attributes X. Additionally, they normalize the attributes of each node to obtain the cosine similarity matrix $\mathrm{W}^{(t)}$. Thus, they obtain the top-k eigenvectors $\mathrm{Y}_{\mathrm{X}}^{(t)} = [\alpha_2, \ldots, \alpha_{k+1}]$ of the generalized eigen-problem corresponding to $\mathrm{W}^{(t)}$.

The two embeddings $\mathrm{Y_X}$ and $\mathrm{Y_A}$ are then fused into a consensus embedding by maximizing their correlation. After this initial step of offline learning, the embeddings are then continuously updated via the matrix perturbation theory. An online update system is required since computation from scratch is very expensive. To do so, a modification in the network structure or node attributes is reflected in the associated matrices, and the embeddings are updated in an online fashion. The timescale in the experiments goes from 10 to 16 timesteps. This setting seems to indicate computations with a sliced *DKG*, but the authors do not say if the complexity may allow for a lot more timesteps as in a streaming *DKG* setting.

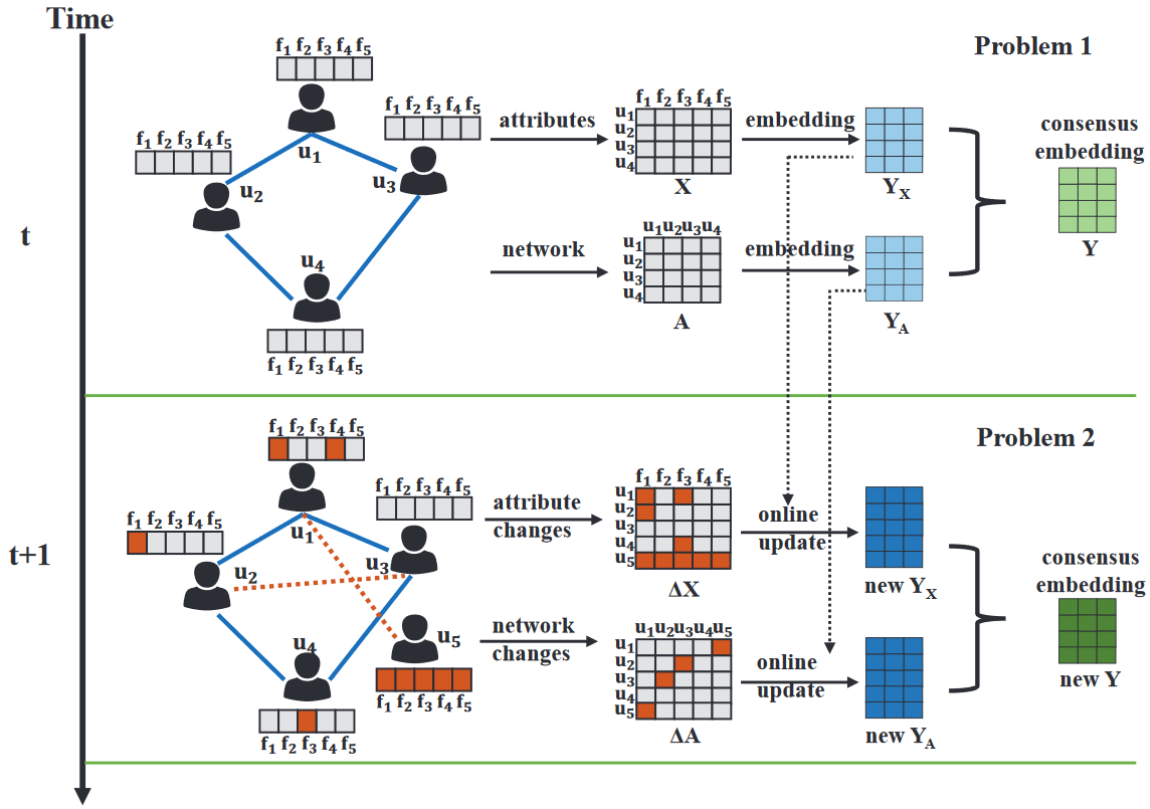


Figure 4.9: The two steps of DANE [Li et al., 2017].The first is the offline phase that leverages the adjacency matrix A for network structure and the features of nodes represented by the matrix X to generate embeddings. Then, the online phase updates the embeddings according to changes in the matrices.

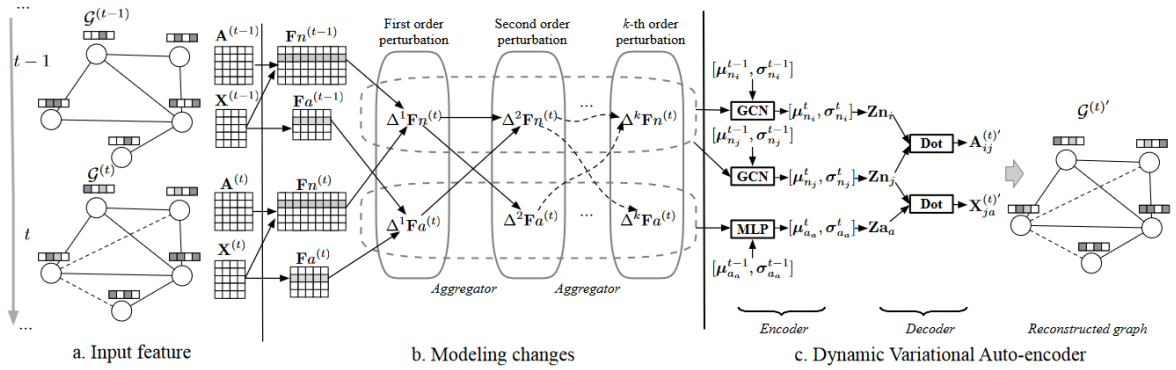


Figure 4.10: CDAN global architecture [Meng et al., 2020]

4.5.2 CDAN

The Co-embedding model for Dynamic Attributed Networks (CDAN) [Meng et al., 2020] is similar to DANE in the way it deals with relations and attributes. It also uses an adjacency matrix for relations and a feature matrix for attributes. However, the matrix perturbation order is of several orders, while DANE only uses first order perturbation. Moreover, CDAN is more neural network oriented, while DANE uses Laplacians and eigenvalues. The first order perturbation for node i and attribute a between time t and $(t-1)$ is defined by:

$$\Delta^1 \mathrm{F}n_i^{(t)} = tanh(\mathrm{W}n^1 \cdot (\mathrm{F}n_i^{(t)} - \mathrm{F}n_i^{(t-1)}))$$

$$\Delta^1 \mathrm{F}a_a^{(t)} = tanh(\mathrm{W}a^1 \cdot (\mathrm{F}a_a^{(t)} - \mathrm{F}a_i^{(t-1)}))$$

where $\mathrm{W}n^1 \in \mathrm{R}^{(|\mathrm{E}|+|\mathrm{A}|)\times \mathrm{H}_p}$ and $\mathrm{W}a^1 \in \mathrm{R}^{\mathrm{N}\times \mathrm{H}_p}$ are weight matrices with $\mathrm{H}_p < |\mathrm{E}|, \mathrm{H}_p < |\mathrm{A}|$ being the dimension of the perturbation features, and $\mathrm{F}n_i^{(t)}, \mathrm{F}a_a^{(t)}$ the features of node i and attribute a at time t. The *k-th order* perturbation is then obtained by learning the aggregation of the lower order perturbations from its neighborhood, using the mean aggregation function.

These perturbations are then fed to a Dynamic Variational Autoencoder that outputs Gaussian embeddings for all nodes and attributes of the network. Figure 4.10 illustrates the global architecture of CDAN.

In CDAN settings, the attributes are dynamic but can be considered as entities (discrete). For instance, in the DBLP dataset, they use the conferences and the key terms as attributes. As for time, the method seems suited for a sequence of graph snapshots G^t and its capacity to deal with continuous time is unclear. The method is tested for node classification and link prediction and only works on homogeneous networks.

4.5.3 SLIDE

[Li et al., 2018] propose a streaming model (SLIDE) on dynamic attributed networks, using a sketching matrix that summarizes the currently observed links and node attributes. When a change occurs in the network, the sketching matrix is updated to include the changes in the network representation. This update is performed using the frequent directions algorithm, which is able to operate in a streaming fashion and makes only one pass of the data. To feed the dynamic attributed network to this algorithm, it must first be converted to a feature representation based on the observed links. For each edge in the graph, and when a new edge is created or an attribute value has changed, a feature vector is added to the matrix as a new

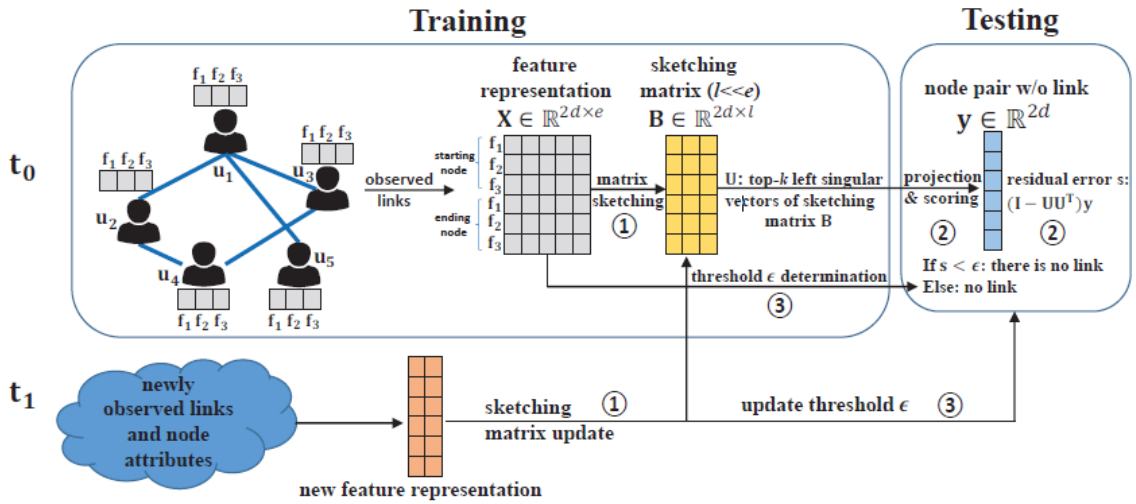


Figure 4.11: SLIDE workflow [Li et al., 2018]

column in the data stream. But this matrix is still too large to be held in memory, hence the sketching matrix to represent the stream of data. With the updated sketching matrix, missing links can be inferred. If the features of the candidate link $y \in \mathrm{R}^{2d}$, with d the size of the node features, is close to the space composed of the columns of the matrix U^t, this link is most likely existing. The residual error of the reconstruction is $\|\mathrm{I} - \mathrm{U}^t(\mathrm{U}^t)^\intercal y\|_2^2$. To know if the link exists, the residual error must be below a threshold automatically obtained from the data stream by picking the largest error among the columns in the stream (of existing links). The global architecture of SLIDE is presented in Figure 4.11.

SLIDE works with cold start nodes, i.e. new nodes that have not been seen before, since it leverages the attributes of the node to perform a prediction and not only its history. It is efficient for link prediction in a streaming fashion. However, it only applies to homogeneous graphs and to discrete attribute features. The performed experiments were made on a restricted number of timesteps (~20), which suggests a discrete time setting, but the streaming capabilities of the framework should make it work in continuous time with much more timesteps. In their conclusion, the authors observe that the structure of the network is more important than features to predict new links, but features do help in the prediction.

4.5.4 Conclusion

The three reviewed techniques are matrix based and have the advantage to work in a streaming fashion. They deal with both relations and attributes in a dynamic setting by summarizing the network features. However, they only deal with homogeneous graphs and the choice of attributes is discrete. We are looking for a model for heterogeneous graphs that can handle continuous attributes (positions, speeds...). Our research line is reinforced by the survey on dynamic graphs by Kazemi et al. [Kazemi et al., 2019], that indicates that “while some initial steps have been taken to extend [existing encoders] to other types of graphs, the best way of extending these approaches to the case of attributed graphs and knowledge graphs (KGs) is still not clear."

4.6 Addition of semantic information

This section presents briefly some additional ways to include or extract knowledge in/from graphs based on semantic information. More details can be found in the survey by [Wang et al., 2017].

4.6.1 Entity types (in embeddings)

When working with nodes only, the class of the node is generally not known by the embedding model. Including this information may help to guide the learning phase, since knowing *what* the nodes are makes the prediction of links between them easier (e.g. a vessel is more likely to enter a port than a parking lot).

A straightforward method to model such information, as investigated in [Nickel et al., 2012], is to take :isA as an ordinary relation and the corresponding triples as ordinary training examples. Guo et al. [Guo et al., 2017] proposed semantically smooth embedding (SSE), which requires entities of the same type to stay close to each other in the embedding space. SSE employs two manifold learning algorithms, i.e., Laplacian eigenmaps [Belkin and Niyogi, 2002] and locally linear embeddings [Roweis and Saul, 2001] to model this smoothness assumption.

Xie et al. [Xie et al., 2016] devised type-embodied knowledge representation learning (TKRL), which can handle hierarchical entity categories and multiple category labels. It achieves good performance in downstream tasks such as link prediction and triple classification, but at the cost of a high space complexity.

4.6.2 Logical rules

Logical rules can represent expert knowledge, and thus give *a priori* information on the graph. They make the learning easier by automatically extracting the known patterns. For instance, first-order Horn clauses such as $\forall x, y : \mathrm{H}as\mathrm{W}ife(x, y) \Rightarrow \mathrm{H}as\mathrm{S}pouse(x, y)$, state obvious knowledge like the fact that two entities linked by the relation HasWife should also be linked by the relation HasSpouse.

To do so, Markov logic networks [Richardson and Domingos, 2006], [Kimmig et al., 2012], [Pujara et al., 2015] can be used. There are also systems such as AMIE [Galárraga et al., 2013], [Galárraga et al., 2015] which can extract logical rules automatically from KGs to apply them later. Wang et al. [Wang et al., 2015] uses rules to refine embedding models during KG completion. Guo et al. [Guo et al., 2016] embed KG facts and logical rules simultaneously. RUGE [Guo et al., 2018] learns embeddings from observed labeled triplets, unlabeled triplets whose labels are going to be predicted iteratively, and soft rules extracted from the KG that guide the learning.

4.7 Conclusion

We reviewed the main models used for static (attributed), dynamic (attributed) graphs. While static graphs are obviously out of scope for the problem at hand (MSA), they lay the basis and give some insight on how to create meaningful graph representations. If a few techniques exist for dynamic attributed graphs, they are not completely adequated to our needs since they deal only with homogeneous graphs. Still, we can get inspired by these on the inclusion

of attributes in a dynamic setting and to work in a streaming fashion. *Our main objective is thus to create an embedding model working on heterogeneous dynamic attributed graphs, ideally in a streaming fashion, to maintain up-to-date the current state of knowledge about a situation and make prediction on future possible links to raise automated alerts on what could happen.*

Part II

Contributions

CHAPTER 5

Problem statement for MSA

In this chapter, we first formalize the link prediction problem in heterogeneous dynamic attributed knowledge graphs, and then present the global architecture of our model, Joint Evolution. Chapter 6 describes the model in detail.

5.1 Problem statement

In this section, we formalize the concepts used to model Dynamic Attributed Knowledge Graphs, and the associated link prediction problem.

5.1.1 Prediction Problems with Knowledge Graphs

Motivated by our application to situation monitoring, we tacitly assume that the relations between entities and attribute values of entities at some timestep t, can be predicted using only the past relations and attribute values, that is, using those quadruplets with a timestamp $\tau < t$.[1] Moreover, because our application scenarios involve very long periods of surveillance with rapidly changing attributes and relations, we cannot afford storing all observations in memory. Hence, we focus on a constant-size representation per entity, namely, its attributive and relational embeddings. At each timestep t and for each entity e, we have an attributive embedding $\boldsymbol{\alpha}_e^{\tau}$ (where $\tau \leq t$ denotes the last timestep when the attributive embedding of e was updated) and a relational embedding $\boldsymbol{\rho}_e^{\tau'}$ (where we have $\tau' \leq t$, and in general $\tau \neq \tau'$).

The problems of interest are thus the following:

- given an attributive quadruple $q = (e, a, v, t)$ observed at time t (called an *attributive event*) and the previous embedding $\boldsymbol{\alpha}_e^{t-}$, compute the new attributive embedding of e, $\boldsymbol{\alpha}_e^{t}$;

[1] As opposed to tasks where we would like to retrieve what the situation was likely to be at some point in the past.

- given a relational quadruple $q = (s, r, o, t)$ observed at time t (*relational event*) and the previous embeddings $\boldsymbol{\rho}_s^{t-}$ and $\boldsymbol{\rho}_o^{t-}$, compute the new relational embeddings $\boldsymbol{\rho}_s^t$ and $\boldsymbol{\rho}_o^t$ of s and o;
- at any timestep t, predict the value of a given attribute a for a given entity e at some given, present or future timestep $t' \geq t$, that is, the value $v \in \mathrm{D}_a$ such that the quadruple $q = (e, a, v, t')$ is most likely to be in DKG^{A};
- similarly, at any timestep t, given two out of s, r, o and some timestep $t' \geq t$, predict the most likely value(s) for the third one (for instance, given s, o, t', predict the relation(s) r such that $q = (s, r, o, t')$ is most likely to be in DKG^{R}.

Observe that we restrict our attention to predictions at given timesteps, that is, we want to be able to predict *what* will occur at a given timestep, but do not address the prediction of *when* a given event will occur. This is left for future work, but that restriction is also motivated by the fact that surveillance typically requires to predict what will happen at the next timesteps (*e.g.*, suspicious or dangerous events). To this end, we will predict the most probable object for a quadruple $q = (s, r, ?, t)$ to answer questions such as *"in which port will this vessel enter next?"* or *"which social network community will this person join?"*. This is also valid for subject and relation prediction, as in *"which vessel will tug this upcoming vessel?"* and *"how will these two vessels interact?"*, i.e. $(?, r, o, t)$ and $(s, ?, o, t)$. These operations constitute the *inference* phase, that is, the production phase when the system is used for online situation assessment.

Thus, we are interested in predicting the missing relations between entities and values of attributes in knowledge graphs. Recall our assumption that for some timestep, they can be predicted from the values of a subset of the relations and attributes at previous timesteps. We thus make the two following assumptions:

Assumption 5.1.1 (determined relation)**.** *Let $DKG = \langle \mathrm{E}, \mathrm{R}, \mathrm{A}, \mathrm{D} \rangle$ be a frame be a dynamic knowledge graph. We assume that the relation r^* is* determined by *DKG in the sense that for all timesteps t^*, function $f : DKG \to 2^{\mathrm{E} \times \mathrm{E}}$ provides the next relational event with*

$$\{(e^s, e^o) \mid (e^s, r^*, e^o, t^*) \in DKG^{\mathrm{R}}\} = f(DKG^{<t^*})$$

where $DKG^{<t^*}$ denotes the restriction of DKG to those quadruplets with a relation with a timestamp less than t^*. Note that $DKG^{<t^*}$ means that we use both relations and attributes.

Assumption 5.1.2 (determined attribute)**.** *Let $DKG^{\mathrm{A}} = \langle \mathrm{E}, \mathrm{A}, \mathrm{D} \rangle$ be an attributive dynamic knowledge graph. We assume that the attribute a^* is* determined by *DKG^{A} in the sense that for all timesteps t^*, the function $g : DKG^{\mathrm{A}} \to 2^{\mathrm{E} \times \mathrm{D}}$ provides the next attributive event for DKG^{A} with*

$$\{(e, v) \mid (e, a^*, v, t^*) \in DKG^{\mathrm{A}}\} = g(DKG^{\mathrm{A},<t^*})$$

where $DKG^{\mathrm{A},<t^*}$ denotes the restriction of DKG to those quadruplets with an attribute in A *and with a timestamp less that t^*.*

With this in hand, the learning problem which we tackle consists, for a given knowledge graph DKG^* (unknown to the learner), and given all the information up to timestep t^* together with some information at timestep t^*, of inducing a given target relation r^* at time t^*. This is also valid for a given target attribute a^*, value v^*, subject s^* or object o^*.

5.1.2 The ideal graph model for maritime applications

After defining formally the settings of our graph, we apply it to the maritime usecase. “Real-world" datasets often have more constraints than the academic ones because of their specificities. Maritime datasets are no exception and the following challenges, presented in more details in Section 2.1, must be overcome: evolution of attributes, event and threat detection, streaming data, uncertainty of data and their sources, explainability of the decision. To answer this, we propose the graph structure illustrated in Figure 5.1 (best viewed in color).

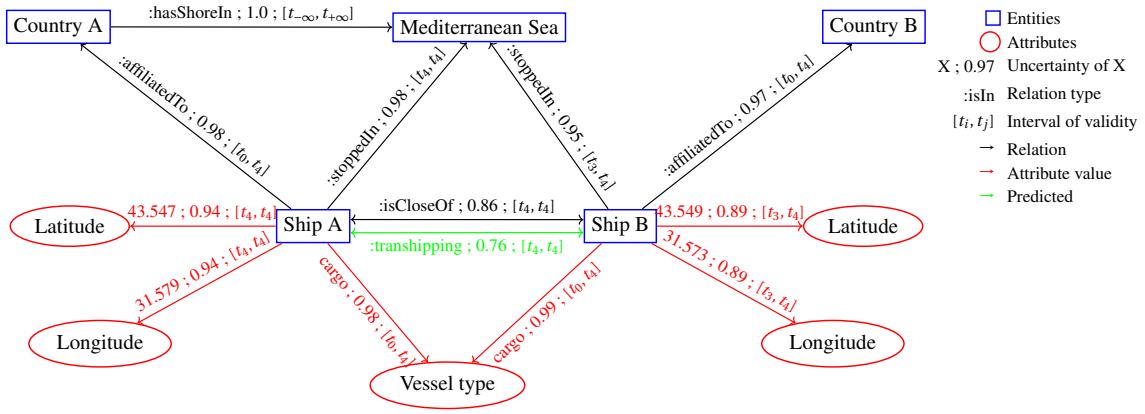


Figure 5.1: Example of DKG^* at time $t^* = t_4$

Events in red represent DKG^A and the other events DKG^R. Ship A and Ship B are vessels from different countries, moving in the Mediterranean Sea. They have a static attribute (vessel type) setting them as cargos and dynamic attributes (latitude and longitude) revealing their positions. Before t^*, the two vessels were moving in the Mediterranean Sea and had different positions, but now (at t^*) they have stopped and are close to each other. A possible link prediction from $DKG^{<t^*}$ is that the two vessels are performing transhipping. Note that the *:stoppedIn* relation can be deduced from the *speed* attribute going to zero, not represented here for the sake of clarity.
If relations such as CountryA having a shore on the Mediterranean Sea are 100% sure, some are more uncertain: the position of Ship B is only 89% sure because the signal was picked up by a satellite in an area with high ship density. Moreover, the uncertainty of predicted relations (*:transhipping*) depends on the uncertainty of the root events. Finally, to predict the transhipping action in time, the model must be updated with the root causes as soon as they are available, hence the need for streaming link prediction. Ideally, the model will have to handle both discrete (port of call, flag...) and continuous data (position, time, wind speed...).

5.1.3 A first restricted model

We will work on a restricted version of the *DKG* presented in Figure 5.1, because several components are research leads on their own to explore. Using a *time interval* for edges boils down to consider not only the edge addition problem, but also the edge deletion one since an edge with a duration becomes invalid as soon as the current timestamp is not in the time interval. At the time of [Kazemi et al., 2019] survey publication, there is no significant work towards the edge deletion task for *DKG*.

Dealing with the uncertainty of the data or its sources is also a problem on its own. Although some work already exists on this topic [Chekol and Stuckenschmidt, 2016, Chekol et al., 2017, Hajiramezanali et al., 2019], it makes the problem more complex and will be

considered in future work. At last, explainability is also left for future work, for which surveys like [Adadi and Berrada, 2018] can be a good starting point.

5.2 A two-headed model

We introduce JointEvolution, a double encoder-decoder system. The global architecture is presented in Figure 5.2 and will be detailed in this section. Note that this is an overview of the model, the full extent of the mechanics of JointEvolution being developed in Chapter 6.

5.2.1 An encoder-decoder system

As stated before, the goal of JointEvolution is to create projections of the entities in an embedding space and make prediction on future links or attribute values from them. The sequence of events, used as input, is processed by an encoder. Each time an entity is met within a quadruplet, a one-hot vector projects it to its embedding space. A one-hot vector is a vector filled with zeroes and ones, where the ones are the index we want to select (each entity is mapped to a unique index). This vector is multiplied by the embedding matrix to access the embedding of the desired entity i.e. the line of the matrix whose index in the one-hot vector has a value of one. The obtained feature vector is then aggregated with the other components of the quadruplet within the encoder and a new, up-to-date embedding is produced.

This new embedding serves two purposes: maintaining the current state of knowledge as complete as possible by including each event received about an entity in its representation, and making hypotheses on what could happen next using the decoder. Since one system is dedicated to DKG^{A} (AttrNet) and the other to DKG^{R} (RelNet), we will now detail the specifics of each encoder and decoder.

5.2.1.1 AttrNet

The input of AttrNet is a sequence of attributive quadruplets $q^{\mathrm{A}} \in DKG^{\mathrm{A}}$. The goal of the network is to learn the behavior of the attributes, so the objective function is set to predict in an accurate manner the next value of the attribute type a given as input. For instance, for the input $q^{\mathrm{A}} = (e_4, a_1, 42.654, t_1)$, AttrNet will try to predict $q'^{\mathrm{A}} = (e_4, a_1, 43.294, t_3)$ using $\boldsymbol{\alpha}_{e_4}^{t_1}$ with a_1 the latitude of the vessel e_4.

To do so, we first include the value of the input $q^{\mathrm{A}} = (e, a, v, t)$ in $\boldsymbol{\alpha}_e^{t-}$ using the encoder. This new embedding is processed through the decoder, that outputs a scalar value: the prediction for the next value. This process learns representations able to predict the next values for attributes.

5.2.1.2 RelNet

For RelNet, we use an encoder structure similar to AttrNet, but with a version for the subject and another slightly different for the object. This is done to consider the position of the entity in the quadruplet during the learning. Indeed, the effect of an event on an entity is not the same whether it is the subject or the object: while some relations are symmetric (e.g. an exchange of goods), others like "entering a place" will most often involve a vessel as its subject and a port or an area as its object. Moreover, learning the right position in the

quadruplet is essential for prediction since the candidate entities are different whether we try to predict a subject or an object. The position of an entity is thus an important feature that must be considered. To be used as input, an event is represented by the concatenation of s, r and o embeddings (from q^{R}), and is given to the encoder to update ρ_s^{t-} and ρ_o^{t-}.

After the update of the embeddings, we compute the compatibility of the entities for a relation type. Here, the representation of an entity e is the concatenation of $\boldsymbol{\rho}_e^t$ and $\boldsymbol{\alpha}_e^t$. The knowledge acquired by AttrNet will then help RelNet to make predictions on future relational edges. Indeed, as stated in Assumption 5.1.1, the attributive behavior of an entity is an additional information that makes relational prediction more precise. For instance, the successive positions of a vessel give clues on its destination, or its speed is an indicator of its current activity (slow while transhipping, fast if under way...). Moreover, the attributive embedding may be the only source of information if no event including an entity happened. Static attributes are also an indicator if no attributive event has happened either. However, as stated in Assumption 5.1.2, we do not use relations to predict the next attribute: since those are mostly time series, we assume that adding the event may unnecessarily complexify the learning process. Of course, there are some cases where relational events may help to predict attributes: entering a port means that the speed is low, or tugging another vessel implies that the two vessels are close. These predictions are not in the scope of Joint Evolution and we focus on the relation prediction problem assisted by attributive knowledge.

Note that both networks work differently during the training and inference phases, about what more details are given in Section 5.2.3.

5.2.2 Input data

As stated before, we use DKG^{A} and DKG^{R} to train two encoder-decoder systems. The events are sorted by timestamp and we use challenges to guide the learning (challenges correspond to ground truths in classification). For an attributive quadruplet $q^{\mathrm{A}} = (e_i, a_j, v_x, t_m) \in DKG^{\mathrm{A}}$, its challenge is the next change of value of the attribute type a_j for entity e_i in the training data, e.g. $q'^{\mathrm{A}} = (e_i, a_j, v_y, t_n)$ with $t_m < t_n$ and $v_x, v_y \in \mathrm{D}(a_j)$. The challenge of a relational quadruplet $q^{\mathrm{R}} = (e_i, r_a, e_j, t_x), i \neq j$, is composed of two events: the next relational event for e_i and the next relational event for e_j (which can be the same if they are both involved in their respective challenges). This time, challenges are not dependent of the type of relation, so $q'^{\mathrm{R}} = ((e_i, r_b, e_k, t_y), (e_j, r_c, e_\ell, t_z))$, with $t_x < t_y$ and $t_y < t_z$. Note that e_i and/or e_j could also be object in their challenge. This way, we learn representation vectors that are tailored to predict the next event that will happen to an entity.

5.2.3 Training and inference

Joint Evolution behaves differently when in training phase or in inference phase. The training phase defines the model and its parameters. It is then used to update the embeddings when new data arrives. These updated embeddings are then used to make predictions (Figure 5.2).

During the training phase, the network is fed with both inputs and their challenges. AttrNet is trained to predict the next value of an attribute as close as possible to the challenge value. For RelNet, we try to increase the compatibility of the entities w.r.t. to the challenge events.

During the inference phase, we distinguish two usecases: streaming and prediction. During the streaming phase, only the encoder is used to update the embeddings with new events.

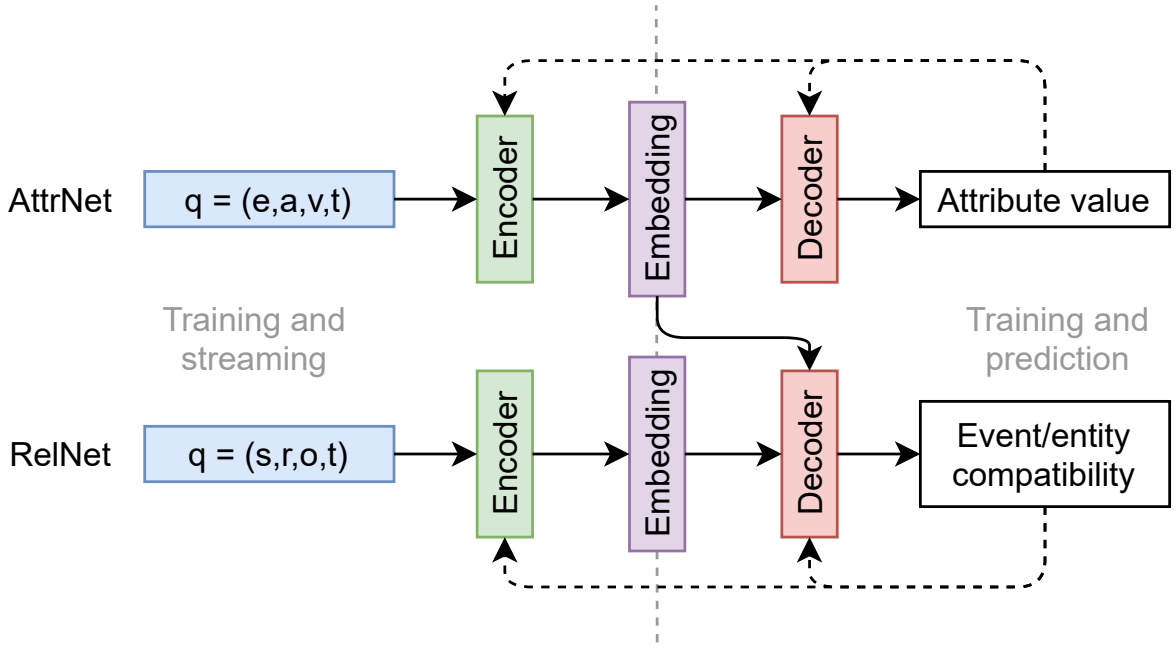


Figure 5.2: Training, streaming and prediction phases.

Then, when a prediction is required, we can ask the model to answer the question $(s, r, ?, t)$ (or $(s, ?, o, t)$, or $(?, r, o, t)$), and it will output the most compatible entity.

In a nutshell:

- We use two encoder-decoder systems: one for DKG^{A} and one for DKG^{R};
- Embeddings are used to predict attribute values and event scores;
- Both embeddings ($\boldsymbol{\rho}_e^t$ and $\boldsymbol{\alpha}_e^t$) are used to predict new relations;
- The state of knowledge can be updated in a streaming fashion.

5.3 Preparing the data: challenges and sequence ordering

To learn accurate representations of the behavior of entities, not only must the data be ordered by timestamp, but also the events occurring at the same time for the same entity must be processed independently. Extracting independent sequences for multi-relational data is not a trivial task and requires some preparation. We will start with the challenging process, followed by the ordering in independent sequences.

5.3.1 Mixing up attributes and relations

One important thing is that AttrNet and RelNet must be trained simultaneously. If an attribute gets a new value between two relational events, this must be taken into account (Figure 5.3). To do that, we need to put attributes and relations in the same sequence of events, even if their quadruplet structures and their encoders are different.

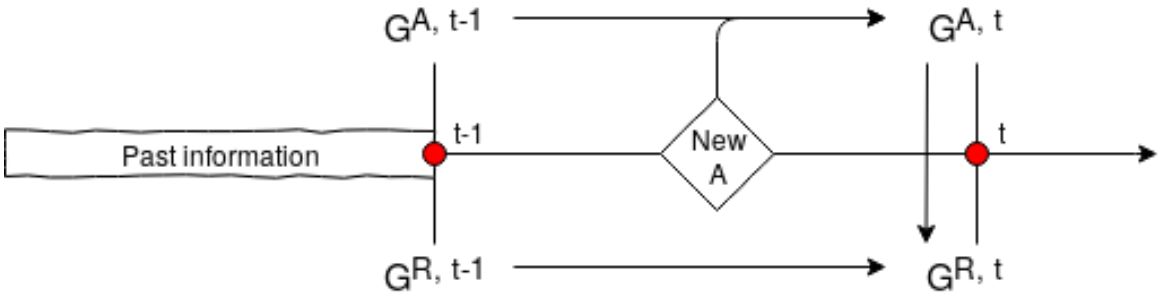


Figure 5.3: If an attribute change occurs between t-1 and t, it must be considered when predicting an event at time t. Here, G refers to a *DKG*.

5.3.2 Assigning challenges

Our input data is the knowledge graph *DKG*. To make it easily readable, we write it as a sequence of quadruplets q^{R} and q^{A}. Each entity, each relation and attribute type is referenced by an integer, and attribute values are floating numbers. Thus, we have input data that looks like $q^{\mathrm{R}} = (2, 1, 23, 1614626755)$ and $q^{\mathrm{A}} = (2, 5, 36.283, 1614626235)$.

As previously mentioned, the challenge of an event is its direct successor in the timeline for the involved entity. It should be noted that even if a relational event happens between two attribute updates, it will not be used as the challenge of the first one: DKG^{A} and DKG^{R} are clearly separated here. This is straightforward for q^{A}, but what about q^{R} that includes two entities?

5.3.2.1 Challenges for DKG^{R}

Events in DKG^{R} involve two entities, s and o. Then, what should their challenges be? The next event including s would be a nice candidate since it is the main actor in the relation (if r is not symmetric), but this completely excludes the future of o. We thus choose to use two challenges per input, one for each entity. A sample of the input data will thus look like this:

$$\{\text{"data"}: (s, r, o, t), \text{"challenges"}:\{(s, r', o', t'), (s'', r'', o, t'')\}, \text{"type"}:\text{"relation"}\}$$

Note that s and o can be either subject or object in their respective challenges, and that r, r' and r'' are not necessarily the same relation. We keep track of the type of the quadruplet to tell the difference between relation and attributes when both will be mixed in the event sequence. The challenge of s is the closest event in DKG^{R} that involves s, with $t < t'$. It means that by construction, no relational event between t and t' happens.

5.3.2.2 Challenges for DKG^{A}

Challenging the attributive graph is easier since only one entity is involved each time. Since we want to learn how an entity behaves, we need to learn how its attributes evolve. Thus, a straightforward way to do this is to learn the time series of each attribute type. For each quadruplet $q^{\mathrm{A}} = (e, a, v, t)$, we assign as challenge the next event in the timeline changing the value of a. A sample of the input data will thus look like this:

$$\{\text{"data"}: (e, a, v, t), \text{"challenges"}:(e, a, v', t'), \text{"type"}:\text{"attribute"}\}$$

with $v, v' \in \mathrm{D}(a)$ and $t < t'$ with t' the smallest timestamp $> t$ where e changed for value of a.

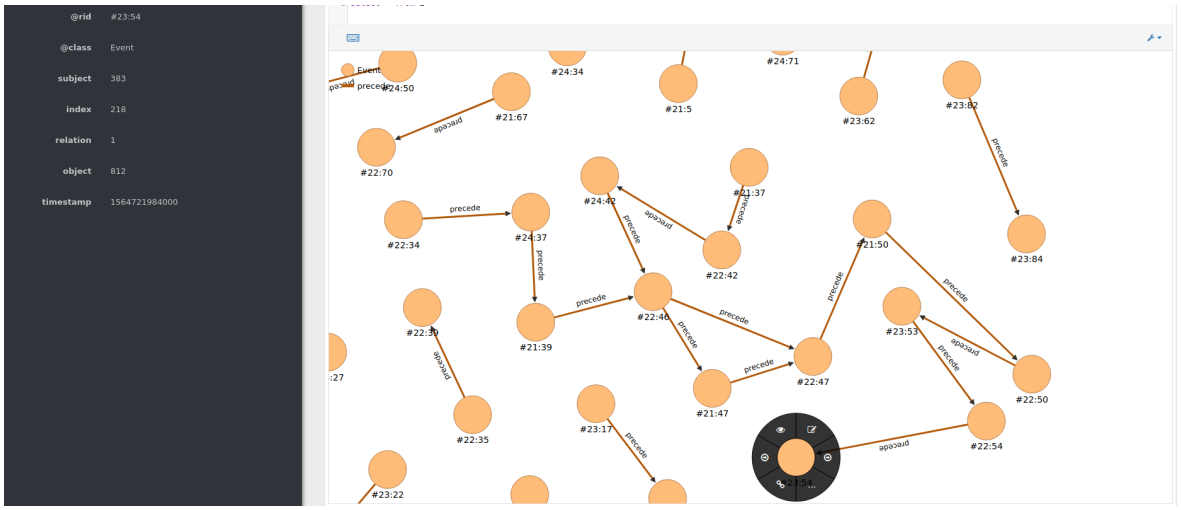


Figure 5.4: Sample of the precedence graph. Each node is a quadruplet and each edge a precedence constraint to resolve before accessing the target node.

5.3.3 Precedence graph

We now have defined challenges for all our inputs, but we still do not know in which order to process them. This is important because events must be included in an embedding in the order in which they happen, and dependencies exist between events.

Our solution is to generate a precedence graph of events, where events are nodes and each directed edge adds a precedence constraint between two nodes, i.e., the target node cannot be processed until all its parents nodes are. This graph is illustrated in Figure 5.4.

The precedence graph is defined as follows:

Definition 12 (precedence graph)**.** *Let DKG be a set of attributive and relational quadruplets, and for each quadruplet* $q \in \mathrm{Q}$*, write* t_q *for the timestamp of* q *and* $ent(q)$ *for the entities involved (i.e.,* $ent(s,r,o,t) = s, o$ *and* $ent(e,a,v,t) = e$*). We define* $\overline{\mathrm{G}}$ *by:* $\forall v_x, v_y \in \mathrm{V}$*,* $(v_x, v_y) \in \mathrm{E}$ *if* $t_x < t_y$ *and* $ent(v_x) \cap ent(v_y) \neq \varnothing$*, and the* precedence graph *of the set of quadruplets to be the transitive reduction of* $\overline{\mathrm{G}}$*.*[2]

The precedence graph generates a link between an event and its direct successor(s). There can be several successors if there are several candidate events with the same timestamp. The dependencies between the events can now be used to extract the events ready to be processed.

With the precedence graph, we can now know which events can be included in the input sequence while meeting the requirements on event ordering. We use a topological sort on the precedence graph to get each node which does not possess any parent node. When a node is taken for processing, the constraint it applies to its target node(s) is lifted and, if all their parents have been processed, they can now be chosen too.

In a nutshell:

- Attributes and relations must be present in the same stream of data;
- The challenge of an event is the closest one in the timeline with a common entity;
- Creating a precedence graph provides the available samples with a topologic sort;

[2]That is, if $(v_x, v_y), (v_y, v_z) \in \mathrm{E}$, then $(v_x, v_z) \notin \mathrm{E}$ despite meeting the requirements (direct successor(s) condition).

5.4 Conclusion

We presented a formalization of the link prediction problem for MSA and a general architecture for a solution based on a double encoder-decoder system. In the next chapter, we will describe each part in more details and explain how we ended up making these choices.

CHAPTER 6

Joint Evolution, a network pair for link prediction on DAKG

6.1 The choice of recurrent networks

To learn on sequential data (dynamic knowledge graphs in our case), recurrent neural networks (RNNs) may prove useful. RNNs are made of non-linear units with recurrent connections and are able to learn features and long-term dependencies. Our model is inspired by [Trivedi et al., 2017] which use RNNs, it is thus a natural choice for Joint Evolution. Besides, many recent models are using these networks to deal with *DKG*s [Ma et al., 2020, Dai et al., 2016, Goyal et al., 2018, Trivedi et al., 2019, Zhao et al., 2020, Hajiramezanali et al., 2019].

6.1.1 RNNs

Popularized by [Elman, 1990], RNNs are neural networks that can process sequences by having a "memory" of the previous computations. Figure 6.1 shows the structure of an RNN: the input is a sequence of vectors, and each vector is a timestep processed by a cell. The black square represents the recurrent step where the previous hidden state is used to compute the current one. Depending on the objective, the output may be taken for each element of the sequence (e.g. for translation) or only at the end of the sequence (e.g. for embeddings).

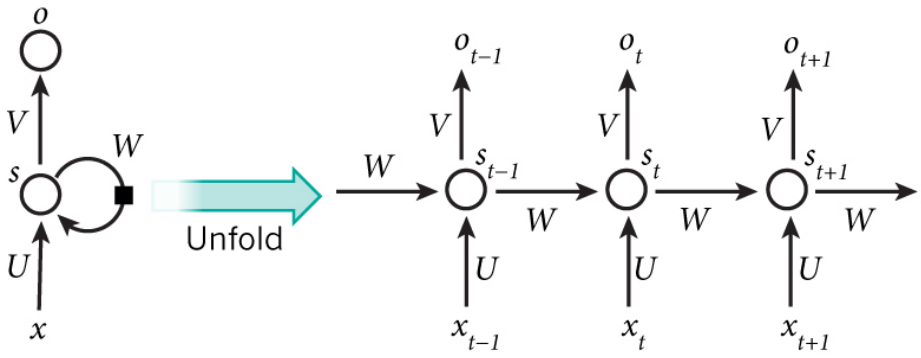


Figure 6.1: A folded and an unfolded RNN [LeCun et al., 2015]

RNNs are formaly defined as follows:

- Let X = $x_0 ... x_t ... x_n$ be a sequence of size $n \in \mathbb{N}$ with $t \in [0, n]$ the current timestep,
- Let U, V and W be weight matrices of the network,
- Let s_t be the cell state at timestep t,
- Let o_t be the output of the cell at timestep t,
- Let b_h be the bias vector of the hidden units and b_o the bias vector of the output layer,
- Let f be a non-linear activation function.

$$s_t = f(\mathrm{U}x_t + \mathrm{W}s_{t-1} + b_h)$$

$$o_t = softmax(\mathrm{V}s_t + b_o)$$

Note that U, V and W are the same matrices throughout the recurrent process. Therefore, such a network can handle input sequences of different lengths because the number of weight matrices is not varying.

However, RNNs have a major drawback: during backpropagation, the gradient often vanishes and sometimes explodes [Bengio et al., 1994], making the learning step harder. And because it fails to propagate the error through deep layers, long-term dependencies are overshadowed by short-term dependencies. The non-linear f function is often a $tanh$ (Equation 6.1) or a R*e*LU (Equation 6.2) function. $tanh$ is in the range [-1,1] and since only near-zero values are mapped to near-zero outputs, it solves the vanishing gradients problem to some extent. R*e*LU is more efficient regarding the vanishing gradients but is has no upper bound and is not differentiable at 0, nor zero-centered.

$$tanh(x) = (e^x - e^{-x})/(e^x + e^{-x}) \tag{6.1}$$

$$\mathrm{R}e\mathrm{LU}(x) = x^+ = max(0, x) \tag{6.2}$$

6.1.2 LSTMs

Hochreiter and Schimdhuber [Hochreiter and Schmidhuber, 1997] introduced in 1997 the Long Short-Term Memory (LSTM) model to cope with the vanishing gradient problem. The classic RNN cells are replaced by memory cells whose inputs and outputs are controlled by gates (input, output, reset). These gates manage the flow of information to preserve or erase the features and states of the previous steps. A chain with only a few linear interactions, the cell state, is introduced, allowing the information to travel along it unchanged if needed. The information within it is regulated by the gates.

The major drawback of LSTMs is the computational complexity in the hidden layers: a standard LSTM requires four times more parameters than a vanilla RNN [Mikolov et al., 2014].

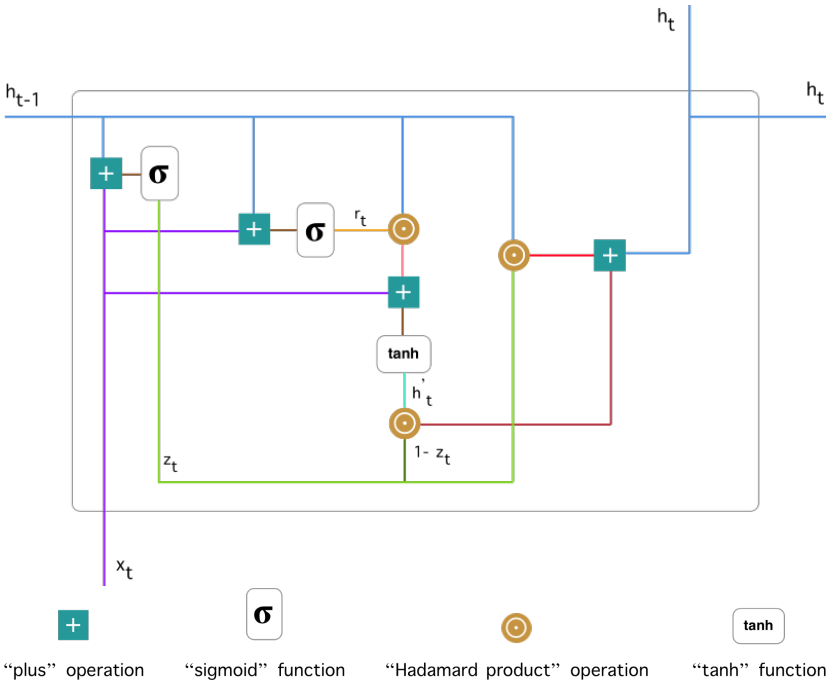


Figure 6.2: A GRU cell

6.1.3 Gated Recurrent Units

To reduce the complexity of LSTMs, Cho et al. [Cho et al., 2014a] proposed the Gated Recurrent Units (GRUs). Like the LSTMs, GRUs have gating units that modulate the flow of information inside the unit, however without having separate memory cells [Chung et al., 2014]. Figure 6.2 illustrates how a GRU cell works[1].

6.1.3.1 The gates

The input and block gate are replaced by a single *update gate* that determines how much of the past information needs to be passed along to the future:

$$z_t = \sigma(\mathrm{W}_z x_t + \mathrm{U}_z h_{t-1} + b_z) \tag{6.3}$$

with W and U weight matrices, b_z the bias, x_t the input and h_{t-1} the previous hidden state.

The forget gate becomes a reset gate that decides how much of the past information to forget. It is written like the update gate but its weights and usage differ:

$$r_t = \sigma(\mathrm{W}_r x_t + \mathrm{U}_r h_{t-1}) \tag{6.4}$$

While LSTMs have a cell state and a hidden state, GRUs only keep the hidden state. To compute the current memory content, the previous hidden state is processed through the *reset gate*, then added to the input and squashed through a $tanh$ activation function:

$$h'_t = tanh(\mathrm{W} x_t + r_t \odot \mathrm{U}_t h_{t-1}) \tag{6.5}$$

[1] https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

with $\odot$ the pointwise multiplication. Note that writing $\mathrm{U}_t(r_t \odot t_{t-1})$ gives the same results as $(r_t \odot \mathrm{U}_t h_{t-1})$ [Chung et al., 2014].

At last, the final memory at the current time stem processes the previous hidden step h_{t-1} and the current memory content through the *update gate*:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \tag{6.6}$$

The higher the values of the update gate $z_t \in [0, 1]$, the more the previous hidden state h_{t-1} is kept and the current memory content h'_t is forgotten (i.e. the more past information is passed along to the future).

6.1.3.2 Performances

If GRUs simplify the LSTM model, this does not mean that they are less efficient: [Chung et al., 2014] show that both models perform better than the other only in certain tasks, which suggests that they can, on average, output a result of the same quality. Since it is not clear which one is better suited to our usecase, we chose the simplest model, namely, GRUs.

6.1.4 Other models

In [Salehinejad et al., 2017], the authors make a list of major RNN architectures. Some recent models seem to be promising but require more research and comparative studies and are thus risky to use for our work. In the context of this thesis, we are looking for a recurrent model to use Know-Evolve [Trivedi et al., 2017] and improve it. GRUs are the choice of simplicity and should perform better than the vanilla RNNs used in [Trivedi et al., 2017].

In a nutshell:

- Recurrent neural networks are often used with *DKG*s;
- They are present in [Trivedi et al., 2017], our base model;
- Gated Recurrent Units provide a good trade-off between complexity and performance.

6.1.5 Batch creation

Training a neural network requires to feed it with batches of data. A batch is a set of samples that is processed simultaneously through the network to speed up the learning process. Indeed, the backpropagation is thus done only every X samples (with X the size of the batch), which improves the error reduction and is faster than backpropagating for each sample. However, some constraints must be respected: to ensure that we process the events in the correct order and that two events about the same entity are not processed at the same time, an entity cannot be present two times in a batch. Such a situation would result in a loss of information since only one $\boldsymbol{\rho}_e^t$ would be kept for two updated $\boldsymbol{\rho}_e^{t-}$.

To create such batches, we could create a slice of the graph for each timestamp, but this would demand too many slices because of the update frequency of AIS. We could also create batches of samples sorted by timestamp and add a unicity constraint on the presence of an entity in the batch. But doing so would require storing in memory all unprocessed samples and do several passes over them until they are all consumed, which is a time-consuming task.

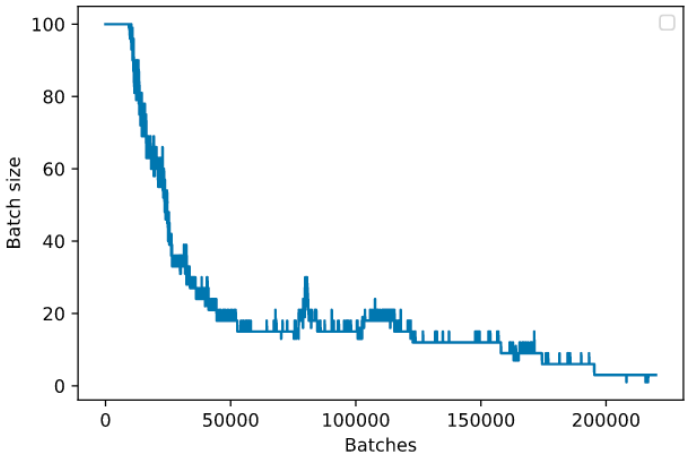


Figure 6.3: Evolution of batch size for datAcron (max size 100)

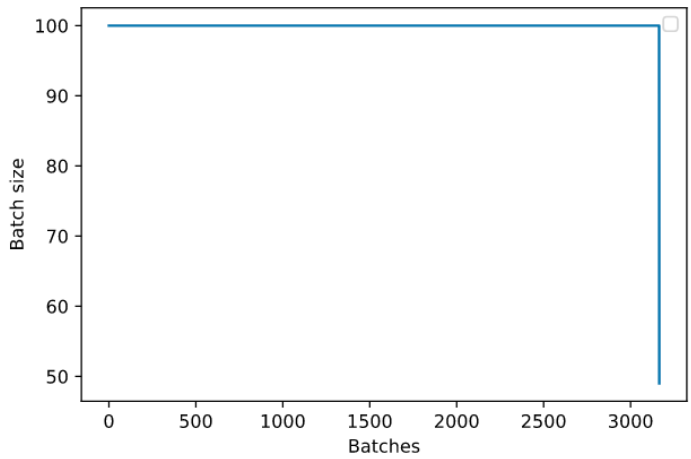


Figure 6.4: Evolution of batch size for DBLP (max size 100)

Our solution is to use the precedence graph presented in Section 5.3.3. Using topological sort, we can extract a chosen number of independent samples. The only requirement is to set the maximum batch size, since they are created during the preprocessing step. This system introduces dynamicity in the batch size, because there are no guarantees that the same number of events will be without constraints each time. This generally means that there will be fewer and fewer constraint-free events as the graph is parsed, leading to a decreasing batch size over time. Figure 6.3 is a perfect example of this: this extract from the datAcron dataset (detailed in Section 7.1) has 364 entities, 764 relational quadruplets and more than 4M attributive quadruplets. Since there are not a lot of entities, they occur more frequently, resulting in more sequences shorter than the fixed maximum batch size. However, this is not the case when there are a lot of entities: our DBLP extract (Section 7.2) has 6.147 entities, 261,324 relational quadruplets and 139,152 attributive quadruplets. DBLP is a lot sparser than datAcron, resulting in a constant batch size since there are always enough unconstrained events to sample, excepted for the last batch (Figure 6.4).

6.2 AttrNet

The objective of AttrNet is to capture the evolution of an entity's attributes. To do so, we build a recurrent neural network that learns the time series of the attributive events.

6.2.1 A first version

We started with an RNN version of AttrNet based on the Know-Evolve relational network. Once adapted, the encoder-decoder structure of AttrNet is defined as:

$$\alpha_e(t_p) = \sigma(\boldsymbol{w}_t(t_p - t_{p-1}) + \mathbf{W}_a^{emb}.\ \alpha_e^{t_{p-1}} + \boldsymbol{w}_a^{in}.\ v + b_a^{in}) \tag{6.7}$$

$$v'(t_{p+1}) = \sigma(\boldsymbol{w}_a.\ \alpha_e^{t_p} + b_a) \tag{6.8}$$

Equation 6.7 updates the attributive embedding of entity e (encoder) and Equation 6.8 predicts the value of the attribute a at time t_{p+1} (decoder). t_{p-1} is the last time entity e was involved in an event before the current event (at time t_p). $(t_p - t_{p-1})$ is the temporal drift between the last and the new event, $\alpha_e^{t_{p-1}} \in \mathbb{R}^m$ is the last state of the dynamic attributive embedding, $\boldsymbol{w}_t, \boldsymbol{w}_a^{in} \in \mathbb{R}^m$ and $\mathbf{W}_a^{emb}, \boldsymbol{w}_a \in \mathbb{R}^{1\times m}$ are weight parameters of the model, v is the value of the attribute and $b_a^{in}, b_a \in \mathbb{R}^m$ are biases. $\boldsymbol{w}_t$ captures variation in temporal drift and is shared across all attributes. $\boldsymbol{w}_a^{in}$ is a weight matrix different for each a: it captures specifically how the values evolve for this attribute over all the entities, along with b_a^{in} and b_a which are also attribute related. $\boldsymbol{w}_a$ considers which part of the embedding is required to predict the attribute value of type a. σ denotes the nonlinear activation function, that will be *tanh* here.

Once the prediction is made, the network must be trained and optimized to be as close as possible from the challenge value. An obvious objective function for this prediction task is Mean Squared Error:

$$\mathrm{L}oss = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (v(t_{p+1})_i - v(t_{p+1})_i^*)^2 \tag{6.9}$$

where $v(t_{p+1})_i^*$ is the ground truth challenge, i.e. the value in the quadruplet given as challenge and N is the size of the batch.

Figure 6.5 shows how the attribute network is trained. Given a new observed value at time t, the old embedding α_e^{t-1} is updated using the network parameters W and the new value. With the freshly updated embedding α_e^t, the value $\widehat{v}$ for time $t+1$ is predicted and compared to the ground-truth value at time $t+1$. In Equations 6.7 and 6.8, $\mathbf{W}_a^{emb}$ and $\boldsymbol{w}_t$ are the parameters that learn how to embed the new information and $\boldsymbol{w}_a^{in}$ learns how to make a prediction value from this new embedding. Thus, the model is trained to update embeddings that can be used to predict next values, creating a representation of the entity behavior. Note that each $\mathbf{W}_a^{emb}$ is different for each a, the attribute type.

6.2.2 Room for improvement

The presented version of AttrNet is well suited for training but has a problem during the prediction phase: we can predict the next value of an attribute from the embedding of the entity at hand, but we have no input for the timestamp of the prediction. Thus, we must add

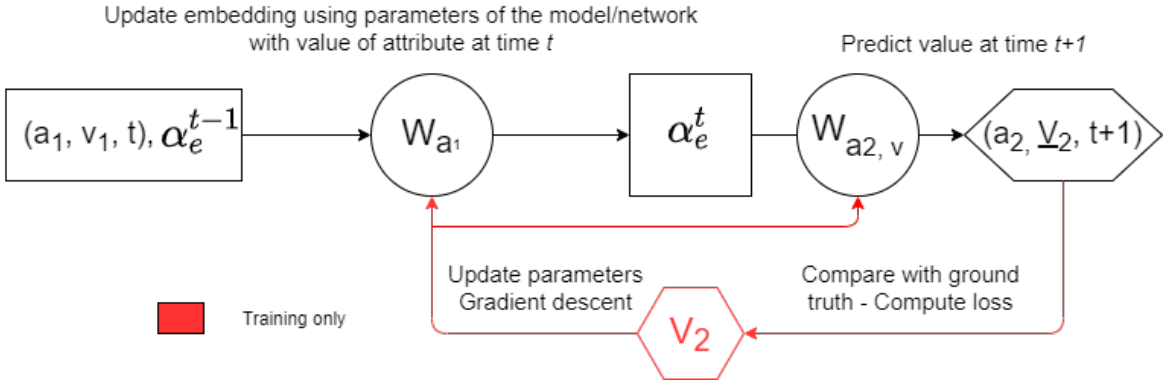


Figure 6.5: Attribute learning process. Note that a_1 and a_2 can be different.

the temporal drift between the last attribute value change t_p and the time of the prediction t_{p+1} (the drift in the encoder is between t_{p-1} and t_p).

$$v'(t_{p+1}) = \sigma((t_{p+1} - t_p)\boldsymbol{w}_t + \boldsymbol{w}_a \cdot \boldsymbol{\alpha}_e^t + b_a) \tag{6.10}$$

In Section 6.1, we discussed the advantages of GRsU over RNNs. The encoder was an RNN, so we switched it to a GRU (Equations 6.11 to 6.14). This solved the vanishing gradient problem that was occurring on long trainings and speeded up the learning (same loss reached in fewer epochs).

$$\boldsymbol{z}_t = \sigma((t_p - t_{p-1})\boldsymbol{w}_t^z + v\boldsymbol{w}_z^a + \mathbf{U}_z\boldsymbol{\alpha}_e^{t_{p-1}} + \boldsymbol{b}_z) \tag{6.11}$$

$$\boldsymbol{r}_t = \sigma((t_p - t_{p-1})\boldsymbol{w}_t^r + v\boldsymbol{w}_r^a + \mathbf{U}_r\boldsymbol{\alpha}_e^{t_{p-1}}) \tag{6.12}$$

$$\boldsymbol{h}'_t = tanh(v\boldsymbol{w}_t^a + \boldsymbol{r}_t \odot \mathbf{U}_t\boldsymbol{\alpha}_e^{t_{p-1}}) \tag{6.13}$$

$$\boldsymbol{\alpha}_e^t = \boldsymbol{z}_t \odot \boldsymbol{\alpha}_e^{t_{p-1}} + (\mathbf{1} - \boldsymbol{z}_t) \odot \boldsymbol{h}'_t \tag{6.14}$$

The last improvement is about the decoder. Predicting the next value directly from the embedding can be a tedious task for a single fully connected layer, especially if the dimension of the embedding is large. To add more expressivity to the model and improve the prediction, we add another hidden layer in the decoder, projecting the embedding to half its size before making the prediction (Equation 6.15). This layer improves the results on attribute value prediction.

$$hidden(\boldsymbol{\alpha}_e^t) = tanh(\boldsymbol{w}_x\boldsymbol{\alpha}_e^{t_p} + b_x)) \tag{6.15}$$

$$v = \sigma((t_{p+1} - t_p)w_t + \boldsymbol{w}_v^a \cdot hidden(\boldsymbol{\alpha}_e^{t_p}) + b_a) \tag{6.16}$$

6.2.3 Asynchronous GRUs

GRUs are usually used with long sequences, and only the output of these sequences is kept. Inpuiting data in a GRU sample by sample is generally less efficient, because predictions are made only with a window of size one in the past (i.e. the last sample), while current methods use more previous samples to predict the next value. However, given our setting, we cannot process the sequence as a whole since it is dependent on other events that happen in between (e.g. the time series of a vessel speed). Since we are using the embeddings as the memory

of the network, we can consider that we process the time series asynchronously: the samples are processed one by one, but with the memory associated to the entity. It may have changed because of another happened event between two samples of the same type, but it will still be more similar to a normal sequence processing than a shared memory between entities.

In a nutshell:

- AttrNet is also inspired from Know-Evolve RNN;
- We use Mean Squared Error to train the network;
- A temporal drift is added to the decoder to enable the choice of the prediction time during inference;
- Some parameters of the decoder are attribute type specific;
- The use of GRUs improves learning;
- More hidden layers improve the prediction results in the decoder.

6.3 RelNet

The architecture of RelNet is inspired by [Trivedi et al., 2017], but we had to do a lot of modifications. For instance, Know-Evolve maximizes the conditional intensity of the input event. In Joint Evolution, we assign challenges to events and try to maximize the conditional intensity of the future event (the challenge). In this section, we will present the architecture of RelNet and how it was designed.

6.3.1 A first version

As a basis for RelNet, we transposed the encoder-decoder system from [Trivedi et al., 2017] to our setting.

- Subject embedding:

$$\boldsymbol{\rho}_s^{t_p} = \sigma(\boldsymbol{w}_t^s(t_p - t_{p-1}) + \mathbf{W}^{hh}.\boldsymbol{h}_s^{t_{p-}}) \tag{6.17}$$

$$h_s^{t_{p-}} = \sigma(\mathbf{W}^h[\rho_s^{t_{p-1}} \oplus \rho_o^{t_p-} \oplus r_p]) \tag{6.18}$$

- Object embedding:

$$\boldsymbol{\rho}_o^{t_q} = \sigma(\boldsymbol{w}_t^o(t_q - t_{q-1}) + \mathbf{W}^{hh}.\boldsymbol{h}_o^{t_{q-})} \tag{6.19}$$

$$\boldsymbol{h}_o^{t_q-} = \sigma(\mathbf{W}^h[\boldsymbol{\rho}_o^{t_{q-1}} \oplus \boldsymbol{\rho}_s^{t_q-} \oplus r_q]) \tag{6.20}$$

- With the score computation defined as:

$$g_r^{s,o}(t_u) = \sigma(\mathbf{W}_r[\boldsymbol{\rho}_s^{t_p} \oplus \boldsymbol{\alpha}_s^{t_x} \oplus \boldsymbol{\rho}_o^{t_q} \oplus \boldsymbol{\alpha}_o^{t_v}]) \tag{6.21}$$

where $\boldsymbol{\rho}_s, \boldsymbol{\rho}_o \in \mathbb{R}^n, \boldsymbol{\alpha}_s, \boldsymbol{\alpha}_o \in \mathbb{R}^m, r^p, r^q \in \mathbb{R}^c, \boldsymbol{w}_t^s, \boldsymbol{w}_t^o \in \mathbb{R}^{n\times 1}, \mathrm{W}^{hh} \in \mathbb{R}^{n\times\ell}, \mathrm{W}^h \in \mathbb{R}^{\ell\times(2n+c)}$ and $\mathrm{W}_r \in \mathbb{R}^{2n+2m}$.

The encoder is pretty much the same (Equation 6.17 to 6.19). In the computation of the bilinear score, we want to include information from AttrNet, and we do so by concatenating

the relational and attributive embeddings of the entities before computing the score. Indeed, as stated in Section 5.2.1.2, we assume that the attributive knowledge will help the prediction of missing or future relations.

6.3.2 Inclusion of challenges

We now add challenges for each input. We start with three challenges: one for the subject, one for the relation type and one for the object. The objective is to be able to predict either of the three members of the quadruplet, and thus to have an objective for each. For subject and object, we do as stated in Section 5.3. For the relation type, we take the input itself as challenge, since it would make no sense to take the next event with this relation type: the way the relation type interacts with the entities does not depend on the time series. Of course, some relationships may have different temporalities, e.g. a vessel will enter a port more often than it changes of owner. But what is more important is how the relation type links the two entities. Figure 6.6 illustrates the learning process with challenges. With the three challenges, there are now three computations of the bilinear score (Equations 6.22 to 6.24).

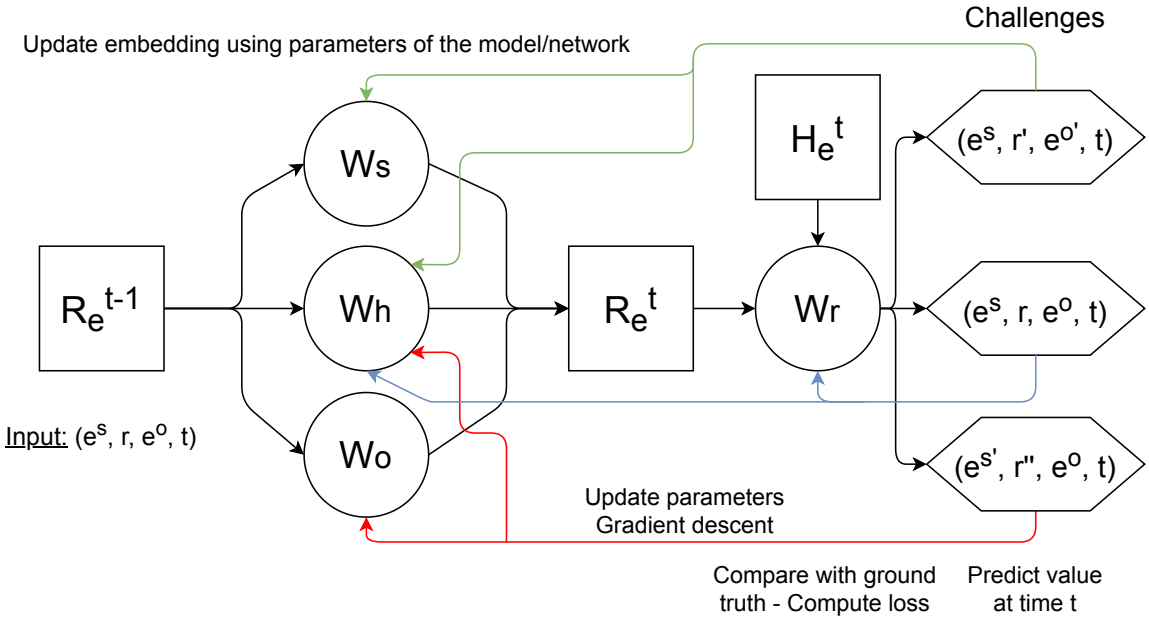


Figure 6.6: Relational learning process.

$$g_r(t_u) = \sigma(\mathbf{W}_r^r[\boldsymbol{\rho}_s^{t_p} \oplus \boldsymbol{\alpha}_s^{t_x} \oplus \boldsymbol{\rho}_o^{t_q} \oplus \boldsymbol{\alpha}_o^{t_v}]) \tag{6.22}$$

$$g_{e^s}(t_u) = \sigma(\mathbf{W}_r^s[\boldsymbol{\rho}_s^{t_p} \oplus \boldsymbol{\alpha}_s^{t_x} \oplus \boldsymbol{\rho}_{o'}^{t_q} \oplus \boldsymbol{\alpha}_{o'}^{t_v}]) \tag{6.23}$$

$$g_{e^o}(t_u) = \sigma(\mathbf{W}_r^o[\boldsymbol{\rho}_{s'}^{t_p} \oplus \boldsymbol{\alpha}_{s'}^{t_x} \oplus \boldsymbol{\rho}_o^{t_q} \oplus \boldsymbol{\alpha}_o^{t_v}]) \tag{6.24}$$

We can note that here, $\boldsymbol{\rho}_{o'}^{t_q} = \boldsymbol{\rho}_{o'}^{t_{q^-}}$ since o' was not involved in the input event. However, it can be processed elsewhere in the batch since the precedence constraint does not apply to challenges, only to input events. Depending on the order in which the events are processed, there is a probability that o' is also updated at t_q, but that the change is not included when

(s, r, o, t) is processed. We currently have no solution for this, and this will be a matter for future work in the input data ordering.

6.3.3 Training loss

Now that we have a score for the predictions, we must find a way to propagate the error and optimize the parameters of the network. We discuss here several solutions to tell the network what a good prediction is.

6.3.3.1 Softmax based

We consider a softmax function to deal with the prediction as a classification problem: we describe here the setting to cast the link prediction problem as a classification one.

Predictions For each part of the quadruple, different predictions need to be made. Thus, each event (e_1, r, e_2, t) has three challenges:

- The next event for e_1, (e_1, r', e_3, t'). Note that it could also be (e_3, r', e_1, t');
- The next event for e_2, (e_2, r'', e_4, t'') or (e_4, r'', e_2, t'');
- The challenge for the relation is the current event (e_1, r, e_2, t).

Now, for each of these challenges, a softmax must be performed to predict the other members of the quadruple. For e_1, e_1 is fixed and the two other members of the triple (time is left aside for now) are removed alternatively before doing the softmax, because this is what we will have during inference.

Prediction for the next event involving e_1 Equations 6.25 and 6.26 represent the softmax layer. Note that we added time as an additional information to the embedding, with a trainable parameter to modulate the temporal drift.

$$pred_o = softmax(\mathbf{W}^e_{soft}[(\boldsymbol{\rho}^t_{e_1} \oplus \boldsymbol{r}'_e) + \boldsymbol{w}^e_t \cdot (t' - t)]) \tag{6.25}$$

$$pred_r = softmax(\mathbf{W}^r_{soft}[(\boldsymbol{\rho}^t_{e_1} \oplus \boldsymbol{\rho}^t_{e_3}) + \boldsymbol{w}^r_t \cdot (t' - t)]) \tag{6.26}$$

where $\mathbf{W}^r_{soft} \in \mathbb{R}^{|\mathrm{R}| \times 2n}$ and $\mathbf{W}^e_{soft} \in \mathbb{R}^{|\mathrm{E}| \times (\ell + n)}$ are the parameters to cast the information to the number of classes, and $\boldsymbol{w}^e_t \in \mathbb{R}^{2n}$, $\boldsymbol{w}^r_t \in \mathbb{R}^{2n}$, $pred_o \in [0,1]^{|\mathrm{E}|}$ and $pred_r \in [0,1]^{|\mathrm{R}|}$. are the predictions made. The predicted class is the one with the highest value, and can be used to complete the quadruplet given as input (e.g. $(e_1, r', ?, t')$ with $pred_o$). Note that we omit α^t_e for the sake of clarity in this section.

Prediction for r The prediction for r is based on the current event to learn the dynamics of the relation type with the entities (Equations 6.27 and 6.28). The time is not required since we work on the same event, and we just want to learn the dynamics of the relation type, not its evolution over time. We then fix r and try to predict s and o.

$$pred_s = softmax(\mathbf{W}^e_{soft}[\boldsymbol{r}_e \oplus \boldsymbol{\rho}^t_{e_2}]) \tag{6.27}$$

$$pred_o = softmax(\mathbf{W}^e_{soft}[\boldsymbol{\rho}_{e_1} \oplus \boldsymbol{\rho}_r]) \tag{6.28}$$

One problem here is that we are using the same parameters than for the prediction of future events. But we do not want to learn the same thing since learning the relation type is to learn on the current event. Moreover, the parts with s and o learn to predict future events, not how two entities interact depending on the relation type. Thus, it may have no sense to use the same weights for these two different tasks.

Prediction for the next event involving e_2 The prediction for the next event of e_2 is very similar to e_1 (Equations 6.29 and 6.30):

$$pred_s = softmax(\mathbf{W}^e_{soft}[(\boldsymbol{\rho}^t_{e_4} \oplus \boldsymbol{\rho}^t_{r''}) + \mathbf{W}^e_t \cdot (t'' - t)]) \tag{6.29}$$

$$pred_r = softmax(\mathbf{W}^r_{soft}[(\boldsymbol{\rho}^t_{e_1} \oplus \boldsymbol{\rho}^t_{e_2}) + \mathbf{W}^r_t \cdot (t'' - t)]) \tag{6.30}$$

6.3.3.2 Score based

This method is based on Know-Evolve decoder system. From the updated embeddings, we compute the bilinear relational score $g^{s,o}_r$ (Equation 6.31), then the conditional intensity function $\lambda^{s,o}_r$ (Equation 6.32). At last, the negative log likelihood loss is applied (Equation 6.33) with N the number of events in the batch.

$$g^{s,o}_r = \boldsymbol{\rho}^{t\top}_s . \mathbf{R}_r . \boldsymbol{\rho}^t_o \tag{6.31}$$

$$\lambda^{s,o}_r(t|\bar{t}) = exp(g^{s,o}_r(\bar{t}))(t - \bar{t}) \tag{6.32}$$

$$-\sum_{p=1}^{\mathrm{N}} \log(\lambda^{s,o}_r(t_p|\bar{t_p})) \tag{6.33}$$

The conditional intensity $\lambda^{s,o}_r$ represents how likely it is for the event to occur. Thus, with the negative log likelihood, we maximize the probability that positive events (i.e. challenges) are predicted as likely.

Score computation If we take the previous example but with a score instead of a softmax, we have:

$$\begin{aligned} g^{e_1,e_3}_{r'}(t') &= \boldsymbol{\rho}^{t\ \top}_{e_1} \cdot \mathbf{R}_{r'} \cdot \boldsymbol{\rho}_{e_3} \\ g^{e_1,e_2}_r(t) &= \boldsymbol{\rho}^{\top}_{e_1} \cdot \mathbf{R}_r \cdot \boldsymbol{\rho}_{e_2} \\ g^{e_4,e_2}_{r''}(t'') &= \boldsymbol{\rho}^{\top}_{e_4} \cdot \mathbf{R}_{r''} \cdot \boldsymbol{\rho}_{e_2} \end{aligned} \tag{6.34}$$

with $\mathbf{R}_x \in \mathbb{R}^{n \times n}$. Then, we apply Equation 6.32 and 6.33 with t the time of the predicted events (t, t' and t'' in Equations 6.25 to 6.30) and $\bar{t}$ is the time of the input event (t in Equations 6.25 to 6.30).

One potential issue when predicting $pred_r$ is that t may be equal to $\bar{t}$ and the time delta in the conditional intensity $g^{e_1,e_2}_r(t)$ is thus equal to 0. We could remove the temporal drift from

the conditional intensity when we predict from r, but this would be inconsistent with the other loss computations. This cannot happen with $pred_s$ and $pred_o$ since $\bar{t} > t$ by construction of the challenges.

6.3.3.3 Softmax or score

The softmax method is less intuitive to use and has no defined process for time, while the score-based prediction uses a temporal point process. Both methods suffer from a lack of clarity when predicting on the current event (for the relation type).

For these reasons, we choose to work with a score and not with a classifier. Also, the challenge used for the relation type is tricky to implement and is not consistent with the two others. Since the relation type is already represented in the encoder (r_p, r_q in Equations 6.18 and 6.20) and in the decoder ($\mathbf{R}_r$ in Equation 6.31) as trainable parameters, a third challenge is not needed to model the interaction type. We thus remove this challenge to only keep the ones for subject and object, consistently with what is presented in Section 5.3.

6.3.4 Survivance

The current loss term maximizes the score of each happened event, but nothing prevents it to maximize each event and still reach its goal. To balance the positive events, we add negative events whose scores must be minimized (as in [Trivedi et al., 2017]). A negative event is an event that did not happen in a timeframe [T, T+t[.

The problem is that computing all possible examples is very expensive, with a complexity of O(|R||E|^2) (Equation 6.35).

$$\mathrm{S} = \sum_{r=1}^{|\mathrm{R}|} \sum_{s=1}^{|\mathrm{E}|} \sum_{o=1}^{|\mathrm{E}|} \int_0^{\mathrm{T}} \lambda_r^{s,o}(\tau|\bar{\tau}) d\tau \tag{6.35}$$

[Trivedi et al., 2017] propose a solution to reduce the complexity to O(|E||R|j) with j the size of the batch:

- For each event e in the batch, compute the survivance of the subject (resp. object)
- To do so, compute the survivance for the subject (resp. object) with every possible object (resp. subject) within the batch, but the resulting triple must not exist in the batch. The relation type may or may not be fixed.
- Add the survivances together.

This is possible because their batches are sliding windows on the temporal sequence and represent all events in the timeframe [T, T+t[of the batch. With our batching system, it is not possible to do so since in general, a batch does not contain all the events happened during its time frame.

To bypass this issue, we propose our own sampling strategy for negative events. Let (s, r, o, t) be an input event and (s, r', o', t') or (s', r', s, t') its challenge for s. We set the time frame of each event in the batch between its own timestamp t and the timestamp of its challenge t', since no events involving s happens in $]t, t'[$ by construction of the batches. Then, we sample events with a random timestamp $t'' \in]t, t'[$ by fixing s as it is in the challenge (subject or object) and replacing the two other slots by all possible values. This sets the

complexity of the sampling strategy to O(|E||R|) for each event of the batch, or O(|E||R|B) if we sample B timestamp(s) in $]t, t'[$. The survivance for s is thus generated using Equation 6.36.

$$S^s = \frac{\sum_{r\in R}\sum_{o'/s'\in E}\sum_{\tau} f(\tau) log(\lambda_r(\tau|t))}{|R| \times |E| \times B} \tag{6.36}$$

with τ the set of sampled timestamps from $]t, t'[$ and λ_r stands for $\lambda_r^{s,o'}$ or $\lambda_r^{s',s}$ depending on whether the challenge has s as its subject or object.

We add a term $f(\tau)$ that depends on the sampled time τ, based on the following hypothesis: the ground truth events at t and t' are based on observations, meaning that it is less likely that we missed an event around these timestamps. In other words, a negative event happening near t or t' is "falser" than an event happening far from them since an observation was made around it and it was not found. Thus, the survivance is lower the furthest t'' is from t and t', as stated in Equation 6.37:

$$f(\tau) = \frac{M_\Delta^{t\ ta}}{min(|\tau - t|, |\tau - t'|)} \tag{6.37}$$

where M_τ^Δ is the mean delta between t' and t over the whole dataset (acting as a normalizer).

To sum it up, S^s penalizes the events including entity s which did not occur in the dataset but could be predicted by RelNet: the higher the rank of the negative event, the higher the penalty in the loss. However, we mitigate this by penalizing less the negative samples that are far from events actually occurring in the dataset (the farthest being $\frac{t'-t}{2}$), so that, for instance, if the dataset is likely to have missing events for some long periods of time, the approach is not penalized too much.

For a batch, the negative log likelihood loss with survivance is thus defined by:

$$L = -\sum_{i=1}^{N} \left(log(\lambda_{r_i}^{s_i,o_i}(t'|t)) - S^{s_i} - S^{o_i} \right) \tag{6.38}$$

6.3.5 GRU

As for AttrNet, we replace the RNN version of the encoder with the GRU version. Given q, $\boldsymbol{\rho}_s^-$, and $\boldsymbol{\rho}_o^-$, the new encoder is defined as follows for updating $\boldsymbol{\rho}_e$ on input (e, r, o, t):

$$\boldsymbol{v}_s = tanh(\mathbf{W}_s^h.[\boldsymbol{\rho}_o^- \oplus \boldsymbol{w}_r]) \tag{6.39}$$

$$\boldsymbol{z}_t^s = \sigma((t - \tau_s^-)\boldsymbol{w}_z^s + \mathbf{W}_z^v.\boldsymbol{v}_s + \mathbf{W}_z^{hh}.\boldsymbol{\rho}_s^- + \boldsymbol{b}_z) \tag{6.40}$$

$$\boldsymbol{r}_t^s = \sigma((t - \tau_s^-)\boldsymbol{w}_r^s + \mathbf{W}_r^v.\boldsymbol{v}_s + \mathbf{W}_r^{hh}.\boldsymbol{\rho}_s^-) \tag{6.41}$$

$$\boldsymbol{h}_t^s = tanh(\mathbf{W}_t^v.\boldsymbol{v}_s + \boldsymbol{r}_t^s \odot \mathbf{W}_t.\boldsymbol{\rho}_s^-) \tag{6.42}$$

$$\boldsymbol{\rho}_s^t = \boldsymbol{z}_t^s \odot \boldsymbol{\rho}_s^- + (1 - \boldsymbol{z}_t^s) \odot \boldsymbol{h}_t^s \tag{6.43}$$

- $\oplus$ denotes concatenation of one-dimensional vectors, and $\odot$ the Hadamard product (pointwise matrix multiplication),
- $\boldsymbol{v}_s$ is the input value representing the event q, $\boldsymbol{z}_t^s$ is the update gate, $\boldsymbol{r}_t^s$ is the reset gate, and $\boldsymbol{h}_t^s$ is the current memory,

- vectors $\boldsymbol{w}_z^s$, $\boldsymbol{w}_r^s \in \mathbb{R}^n$ weigh the temporal drift between two events, and $\boldsymbol{w}_r \in \mathbb{R}^d$ is a parameter specific to the relation type r,
- matrix $\mathbf{W}_s^h \in \mathbb{R}^{n\times(n+d)}$ captures the current relation in the history of the entities,
- vector $\boldsymbol{b}_z \in \mathbb{R}^n$ is the bias vector of the update gate,
- $\mathbf{W}_z^{hh}, \mathbf{W}_r^{hh}, \mathbf{W}_t^{hh} \in \mathbb{R}^{n\times n}$ are parameters governing which part of the embedding should be kept in the gate,
- $\mathbf{W}_z^v, \mathbf{W}_r^v, \mathbf{W}_t^v \in \mathbb{R}^{n\times n}$ monitor the input value.

The embedding of o is updated similarly, swapping s and o in Equations 6.39–6.43. Equation 6.39 encodes the input event, but we excluded the subject on purpose because it is already included in the memory of the network.

6.3.6 Temporal point process

[Mei and Eisner, 2017] proposes a Neural Hawkes Process based on Hawkes process. With this one, a happened event can excite the probability for other events, which suits our usecase better. For instance, a vessel entering a port should increase the probability for the exit event. In [Mei and Eisner, 2017], the intensity function of the Hawkes process is defined by:

$$\lambda_r(t) = \mu_r + \sum_{q:t_q<t} \alpha_{r_q,r} exp(-\delta_{r_q,r}(t-t_q)) \tag{6.44}$$

"where $\mu_r \geq 0$ *is the base intensity of event type r,* $\alpha_{j,r} \geq 0$ *is the degree to which an event of type j initially excites type r, and* $\delta_{j,r} \geq 0$ *is the decay rate of that excitation. When an event occurs, all intensities are elevated to various degrees, but then will decay toward their base rates* μ*".*

However, though more adapted to the problem at hand, this process will be left for future work since it is not trivial to add and implement, in particular with the parameters to tune. Still, we make some changes to the intensity function in Equation 6.45:

$$\lambda_r^{s,o}(t'|t) = w_r^t(t-t')(\log(1+\exp(g_r^{s,o})) \tag{6.45}$$

We replace the exponential by a softplus function as suggested in [Mei and Eisner, 2017]. This approximates the ReLU function $f(x) = max(0,x)$ while being strictly greater than zero. We also add a parameter $w_r^t \in \mathrm{R}^1$ on the temporal drift to use it differently depending on the type of relation r, since different relations may have different temporal processes. This is done to consider that, for instance, the dynamic of a vessel entering a port is very different from a vessel exchanging goods or engaged in fishing activities.

6.3.7 Another loss

We devised another possible loss function for Joint Evolution. Our objective is to have the maximum conditional intensity for the challenge event compared to all possible events. To do so, we compute the conditional intensity for all possible negative event (as we do for the survivance) and keep the maximum value. We then subtract the conditional intensity of the ground truth event (the challenge) and try to minimize the difference between the two, i.e., to make the conditional intensity of the challenge higher than the conditional intensity of the negative events (Equation 6.46):

$$\mathrm{L} = (max(\lambda^{s}) - \lambda^{s'}) + (max(\lambda^{o}) - \lambda^{o'}) \tag{6.46}$$

with $s \in \mathrm{N}_s$ where N_s is the negative sampling for subject and $o \in \mathrm{N}_s$ where N_o is the negative sampling for object. We can choose to bound this loss positively.

6.4 Streaming and unseen nodes

Maritime Situational Awareness requires constant surveillance and a continuous stream of information. As new AIS and events are coming, they must be included in the entity representations, even after the model is trained. Thanks to the encoder, upcoming events can be added to update the embeddings as they arrive. The decoder is only used when a prediction needs to be made.

Joint Evolution is also able to deal with unseen nodes. If an entity $e \notin \mathrm{E}$ is encountered, it is added to $|\mathrm{E}|$ with an embedding initialized with zeroes. Prediction will not be accurate at first because the model has no prior information on the entity (cold start), but it gets better when it is encountered multiple times in an event.

6.5 Wrapping up: Joint Evolution

6.5.1 The framework

Joint Evolution is composed of two specialized neural networks, *AttrEmb* and *RelEmb*, which respectively deal with DKG^{A} and DKG^{R}. We assume that attribute values for an entity e can be predicted from the attributional history of e only, so that *AttrEmb* receives input only from DKG^{A} (that is, only the observed changes of attribute values for e are fed into *AttrEmb*).

On the other hand, one specificity of our approach is to use values of attributes for predicting relations, so that *RelEmb* receives both input from DKG^{R} (the observed relational events involving the entities at hand) and the attributional features generated by *AttrEmb* to make its predictions. The global architecture is illustrated on Figure 6.7.

The input of the model consists of events $q^{\mathrm{A}} \in DKG^{\mathrm{A}}$ and $q^{\mathrm{R}} \in DKG^{\mathrm{R}}$, arriving in a timely manner. Each network is composed of an *encoder* and a *decoder*. The encoder updates an *embedding* for each entity involved in the processed event, so as to capture the behavior of these entities given the past events involving them. When a prediction is requested, these embeddings are fed into the decoder of *AttrEmb* (resp. *RelEmb*), which uses this information to predict the attribute value (resp. the subject, relation, or object).

6.5.2 AttrNet

The network *AttrEmb* encodes DKG^{A} to make predictions on the future values of attributes.

6.5.2.1 Encoder

When a new attributional event $q = (e, a, v, t)$ is observed, the encoder of *AttrEmb* takes as input both q and the current attributional embedding $\boldsymbol{\alpha}_e^-$ of e, and produces the new attributional embedding $\boldsymbol{\alpha}_e^t$ of e. To keep track of the chronological order of the changes of

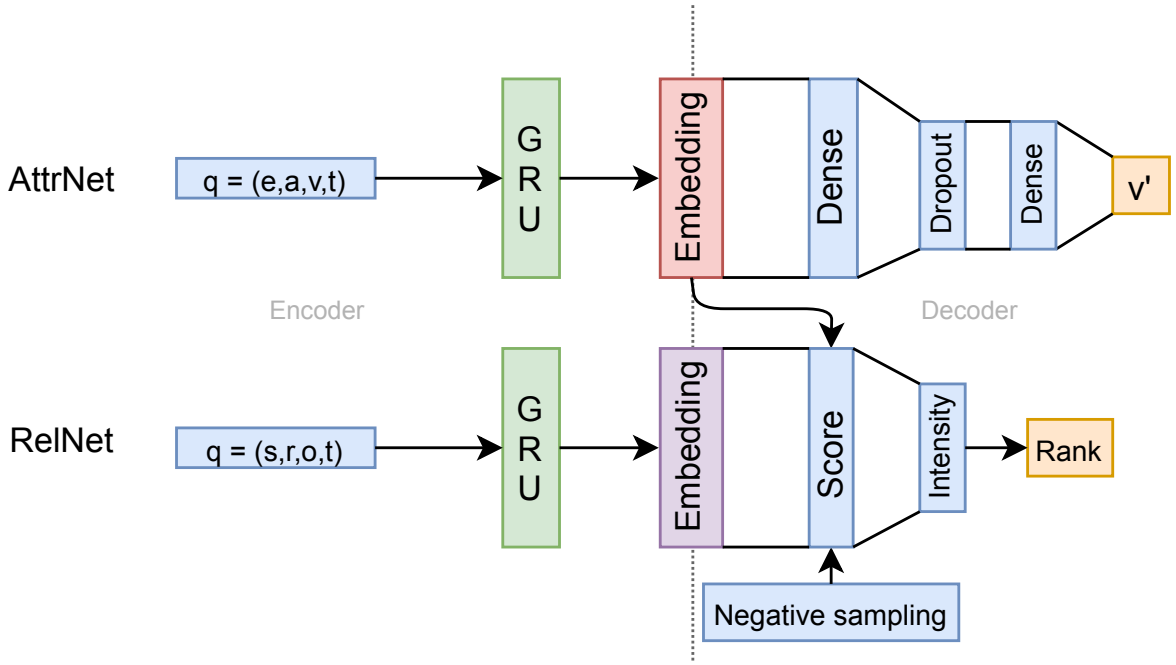


Figure 6.7: Joint Evolution global architecture

attribute values, we use a GRU neural network as this encoder [Cho et al., 2014b]. Given $q = (e, a, v, t)$ and $\boldsymbol{\alpha}_e^{t-}$, the encoder is defined as follows:

$$\boldsymbol{z}_t = \sigma((t-\tau^-)\boldsymbol{w}_t^z + v\boldsymbol{w}_z^a + \mathbf{U}_z\boldsymbol{\alpha}_e^{t-} + \boldsymbol{b}_z) \tag{6.47}$$

$$\boldsymbol{r}_t = \sigma((t-\tau^-)\boldsymbol{w}_t^r + v\boldsymbol{w}_r^a + \mathbf{U}_r\boldsymbol{\alpha}_e^{t-}) \tag{6.48}$$

$$\boldsymbol{h}'_t = tanh(v\boldsymbol{w}_t^a + \boldsymbol{r}_t \odot \mathbf{U}_t\boldsymbol{\alpha}_e^{t-}) \tag{6.49}$$

$$\boldsymbol{\alpha}_e^t = \boldsymbol{z}_t \odot \boldsymbol{\alpha}_e^- + (\mathbf{1} - \boldsymbol{z}_t) \odot \boldsymbol{h}'_t) \tag{6.50}$$

In words, the new embedding for e is obtained as a function of the temporal drift (with a parameter w_t which is shared by all attributes), of the new observed value (with a parameter specific to the attribute type), and of the previous embedding (with a parameter specific to the attribute type). Figure 6.8 illustrates Equations 6.47 to 6.50.

Here $\boldsymbol{z}_t$ is called the *update gate*, $\boldsymbol{r}_t$ the *reset gate*, $\boldsymbol{h}'_t$ the *current memory*, and σ is the sigmoid function; $v \in \mathrm{D}_a$ is the scalar value of the attribute a, $\odot$ denotes the Hadamard product, and σ and $\boldsymbol{z}_t \mapsto \mathbf{1} - \boldsymbol{z}_t$ are applied coordinatewise.

Vectors $\boldsymbol{w}_z^a, \boldsymbol{w}_r^a, \boldsymbol{w}_t^a \in \mathbb{R}^m$ are attribute-specific parameters of the network. Matrices $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_t \in \mathbb{R}^{m \times m}$ are parameters governing which part of the embedding should be kept in the gate. Finally, $\boldsymbol{b}_z \in \mathbb{R}^m$ is the bias layer of the update gate.

6.5.2.2 Decoder

When at timestep t a prediction is required for the value of attribute a for entity e at some timestep $t' \geq t$, the decoder is given as input the attributional embedding $\boldsymbol{\alpha}_e^t$ of entity e, and is required to predict the most likely value v such that (e, a, v, t') is the real quadruple. We use a fully connected layer to predict the value v as follows:

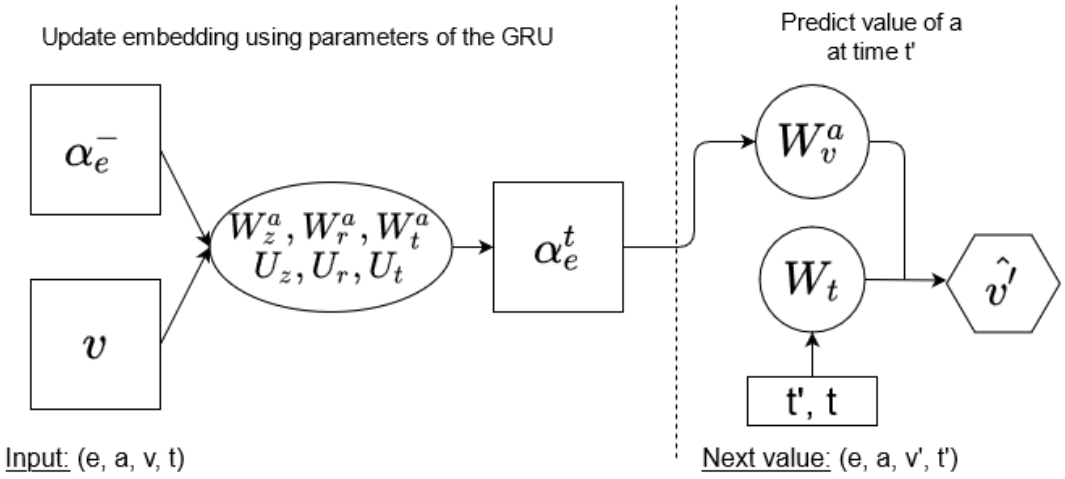


Figure 6.8: Inference with AttrEmb

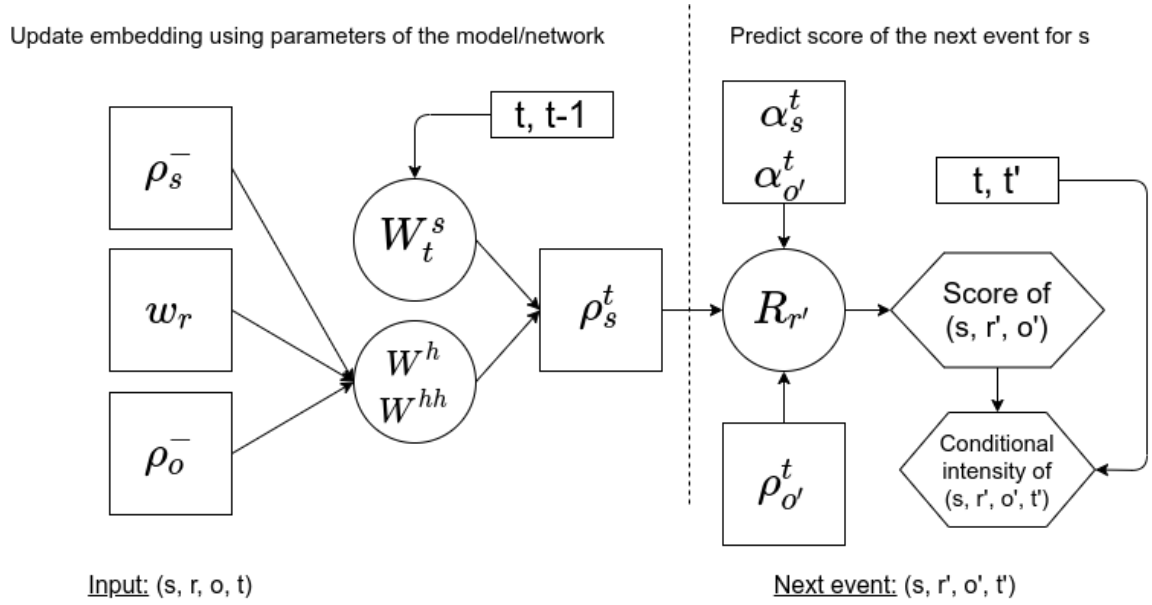


Figure 6.9: Inference with RelEmb for s. The same process is performed for o following Equations 6.57, 6.58 and 6.59.

$$hidden(\boldsymbol{\alpha}_e^t) = tanh(\boldsymbol{w}_x \boldsymbol{\alpha}_e^t + b_x)) \tag{6.51}$$

$$v = tanh((t' - t)\, w_t + \boldsymbol{w}_a \cdot hidden(\boldsymbol{\alpha}_e^t) + b_a) \tag{6.52}$$

Figure 6.8 (right) illustrates this step.

Here, $w_t \in \mathbb{R}$ is a parameter that modulates the prediction according to the time elapsed since the last change of value, $\boldsymbol{w}_a \in \mathbb{R}^m$ is an attribute-specific parameter, and $b_a \in \mathbb{R}$ is the bias.

6.5.3 RelNet

6.5.3.1 Encoder

When a new relational event $q = (s, r, o, t)$ is observed, the encoder of *RelEmb* takes as input q and the current relational embeddings $\boldsymbol{\rho}_s^{t-}$ and $\boldsymbol{\rho}_o^{t-}$ of s, o, and produces the new relational embeddings $\boldsymbol{\rho}_s^t$ and $\boldsymbol{\rho}_o^t$ of s, o. Given the event q and the current embeddings $\boldsymbol{\rho}_s^{t-}, \boldsymbol{\rho}_o^{t-}$, we update the embeddings as follows:

$$\boldsymbol{v}_s = tanh(\mathbf{W}_s^h.[\boldsymbol{\rho}_o^{t-} \oplus \boldsymbol{w}_r]) \tag{6.53}$$

$$\boldsymbol{z}_t^s = \sigma((t - \tau_s^-)\boldsymbol{w}_z^s + \mathbf{W}_z^v.\boldsymbol{v}_s + \mathbf{W}_z^{hh}.\boldsymbol{\rho}_s^- + \boldsymbol{b}_z) \tag{6.54}$$

$$\boldsymbol{r}_t^s = \sigma((t - \tau_s^-)\boldsymbol{w}_r^s + \mathbf{W}_r^v.\boldsymbol{v}_s + \mathbf{W}_r^{hh}.\boldsymbol{\rho}_s^{t-}) \tag{6.55}$$

$$\boldsymbol{h}_t^s = tanh(\mathbf{W}_t^v.\boldsymbol{v}_s + \boldsymbol{r}_t^s \odot \mathbf{W}_t.\boldsymbol{\rho}_s^{t-}) \tag{6.56}$$

$$\boldsymbol{\rho}_s^t = \boldsymbol{z}_t^s \odot \boldsymbol{\rho}_s^- + (1 - \boldsymbol{z}_t^s) \odot \boldsymbol{h}_t^s \tag{6.57}$$

6.5.3.2 Decoder

When at timestep t a relational prediction is required for some timestep $t' \geq t$, recall that we assume that two out of s, r, o are given, and the prediction concerns the third component. Whatever the missing component, we explore all candidates and rank them, so that prediction amounts to computing a score for a number of quadruples (s', r', o', t').

We compute the bilinear relational score $g_r^{s',o'}$ (Equation 6.58) from the updated embeddings for a challenge event $q' = (s', r', o', t')$ that involves one of the two entities from q.

$$g_{r'}^{s',o'} = (\boldsymbol{\rho}_{s'}^t \oplus \boldsymbol{\alpha}_{s'}^t)^{\mathrm{T}}.\,\mathbf{R}_{r'}\,.\,(\boldsymbol{\rho}_{o'}^t \oplus \boldsymbol{\alpha}_{o'}^t) \tag{6.58}$$

where $\mathbf{R}_{r'} \in \mathbb{R}^{(n+m)\times(n+m)}$ is a relation specific matrix learnt during training. This score is then used to compute a conditional intensity function λ (Equation 6.59), that will be used to rank the possible events during the prediction step (the higher λ, the more likely the quadruple (s, r, o, t')):

$$\lambda_r^{s,o}(t'|t) = w_t^r(t - t')(\log(1 + \exp(g_r^{s,o})) \tag{6.59}$$

This is an adaptation of the modified softplus function found in [Trivedi et al., 2019] and [Mei and Eisner, 2017]. Using the representation of the current event, *Joint Evolution* will thus learn which event to predict next. The delta of time $(t - t')$ illustrates the Rayleigh process used in Know-Evolve: the longer an event has not happened, the more probable its occurrence is.

6.5.4 Usage

6.5.4.1 Update

During training, the networks learn how to build embeddings that can be used to make predictions. Once trained, all parameters are frozen but the embeddings are continuously updated as new events happen. By using the embeddings of the members of the current event (time t) in the encoder, the new knowledge is directly included in $\boldsymbol{\rho}_e$ and $\boldsymbol{\alpha}_e$, so that the model has all available data to perform a prediction at a time $t' > t$. It can thus be fed with a stream of data, update the embeddings continuously, and at any time be ready to predict next events.

6.5.4.2 Prediction

The prediction of the next value for an attribute is done directly at the output of *AttrEmb*: this is a scalar value computed using $\boldsymbol{\alpha}_e$ (Equation 6.52). To predict the next relational event for an entity, we set the time and two of the three other components, e.g., $(s, r, ?, t)$, and we look for the most probable object for relation r with subject s and time t. We thus replace the object by each possible entity, compute their bilinear score and conditional intensity with Equations 6.58 and 6.59, and rank them. The most probable events are those with the highest conditional intensities.

6.5.5 Losses

The loss used for AttrNet is a Mean Squared Error, a common loss for regression:

$$\mathrm{L}oss = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (v(t_{p+1})_i - v(t_{p+1})_i^*)^2 \tag{6.60}$$

As for RelNet, we kept the second presented loss:

$$\mathrm{L} = (max(\lambda^s) - \lambda^{s'}) + (max(\lambda^o) - \lambda^{o'}) \tag{6.61}$$

6.6 Conclusion

We presented Joint Evolution, a double recurrent network encoder-decoder system for Dynamic Attributed Knowledge Graphs. During its construction, we made the choices that are the more suited to MSA, namely, a representation for each entity, a decoder that can get the likeliness of an event on-the-fly and a continuous update of the representations as new data comes through the encoder. With its two recurrent networks, Joint Evolution can deal with both relations and attributes. We are thus able to test our hypothesis on the impact of the attributes on link prediction. In Part III, we present the experiments run to evaluate Joint Evolution.

Part III

Experiments

CHAPTER 7

Datasets

Our work focuses on Maritime Situational Awareness and as such, we test JointEvolution on a maritime dataset. This experiment is a proof of concept on real data, which aims to demonstrate the effectiveness of involving dynamic attributes in the task of link prediction. However, to test JointEvolution in a more academic context and make a better comparison with other state-of-the-art techniques, we try JointEvolution against DBLP, a citation network frequently used in graph analysis.

7.1 datAcron

The first dataset we use contains maritime data, the objective being to predict future interactions between vessels. As a basis, we use the dataset given by [Ray et al., 2019] and part of the datAcron project, which contains ship information collected through AIS. The dataset extends over six months, from October 1st, 2015 to March 31st, 2016, and covers the Celtic sea, the North Atlantic Ocean, the English Channel and the Bay of Biscay (France). Along with this dataset, we use the Composite Maritime Events dataset [Pitsikalis and Artikis, 2019], extracted from the AIS of [Ray et al., 2019] using a rule-based engine.

7.1.1 Dataset breakdown

Within all the available data in the dataset, we are particularly interested in the AIS positions and the composite events. There are 3,970,370 events, but only 4,405 events involving two entities: *:pilotBoarding* (730), *:rendezVous* (3,548) and *:tugging* (127). The others are self-centered events, e.g. when a vessel makes a sudden change of speed or when it stops. The pilot boarding event occurs when a vessel enters a port: it is not its pilot that makes the maneuver but a pilot from the destination port that boards the vessel. A rendez-vous between two vessels is a meeting, usually for transshipping, and tugging is when a tug boat is towing another vessel.

The dataset contains 19,035,630 AIS messages which refer to 5.055 vessels. The vessels are the only type of entity under study, and they are identified by their MMSI. From this identifier, it is straightforward to map the vessels to their attributes using the AIS values. We only keep the latitude, the longitude, the speed and the timestamp. The course is left out

because it is in degrees, and values such as 2 and 350 may be considered to be far apart from each other by the network while there are in fact very close. An option is to use the cosine and the sine instead, but this means that the course has to be learned from two dependent values. It would make the task more difficult, and this problem already arises with position (but position is mandatory to get the movements of the vessels).

In these 4,405 events, there are only 363 unique MMSI and the subjects and objects are two separate sets: there are 174 unique subjects and 189 unique objects (and 174 + 189 = 363). Since we are interested in the link prediction task, we have no use for vessels that are only involved in attributive events. We can thus only use these 363 MMSI and discard the rest.

A major issue is that we do not have any AIS for object vessels. The AIS of the 174 subject vessels include 10,735,097 observations (one observation contains the four attributes), i.e. 56,39% of all the observations (from 5,055 vessels). We hence have a lot of information for these subjects, but only a little for the objects. Once we exclude the course over ground, we have 32,205,291 attributional events.

7.1.2 About our datAcron version

For scalability reasons, and because it is the only part manually verified by experts, we only use the first month of the dataset. Among all detected events, we only consider those which include two different entities (*pilotBoarding*, *rendezVous*, *tugging*). In terms of dynamic attributes, we consider the *latitude*, the *longitude*, and the *speed* of the vessels. Some statistics about this projection of the dataset are reported in Table 7.1.

The relation/attribute ratio of datAcron is very unbalanced with more than 4M attributives event for only 764 relational ones. Though it might seem problematic, it is an opportunity to test AttrNet capability to improve an entity representation and the subsequent relation prediction. DBLP (Section 7.2) allows us to evaluate Joint Evolution on a different setting.

7.2 DBLP

To test JointEvolution, we use the Digital Bibliography & Library Project (DBLP) to learn and predict citation and co-autorship between two authors. Specifically, we use the DBLP v12 dataset from AMiner [Tang et al., 2008]. It contains 4,894,081 papers and 45,564,149 citation relationships as of 2020-04-09. The detailed content of the dataset can be found on the website[1].

Our objective is to predict new relations between authors: citation or co-authorship relations. To do so, we need to preprocess the dataset since these links are not explicitely formatted. Since the file is very large (12GB), we filter out the irrelevant information to save some space. First, we only keep the field of study (FoS) of the article, its id, its authors, its title, its year of publication and its references. Then, we extract several dictionaries from the file so that we can construct the graph:

- **Author information** id_author: [name, org]
- **Organization** name_org: id_org
- **Article information** id_article: [title, year, author_list]

[1] https://www.aminer.cn/citation

- **Co-authorship** id_author: [(coauthor_id, year), (coauthor_id, year),...]
- **Write article** id_author: [(id_article, year), (id_article, year),...]
- **Cite article** id_author: [(id_ref, year), (id_ref, year),...]
- **Cite author** id_author: [(id_author, year), (id_author, year),...]

7.2.1 First version

In our first version of the DBLP dataset, we take the subset between 1995 and 2011 restricted to authors with at least three publications.

7.2.1.1 Relations

We consider three types of relations: (author, :cite, author, t) that happens when an author cites the article of another author ; (author, :citedBy, author, t) when an author is cited by another author ; and (author, :coauthor, author, t) when two authors write the same paper.

7.2.1.2 Attributes

We consider three kinds of attribute events. The first one is the number of citations of an author, (author, : $nbCitations$, x). For each year where the author is cited at least once, we compute the total number of citations at time t. Note that this number of citations is the number of *articles* that cite the author, not the number of *authors* unlike the : $citedBy$ relational event. The second one is the number of publications of the author, (author, : $nbPubli$, x, t). It is incremented each time the author writes a new paper. Finally we have the h_index of the author, (author, : $h-index$, x, t), which is computed each time the author is cited in an article or has published a new paper. These attributes are chosen because they reflect the authors' characteristics regarding publication.

7.2.1.3 Overview and discussion

With the selected relations and attributes in the selected timeframe, we obtain the following numbers of edges:

- 1.445.901 articles
- 412.485 authors
- 59.598.155 citation links (x2 if we add the reversed relations :citedBy)
- 5.353.468 coauthor links
- 1.885.578 citation number changes
- 1.510.173 number of publications and h-index changes

This first version has three relations and attribute types with millions of edges, which is what we are looking for despite the lack of balance between the relation types. However, the size of the dataset makes it too heavy to preprocess and generate its precedence graph. The same constraint applies to training time with JointEvolution, which is extensively long with

so many relational quadruplets. Moreover, we came to the conclusion that these attributes can be derived from the relational dataset and do not add valuable information to the learning. Moreover, if the publication behavior of an author helps to characterize the entity, it is not straightforwardly clear that it can help to predict a citation or co-authorship link between two authors. To consider the scalability and attribute usefulness issues better, we designed a second version of the dataset.

7.2.2 Second version

Since the first dataset is too large, we extract authors and articles only between 1995 and 2000 in the field of Artificial Intelligence, and only authors that have at least 5 publications. We made this choice based on the rapidly evolving nature of this field, and we did not take more recent years into account due to the too large number of publications and authors that would throw us back to our previous issue. In this setup, we consider two relation types, :cite and :coauthor. We discard the :citedBy relation type, because it is the symmetric relation of :cite and doubles the number of edges for :cite. Concerning the attributes, we completely change the previous selection: since scalar values that describe the publication rate and impact are not that useful to predict citation and co-authorship relations, we use the articles themselves as attributes.

Indeed, an article contains useful information on the field of study and the specific notions an author is working on. Of course, the field of study is already known, but it is not a discriminant factor here since we restricted ourselves to Artificial Intelligence. Thus, we link articles to authors as attribute values using two attribute types, :write and :cite. The first one provides information on the current work of the author, and the second one on the related works.

The problem is that an article is not represented by a value as it is. There exist several ways to get a representation for an article: [Liu et al., 2019, Meng et al., 2020, Meng et al., 2019] and [Li et al., 2018] use the keyterms extracted from the titles or abstracts as attributes (bag-of-words), and [Meng et al., 2020] also uses the conferences.

Since the abstracts of the articles were not always available, we used the title of the articles. We try to go further than keyterms by projecting the whole title into 512-dimensional embeddings created with Google's Universal Sentence Encoder [Cer et al., 2018]. Each time an author cites or write an article, an associated attributional event is created with this embedding as a value. Thus, the attributional embedding of an author will reflect his/her field of publication by aggregating the titles.

7.2.3 About the dataset

The final version of the dataset has 6,147 authors, 37,097 articles, 351,365 relations divided between :coauthor (40,786) and :cite (310,579), and 139,152 attributes divided between :write (43,648) and :cite (95,504). Regarding the link prediction problem on dynamic attributed graphs, DBLP is a reduction of it since the time scale is year by year, which is equivalent to a sequence of static graphs of reasonable length (5 slices). With a maritime dataset that contains AIS data, the timescale is the second and it would be much harder to use such a sequence, hence the streaming scenario. But Joint Evolution can still be applied since it can work with any timescale.

7.2.4 Adaptation of Joint Evolution

AttrNet takes scalar values as input to update the attributive embeddings of entities, but DBLP provides embeddings. To adapt AttrNet, we cast this embedding to the size of the embeddings of the entities, i.e. m. The encoder thus becomes:

$$\boldsymbol{v'} = \mathbf{W}^{cast}\boldsymbol{v} \tag{7.1}$$

$$\boldsymbol{z}_t = \sigma((t-\tau^-)\boldsymbol{w}_t^z + \mathbf{W}_z^a\boldsymbol{v'} + \mathbf{U}_z\boldsymbol{\alpha}_e^{t-} + \boldsymbol{b}_z) \tag{7.2}$$

$$\boldsymbol{r}_t = \sigma((t-\tau^-)\boldsymbol{w}_t^r + \mathbf{W}_r^a\boldsymbol{v'} + \mathbf{U}_r\boldsymbol{\alpha}_e^{t-}) \tag{7.3}$$

$$\boldsymbol{h}_t' = tanh(v\boldsymbol{w}_t^a + \boldsymbol{r}_t \odot \mathbf{U}_t\boldsymbol{\alpha}_e^{t-}) \tag{7.4}$$

$$\boldsymbol{\alpha}_e^t = \boldsymbol{z}_t \odot \boldsymbol{\alpha}_e^- + (\mathbf{1}-\boldsymbol{z}_t) \odot \boldsymbol{h}_t') \tag{7.5}$$

with $\boldsymbol{v} \in \mathbb{R}^{512}$, $\boldsymbol{v'} \in \mathbb{R}^m$, $\mathbf{W}^{cast} \in \mathbb{R}^{m\times 512}$ and $\mathbf{W}_z^a, \mathbf{W}_r^a \in \mathbb{R}^{m\times m}$.

Thus, each embedding of a title is included in the attributive embedding of the entity. However, a new issue arise: we formerly predicted the next attribute value to train AttrNet, but what are we going to predict now? Predicting the embedding or title of the next article an author will write or cite makes no sense and these attributes are only here to characterize an author field and help the relational prediction. So instead of decoding $\boldsymbol{\alpha_e}$ to train AttrNet, we only use RelNet to train AttrNet encoder. When a link prediction is made and the loss is computed by RelNet, it backpropagates to the RelNet encoder and decoder, but also to AttrNet encoder. To sum it up, we train AttrNet encoder directly on the RelNet decoder that uses $\boldsymbol{\alpha_e}$ to predict new links.

Dataset	Entities	Rel types	Attr types	Nb Rel	Nb Attr
datAcron	363	3	3	764	4,596,204
DBLP	6,147	2	2	261,324	139,152

Table 7.1: Statistics for DBLP and datAcron

7.3 Conclusion

We test Joint Evolution on two datasets, datAcron and DBLP. datAcron has less entities and relations but has a lot of fast evolving attributes. It is useful to test Joint Evolution dynamic capacity as well as the effect of attributes on link prediction, especially when there a only a few relations to learn. Our hypothesis is that the large number of attributes will compensate the lack of relations. On the other side, DBLP has less attributes than datAcron but the dataset is more balanced. However, its dynamicity is at the scale of a year where datAcron can work at the scale of a second. DBLP is thus closer to a sliced *DKG* than to a streamed *DKG*.

CHAPTER 8

Experimental settings

8.1 Dataset settings

Preparing the data for the experiments requires several preparatory steps. The major ones are described in Section 5.3: assigning the challenges, generating the precedence graph and extracting the batches. In addition to this, we normalize all values of datAcron attributes to $[-1, 1]$, which is the range of the *tanh* function we use in AttrNet. The preprocessing of the datasets is not a trivial task and requires several hours of computing because of the complexity of relational data. To train Joint Evolution, we use 90% of the dataset as the training set and the remaining 10% as the validation set. Since we build batches of independent sequences, the training set is always the same (namely, the 90% earliest events) since we cannot perform a cross-validation: this would shuffle the events and negate the benefit of the precedence graph.

8.2 Learning task

We train *Joint Evolution* to predict the new entity in the challenge, i.e. the one that is not in the input (as described in Section 5.1). By doing so, the model will be able to answer questions such as $(s, r, ?, t)$. However, datAcron does not contains any AIS message for the object entities, and there is no intersection between the sets of subject (174) and object entities (189). The model thus runs essentially without $\boldsymbol{\alpha}_o^t$.

We test both datAcron and DBLP with and without attributional embeddings so as to determine whether the addition of $\boldsymbol{\alpha}_e^t$ improves the prediction results. The tests that use only *RelEmb* are thus our baseline to evaluate the addition of attributes. Note that during validation, embeddings are still updated with incoming events. When training with relations only, *AttrEmb* is not used and all $\boldsymbol{\alpha}_e$ stay at zero, thus negating any parameter associated to them during the computation of the scores.

Since DBLP uses embeddings and not scalars as attributional values, there is no task on *AttrNet* for prediction. Thus, we learn *AttrNet* directly from the link prediction task of *RelNet*, i.e. we only train and use the encoder of *AttrNet* by feeding *RelNet*.

8.3 Model comparison

To compare our method, we use two models implemented with the StellarGraph library [Data61, 2018]. It offers state-of-the-art methods for graph machine learning and supports many kinds of graphs: homogeneous, heterogeneous, graphs with or without data associated to nodes (the attributes), and graphs with edge weights.

8.3.1 node2vec

node2vec [Grover and Leskovec, 2016] is an embedding method on static, homogeneous graphs that we already presented in Section 4.2.2.3. We test against node2vec for two reasons: it is a largely known and used model, and it allows to evaluate the gain of performance provided by dynamic models over static models on dynamic relational data.

8.3.2 CTDNE

CTDNE [Nguyen et al., 2018] is an embedding method on dynamic, homogeneous graphs using temporal random walks without slicing the graph. We presented this model in Section 4.4.3. It allows us to test Joint Evolution against a dynamic method. The embeddings are generated through temporal random nodes and can be used for downstream machine learning tasks.

8.3.3 Adaptations

Both node2vec and CTDNE learn embeddings that cannot be easily updated: they are created by a selection of random walks, and incorporating new data implies to perform the walks selection again. It means that with these two techniques, the embeddings are learned once and for all. On a long-term use, this may requires to generate new embeddings to update the behavior of the entities.

As these methods cannot update the embeddings in a streaming fashion, we create the embeddings using 75% of the datasets. The next 15% are then used to train link prediction using the RelNet decoder and the last 10% to test the learning. Moreover, they both deal only with homogeneous data when learning the embeddings. However, the decoder of RelNet takes the relation type into account when using the embeddings, which allows a fairer comparison with our embedding method since node2vec and CTDNE initially cannot. Also, we removed all timestamps when using node2vec and did not use deltas of time in the decoder since it is static.

8.4 Measures of performance

To evaluate the performance of Joint Evolution, we watch the score of the ground truth event compared to other events.

8.4.1 Score and rank

The output of Joint Evolution provides a score for a given event, that allows to measure the capability of the model to predict correctly. The score predicted for the ground truth is compared to the score predicted for negative (generated) events. This way, we can evaluate

how the model fares to give the best possible score to the ground truth. By sorting the event scores by descending order, we obtain a ranking for the ground truth. Our objective is thus to optimize the rank of the true event to get better prediction during inference.

8.4.2 MAR and Hits@X

Two measures are commonly used for link prediction: Mean Average Rank (MAR) and Hits@X. The first one is the mean rank of the ground truth value for all events in the set. It tells how good the model is on average, but what defines a good value depends of the number of candidate entities. The Hits@X measures how many ground truths are ranked in the top X scores. For instance, Hits@10 is the proportion of true events that are in the top 10 in terms of score. It should be noted that the better rank is 1 and that Hits@X is in $[0, 1]$ (hence the higher the better: @Hits10 = 1 means that all samples are classified in the top ten, and none if Hits@10 = 0). We thus look for low values of MAR and high values of Hits@X.

MAR is representative of how the model fares in general, but Hits@X is more about the proportion of useful predictions: in an operational use such as MSA, it is better to have a good Hits@10 rather than a good MAR - though the two are linked. This is because it is easier to verify the 10 best predictions manually by browsing the suggested vessels than to verify 50 or 100 predictions.

In these experiments, we report Hits@1, Hits@10, Hits@30, Hits@100 and Mean Average Rank (MAR) for object prediction. Given a quadruplet $(s, r, ?, t)$, we compute the score of all possible quadruplets by replacing the object by all possible entities. We then rank the quadruplet by score and report the mean rank for the ground truth quadruplet as the MAR score (hence the lower the better).

8.5 Software and hardware

Joint Evolution was implemented from scratch using the PyTorch[1] library. It is an open source machine learning framework whose two main features are tensor computing with strong acceleration via graphics processing units (GPU) and deep neural networks built on a type-based automatic differentiation system.

To preprocess the *DKG*, we use OrientDB. It is a graph and document-oriented database made to manage relational data. It allows easy storage and manipulation of entities and edges, and can create visualizations of the graph such as Figure 5.4 to navigate through it. For the precedence graph, we used networkX, a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

As for hardware, preprocessing of the data and trainings are done on a server at the GREYC laboratory. The machine is built with two Intel Xeon E5-2620 v4 at 2.10GHz that totalizes 32 cores, 512GB RAM, 4 GeForce GTX 1080 Ti as GPUs with 11GB of memory each to contain the models. A single GPU is sufficient to run the experiments, but we can run several experiments at the same time with more GPUs.

8.6 Parameters

We present here the parameters used for training and their effect:

[1] `https://pytorch.org/`

- *Batch size*: The size of the batches of input data. We set it to 100 and did not make many changes because the sequence of batches is precomputed from the precedence graph. Unlike many machine learning input pipelines, we cannot change the size of batches on-the-fly.
- *Training set*: The percentage of the dataset used for training. As usually done, we use 90% for training and 10% for validation. It should be noted that since we work with specifically ordered sequence (thus with no cross-validation), the test and validation sets are the same set.
- *Epochs*: the number of times the full dataset is processed. The embeddings are set to full zeroes at initialization and between two epochs: we train the weights, not the embeddings, and the history must be reset at each new start.
- *Embedding size*: the size of $\boldsymbol{\alpha}_e$ and $\boldsymbol{\rho}_e$.
- *Learning rates*: the learning rates for AttrNet and RelNet.
- *Schedulers*: a scheduler decreases the learning rate every X epochs and has two parameters: the number of epochs X and the multiplier for the learning rate ($\gamma \in [0, 1]$).
- *Weight decay*: how much the weights decay between epochs. Decaying prevents the network from being stuck in a local optimum, but can also negate the learning if too high (which happens in particular with a decaying learning rate).
- *Dropout*: the percentage of neurons deactivated in the dense layer of the AttrNet decoder. This forces the network to use different paths and not to always reinforce the same neurons.
- *Gradient clipping*: sometimes, the gradient can explode during training. To avoid this and not end up with infinite or NaN (Not a Number) values, we clip the gradient to a maximum value to keep it stable.

We did not perform a grid search to find the best set of parameters for two reasons: there are too many of them, and the computing time for one run is high. In Chapter 9, we empirically test several combinations of these parameters.

8.7 Conclusion

We train Joint Evolution to predict the missing entity in a quadruplet $(s, r, ?, t)$ or $(?, r, o, t)$ and compare the results with the two other models, node2vec and CTDNE. The performances of all models is measure using Mean Average Rank and Hits@X metrics, and various sets of parameters. In the next chapter, we present the results of these experiments.

CHAPTER 9

Results and discussion

In this chapter, we present the results of the experiments and discuss them. Section 9.1 successively presents the results for each method with different sets of parameters to deduce what are the best results for the method. Section 9.2 makes the comparison between the methods and their best results to deduce which one works better overall. To understand the curves, the x-axis reads the number of epochs and the y-axis reads the value of the metric. The type of the metric is written above the curves. For instance, in Figure 9.1, the blue curve of Hits@10 reaches 0.15 at the epoch number 15.

9.1 Final results

We present here the results for Joint Evolution, node2vec and CTDNE on datAcron and DBLP. The tests are performed with and without attributes. We report the MAR and the Hits@1, 10, 20, 30 and 100 to get an accurate view of the performances of the models. We chose several values of Hits@X because a model could fail in the top 10 but be very good on the top 100, which is still a good result if there are thousands of entities. The resulting curves represent the metrics over the number of epochs. We also report the loss during training and validation.

9.1.1 Joint Evolution

All runs are done with an embedding size of 50 for both attributes and relations: with a higher embedding size, the space is too large and the features are not learned properly.

9.1.1.1 datAcron

Full set The results for datAcron are a little unstable due to the low number of samples but they stay within the same range. In Figure 9.1, the blue curve starts well but the results decrease dramatically after 10 epochs. On the other hand, the green and pink curves are more stable despite having lower results (if we do not consider the spike at the beginning). We remark that the model performs equally at Hits@10 and Hits@100, meaning that there are few results ranked between 10 and 100. The main difference between the two behaviors

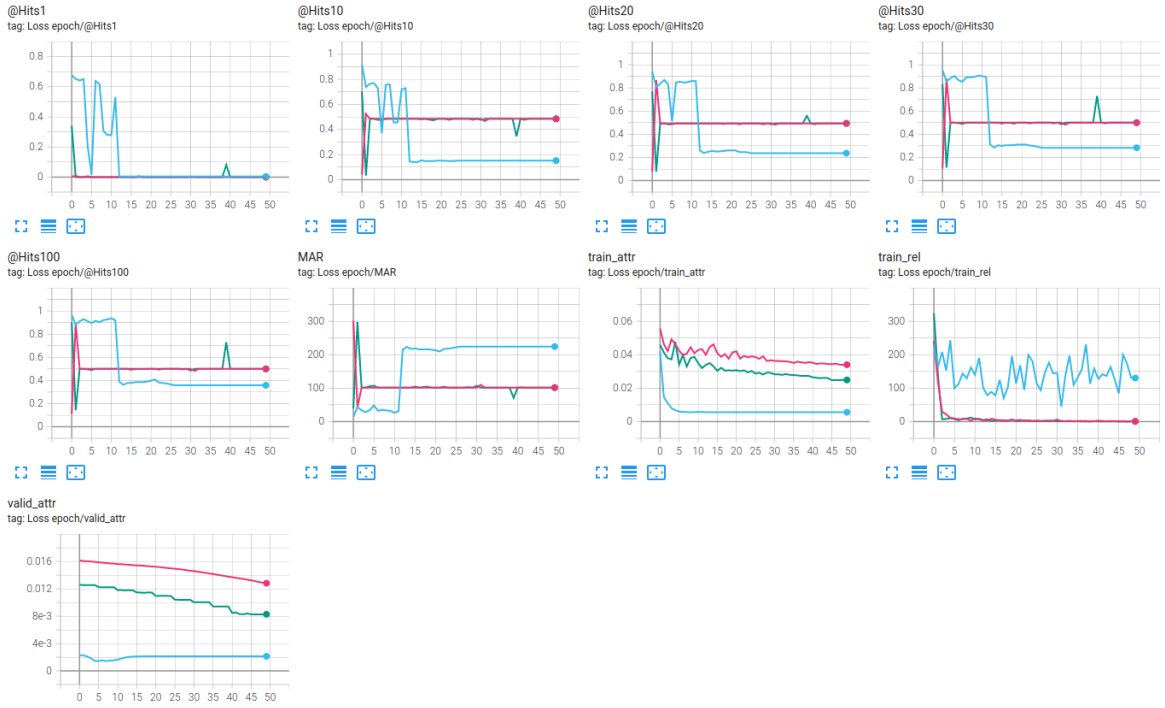


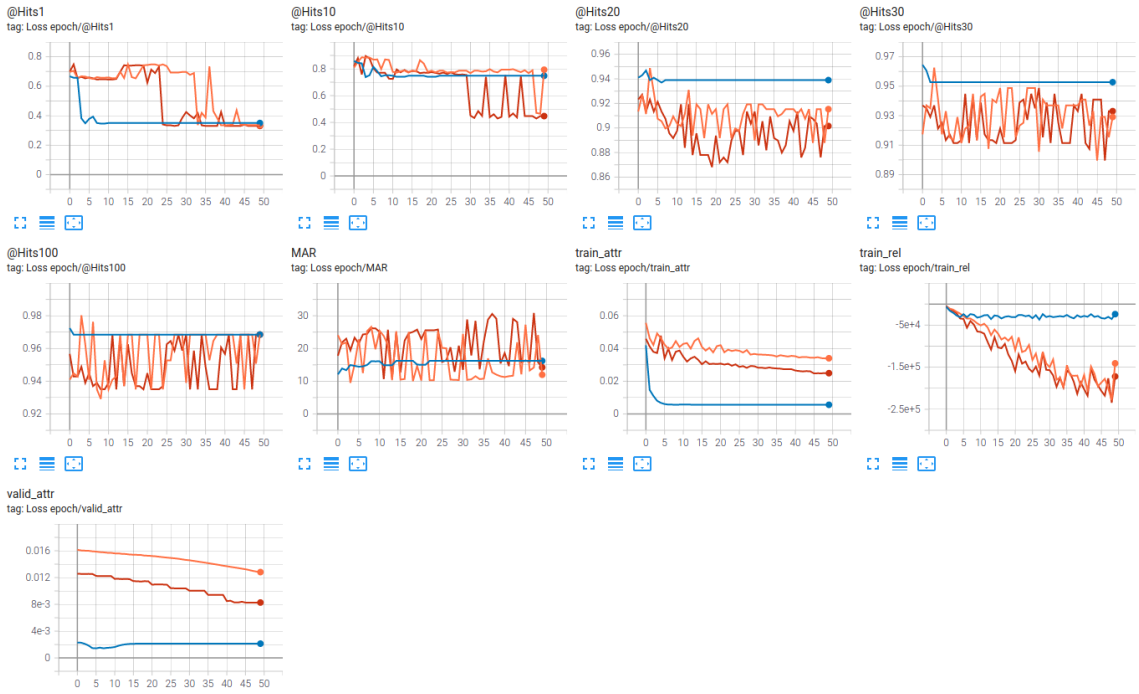
Figure 9.1: Joint Evolution on datAcron

is the weight decay: it is set at 0.005 for the blue curve and respectively at 0.2 and 0.3 for the green and pink ones. The model may require a higher weight decay to not overfit, especially with the low number of relational samples.

In Equation 6.61, we positively define the loss by choosing the maximum between the loss and zero. Blocking it at zero may prevent the model to learn further, so we remove this constraint to see if it improves the results. According to Table 9.1, the results are indeed improved. Looking at the blue line, the metrics are doubled. Surprisingly, it is this time the lower weight decay that gives the better results (0.005 opposed to 0.2 and 0.3). As a final result, we keep the blue curve for its stability. The chosen parameters are reported in Table 9.1 and the results in Table 9.2.

Relation only We then test Joint Evolution on datAcron, but without DKG^{A}. Only RelNet is trained here using the 764 relational quadruplets. We report the curves in Table 9.3. This version of the dataset is less sensitive to parameters since all the runs are rather close in terms of metrics. Hits@1 is still unstable due to the low number of samples. This time, we keep the positivity constraint from Equation 6.61 since it gives good results and is also used on DBLP. We report the results of the blue curve in Table 9.4.

With and without attributes We now make the comparison between the two version of the dataset and check our hypothesis about the role of attributes in the learning process (Figure 9.2). On each metric, the version without the attributes outperforms the one that uses them. Our intuition is that the four million attributive quadruplets add too much information compared to the raw relational quadruplets and disrupt the learning.

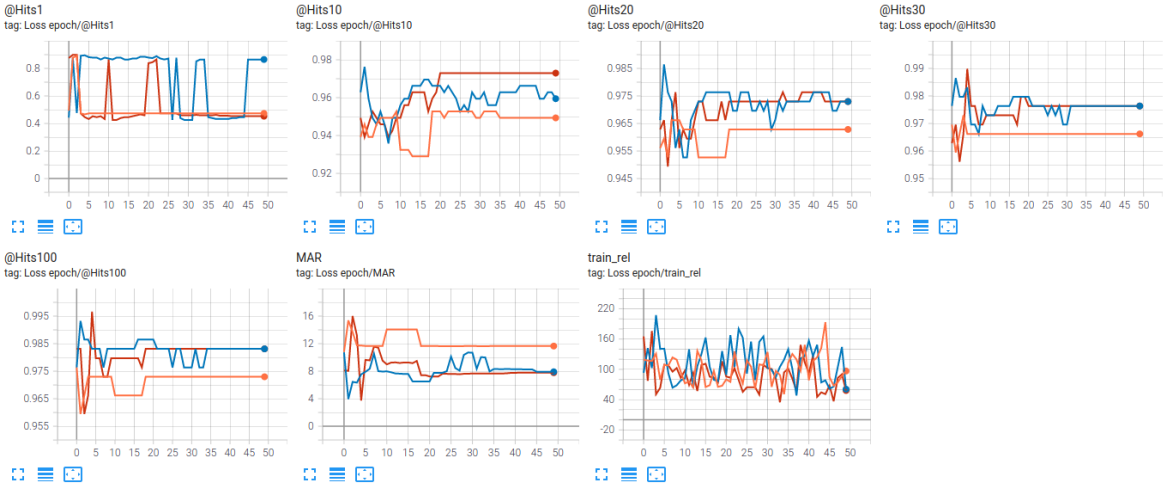


Curve	LR	Sched. steps	LR multiplier γ	Grad. clipping	Weight decay
Blue	0.001	2	0.5	0.1	0.005
Orange	0.001	1	0.95	0.1	0.005
Red	0.001	5	0.7	0	0.2

Table 9.1: Curves and parameters used for Joint Evolution on datAcron, but the loss is not positively defined

MAR	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100
16.21	0.3504	0.748	0.939	0.9528	0.9685

Table 9.2: Results for Joint Evolution on datAcron (blue curve)



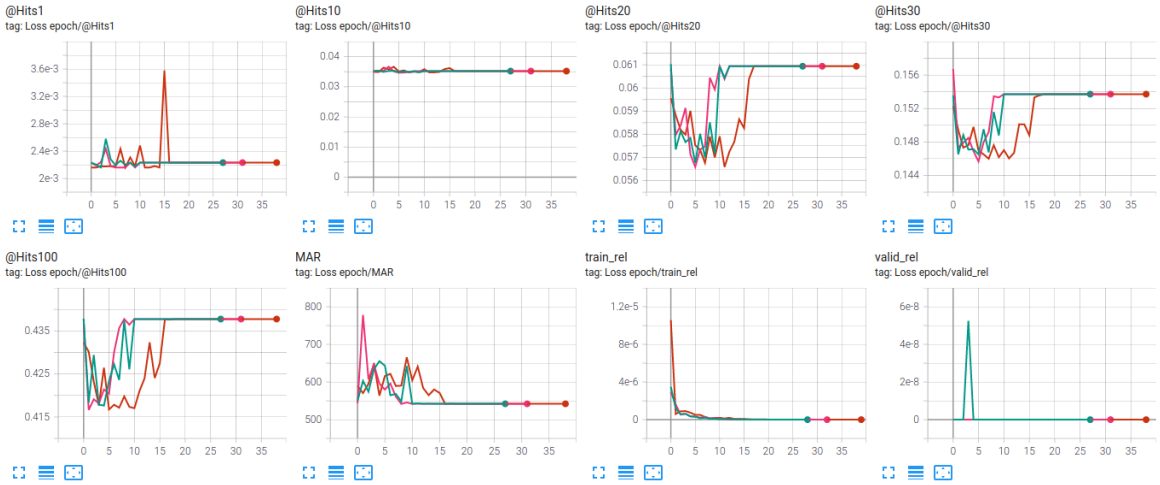
Curve	LR	Sched. steps	LR multiplier γ	Grad. clipping	Weight decay
Blue	0.001	1	0.5	0.1	0.3
Orange	0.001	2	0.5	0.1	0.005
Red	0.001	5	0.7	0	0.2

Table 9.3: Curves and parameters for Joint Evolution on relational datAcron

MAR	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100
7.58	0.8784	0.9662	0.9764	0.9764	0.9831

Table 9.4: Results for Joint Evolution on relational datAcron (blue curve)

Figure 9.2: Joint Evolution comparison on datAcron with (blue) and without (pink) attributes



Curve	LR	Sched. steps	LR multiplier γ	Grad. clipping	Weight decay
Green	0.1	2	0.5	0.1	0.005
Pink	0.01	2	0.5	0.1	0.005
Red	0.01	1	0.8	0.1	0.005

Table 9.5: Curves and parameters for Joint Evolution on DBLP - Part 1

9.1.1.2 DBLP

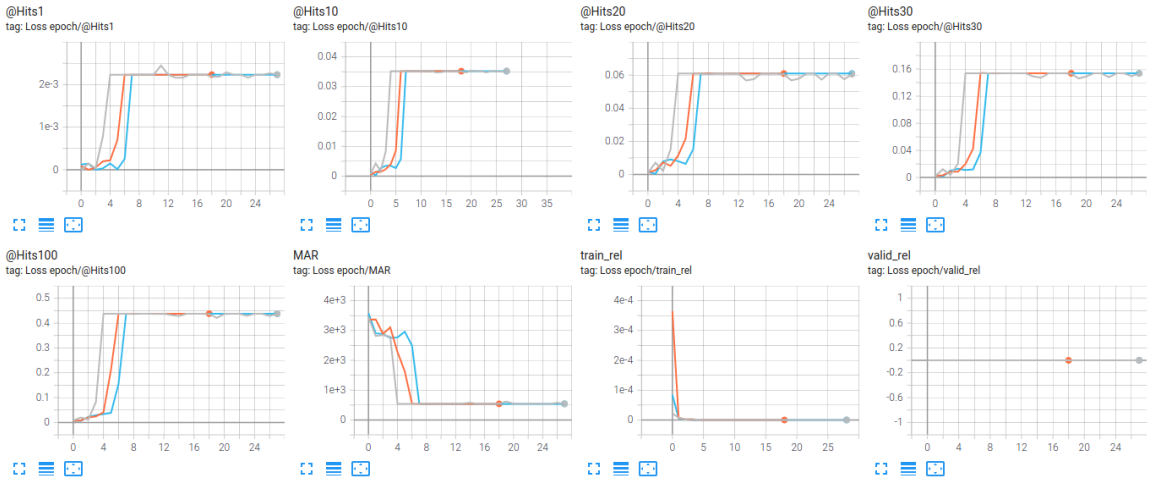
Full set On the full set of DBLP, Joint Evolution has almost the same behavior with different parameters: it always reaches the same plateau, the difference being the time taken to reach it. In the curves of Table 9.5, the plateau is reached from the first epoch, then the metrics decrease to then reach the plateau again. This is due to the high value of learning rate (parameters in Table 9.5). For instance, the red curve takes more time to get better because the learning rate stays higher during a longer time (higher multiplier). Since the learning rate needs to decrease over time to reach an optimum without diverging, it is slower than the other runs. The weight decay is set at 0.005 for all runs because higher values will erase what has been learned too much.

In Table 9.6, we present other runs with a lower learning rate. The learning is more stable with a natural progression and reaches the same maximum values as the previous runs. Because of this stability, we choose to retain these curves. Note that the orange and blue curves have the same parameters but a different seed for the training (the seed impacts the random initialization of the weights).

The results for Joint Evolution on the full set of DBLP are reported in Table 9.7.

Relations only We now test Joint Evolution on the DKG^{R} of DBLP only and report the curves and parameters in Table 9.8 and the results in Table 9.9. The learning rate has the same effect as on the full dataset, and we retain the pink curve for this very reason.

With and without attributes We now make the comparison between the two versions of DBLP, with and without the attributive part (Figure 9.3). The difference between the two seems large on Hits@1, but it is negligible given how close to zero both metrics are. Idem for the MAR which shows little difference. However, the gap is more significant on other Hits@X metrics, in favor of the strictly relational dataset. Thus, our hypothesis is false on the two datasets.



Curve	LR	Sched. steps	LR multiplier γ	Grad. clipping	Weight decay
Orange	0.001	4	0.1	0.1	0.005
Grey	0.001	4	0.9	0.5	0.005
Blue	0.001	4	0.1	0.1	0.005

Table 9.6: Curves and parameters for Joint Evolution on DBLP - Part 2

MAR	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100
542.1	2.2322e-3	0.035	0.061	0.1537	0.4379

Table 9.7: Results for Joint Evolution on DBLP (grey curve)

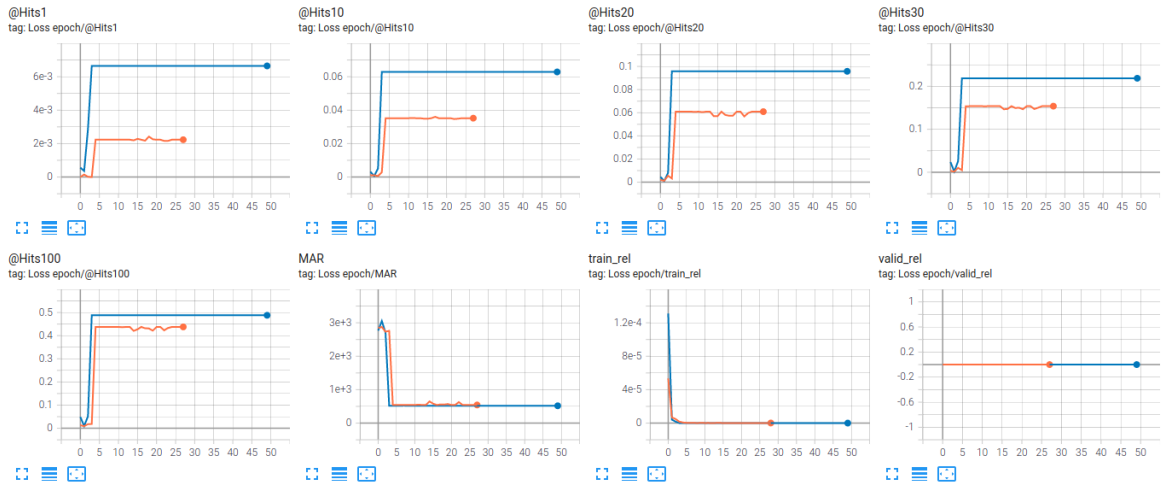
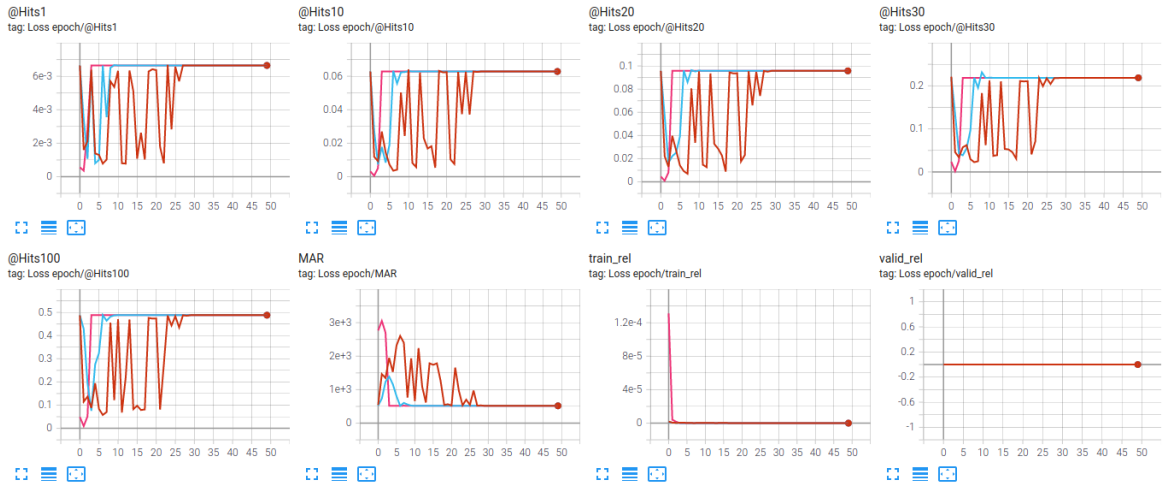


Figure 9.3: Comparison between full (orange) and relational (blue) DBLP with Joint Evolution



Curve	LR	Sched. steps	LR multiplier γ	Grad. clipping	Weight decay
Red	0.01	1	0.9	0.2	0.005
Blue	0.01	2	0.5	0.1	0.005
Pink	0.001	4	0.5	0.1	0.005

Table 9.8: Parameters for Joint Evolution on relational DBLP

MAR	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100
517.6	6.6466e-3	0.0629	0.0958	0.2187	0.488

Table 9.9: Results for Joint Evolution on DBLP relational quadruplets (pink curve)

9.1.2 CTDNE

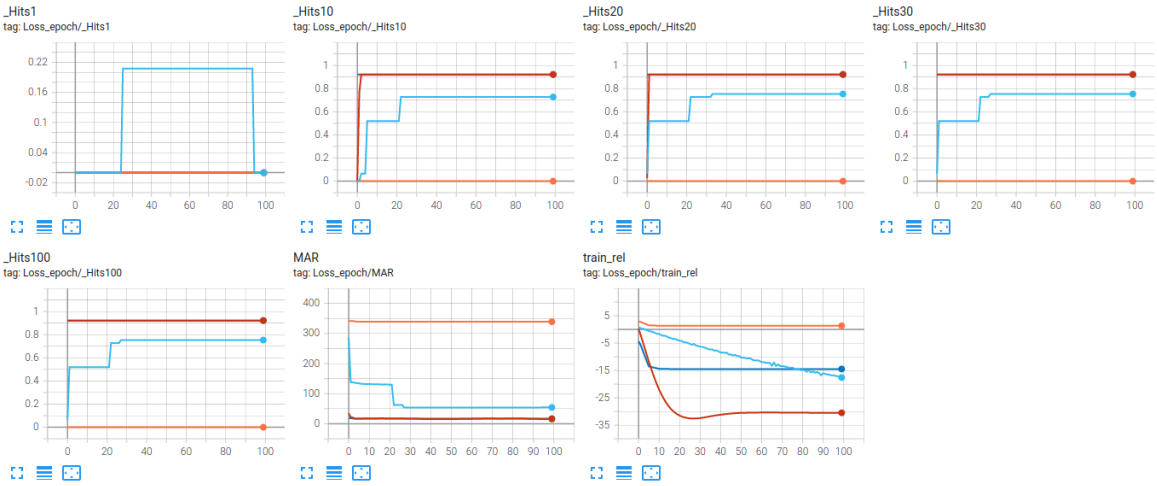
We now apply CTDNE to the two datasets. Since it does not deal with attributes, we only test it on DKG^{R}.

9.1.2.1 datAcron

Since the embedding does not evolve in this setting and because there are no attributes to learn, the training of CTDNE is very quick (a few seconds per epoch) and reaches its optimal points in a few epochs (6). We report the curves and parameters in Table 9.10.

The effects of the parameters is hard to evaluate on CTDNE. The most discriminant one is the learning rate since the two best overall curves, the red and dark blue ones, have the same, higher learning rate. However, the differences between the blue and orange ones are not explainable and require more experiments to clarify. We chose the red curve as the best one over the blue one because despite the large advantage on Hits@1 for the latter, this result is not stable and the performance is not as good on other metrics. Lastly, we chose the red over the dark blue one for its behavior regarding the loss: it is better on the red one even if the metrics are the same. This can be explained by the use of a scheduler on the dark blue one while there is none on the red. We can clearly see that the loss stops improving after five epochs, meaning that the learning rate becomes too low to improve the model further. This would also explain the lesser performance of the blue and orange settings.

We report the best results, represented by the red curve, in Table 9.11. It seems that there are no results between Hits@10 and Hits@100, meaning that 92.21% of the predictions fall between 2 and 10 and the remaining 7.79% between 101 and 363.

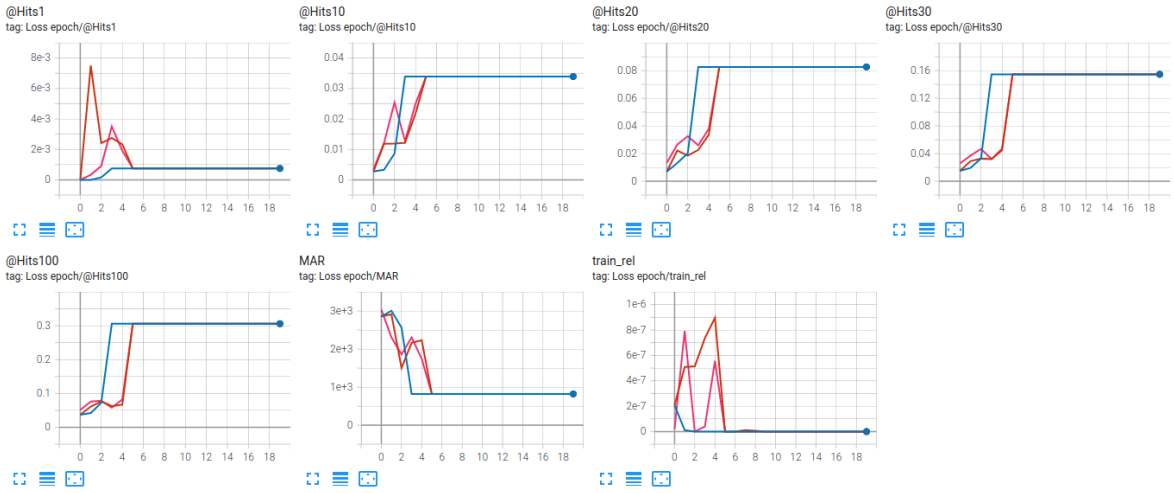


Curve	LR	Sched. steps	LR multiplier γ	Grad. clipping	Weight decay
Dark blue	0.1	5	0.1	0.1	0.1
Blue	0.01	4	0.1	0.1	0.1
Orange	0.01	5	0.1	0.1	0.1
Red	0.1	0	0.1	0.1	0.1

Table 9.10: Curves and parameters for CTDNE on relational datAcron

MAR	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100
15.3	0	0.9221	0.9221	0.9221	0.9221

Table 9.11: Results for CTDNE on datAcron relational quadruplets (red curve)



Curve	LR	Sched. steps	LR multiplier	Grad. clipping	Weight decay
Pink	0.01	5	0.1	0.1	0.005
Blue	0.01	1	0.1	0.1	0.1
Red	0.01	5	0.1	0.1	0.1

Table 9.12: Curves and parameters for CTDNE on relational DBLP

MAR	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100
822.6	7.5e-4	0.0339	0.0826	0.1545	0.3061

Table 9.13: Results for CTDNE on DBLP relational quadruplets (blue curve)

9.1.2.2 DBLP

The training on DBLP is much longer due to the larger number of relational quadruplets (two hours per epochs). Still, the optimum is also reached in a few epochs. We report the curves and parameters in Table 9.12.

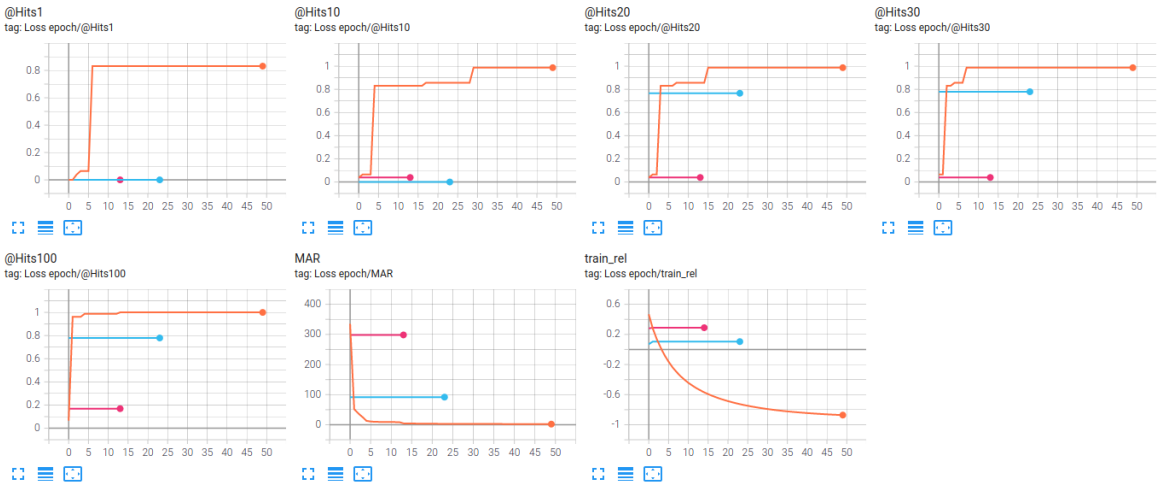
We first notice that the three curves end up with the same result despite some differences in the evolution of their learning rate and in the weight decay. The weight decay does not seem to have much effect given the similarity between the red and pink curves between which it is the only difference. More frequent learning rate reductions stabilize the learning as observed on the blue curve: we do not see the same oscillations with the two others. For these reasons, we retain the blue curve and report its results in Table 9.13.

9.1.2.3 Summary

The only parameter CTDNE seems sensitive to is the learning rate. Otherwise, all other settings give the same results. This may be because we only train the decoder part from pre-trained embeddings: there are fewer parameters to train for link prediction.

9.1.3 node2vec

At last, we train a decoder on node2vec embeddings. With this experiment, we have a reference for static methods.



Curve	LR	Sched. steps	LR multiplier	Grad. clipping	Weight decay
Orange	0.01	2	0.9	0.1	0
Blue	0.001	5	0.9	0.1	0.1
Pink	0.01	2	0.9	0.1	0.3

Table 9.14: Curves and parameters for node2vec on relational datAcron

MAR	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100
2.117	0.8312	0.987	0.987	0.987	1

Table 9.15: Results for node2vec on datAcron relational quadruplets (blue curve)

9.1.3.1 datAcron

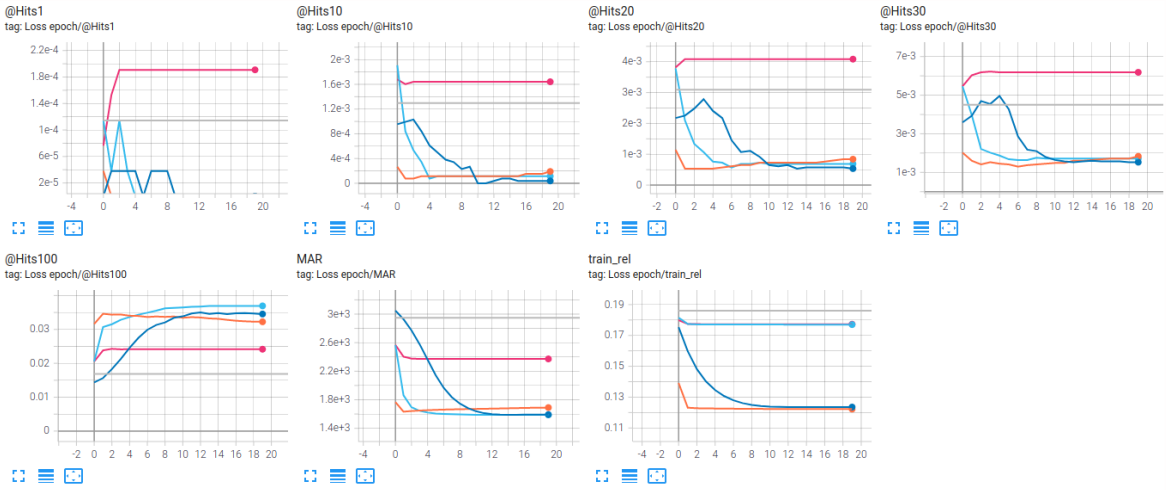
As for CTDNE, the training of node2vec on datAcron takes only a few seconds per epoch. However, we must this time wait 30 epochs to reach the optimum, but this is not a problem since it remains a matter of minutes. We report the curves and parameters in Table 9.14.

Undoubtedly, the determining parameter to train node2vec is the weight decay. The higher it is, the worse are the results: the decay of the weight must be too strong compared to the learning and the best setting is with no decay at all. The learning rate seems to have little effect: orange and pink have the same value and are diametrically opposed, while the blue one is in between while having a different value. Looking at the curve, the motto here is the lower the weight decay, the better the results. We report them for the orange curve in Table 9.15.

9.1.3.2 DBLP

The training on DBLP is longer than on datAcron (one hour/epoch), but still faster than CTDNE on the same dataset, and it takes 13 epochs to reach its optimum. We report the curves and parameters in Table 9.16.

Apart from the pink and grey curves, all settings behave similarly. In the grey case, the metrics do not evolve at all: the only differences laying in the gradient clipping and weight decay, further experiments are needed to see which one (or maybe both) is responsible for the learning impediment. The difference between the pink and blue curves, that have exactly the same setting, cannot be explained with the current experiments and requires further testing too. The most unsettling result is the decreasing values for Hits@1, 10, 20 and 30 over time



Curve	LR	Sched. steps	LR multiplier γ	Grad. clipping	Weight decay
Orange	0.01	0	0.9	0.1	0.1
Blue	0.01	1	0.9	0.1	0.1
Dark blue	0.001	0	0.9	0.1	0.1
Pink	0.01	1	0.9	0.1	0.1
Grey	0.01	1	0.1	0	0

Table 9.16: Curves and parameters for node2vec on relational DBLP

MAR	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100
1676	0	1.145e-4	7.252e-4	1.603e-3	0.0335

Table 9.17: Results for node2vec on DBLP relational quadruplets (orange curve)

for blue, dark blue and orange curves. While Hits@100, MAR and loss behave as expected, the other four are not supposed to drop. This leads to poor results on these metrics, but we still focus on these three settings since grey and pink have an awful MAR.

Out of these three, we exclude the blue one for having a higher loss, and orange and dark blue are basically the same curve going at different speeds (directly linked to their learning rate). We thus stick with orange as best result since it learns faster, and we report the results in Table 9.17.

9.1.3.3 Summary

node2vec is much more sensitive to parameters than CTDNE, mainly to the weight decay. It captures very well the interactions in datAcron but fails to do so on DBLP. This difference may come from the low number of samples in datAcron that makes a static method still efficient since there are not a lot of changes. Regarding DBLP, the large number of entities and relations over time makes the task more difficult for a static model.

9.2 Score comparison

We now compare all four methods on both datasets to evaluate both the results of Joint Evolution over the two other methods and the effect of the addition of attributes on the link

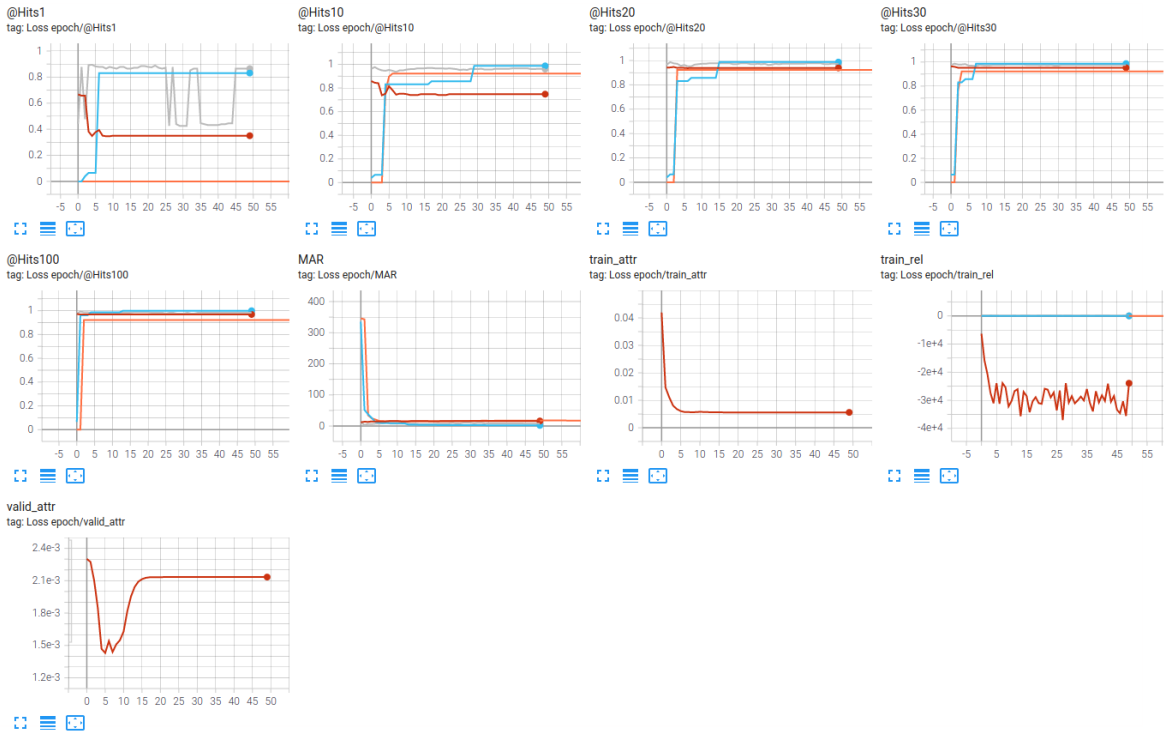


Figure 9.4: Comparison of Joint Evolution (red), Joint Evolution without attributes (grey), CTDNE (orange) and node2vec (blue) for datAcron

prediction task.

9.2.1 datAcron

We report the results for datAcron on Figure 9.4 and Table 9.18. The best method overall is node2vec, closely followed by Joint Evolution without the attributes. Still, all methods have similar performances in all metrics excepted for Hits@1, which suffers of instability caused by the low number of samples. We also noticed lower results for Joint Evolution on Hits@10.

	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100	MAR
node2vec	0.831	0.987	0.987	0.987	1	2.117
CTDNE	0	0.9221	0.9221	0.9221	0.9221	15.3
JE (RelNet)	0.8784	0.9662	0.9764	0.9764	0.9831	7.58
JE	0.3504	0.748	0.939	0.9528	0.9685	16.21

Table 9.18: Hits@ and MAR results on datAcron

9.2.2 DBLP

We report the results for DBLP on Figure 9.5 and Table 9.19. Joint Evolution without attributes has the best overall results, followed by Joint Evolution, then CTDNE and node2vec. CTDNE is close to Joint Evolution on Hits@10 and Hits@30 and even surpasses it on Hits@20, but the large gap on MAR in favor of Joint Evolution makes it better. node2vec,

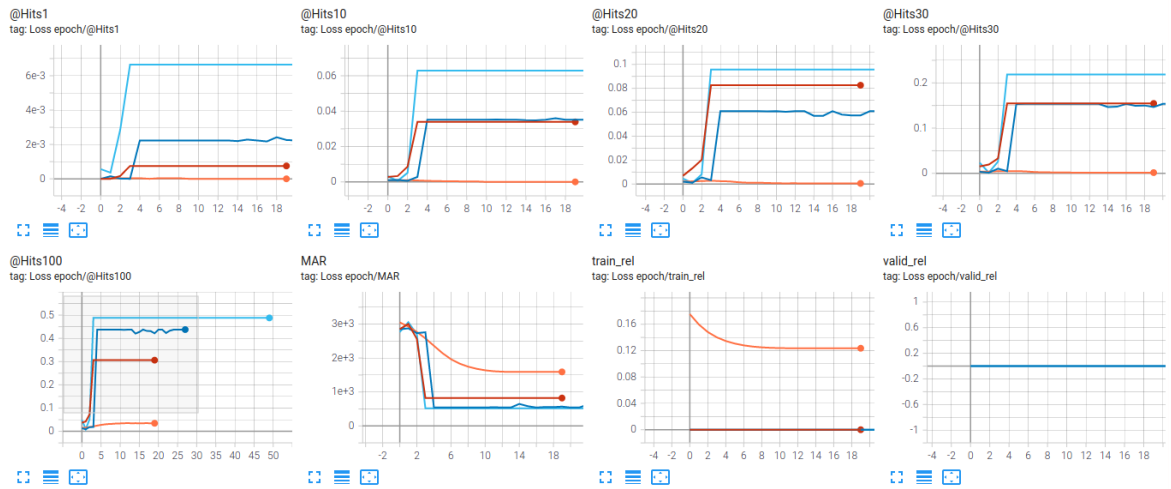


Figure 9.5: Comparison of Joint Evolution (dark blue), Joint Evolution without attributes (blue), CTDNE (red) and node2vec (orange) for DBLP

previously able to stand side by side with the other methods, fails completely at predicting links in DBLP.

	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100	MAR
node2vec	0	$1.145e^{-4}$	$7.252e^{-4}$	$1.603e^{-3}$	0.0335	1,676
CTDNE	$7.5e^{-4}$	0.0339	0.0826	0.1545	0.3061	822.6
JE (RelNet)	$6.646e^{-3}$	0.0629	0.0958	0.2187	0.488	517.6
JE	$2.232e^{-3}$	0.035	0.061	0.1537	0.4379	542.1

Table 9.19: Hits@ and MAR results on DBLP

9.2.3 Analysis

We ran our experiments on two datasets: one adapted to our usecase (MSA) and the other one more suited to an academic evaluation and comparison to other models.

9.2.3.1 Datasets

The first one, datAcron, is a small dataset, easier to learn. All models were able to learn well on it, probably due to its small size. We need to run more experiments on a larger dataset of this kind to evaluate if the models can generalize well. Joint Evolution (with attributes) is the model that struggled the most on datAcron: our hypothesis is that the dataset is too simple compared to what Joint Evolution is made to handle regarding the number of entities, relations and relation types, hence the difficulties to learn. The number of parameters is too large for the model to be trained correctly. Another possibility, given that Joint Evolution without attributes comes in second place, is that the attributive embeddings add noise and do not help the learning. For all datasets, the Hits@1 is unstable and not very representative of the model performances because of the limited dataset size. This metric should have more sense with a larger dataset.

On the second dataset, DBLP, the ratio between relational and attributive links is far more balanced. Joint Evolution has the best results on this dataset, with a small advantage for the

version without attributes. Despite the size of the dataset being more adequate, the version with attributes may suffer from its large number of parameters: a 512×50 matrix is needed to scale the title embeddings to the size of the attributive embeddings. Also, Google's sentence encoder is trained to output vectors of size 512, which is a lot considering that we only embed the titles of the articles (the embedding may actually be longer than the title itself). It may have been better to embed the abstracts, but they were not always available. The easiest option here would be to reduce the output size of the sentence encoder.

CTDNE has results close to Joint Evolution, apart from its higher value of MAR. It status of dynamic model allows it to learn more efficiently than node2vec, but the results might be impeded by the absence of update for embeddings and the homogeneous learning of embeddings. At last, node2vec cannot learn because the larger dataset size implies more temporal dependencies. node2vec being a static method, it cannot capture the evolving interactions between entities.

9.2.3.2 On the use of attributes

Since Joint Evolution with RelNet always has better results than Joint Evolution, our hypothesis that attributes improve results on link prediction is rejected. Unfortunately, the addition of attributes in the link prediction task degrades the results compared to a purely relational approach. However, Joint Evolution without attributes still performs better than CTDNE and node2vec, and Joint Evolution has streaming capabilities with embeddings that can be updated over time, which is an important requirement regarding MSA.

Despite these results, we still hope that attributes can improve the prediction of new relations, since additional information should always be beneficial to infer new links. Here are several leads on why we did not get the expected results:

- The training of Joint Evolution involves a large set of hyperparameters, and every combination of them could not be tested because of the computation time it would require. Thus, we may have missed the perfect set of parameters that would have made Joint Evolution work better with attributes;
- datAcron is an unbalanced dataset, and its main purpose was thus to check if a large number of attributes can alleviate the lack of relational links. However, maybe a minimum number of links is needed and we did not reach this minimum with datAcron. Since we will always have more AIS than relational events, this experiment should be run again on a larger set of events;
- In datAcron, the dynamic attributes were natural and straightforward: a vessel has a position and a speed that evolve over time. In DBLP, the notion of attribute is a little more farfetched: the articles used as attributes can also be considered as entities, and they can be represented by multiple values. We chose to encode the title because it was the most straightforward way to do it and it represents the article field, but other values may have worked better. By wanting to be more precise in the information given to Joint Evolution, we may have made learning noisy. Using the conference of publication is another lead than can be explored, although some conference themes can be large and not give enough information about the theme of the publications;
- The two datasets we used have a limited number of relation and attribute types: we only have three of each in datAcron and two of each in DBLP. A knowledge graph is the most extreme case of heterogeneous graph that exists, with possibly hundreds

or thousands of relations and attributes types. It would have been more interesting to compare Joint Evolution and other methods on such a graph, but dynamic attributed graph datasets are not easy to come by.

9.3 Conclusion

We tested four methods on datAcron and DBLP: Joint Evolution, Joint Evolution without the attributive network, CTDNE and node2vec. The two datasets are very different in nature and make the results of the four methods different. All methods perform well on datAcron while only dynamic methods work on DBLP. The best overall method is Joint Evolution without attributes, which dismisses our hypothesis on the benefits of attributes in the relation prediction process. We presented possible causes and leads to improve the results, and still hope that, in the correct setting, we can use attributes to improve the prediction of relational links.

Conclusion

CHAPTER 10

Summary and perspectives

10.1 Summary

Since this thesis is in the scope of the CIFRE program with GREYC and Airbus Defence and Space, we started by presenting the industrial problem this thesis aims to solve. The main objective is to improve Maritime Situational Awareness and provide a decision aid to surveillance operator who have to navigate through a lot of incoming hard and soft data. To improve the understanding of a situation, we analyze historical data to predict next events and raise early alerts. In order to do this, we needed to create a representation of the past and current maritime situation and turned to knowledge graphs to do so. Their semantic descriptive capabilities coupled with the graph-based interactions make them a great tool for this purpose. We then started by assessing the state-of-the-art on two topics: how to improve maritime situational awareness and link prediction methods on dynamic attributed graphs. In the first one, we found many approaches on route generation and anomaly detection. Some deal with events, but mostly with rule-based engines. The field of MSA improvement using event-based link prediction methods on knowledge graphs is thus not actively studied and that is where we position this work. On the second topic, we found out that there are many contributions about link prediction in static, static attributed and dynamic graphs, but fewer in dynamic attributed graphs. We thus take inspiration from the dynamic methods to adapt them to a new setting including attributes, while making the hypothesis that attributes can improve the results of link prediction.

Based on this, we proposed Joint Evolution, an encoder-decoder system composed of two parts: one that learns embeddings from changes of values for attributes to predict the next ones, and the other that learns relational embeddings to predict new relations in the graph using both types of embeddings. We designed a preprocessing pipeline to create independent sequences of events that ensure that events happening to an entity are processed in the correct order. We then made the choice to use Gated Recurrent Units to process these sequences and generate two embeddings for each entity: the attributive one and the relational one. Once the model is trained to update embeddings that can be used to predict the next event, embeddings can be updated on the fly as new events are observed using only the encoder parts of Joint Evolution. When a prediction is needed, only the decoder parts are used to answer the question at hand, generally in the shape of a quadruplet like (subject, relation, ?, timestamp).

The decoder will then find the object that best fits the quadruplet by computing a score for each possibility.

We tested Joint Evolution on two datasets, datAcron and DBLP. datAcron is a set of events and AIS messages from vessels in the Bay of Biscay near Brest, and DBLP is a citation network that links authors and articles with authorship and co-authorship relations. The main differences between the two datasets lie in the number of events and in the type of the attributes, which gives us two settings to experiment with. We compared Joint Evolution with two other methods: node2vec, a well-known random walk approach to embed static graphs and CTDNE, its dynamic version. We also ran experiments with a version of Joint Evolution that considers relations only to evaluate the effect of attribute inclusion. In the end, Joint Evolution without the attributes is overall the best out of the four models, which rejects our hypothesis that the addition of attributes can improve the results on relational link prediction. Still, Joint Evolution has better results than node2vec and CTDNE and can deal with heterogeneous graphs in a streaming fashion. We still hope that any addition of information may be valuable to the learning and explain these results with several factors, the main ones being the large set of possible parameters and the configuration of the datasets.

10.2 Perspectives

With these results at hand, we now describe the limitations of Joint Evolution and pave the way for future work.

10.2.1 Limitations

We defined five essential items for MSA: follow the evolution of entities, detect events and threats, process streaming data, deal with uncertainty and make the output of the model explainable. Unfortunately, we did not have the time to work on uncertainty and explainability, which are left for future works. About the link prediction itself, Joint Evolution is limited to subject and object prediction. We are not able to predict *when* an event will happen, only which entity will connect with another at a given time. We focused on this because it is the main usecase for MSA but being able to predict time may also be a useful feature. The other prediction we are not able to make is the relation type: as stated in Section 6.3.3, training Joint Evolution to predict a relation type is different because the notion of challenge is not the same. We do take the relation type into account with the specific weights, but it does not give the capability to predict it. To be able to predict the time and relation type of an event, a new training setup must be devised.

Another problem in our setting is that the update is only performed on the entities involved in an event, while this event could have an effect on other entities. For instance, a conflict between two countries (entities c and c') cannot influence the relation between a vessel from country c (entity v) and a vessel from country c' (entity v') since the embeddings of v, v' are not updated when the conflict arises. Methods that take neighborhood and context into account in the knowledge graph could bridge this gap [Jin et al., 2019, Heidari and Papagelis, 2019].

The last limitation worth mentioning is the nature of the attributes we use. The attributes in datAcron are scalar values while DBLP uses vectors. It required an adaptation of Joint Evolution between the two. If a dataset were to mix both types of attributes, we would have to find a way to deal with both, for instance by deactivating the AttrNet decoder when vector attributes is encountered.

10.2.2 Short term perspectives

Our short term perspectives are mainly about the datasets and the experiments. To assess in a larger extent the effect of attributes, we would like to train Joint Evolution on a larger version of datAcron with more events and entities. We may also rethink the way we create attributes for DBLP in a more straightforward, easy to learn manner. This could start with the reduction of the sentence encoder dimensionality, the addition of the conferences where the articles are published and the organization the authors come from (though the latter implies dealing with static attributes).

Finding an evaluation dataset for dynamic attributed graphs is not easy since there is no dataset of reference in this field like MNIST[1] is in image analysis. Some known graph datasets such as Epinions [Richardson et al., 2003], Freebase or WordNet[2] are often used to evaluate static graphs and are either undirected or homogeneous. Citation networks such as Citeseer [Bhattacharya and Getoor, 2007] and Cora [Getoor, 2005] are either static, homogeneous or without attributes, and are mainly used for article classification. Some well-known dynamic datasets are GDELT [Leetaru and Schrodt, 2013] and ICEWS [Boschee et al., 2015] but it is difficult to find dynamic attributes in them. Enron [Klimt and Yang, 2004] could be used as we used DBLP since it is an email exchange network, but it only has 150 entities and is homogeneous. Moreover, the complexity of the email contents may not help to predict a mail exchange between two employees. The creation of a reference dataset for dynamic attributed graphs would be of great help in this field of study.

Our last short-term perspective is to run more experiments on the current settings, especially regarding the sets of parameters. With more time, we could improve the choice of parameters and find the best setting for each method. We could also do better to evaluate the effectiveness of AttrNet, which was not extensively tested apart from optimizing its loss.

10.2.3 Long term perspectives

On the longer term, the objective is to fill the void in the requirements for MSA, namely uncertainty and explainability. Our major lead on uncertainty is the use of Markov Logic Networks and Probabilistic Soft Logics [Chekol et al., 2017, Chekol and Stuckenschmidt, 2016]. About explainability, the study by Adadi and Berrada [Adadi and Berrada, 2018] is a solid basis to find solutions adapted to our setting.

Another envisioned addition to Joint Evolution is a true semantic description of the entities and their interactions. We now have relation types, but currently no way to tell the system that an entity is a vessel or a port, which may play an important role in the learning process. Knowing in advance what an entity is, what its full set of attributes is, and what interactions it can have, reduces the number of possibilities when making prediction and could make the generation of new links more accurate. We would then have to find a way to include full ontologies within the model to provide their descriptive power to the service of the learning process.

[1] `http://yann.lecun.com/exdb/mnist/`

[2] `https://everest.hds.utc.fr/doku.php?id=en:transe`

Part IV

Appendices

APPENDIX A

Résumé

A.1 Contexte

Comment surveiller efficacement les activités maritimes et anticiper les comportements anormaux des bateaux qui naviguent à travers le monde contre vents et marées? C'est la question à laquelle j'apporte une réponse dans ce manuscrit qui parle aussi bien de données opérationnelles et de séries temporelles que de graphes de connaissance et de machine learning. Il présente également comment utiliser les données transmises par les bateaux afin de prédire leurs interactions, présentes ou futures.

Ces travaux s'effectuent dans le cadre d'une collaboration CIFRE entre Airbus Defence and Space (Elancourt, France) et le laboratoire GREYC de l'Université de Caen Normandie. Ce dispositif est géré par l'Agence Nationale de la Recherche et de la Technologie (ANRT), qui soutient les entreprises françaises souhaitant embaucher un doctorant et créer une collaboration de recherche avec un laboratoire public. Ceci a pour but de faciliter les échanges entre les mondes industriel et académique.

Le sujet principal de ces travaux est la représentation de l'évolution de situations afin d'extraire de nouvelles informations. Pouvoir représenter une situation et sa dynamique est très intéressant pour les opérateurs de surveillance et de renseignement, car ils ont besoin d'en savoir le plus possible sur une zone ou une personne d'intérêt. Cette connaissance d'une situation est également appelée *Situational Awareness*, et est un élément critique dans de nombreux domaines, tels que la protection de la vie humaine et de la propriété, le respect de la loi, l'aviation, le contrôle du trafic aérien, la navigation, la santé, les urgences ainsi que les opérations militaires. Trois piliers sont nécessaires à son efficacité: la perception de l'environnement, la comprehension de la situation et la prédiction des états futurs.

Le premier pilier se concentre sur la collecte de données hétérogènes concernant une zone ou des entités d'intérêt. Elles peuvent être des données "hard" venant de capteurs physiques (quantitatives, Figure A.1) ou des données "soft", i.e. des données linguistiques produites par des humains (qualitatives). Le second pilier utilise ces données pour en extraire des informations sur la situation étudiée et en connaître tous les tenants et aboutissants. Enfin, le troisième utilise ce qui a été compris de la situation actuelle pour prédire ce qui pourrait arriver ensuite. Pouvoir inférer la suite des événements à partir des connaissances courantes

procure une capacité de prévision aux opérateurs, qui pourront alors mieux anticiper les problèmes et menaces.

Grâce à la Situational Awareness, un opérateur de surveillance peut analyser la situation pour mieux la comprendre. Durant cette analyse, il pourra établir des liens entre les entitiés représentant leurs interactions (par exemple, des liens d'affiliation, de commandement, de subordination ou de conflit) mais aussi créer de nouvelles entités. Actuellement, cette analyse est très peu automatisée et ne couvre qu'une petite partie des données disponibles. Avec l'augmentation du nombre d'entités à surveiller (personnes, véhicules, appareils...) et le déploiement de plus en plus de capteurs qui produisent massivement des données, il est devenu presque impossible pour un opérateur d'analyser et d'évaluer manuellement une situation. Il y a ici un besoin d'automatisation du processus de surveillance, pour que les opérateurs et experts n'aient qu'à regarder les événements d'intérêt proposés par un système automatisé, plutôt que de parcourir l'ensemble des données à la main.

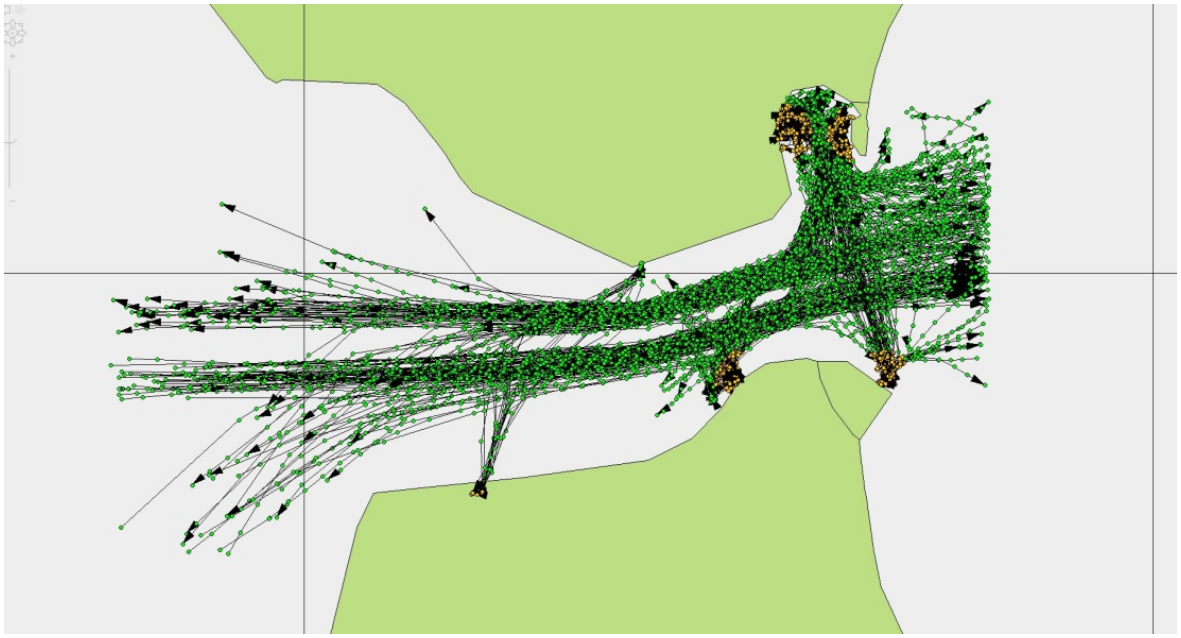


Figure A.1: Un jour de trafic maritime dans le détroit de Gibraltar. Le trafic est dense et difficile à analyser.

Pour ce faire, la représentation de la connaissance est de la plus grande importance et nécessite de relever plusieurs défis. La structure de données doit être dynamique pour suivre l'évolution de la situation, contenir les attributs et caractéristiques de chaque entité et représenter leurs interactions avec leur environnement. Puisqu'il y a toujours un humain dans la boucle, l'information doit être facile à visualiser, comprendre et parcourir. De plus, toute décision prise par un système automatisé doit être interprétable ou explicable pour avoir la confiance de ses utilisateurs, i.e. l'opérateur qui l'utilise doit connaître les raisons de sa décision. Dans un contexte de défense et de surveillance, il n'est pas possible pour les décideurs d'accepter et d'utiliser les résultats d'un tel système s'il n'est pas fiable et expliqué. De plus, l'incertitude est présente à plusieurs niveaux: les prédictions ont un degré de fiabilité, et les données elles-mêmes sont sujettes à des erreurs: un rapport peut contenir une faute de frappe, les capteurs ont une portée et une précision, ou des collisions de messages peuvent perturber leur réception par des satellites. Enfin, la surveillance d'une situation donnée doit être continue et les prédictions faisables à n'importe quel moment: la perception de la situ-

ation (i.e. l'ajout de nouvelles informations sur l'environnement) doit idéalement être faite en temps réel et le temps de calcul des prédictions fait en un temps raisonnable pour que la réaction appropriée puisse être déployée aussi vite que possible.

Les graphes de connaissance sont de plus en plus utilisés pour représenter un champ de connaissances: des paires de noeuds dans le graphe sont reliées par un arc, formant une relation entre les deux noeuds sous la forme de triplets (sujet, relation, objet). Ce réseau de relations constitue un graphe de connaissance, et modélise des informations telles que (RMS TItanic, :construitPar, WhiteStarLine). Le plus souvent, cette connaissance est statique et n'évolue pas au cours du temps. Mais le graphe peut également indiquer à quel moment un événement s'est produit, e.g. (RMS Titanic, :construitPar, WhiteStarLine, 1909). Avec les graphes dynamiques, de nouveaux liens peuvent être inférés grâce à l'apprentissage de dépendances temporelles, et aider à comprendre le comportement des entités au cours du temps.

De plus, les noeuds des graphes dynamiques peuvent être entichis avec des attributs qui caractérisent chaque entité. Généralement, les attributs sont attachés aux noeuds, comme par exemple (RMS Titanic, :taille, 269.1), et donnent plus d'informations sur les entités. La dynamicité s'applique également aux attributs, et tout particulièrement lorsque l'on s'intéresse à des entités mouvantes telles que des avions, des bateaux ou des personnes, e.g. (RMS Titanic, :latitude, 40.963, 23:40 1912/04/14).

A partir de tels graphes, l'apprentissage relationnel vise à raisonner et à inférer de nouveaux liens entre les entités, ou des liens qui ont été manqués dans le passé. Les applications directes sont la complétion de graphes de connaissance et les systèmes de recommandation. Générer de nouveaux liens dans un graphe est maintenant un usage commun de l'IA, mais beaucoup de méthodes de l'état de l'art se concentrent sur les liens statiques (avec ou sans attributs) ou sur les liens dynamiques. En revanche, peu d'entre elles traitent des liens dynamiques *et* des attributs. Quand il n'y a que quelques relations entre des entités, l'évolution de leurs attributs peut être d'une grande aide pour les caractériser.

En gardant ceci à l'esprit, les bateaux, ports, pays et compagnies maritimes ont des interactions variées, et l'état des bateaux évolue rapidement. Une variété de données, telles que les positions, vitesses, directions ou destinations peuvent être collectées. Ces données peuvent donner des connaissances plus précises sur ce qui est en train de se passer: un bateau entre dans un port, deux navires se rencontrent, ou un motif de déplacement indique une activité de pêche. Sachant cela, un opérateur de surveillance voudrait savoir à quel moment les deux bateaux vont se rencontrer, ou l'heure d'arrivée au port d'un bateau en fonction du contexte environnant. Cela s'applique également à la piraterie, à la contrebande et à la pêche illégale. De telles situations maritimes peuvent être représentées par un graphe de connaissance dynamique avec attributs. L'objectif est d'utiliser ce graphe pour inférer de nouvelles relations entre les entités maritimes, et donc d'améliorer la *Situational Awareness* de l'opérateur en détectant en avance de nouveaux événements.

C'est sur cette tâche que ce concentrent les contributions de ces travaux: nous proposons une structure de graphe de connaissance dynamique avec attributs qui représente une situation en évolution, et y appliquons un système d'apprentissage profond composé de deux réseaux de neurones temporels. Ces réseaux apprennent des représentations pour chaque entité et prédisent leur comportement futur en se basant sur les informations passées. Puisque les situations maritimes impliquent des entités mouvantes, des changements environnemen-

taux, des interactions, des capteurs, de l'incertitude et des opérateurs humains, notre cas d'usage principal est le contexte maritime. Ces travaux peuvent aussi s'appliquer à l'analyse de réseaux sociaux et nous les testerons également sur un réseau de citations bien connu, DBLP, pour s'assurer qu'ils sont généralisables.

La première partie de ce manuscrit est dédiée à l'état de l'art et est divisée en trois chapitres décrivant respectivement les méthodes pour améliorer la *Situational Awareness* maritime, les structures de graphes de connaissance ainsi que comment créer des représentations pour ces graphes et en prédire de nouveaux liens. La deuxième partie présente nos contributions. Le premier chapitre formalise le problème de prédiction de liens dans les graphes de connaissance dynamiques avec attributs. Le second introduit notre système, Joint Evolution et comment nous l'avons construit. La troisième partie est dédiée à nos expérimentations sur des jeux de données académiques et opérationnels. La dernière partie résume le travail effectué et conclut avec des perspectives.

A.2 Introduction à la Situational Awareness maritime

La surveillance maritime est fortement liée à la *Situational Awareness*, produit de grands volumes de données hétérgènes et a besoin de traitements automatisés. Ce sera donc notre cas d'application principal.

La Situational Awareness maritime (SAM) est définie par l'Organisation Maritime Internationale comme étant "la compréhension de tout ce qui est associé avec l'ensemble de l'environnement maritime qui pourrait impacter la sécurité, la sûreté, l'économie ou l'environnement". Le but est d'établir une image de la situation maritime à un moment donné, en percevant les différents éléments qui agissent dans l'environnement et en comprenant leur signification, pour prendre les bonnes décisions et prévoir ce qui va arriver.

Pour être efficace, la SAM a besoin d'un flux constant de données récentes, d'opérateurs formés et ayant de la connaissance experte ainsi que d'une prise de décision efficace. Ainsi, les défis suivant doivent être relevés pour une SAM efficace:

- *Fusion de données hétérogènes*: pour avoir la meilleure vue d'une situation, des données de diverses sources doivent être aggregées dans une structure de données commune. Ces sources peuvent inclure des données de navigation, des données de capteurs comme la température, les courants ou des sons, des images satellite ou radar, des rapports des autorités portuaires... Les trajectoires, images, textes et sons doivent être assemblés pour créer la photographie la plus complète possible pour l'opérateur de surveillance.

- *Evolution des entités*: une situation maritime est un monde en constante évolution avec peu de temps entre deux événements. Un bon modèle d'évolution est donc requis pour détecter les changements.

- *Détection d'événements et de menaces*: un événement peut être représenté par une relation entre deux entités ou une évolution d'attributs. la définition de menace dépend des faits mais aussi du contexte: une nation peut ne pas considérer un échange entre deux navires de pêche comme une menace, mais une ONG traitant de la conservation des océans pourra y voir de la pêche illégale, et donc une menace. Faire la distinction nécessite de la connaissance experte, éventuellement sous la forme d'étiquettes sur les données.

- *Streaming*: la SAM a besoin d'une surveillance constante des zones maritimes, ce qui veut dire que le modèle de surveillance doit être capable de traiter un flux continu de données.

- *Incertitude*: les données maritimes proviennent souvent de la fusion de données capteurs hard (capteurs) et soft (sites web, renseignement). Cependant, cette donnée est rarement précise à 100 %: l'incertitude doit être prise en compte pour prendre une décision.

- *Explicabilité*: les modèles d'IA sont souvent des boîtes noires, rendant difficile l'explicitation des causes d'une alerte. Cependant, un opérateur a besoin d'une justification pour sa décision. Puisque les opérateurs ne font pas complètement confiance aux systèmes d'IA, l'explicabilité des modèles d'aide à la décision est un critère déterminant pour la SAM [Adadi and Berrada, 2018].

Lorsque l'on regarde le trafic maritime dans sa globalité, les événements dangereux ou illégaux sont considérés comme rares et inhabituels et correspondent donc à la définition d'anomalie. Les détecter n'est pas aisé car il y a beaucoup de données dont il faut extraire les éléments importants, d'où le besoin d'un outil de détection automatique pour les événements anormaux. Voici une liste non-exhaustive des événements pouvant être détectés à partir des données de positions et de contexte:

- Navire endommagé ou immobilisé;
- Perte de cargaison;
- Pollution marine;
- Des tempêtes ou toute météo qui pourrait ralentir le voyage;
- Une collision;
- Echange de marchandises/d'équipage;
- Direction ou vitesse inhabituelle;
- Pêche dans une zone protégée;
- ...

Objectif de la thèse

Ces travaux se concentrent sur la détection et la prédiction d'événements: notre objectif est de créer un modèle qui apprend le comportement des navires en utilisant des événements étiquetés entre des entités de la situation (navires, ports, compagnies maritimes...) et qui détecte automatiquement les causes de ces évenements afin de trouver des patterns similaires ailleurs dans les données. Au lieu d'indiquer au modèle comment trouver les événements, nous lui donnons des événements, de l'AIS et des informations sur l'environnement pour qu'il puisse apprendre les causes par lui-même.

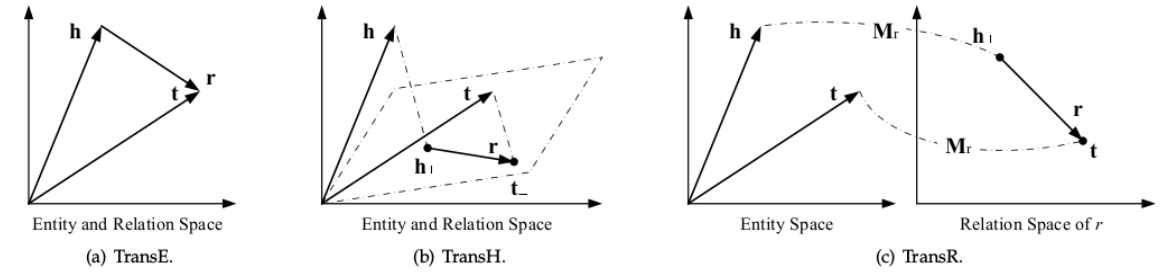


Figure A.2: Illustrations simples de TransE, TransH, et TransR de [Wang et al., 2017]. Les figures sont adaptées de [Wang et al., 2014], [Lin et al., 2015b]

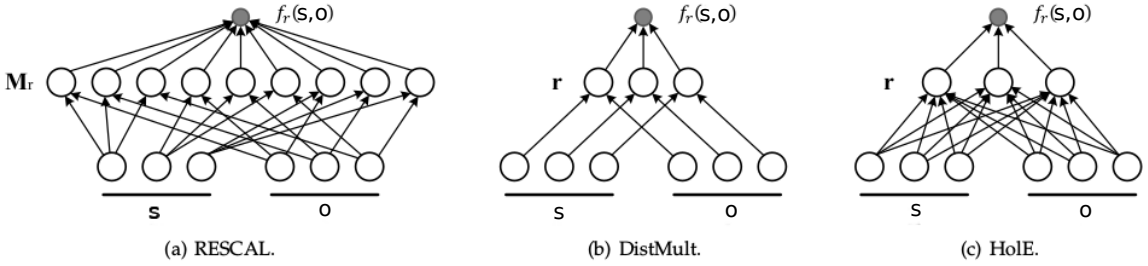


Figure A.3: Illustrations simples de RESCAL, DistMult, et HolE adaptés de [Wang et al., 2017].

A.3 Etat de l'art

A.3.1 Graphes statiques

Cette section décrit l'état de l'art sur les graphes statiques. Pour plus de détails sur ces méthodes, voir les études de [Wang et al., 2017] et [Cai et al., 2018].

A.3.1.1 Modèles translationnels

Dans un contexte statique, chaque noeud est représenté par un seul vecteur. Une grande variété de techniques existent, dont les modèles translationnels qui évaluent la probabilité d'un événement (sujet, relation, objet) en mesurant la distance entre les deux entités. TransE [Bordes et al., 2013] en est le représentant le plus connu et a fait l'objet de plusieurs extensions. Un résumé de ces méthodes est présenté en Figure A.2.

A.3.1.2 Modèles de correspondance sémantique

Les modèles de correspondance sémantique utilisent des scores pour comparer les plongements des entités et des types de relations en se basant sur leur similarité. Une illustration de ces méthodes est donnée en Figure A.3.

RESCAL et ses extensions RESCAL [Nickel et al., 2011] a été le premier modèle de correspondance sémantique. Il affecte un plongement à chaque entité et une matrice à chaque type de relation, qui représente l'interaction entre deux entités. Le score d'un fait (sujet, relation, objet) est défini par une fonction bilinéaire. Comme TransE, RESCAL a été étendu plusieurs fois [Nickel et al., 2015, Liu et al., 2017].

A.3.1.3 Correspondance par réseaux de neurones

Plusieurs modèles utilisent les réseaux de neurones pour calculer la correspondance entre des entités: NTN [Socher et al., 2013], MLP [Dong et al., 2014], RDFDNN [Murata et al., 2017], Graph Convolutional Network (GCN) [Kipf and Welling, 2016].

A.3.1.4 Marches à travers le graphe

Les méthodes suivantes utilisent des marches aléatoires pour générer les caractéristiques des noeuds. Ces techniques sont moins centrées sur les entités et explorent également le voisinage i.e. le contexte. Les modèles les plus représentatifs sont node2vec [Grover and Leskovec, 2016], GraphSAGE [Hamilton et al., 2017] et LINE [Tang et al., 2015]

A.3.2 Graphes avec attributs

Dans le cadre de la SAM, nous avons besoin d'utiliser les attributs des entités tels que la position ou la vitesse pour inférer de nouvelles relations. Puisque les graphes avec attributs sont plus adaptés à nos besoins, voici comment ils sont traités dans la littérature.

L'approche de Lin et al. [Lin et al., 2016] peut prédire les valeurs d'attributs discrets et trouver des corrélations entre eux. En revanche, les attributs ne sont pas inclus lors de l'apprentissage des relations, et les relations ne sont pas incluses dans l'apprentissage des attributs. Tay et al. [Tay et al., 2017] proposent un modèle qui apprend de manière jointe les relations et les attributs grâce à un réseau de neurones, et prédit des valeurs continues d'attributs avec une tâche de régression. attri2vec [Zhang et al., 2019] étend le modèle Deepwalk [Perozzi et al., 2014] (proche de node2vec [Grover and Leskovec, 2016]) pour capturer la structure du graphe ainsi que le contenu des noeuds. Co-embedding Attributed Networks (CAN) [Meng et al., 2019] produit un plongement pour chaque événement relationnel et attributif en utilisant la moyenne et la variance d'une distribution gaussienne grâce à un autoencodeur variationnel. Cependant, aucun de ces modèles ne peut traiter les données temporelles.

A.3.3 Graphes dynamiques

Dans un contexte dynamique, chaque noeud est représenté par une série temporelle de vecteurs qui représentent son évolution. Avec l'essor des jeux de données temporels (comme GDELT [Leblay and Chekol, 2018] et ICEWS [Boschee et al., 2015]), l'étude des graphes dynamiques est un sujet émergent et possède moins de contributions que les graphes statiques, mais certaines avancées ont déjà été faites. Leblay et al. [Leblay and Chekol, 2018] prédisent la validité d'une date pour des arcs non-annotés. Esteban et al. [Esteban et al., 2016] mettent à jour le graphe de connaissance en utilisant un graphe dédié aux événements, et Trivedi et al. [Trivedi et al., 2017] étendent le modèle bilinéaire (RESCAL) grâce à un réseau LSTM pour apprendre l'évolution non-linéaire des entités. Jiang et al. [Jiang et al., 2016] incorporent la durée de validité des faits en utilisant un modèle d'inférence conjoint tenant compte du temps, basé sur la progammation linéaire en nombres entiers. Les réseaux auto-attentifs on été utilisés par Sankar et al. [Sankar et al., 2018]. Goyal et al. [Goyal et al., 2018] utilisent des autoencodeurs profonds pour traiter des captures instantanées de graphes et apprendre les dépendances temporelles. Continuous-Time Dynamic Network Embeddings [Nguyen et al., 2018] est une méthode de plongements sur les graphes dynamiques et homogènes utilisant des marches aléatoires temporelles en temps continu.

A.3.4 Graphes dynamiques avec attributs

Jusqu'à présent nous avons examiné les techniques qui traitent des graphes statiques relationnels et attributifs ainsi que des graphes relationnels dynamiques. Cependant, les graphes dynamiques attributifs sont ceux qui correspondent le mieux à la SAM. Ce domaine est relativement nouveau et peu de modèles existent.

DANE [Li et al., 2017] est une technique de plongement qui utilise la structure du graphe (relations) et les attributs des noeuds. Le Co-embedding model for Dynamic Attributed Networks (CDAN) [Meng et al., 2020] est similaire à DANE dans sa façon de traiter les relations et les attributs. [Li et al., 2018] proposent un modèle de streaming (SLIDE) sur les graphes dynamiques avec attributs en utilisant une matrice d'esquisse (sketching matrix) qui résume les liens et attributs observés.

A.4 Formalisation du problème

Nous formalisons maintenant le problème de prédiction de liens dans un graphe dynamique hétérogène avec attributs et présentons l'achitecture globale de notre modèle, Joint Evolution.

Les problèmes d'intérêt sont les suivants:

- étant donné un quadruplet attributif $q = (e, a, v, t)$ observé à l'instant t et le plongement précédent $\boldsymbol{\alpha}_e^{t-}$, calculer le nouveau plongement attributif de e, $\boldsymbol{\alpha}_e^t$;
- étant donné le quadruplet relationnel $q = (s, r, o, t)$ observé à l'instant t et les plongements précédents $\boldsymbol{\rho}_s^{t-}$ et $\boldsymbol{\rho}_o^{t-}$, calculer les nouveaux plongements relationnels $\boldsymbol{\rho}_s^t$ et $\boldsymbol{\rho}_o^t$ de s et o;
- à l'instant t, prédire la valeur d'un attribut a pour une entité e à un instant $t' \geq t$ présent ou futur, c'est-à-dire, la valeur $v \in \mathrm{D}_a$ telle que le quadruplet $q = (e, a, v, t')$ a de fortes chances d'appartenir à la partie attributive du graphe;
- de façon similaire, à chaque instant t, avec deux éléments de s, r, o et un instant $t' \geq t$, prédire la valeur la plus probable pour le troisième élément.

A.4.1 Le modèle idéal de graphe pour les applications maritimes

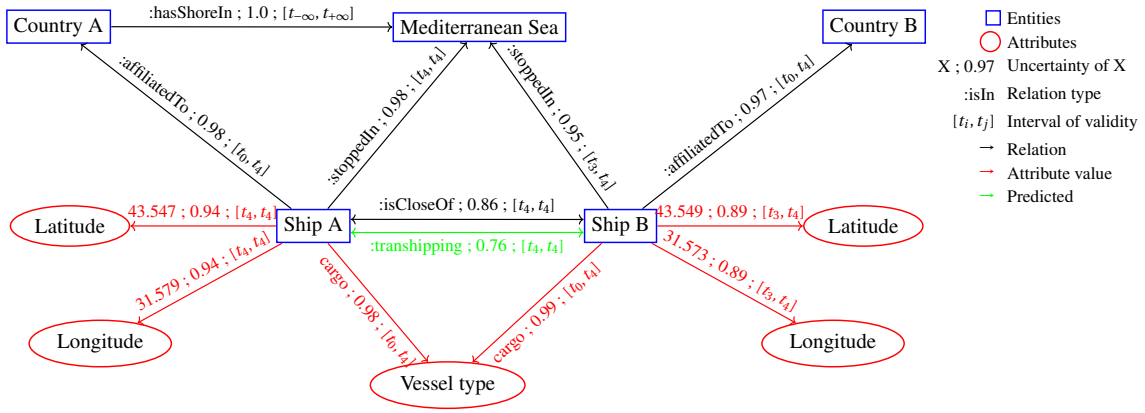


Figure A.4: Exemple de graphe à l'instant $t^* = t_4$

Pour répondre aux défis posés par la SAM, nous proposons la structure de graphe illutrée en Figure A.4. Les événements en rouge représentent la partie attributive et les autres la partie relationnelle. Le navire A et le navire B sont de différents pays et se déplacent dans la mer Méditerranée. Ils ont des attributs statiques (type de bateau) qui les définissent comme des cargos ainsi que des attributs dynamiques (latitude et longitude) qui révèlent leurs positions. Avant t^*, les deux navires se déplaçaient dans la mer Méditerranée et avaient différentes positions mais maintenant (à t^*), ils sont à l'arrêt et proches l'un de l'autre. Une prédiction de lien possible est que les deux bateaux sont en train d'effectuer un échange de marchandises. Notez que la relation :stoppedIn peut être déduite de l'attribut *speed* (vitesse) qui est alors proche de 0, non représenté ici pour plus de clarté.

Si les relations telle que le pays A ayant une côte sur la mer Méditerranée est sûre à 100 %, certaines sont moins évidentes: la position d'un navire est seulement sûre à 89 % car le signal a été capté par un satellite dans une zone avec un trafic important. De plus, l'incertitude des relations prédites (e.g. l'échange de marchandises) dépend également de l'incertitude des informations à la source. Enfin, pour prédire l'échange à temps, le modèle doit être mis à jour avec les causes de l'événement dès que possible, d'où le besoin d'une prédiction de liens en flux constant. Idéalement, le modèle doit pouvoir gérer les données discrètes (port d'arrivée, drapeau...) et les données continues (positions, date, vitesse du vent...).

A.5 Joint Evolution

A.5.1 Le framework

Joint Evolution est composé de deux réseaux de neurones spécialisés *AttrEmb* et *RelEmb*. L'architecture globale du système est illustrée en Figure A.5.

L'entrée du modèle est composée d'événements relationnels et attributifs arrivant dans l'ordre de leur timestamp. Chaque réseau est composé d'un encodeur et d'un décodeur. L'encodeur met à jour le plongement de chaque entité impliquée dans l'événement traité afin de saisir le comportement de ces entités en fonction de leurs événements passés. Quand une prédiction est demandée, ces plongements sont donnés au décodeur qui va utiliser ces informations pour prédire la prochaine valeur d'un attribut (AttrNet) ou le sujet (ou l'objet) d'un événement (RelNet).

A.5.2 AttrNet

Le réseau *AttrNet* encode les événements attributifs pour faire des prédictions sur les valeurs des attributs.

A.5.2.1 Encodeur

Lorsqu'un nouvel événement attributif $q = (e, a, v, t)$ est observé, l'encodeur d'*AttrEmb* prend en entrée q ainsi que le plongement actuel de e, $\boldsymbol{\alpha}_e^-$, et produit un nouveau plongement $\boldsymbol{\alpha}_e^t$. Pour prendre en compte la séquence d'événements sur les attributs, nous utilisons un réseau de neurones GRU comme encodeur [Cho et al., 2014b]. Etant donné $q = (e, a, v, t)$ et $\boldsymbol{\alpha}_e^{t-}$, l'encodeur est défini comme suit:

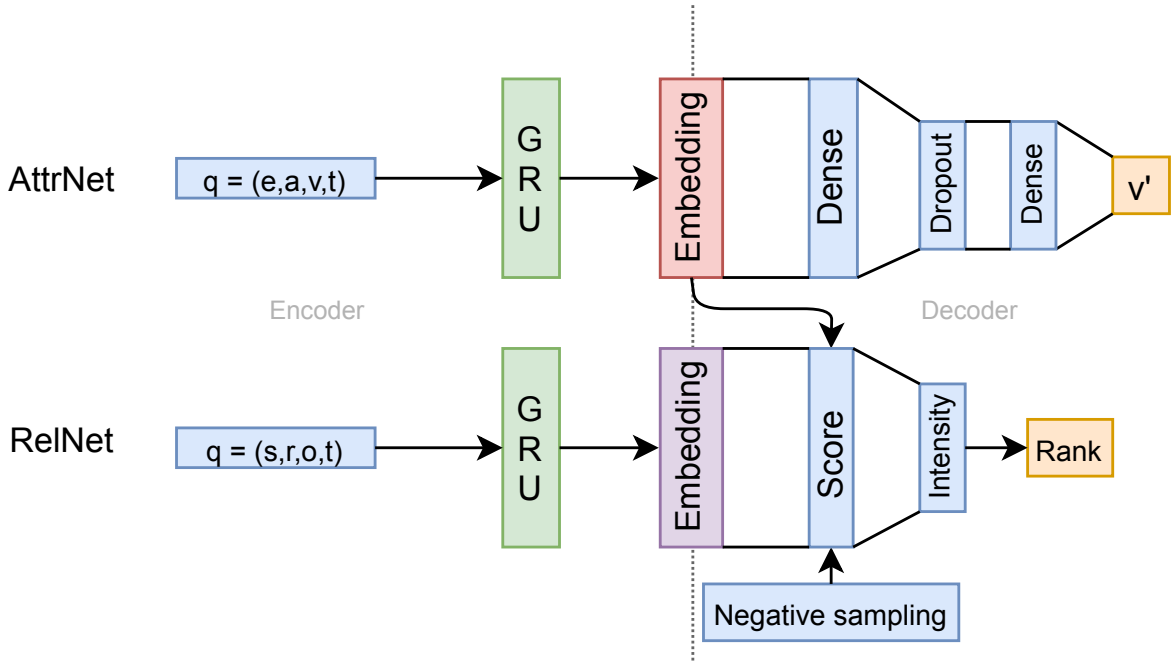


Figure A.5: Joint Evolution

$$\boldsymbol{z}_t = \sigma((t-\tau^-)\boldsymbol{w}_t^z + v\boldsymbol{w}_z^a + \mathbf{U}_z\boldsymbol{\alpha}_e^{t-} + \boldsymbol{b}_z) \quad \text{(A.1)}$$

$$\boldsymbol{r}_t = \sigma((t-\tau^-)\boldsymbol{w}_t^r + v\boldsymbol{w}_r^a + \mathbf{U}_r\boldsymbol{\alpha}_e^{t-}) \quad \text{(A.2)}$$

$$\boldsymbol{h}_t' = tanh(v\boldsymbol{w}_t^a + \boldsymbol{r}_t \odot \mathbf{U}_t\boldsymbol{\alpha}_e^{t-}) \quad \text{(A.3)}$$

$$\boldsymbol{\alpha}_e^t = \boldsymbol{z}_t \odot \boldsymbol{\alpha}_e^- + (\mathbf{1}-\boldsymbol{z}_t) \odot \boldsymbol{h}_t') \quad \text{(A.4)}$$

Autrement dit, le nouveau plongement pour e est obtenu en fonction du décalage temporel (avec un paramètre w_t partagé par tous les attributs), de la nouvelle valeur observée (avec un paramètre spécifique au type d'attribut) et du plongement précédent (également avec un paramètre lié au type d'attribut). La Figure A.6 illustre les Equations A.1 à A.4.

Ici, $\boldsymbol{z}_t$ est appelé l'*update gate*, $\boldsymbol{r}_t$ la *reset gate*, $\boldsymbol{h}_t'$ la *mémoire courante*, et σ est la fonction sigmoïde; $v \in \mathrm{D}_a$ est la valeur scalaire de l'attribut a, $\odot$ est le produit d'Hadamard, et σ et $\boldsymbol{z}_t \mapsto \mathbf{1}-\boldsymbol{z}_t$ sont appliqués coordonnée par coordonnée.

Les vecteurs $\boldsymbol{w}_z^a, \boldsymbol{w}_r^a, \boldsymbol{w}_t^a \in \mathbb{R}^m$ sont des paramètres du réseau qui dépendent du type d'attribut. Les matrices $\mathbf{U}_z, \mathbf{U}_r, \mathbf{U}_t \in \mathbb{R}^{m\times m}$ sont des paramètres qui indiquent quelle partie du plongement doit être gardée dans les *gates*. Enfin, $\boldsymbol{b}_z \in \mathbb{R}^m$ est la couche de biais de l'*update gate*.

A.5.2.2 Décodeur

Le décodeur d'AttNet est défini comme suit:

$$hidden(\boldsymbol{\alpha}_e^t) = tanh(\boldsymbol{w}_x\boldsymbol{\alpha}_e^t + b_x)) \quad \text{(A.5)}$$

$$v = tanh((t'-t)w_t + \boldsymbol{w}_a \cdot hidden(\boldsymbol{\alpha}_e^t) + b_a) \quad \text{(A.6)}$$

La Figure A.6 (à droite) illustre cette étape.

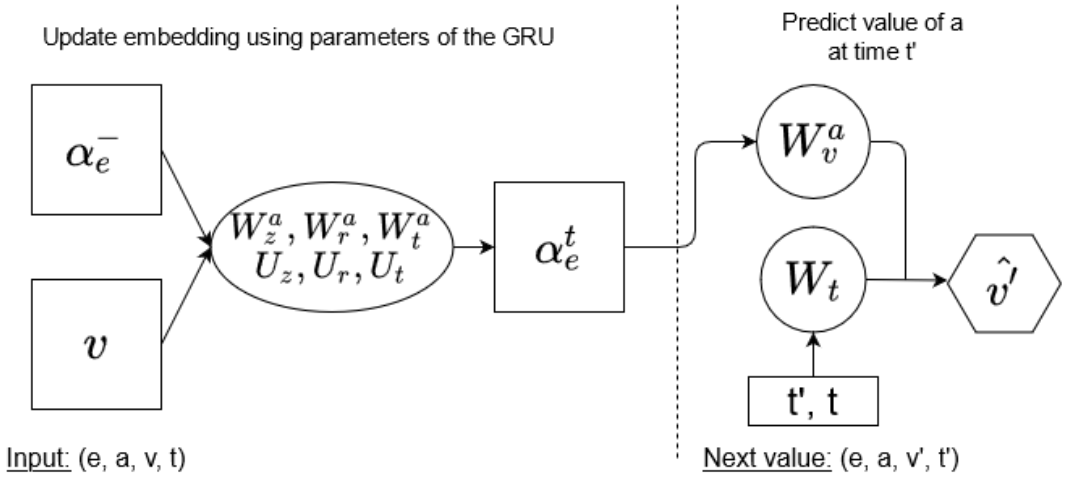


Figure A.6: Inférence avec AttrEmb

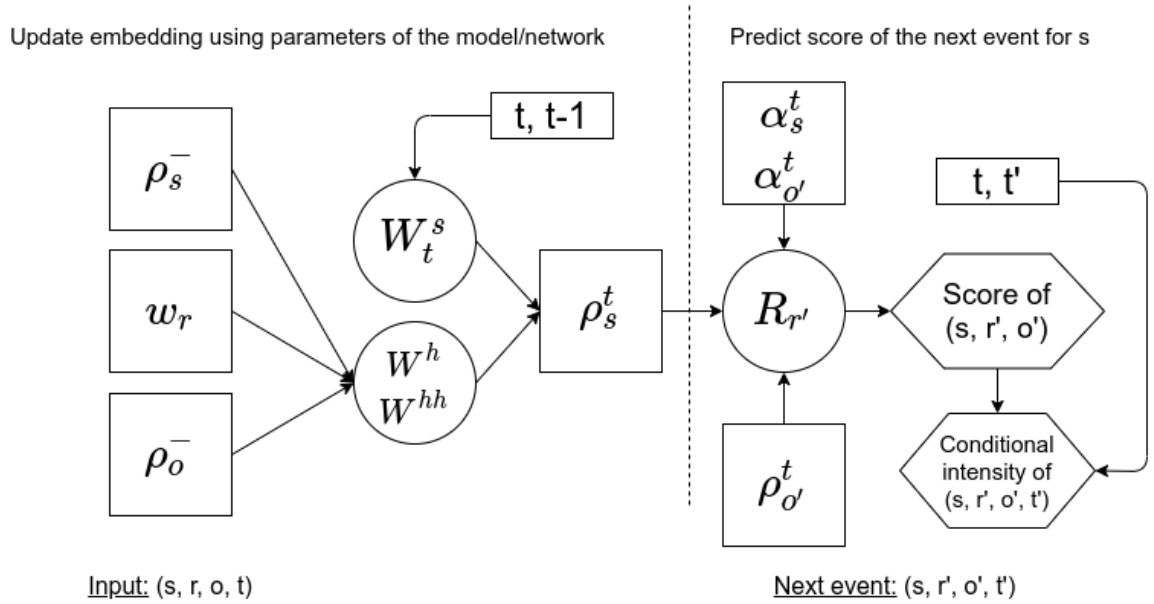


Figure A.7: Inférence avec RelEmb pour s. Le même processus est appliqué à o en suivant les Equations A.11, A.12 et A.13.

Ici, $w_t \in \mathbb{R}$ est un paramètre qui module la prédiction suivant le temps écoulé depuis le dernier changement de valeur, $\boldsymbol{w}_a \in \mathbb{R}^m$ est un paramètre spécifique au type d'attribut et $b_a \in \mathbb{R}$ est le biais.

A.5.3 RelNet

A.5.3.1 Encodeur

Lorsqu'un nouvel événement $q = (s, r, o, t)$ est observé, l'encodeur de RelEmb prend en entrée q et les plongements courants de s et o, $\boldsymbol{\rho}_s^{t-}$ et $\boldsymbol{\rho}_o^{t-}$, et produit leurs nouveaux plongements $\boldsymbol{\rho}_s^t$ and $\boldsymbol{\rho}_o^t$.

La mise à jour se fait comme suit:

$$\boldsymbol{v}_s = tanh(\mathbf{W}_s^h.[\boldsymbol{\rho}_o^{t-} \oplus \boldsymbol{w}_r]) \tag{A.7}$$

$$\boldsymbol{z}_t^s = \sigma((t-\tau_s^-)\boldsymbol{w}_z^s + \mathbf{W}_z^v.\boldsymbol{v}_s + \mathbf{W}_z^{hh}.\boldsymbol{\rho}_s^- + \boldsymbol{b}_z) \tag{A.8}$$

$$\boldsymbol{r}_t^s = \sigma((t-\tau_s^-)\boldsymbol{w}_r^s + \mathbf{W}_r^v.\boldsymbol{v}_s + \mathbf{W}_r^{hh}.\boldsymbol{\rho}_s^{t-}) \tag{A.9}$$

$$\boldsymbol{h}_t^s = tanh(\mathbf{W}_t^v.\boldsymbol{v}_s + \boldsymbol{r}_t^s \odot \mathbf{W}_t.\boldsymbol{\rho}_s^{t-}) \tag{A.10}$$

$$\boldsymbol{\rho}_s^t = \boldsymbol{z}_t^s \odot \boldsymbol{\rho}_s^- + (1-\boldsymbol{z}_t^s) \odot \boldsymbol{h}_t^s \tag{A.11}$$

A.5.3.2 Décodeur

Nous calculons le score relationnel bilinéaire $g_r^{s',o'}$ (Equation A.12) du plongement qui vient d'être mis à jour. Nous utilisons pour cela un *challenge* $q' = (s', r', o', t')$ qui implique une des deux entités de q.

$$g_{r'}^{s',o'} = (\boldsymbol{\rho}_{s'}^t \oplus \boldsymbol{\alpha}_{s'}^t)^{\mathrm{T}}.\,\mathbf{R}_{r'}\,.\,(\boldsymbol{\rho}_{o'}^t \oplus \boldsymbol{\alpha}_{o'}^t) \tag{A.12}$$

où $\mathbf{R}_{r'} \in \mathbb{R}^{(n+m)\times(n+m)}$ est une matrice spécifique au type de relation apprise durant l'entraînement.

Ce score est ensuite utilisé pour calculer une fonction conditionnelle d'intensité λ (Equation A.13), qui sera utilisée pour classer les événements possibles durant la phase de prédiction (plus λ est élevé, plus le quadruplet (s, r, o, t) est probable).

$$\lambda_r^{s,o}(t'|t) = w_t^r(t-t')(\log(1+\exp(g_r^{s,o})) \tag{A.13}$$

Ceci est une adaptation de la fonction softplus décrite par [Trivedi et al., 2019] et [Mei and Eisner, 2017]. En utilisant la représentation de l'événement courant, Joint Evolution va ainsi apprendre quel événement prédire ensuite. Le delta de temps $(t - t')$ illustre le processus de Rayleigh utilisé dans Know-Evolve: plus le temps passe sans qu'un événement ne se produise, plus il y a de chances qu'il arrive.

A.5.4 Utilisation

A.5.4.1 Mise à jour

Durant l'entraînement, les réseaux apprennent comment construire des plongements qui peuvent être utilisés pour faire des prédictions. Une fois entraînés, tous les paramètres sont gelés, mais les plongements sont toujours mis à jour au fur et à mesure que les événements arrivent. En utilisant les plongements des entités d'un événement (à l'instant t) dans l'encodeur, la connaissance obtenue est directement incluse dans $\boldsymbol{\rho}_e$ et $\boldsymbol{\alpha}_e$ pour que le modèle ait toutes les informations nécessaires pour faire une prédiction à un moment $t' \geq t$. Le modèle peut ainsi recevoir un flux de données, mettre continuellement à jour les plongements et être prêt à prédire les prochains événements à n'importe quel moment.

A.5.4.2 Prédiction

La prédiction de la valeur suivante d'un attribut est faite directement en sortie d'AttrNet: c'est une valeur scalaire calculée en utilisant $\boldsymbol{\alpha}_e$ (Equation A.6). Pour prédire le prochain événement relationnel pour une entité, nous fixons le moment ainsi que deux de ses trois composants, e.g. $(s, r, ?, t)$ et nous cherchons l'objet le plus probable pour la relation r avec le sujet s à l'instant t. Nous remplaçons donc cet objet par toutes les entités possibles,

calculons le score bilinéaire et l'intensité conditionnelle des événements obtenus avec les Equations A.12 et A.13, et les classons. Les événements les plus probables sont ceux ayant l'intensité conditionnelle la plus élevée.

A.5.5 Fonctions de coût

La fonction de coût pour AttrNet est l'erreur quadratique moyenne, commune pour les tâches de régression:

$$\mathrm{L}oss = \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} (v(t_{p+1})_i - v(t_{p+1})_i^*)^2 \tag{A.14}$$

Pour RelNet, nous cherchons à avoir la plus grande intensité conditionnelle pour le *challenge* comparé à tous les événements possibles. Nous calculons donc l'intensité conditionnelle de tous les événements "négatifs" (qui ne sont pas arrivés) et en gardons la valeur maximale. Nous cherchons ensuite à minimiser la différence entre cette valeur et l'intensité conditionnelle du *challenge*, i.e. nous faisons en sorte que celle du *challenge* soit plus élevée que toutes celles des événements négatifs (Equation A.15.

$$\mathrm{L} = (max(\lambda^s) - \lambda^{s'}) + (max(\lambda^o) - \lambda^{o'}) \tag{A.15}$$

A.6 Expérimentations

Nos travaux se concentrent sur la SAM et nous testons donc Joint Evolution sur un jeu de données maritimes. Cette expérimentation est une preuve de concept sur des données réelles qui vise à démontrer l'efficacité des attributs dynamiques dans la prédiction de liens. Cependant, pour tester Joint Evolution dans un contexte plus académique et pouvoir le comparer à d'autres méthodes, nous l'appliquons également à DBLP, un graphe de citations fréquemment utilisé en analyse de graphes.

A.6.1 datAcron

Ce premier jeu de données contient des données maritimes, l'objectif étant de prédire les futures interactions entre navires. Nous utilisons le jeu de données fourni par [Ray et al., 2019] et faisant partie du projet datAcron. Il contient six mois de données AIS d'octobre 2015 à mars 2016 sur la mer Celtique, le nord de l'océan Atlantique, le channel anglais et la baie de Biscay. Nous utilisons également le jeu de données Composite Maritime Events [Pitsikalis and Artikis, 2019] qui contient des événements extraits de [Ray et al., 2019] grâce à un moteur de règles.

A.6.2 DBLP

Pour tester Joint Evolution, nous utilisons également Digital Bibliography & Library Project (DBLP) pour apprendre et prédire des relations de citation et de co-auteur entre deux auteurs. Plus spécifiquement, nous utilisons DBLP v12 d'AMiner [Tang et al., 2008]. Il contient 4.894.081 articles et 45.564.149 relations de citation au 09-04-2020. Le contenu détaillé du jeu de données peut être trouvé sur son site web[1]. Nous ne gardons que les publications

[1] https://www.aminer.cn/citation

d'intelligence artificielle entre 1995 et 2000 et seulement les auteurs ayant publié au moins cinq fois.

Les caractéristiques des deux jeux de données sont récapitulées en Figure A.1.

Dataset	Entités	Rel types	Attr types	Nb Rel	Nb Attr
datAcron	363	3	3	764	4,596,204
DBLP	6,147	2	2	261,324	139,152

Table A.1: Caractéristiques de DBLP et datAcron

Nous testons Joint Evolution sur ces deux jeux de données. datAcron a peu d'entités et peu de relations et servira à vérifier si son grand nombre d'attributs peut compenser ce manque. DBLP est plus équilibré mais moins dynamique car à l'échelle des années et non des secondes.

A.7 Comparaison des modèles

Nous comparons Joint Evolution avec deux modèles implémentés avec la bibliothèque StellarGraph [Data61, 2018]. Elle propose des modèles de l'état de l'art sur étagère et peut traiter toutes sortes de graphes: homogènes, hétérogènes, graphes avec ou sans attributs, graphes avec arcs pondérés.

A.7.1 node2vec

node2vec [Grover and Leskovec, 2016] est une méthode de plongement sur les graphes statiques et homogènes que nous avons évoquée dans l'état de l'art. Elle a l'avantage d'être très utilisée, et nous permet d'évaluer l'avantage qu'ont les méthodes dynamiques sur les méthodes statiques pour traiter des données temporelles.

A.7.2 CTDNE

CTDNE [Nguyen et al., 2018] est une méthode de plongement sur les graphes dynamiques et homogènes, utilisant des marches aléatoires temporelles. Elle nous permet de comparer Joint Evolution à une méthode dynamique.

A.8 Comparaison de scores

Nous comparons maintenant les résultats de Joint Evolution, Joint Evolution relations seules, node2vec et CTDNE.

A.8.1 datAcron

Nous rapportons les résultats de datAcron sur la Figure A.8 et la Table A.2. La meilleure méthode est ici node2vec, suivie de près par Joint Evolution sans les attributs. Cependant, toutes les autres méthodes ont des résultats similaires sur toutes les métriques sauf pour le Hits@1 qui souffre de l'instabilité causée par le faible échantillon de données. Nous remarquons aussi des résultats plus faibles sur le Hits@10 pour Joint Evolution.

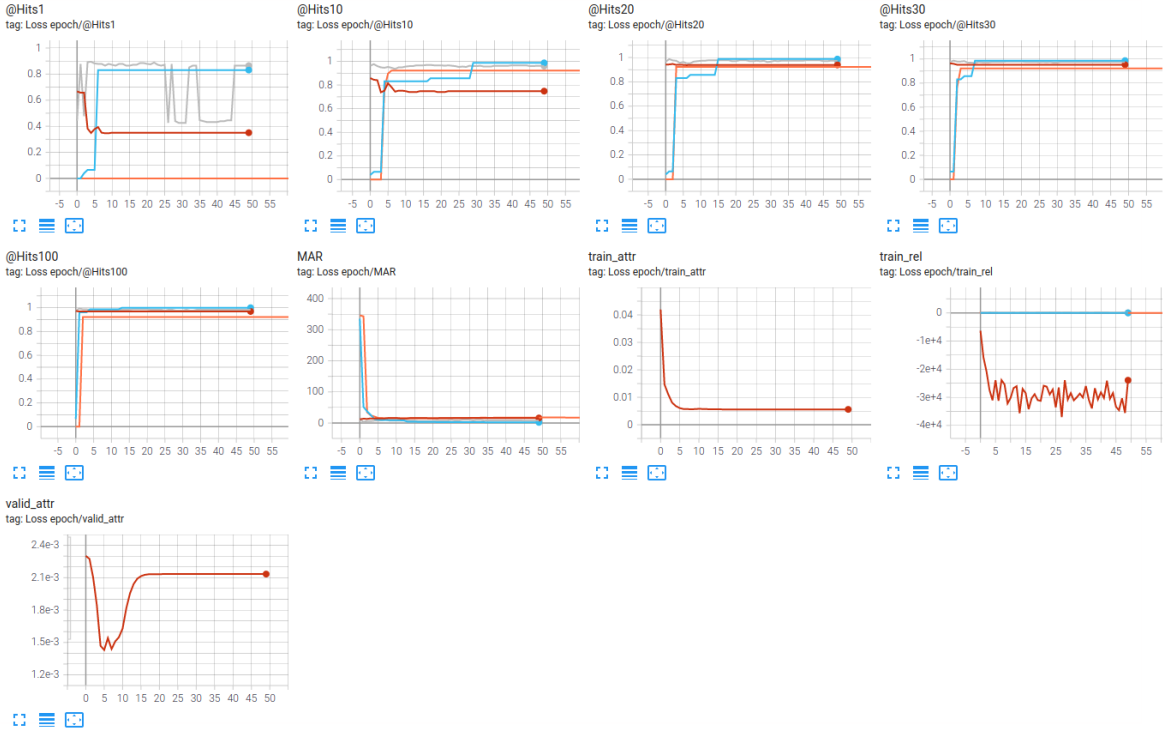


Figure A.8: Comparaison de Joint Evolution (rouge), Joint Evolution sans attributs (gris), CTDNE (orange) et node2vec (bleu) pour datAcron

	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100	MAR
node2vec	0.831	0.987	0.987	0.987	1	2.117
CTDNE	0	0.9221	0.9221	0.9221	0.9221	15.3
JE (RelNet)	0.8784	0.9662	0.9764	0.9764	0.9831	7.58
JE	0.3504	0.748	0.939	0.9528	0.9685	16.21

Table A.2: Résultats de Hits@ et MAR sur datAcron

A.8.2 DBLP

Nous rapportons les résultats pour DBLP sur la Figure A.9 et la Table A.3. Joint Evolution sans attributs donne globalement les meilleurs résultats, suivi par Joint Evolution, puis CTDNE puis node2vec. CTDNE est proche de Joint Evolution sur le Hits@10 et le Hits@30 et le surpasse même sur le Hits@20, mais le grand écart sur le MAR en faveur de Joint Evolution le rend plus efficace. node2vec, auparavant capable de tenir tête aux autres méthodes, échoue complètement à prédire des liens sur DBLP.

	Hits@1	Hits@10	Hits@20	Hits@30	Hits@100	MAR
node2vec	0	$1.145e^{-4}$	$7.252e^{-4}$	$1.603e^{-3}$	0.0335	1,676
CTDNE	$7.5e^{-4}$	0.0339	0.0826	0.1545	0.3061	822.6
JE (RelNet)	$6.646e^{-3}$	0.0629	0.0958	0.2187	0.488	517.6
JE	$2.232e^{-3}$	0.035	0.061	0.1537	0.4379	542.1

Table A.3: Hits@ and MAR results on DBLP

Puisque Joint Evolution avec seulement RelNet a toujours de meilleurs résultats que Joint Evolution classique, notre hypothèse que les attributs améliorent les résultats sur la prédic-

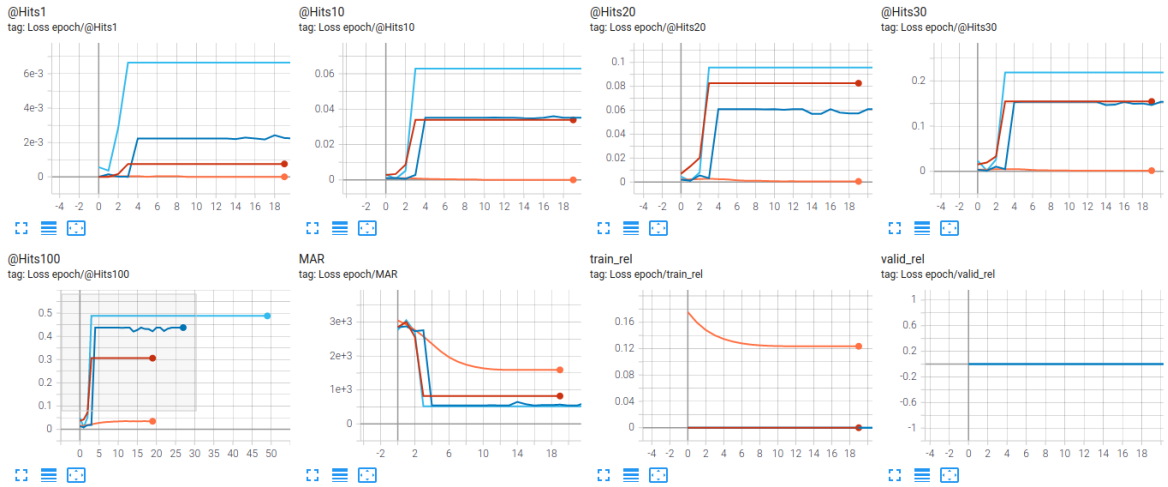


Figure A.9: Comparaison de Joint Evolution (bleu foncé), Joint Evolution sans attributs (bleu), CTDNE (rouge) et node2vec (orange) pour DBLP

tion de liens est rejetée. Malheureusement, l'ajout d'attributs dans la tâche de prédiction dégrade les résultats en comparaison d'une approche purement relationnelle. Cependant, Joint Evolution sans les attributs reste meilleur que CTDNE et node2vec sur ces tâches et possède des capacités de streaming avec des plongements qui peuvent être mis à jour au cours du temps, ce qui est un critère important pour la SAM.

A.9 Bilan et perspectives

A.9.1 Bilan

Puisque cette thèse se déroule dans le cadre d'un programme CIFRE entre le GREYC et Airbus Defence and Space, nous avons commencé par présenter le problème industriel que cette thèse vise à résoudre. L'objectif principal est d'améliorer la *Situational Awareness* et de fournir une aide à la décision aux opérateurs de surveillance qui doivent parcourir de grande quantité de données hard et soft. Pour améliorer la compréhension de la situation, nous analysons les données historiques pour prédire les prochains événements et lever des alertes. Pour ce faire, nous avions besoin de créer une représentation des situations maritimes passées et présentes et avons choisi d'utiliser les graphes de connaissance. Leur capacité de description sémantique couplée à la modélisation des interactions entre entités en fait un outil adapté. Nous avons ensuite commencé par faire l'état de l'art sur deux principaux sujets: comment améliorer la *Situational Awareness* et comment prédire des liens dans des graphes dynamiques avec attributs. Sur le premier, nous avons trouvé beaucoup d'approches sur la génération de routes et la détection d'anomalies. Certaines traitent des événements, mais la plupart sont des moteurs de règles. L'utilisation de graphes et de prédictions d'événements pour la *Situational Awareness* maritime n'est donc pas un domaine très étudié et c'est sur ce point que nous positionnons nos travaux. Sur le second sujet, nous avons remarqué qu'il y a beaucoup de contributions sur les graphes statiques avec et sans attributs ainsi que sur les graphes dynamiques, mais peu sur les graphes dynamiques avec attributs. Nous nous sommes donc inspirés des méthodes de graphes dynamiques pour les adapter à l'inclusion d'attributs, en faisant l'hypothèse que les attributs peuvent améliorer les résultats de la prédiction de liens.

Sur ce constat, nous proposons Joint Evolution, un système encodeur-décodeur en deux parties: une qui apprend des plongements à partir des changements de valeur d'attributs pour prédire les suivants, et l'autre qui apprend des plongements relationnels pour prédire de nouvelles relations en utilisant les deux types de plongements. Nous aovns conçu un pipeline de pré-traitement pour créer des séquences d'événements indépendants qui respectent l'ordre dans lequel les événements s'appliquent à une entité. Nous avons ensuite fait le choix d'utiliser des GRU pour traiter ces séquences et générer deux plongements pour chaque entité: celui sur les attributs et celui sur les relations. Une fois que le modèle est entraîné à mettre à jour les plongements qui peuvent ensuite être utilisés pour prédire le prochain événement, les plongements peuvent être mis à jour à la volée lorsque de nouveaux événements sont reçus en utilisant la partie encodeur de Joint Evolution. Quand il est nécessaire de faire une prédiction, nous utilisons uniquement la partie décodeur pour répondre à la question posée, généralement sous la forme d'un quadruplet tel que (sujet, relation, ?, timestamp). Le décodeur va ensuite trouver l'objet qui correspond le mieux au quadruplet en calculant un score pour chaque possibilité.

Nous avons testé joint Evolution sur deux jeux de données, datAcron et DBLP. datAcron est un ensemble d'événements et de messages AIS provenant de bateaux dans la baie de Biscay, proche de Brest, et DBLP est un réseau de citation qui relie des auteurs et des articles avec des liens d'auteurs et de co-auteurs. La principale différence entre les deux jeux de données repose dans le nombre d'événements et le type des attributs, ce qui nous donne deux contextes d'expérimentation différents. Nous avons comparé Joint Evolution avec deux autres méthodes: node2vec, une approche connue de *random walk* pour représenter les graphes statiques, et CTDNE, sa version dynamique. Nous avons également testé une version de Joint Evolution qui utilise seulement les relations pour évaluer l'impact des attributs sur le résultat. Au final, le modèle qui donne les meilleurs résultats est Joint Evolution sans les attributs, ce qui rejette notre hyptohèse que les attributs peuvent améliorer la prédiction de liens. Cependant, Joint Evolution a tout de même de meilleurs résultats que node2vec et CTDNE dans les deux cas et peut traiter des graphes hétérogènes en streaming. Nous espérons toujours que l'ajout d'informations peut être utile à l'apprentissage et expliquons les résultats obtenus avec plusieurs facteurs, notamment le nombre élevés de combinaisons pour les hyperparamètres et la structure des jeux de données.

A.9.2 Perspectives

Avec ces résultats, nous décrivons maintenant les limites de Joint Evolution et les travaux futurs envisagés.

A.9.2.1 Limites

Nous définissons cinq critères pour la *Situational Awareness* maritime: suivre l'évolution des entités, détecter les événements et menaces, pouvoir traiter un flux de données, gérer l'incertitude et rendre la sortie du modèle explicable. Malheureusement, nous n'avons pas eu le temps de travailler sur l'incertitude et l'explicabilité, qui seront traités lors de travaux futurs.

Sur la prédiction de liens, Joint Evolution est limité à la prédiction de sujets et d'objets. Nous ne pouvons pas prédire *quand* un événement va arriver, seulement quelle entité va interagir avec une autre à un moment donné. Nous nous sommes concentrés sur ce cas car il s'agit de la principale utilisation dans notre contexte, mais pouvoir prédire la date d'un

événement pourrait également être utile. Il nous est également impossible de prédire le type de relation: entraîner Joint Evolution à prédire un type de relation est différent car la notion de challenge n'est pas la même. Nous prenons en compte le type de relation en utilisant des paramètres spécifiques dans le réseau, mais cela ne nous donne pas la possibilité de les prédire. Pour prédire la date et le type d'un événement, un nouveau cadre d'entraînement doit être défini.

Un autre problème rencontré est la mise à jour d'une entité qui ne s'effectue que lorsqu'un événement la concernant est rencontré. Cependant, d'autres événements peuvent influer sur ces entités sans pour autant l'inclure directement. Par exemple, un conflit entre deux pays (entités c et c') ne peut pas influencer la relation entre un navire du pays c (entité v) et un navire du pays c' (entité v') puisque les plongements des entités v et v' ne sont pas mis à jour lorsque le conflit survient. Les méthodes qui prennent en compte le voisinage et le contexte dans les graphes de connaissance pourraient combler ce vide [Jin et al., 2019, Heidari and Papagelis, 2019].

La dernière limite qui mérite d'être mentionnée est la nature des attributs utilisés. Les attributs de datAcron sont des scalaires alors que DBLP utilise des vecteurs. Cela requiert une adaptation de Joint Evolution suivant le cadre. Si un jeu de données devait contenir les deux types d'attributs, il faudrait trouver un moyen des traiter les deux types, par exemple en désactivant AttrNet lorsqu'un attribut vectoriel est rencontré.

A.9.2.2 Perspectives à court terme

Nos perspectives à court terme concernent principalement les jeux de données et les expérimentations. Pour évaluer dans une plus large mesure l'effet des attributs, nous souhaitons entraîner Joint Evolution sur une version plus grande de datAcron, avec plus d'événements et d'entités. Nous souhaitons également repenser la façon dont les attributs de DBLP sont créés, d'une manière plus directe et facile à apprendre. Cela pourrait commencer par une dimension réduite pour l'encodeur de titres, l'ajout des conférences dans lesquelles les articles sont publiés et l'organisation auxquelles les auteurs appartiennent (bien que cette dernière implique de gérer les attributs statiques).

Trouver un jeu de données d'évaluation pour les graphes dynamiques avec attributs n'est pas facile puisqu'il n'y a pas de jeu de données de référence dans ce domaine comme l'est MNIST[2] pour l'analyse d'images. Certains graphes comme Epinions [Richardson et al., 2003], Freebase ou WordNet[3] sont souvent utilisés pour évaluer les graphes statiques et sont soit non-orientés soit homogènes. Les réseaux de citation tels que Citeseer [Bhattacharya and Getoor, 2007] et Cora [Getoor, 2005] sont soit statiques, soit sans attributs, et sont principalement utilisés pour de la classification d'articles. GDELT [Leetaru and Schrodt, 2013] et ICEWS [Boschee et al., 2015] sont des jeux de données dynamiques connus, mais il est difficile d'y trouver des attributs dynamiques. Enron [Klimt and Yang, 2004] pourrait être utilisé comme DBLP puisque c'est un réseau d'échange de mails, mais il ne possède que 150 entités et est homogène. De plus, la complexité du contenu des emails pourrait ne pas aider à prédire un échange de mail entre deux employés. la création d'un jeu de données de référence pour les graphes dynamiques avec attributs serait d'une grande aide dans ce domaine.

Notre dernière perspective à court terme est de faire plus d'expériences dans le cadre actuel, notamment à propos du jeu d'hyperparamètres. Avec plus de temps, nous pourrions

[2] `http://yann.lecun.com/exdb/mnist/`

[3] `https://everest.hds.utc.fr/doku.php?id=en:transe`

améliorer le choix des hyperparamètres et trouver le meilleur ensemble pour chaque méthode. Nous pourrions aussi évaluer avec plus de détail l'efficacité d'AttrNet, qui n'a pas été beaucoup testé en dehors de l'optimisation de sa fonction d'erreur.

A.9.2.3 Perspectives à long terme

Sur le plus long terme, l'objectif est de combler le vide laissé par l'incertitude et l'explicabilité dans les critères de *Situational Awareness*. Notre piste principale sur l'incertitude est l'utilisation des *Markov Logic Networks* et des *Probabilistic Soft Logics* [Chekol et al., 2017, Chekol and Stuckenschmidt, 2016]. Pour l'explicabilité, l'étude d'Adadi et Berrade [Adadi and Berrada, 2018] est une base solide pour trouver des solutions adaptées à notre cas.

Un autre ajout envisagé pour Joint Evolution est une vraie description sémantique des entités et de leurs interactions. Nous avons des types de relations, mais nous n'avons aucun moyen d'indiquer au modèle qu'une entité est un navire ou un port, ce qui peut jouer un rôle important dans le processus d'apprentissage. Savoir à l'avance le type d'une entité, les attributs qu'elle peut avoir et quels interactions sont possibles devrait rendre la création de nouveaux liens plus précise. Il faudrait alors trouver un moyen d'inclure des ontologies complètes dans le modèle pour mettre leurs capacités descriptives au bénéfice de l'apprentissage.

Bibliography

[Adadi and Berrada, 2018] Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160. 8, 60, 119, VII, XXI

[A.Holst and Ekman, 2005] A.Holst and Ekman, J. (2005). Anomaly detection in vessel motion. 17

[Alvarez et al., 2016] Alvarez, M., Arguedas, V. F., Gammieri, V., Mazzarella, F., Vespe, M., Aulicino, G., and Vollero, A. (2016). AIS event-based knowledge discovery for Maritime Situational Awareness. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1874–1880. 18

[Artikis et al., 2015] Artikis, A., Sergot, M., and Paliouras, G. (2015). An Event Calculus for Event Recognition. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):895–908. Conference Name: IEEE Transactions on Knowledge and Data Engineering. 18

[Belkin and Niyogi, 2002] Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in Neural Information Processing System*, 14. 53

[Bengio et al., 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166. 68

[Berger, 2018] Berger, D. (2018). Guidelines for the CISE data model. `http://emsa.europa.eu/cise-documentation/cise-data-model-1.5.3/model/guidelines/687507181.html`. 9

[Bhattacharya and Getoor, 2007] Bhattacharya, I. and Getoor, L. (2007). Collective entity resolution in relational data. *ACM Trans. Knowl. Discov. Data*, 1(1):5–es. 119, XX

[BigOceanData, 2016] BigOceanData (2016). The definitive ais handbook. On website: `https://www.marineinsight.com/wp-content/uploads/2016/11/AiS-Whitepaper.pdf`. 9, 10

[Bordes et al., 2014] Bordes, A., Glorot, X., Weston, J., and Bengio, Y. (2014). A semantic matching energy function for learning with multi-relational data: Application to word-sense disambiguation. *Machine Learning*, 94(2):233–259. 35

[Bordes et al., 2013] Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013). Translating Embeddings for Modeling Multi-relational Data. *NIPS*. 26, 34, 35, 40, VIII

[Boschee et al., 2015] Boschee, E., Lautenschlager, J., O'Brien, S., Shellman, S., Starz, J., and Ward, M. (2015). ICEWS Coded Event Data. 44, 119, IX, XX

[Cai et al., 2018] Cai, H., Zheng, V. W., and Chang, K. (2018). A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1. 35, VIII

[Cer et al., 2018] Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y.-H., Strope, B., and Kurzweil, R. (2018). Universal Sentence Encoder. *arXiv:1803.11175 [cs]*. arXiv: 1803.11175. 92

[Chandola et al., 2009] Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Comput. Surv.*, 41. 15

[Chekol et al., 2017] Chekol, M. W., Pirro, G., Schoenfisch, J., and Stuckenschmidt, H. (2017). Marrying Uncertainty and Time in Knowledge Graphs. page 7. 59, 119, XXI

[Chekol and Stuckenschmidt, 2016] Chekol, M. W. and Stuckenschmidt, H. (2016). Uncertain Temporal Knowledge Graphs. page 12. 59, 119, XXI

[Cho et al., 2014a] Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259. 69

[Cho et al., 2014b] Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014b). On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv:1409.1259 [cs, stat]*. arXiv: 1409.1259. 82, XI

[Chung et al., 2014] Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555. 69, 70

[Cox and Lewis, 1972] Cox, D. R. and Lewis, P. A. W. (1972). Multivariate point processes. In *Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Volume 3: Probability Theory*, pages 401–448, Berkeley, Calif. University of California Press. 46

[Dai et al., 2016] Dai, H., Wang, Y., Trivedi, R., and Song, L. (2016). Deep Coevolutionary Network: Embedding User and Item Features for Recommendation. *arXiv:1609.03675 [cs]*. arXiv: 1609.03675. 67

[Daley and Vere-Jones, 2008] Daley, D. J. and Vere-Jones, D. (2008). *An Introduction to the Theory of Point Processes: Volume II: General Theory and Structure*. Probability and Its Applications, An Introduction to the Theory of Point Processes. Springer-Verlag, New York, 2 edition. 47

[Data61, 2018] Data61, C. (2018). Stellargraph machine learning library. `https://github.com/stellargraph/stellargraph`. 96, XVI

[Divakaran and Mohan, 2019] Divakaran, A. and Mohan, A. (2019). Temporal Link Prediction: A Survey. *New Generation Computing*. 48

[Dong et al., 2014] Dong, X. L., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. (2014). Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. (Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining):601–610. 37, 43, IX

[Elman, 1990] Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science*, (14):179–211. 67

[Esteban et al., 2016] Esteban, C., Tresp, V., Yang, Y., Baier, S., and Krompaß, D. (2016). Predicting the Co-Evolution of Event and Knowledge Graphs. page 8. 44, 45, IX

[Fan et al., 2014] Fan, M., Zhou, Q., Chang, E., and Zheng, T. (2014). Transition-based knowledge graph embedding with relational mapping proper- ties. *Proc. 28th Pacific Asia Conf. Language, Inf., Comput.*, pages 328–337. 36

[Feng et al., 2016a] Feng, J., Huang, M., Wang, M., Zhou, M., Hao, Y., and Zhu, X. (2016a). Knowledge Graph Embedding by Flexible Translation. page 4. 36

[Feng et al., 2016b] Feng, J., Huang, M., and Yang, Y. (2016b). GAKE: Graph Aware Knowledge Embedding. page 11. 30

[Forti et al., 2020] Forti, N., Millefiori, L. M., Braca, P., and Willett, P. (2020). Prediction oof Vessel Trajectories From AIS Data Via Sequence-To-Sequence Recurrent Neural Networks. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8936–8940, Barcelona, Spain. IEEE. 18

[Galárraga et al., 2015] Galárraga, L., Teflioudi, C., Hose, K., and Suchanek, F. M. (2015). Fast rule mining in ontological knowledge bases with AMIE $$+$$ +. *The VLDB Journal*, 24(6):707–730. 53

[Galárraga et al., 2013] Galárraga, L. A., Teflioudi, C., Hose, K., and Suchanek, F. (2013). AMIE: association rule mining under incomplete evidence in ontological knowledge bases. pages 413–422. ACM Press. 53

[Garagic et al., 2009] Garagic, G., Rhodes, B., Bomberger, N., and Zandipour, M. (2009). Adaptive mixture-based neural network approach for higher-level fusion and automated behavior monitoring. 17

[García-Durán et al., 2014] García-Durán, A., Bordes, A., and Usunier, N. (2014). Effective Blending of Two and Three-way Interactions for Modeling Multi-relational Data. In Calders, T., Esposito, F., Hüllermeier, E., and Meo, R., editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8724, pages 434–449. Springer Berlin Heidelberg, Berlin, Heidelberg. 37

[García-Durán et al., 2018] García-Durán, A., Dumančić, S., and Niepert, M. (2018). Learning Sequence Encoders for Temporal Knowledge Graph Completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, Brussels, Belgium. Association for Computational Linguistics. 48

[Getoor, 2005] Getoor, L. (2005). Link-based classification. In *Advanced methods for knowledge discovery from complex data*, pages 189–207. Springer. 119, XX

[Getoor and Diehl, 2005] Getoor, L. and Diehl, C. P. (2005). Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12. 25

[Goyal et al., 2018] Goyal, P., Kamra, N., He, X., and Liu, Y. (2018). Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*. 67, IX

[Grover and Leskovec, 2016] Grover, A. and Leskovec, J. (2016). node2vec: Scalable Feature Learning for Networks. pages 855–864. ACM Press. 38, 44, 48, 96, IX, XVI

[Guo et al., 2017] Guo, S., Wang, Q., Wang, B., Wang, L., and Guo, L. (2017). SSE: Semantically Smooth Embedding for Knowledge Graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(4):884–897. 53

[Guo et al., 2016] Guo, S., Wang, Q., Wang, L., Wang, B., and Guo, L. (2016). Jointly Embedding Knowledge Graphs and Logical Rules. pages 192–202. Association for Computational Linguistics. 53

[Guo et al., 2018] Guo, S., Wang, Q., Wang, L., Wang, B., and Guo, L. (2018). Knowledge Graph Embedding with Iterative Guidance from Soft Rules. *AAAI*, page 8. 53

[Hajiramezanali et al., 2019] Hajiramezanali, E., Hasanzadeh, A., Narayanan, K., Duffield, N., Zhou, M., and Qian, X. (2019). Variational Graph Recurrent Neural Networks. *Advances in Neural Information Processing Systems*, 32. 59, 67

[Hamilton et al., 2017] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive Representation Learning on Large Graphs. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30, pages 1024–1034. Curran Associates, Inc. 39, IX

[Harati-Mokhtari et al., 2008] Harati-Mokhtari, A., Wall, A., Brooks, P., and Wang, J. (2008). Automatic Identification System (AIS): A Human Factors Approach. *The Nautical Institute AIS Forum*, page 12. 12

[Heidari and Papagelis, 2019] Heidari, F. and Papagelis, M. (2019). EvoNRL: Evolving Network Representation Learning Based on Random Walks. In *Complex Networks and Their Applications VII*, volume 812, pages 457–469. Springer International Publishing, Cham. 118, XX

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9:1735–1780. 68

[Hodge and Austin, 2004] Hodge, V. and Austin, J. (2004). A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:85–126. 13

[Iphar et al., 2016] Iphar, C., Napoli, A., Ray, C., Alincourt, E., and Brosset, D. (2016). Risk Analysis of falsified Automatic Identification System for the improvement of maritime traffic safety. pages 606–613 – ISBN 978–1–138–02997–2. Taylor & Francis. 12

[Ji et al., 2015] Ji, G., He, S., Xu, L., Liu, K., and Zhao, J. (2015). Knowledge Graph Embedding via Dynamic Mapping Matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, Beijing, China. Association for Computational Linguistics. 36

[Ji et al., 2016] Ji, G., Liu, K., He, S., and Zhao, J. (2016). Knowledge Graph Completion with Adaptive Sparse Transfer Matrix. page 7. 36

[Jiang et al., 2016] Jiang, T., Liu, T., Ge, T., Sha, L., Li, S., Chang, B., and Sui, Z. (2016). Encoding Temporal Information for Time-Aware Link Prediction. pages 2350–2354. Association for Computational Linguistics. IX

[Jin et al., 2019] Jin, W., Zhang, C., Szekely, P., and Ren, X. (2019). Recurrent Event Network for Reasoning over Temporal Knowledge Graphs. *arXiv:1904.05530 [cs, stat]*. arXiv: 1904.05530. 118, XX

[Johansson and Falkman, 2007] Johansson, F. and Falkman, G. (2007). Detection of vessel anomalies - a Bayesian network approach. In *Sensor Networks and Information 2007 3rd International Conference on Intelligent Sensors*, pages 395–400. 17

[Joubert, 2020] Joubert, L. (2020). The state of maritime piracy 2019. *One Earth Future*. 7

[Kaluza et al., 2010] Kaluza, P., Kölzsch, A., Gastner, M. T., and Blasius, B. (2010). The complex network of global cargo ship movements. *Journal of The Royal Society Interface*, 7(48):1093–1103. 7

[Kanjir et al., 2018] Kanjir, U., Greidanus, H., and Oštir, K. (2018). Vessel detection and classification from spaceborne optical images: A literature survey. *Remote Sensing of Environment*, 207:1–26. 12

[Katsilieris et al., 2013] Katsilieris, F., Braca, P., and Coraluppi, S. (2013). Detection of malicious AIS position spoofing by exploiting radar information. In *Proceedings of the 16th International Conference on Information Fusion*, pages 1196–1203. 12, 18

[Kazemi et al., 2019] Kazemi, S. M., Goel, R., Jain, K., Kobyzev, I., Sethi, A., Forsyth, P., and Poupart, P. (2019). Relational Representation Learning for Dynamic (Knowledge) Graphs: A Survey. *arXiv:1905.11485 [cs, stat]*. arXiv: 1905.11485. 29, 48, 52, 59

[Khatkhate et al., 2007] Khatkhate, A. M., Ray, A., and Keller, E. (2007). Modelling and system identification of an experimental apparatus for anomaly detection in mechanical systems. *Applied Mathematical Modelling*, 31(4):734–748. 13

[Kimmig et al., 2012] Kimmig, A., Bach, S. H., Broecheler, M., Huang, B., and Getoor, L. (2012). A Short Introduction to Probabilistic Soft Logic. page 4. 53

[Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-Supervised Classification with Graph Convolutional Networks. *arXiv:1609.02907 [cs, stat]*. arXiv: 1609.02907. 38, 39, IX

[Klimt and Yang, 2004] Klimt, B. and Yang, Y. (2004). The Enron Corpus: A New Dataset for Email Classification Research. In Boulicaut, J.-F., Esposito, F., Giannotti, F., and Pedreschi, D., editors, *Machine Learning: ECML 2004*, pages 217–226, Berlin, Heidelberg. Springer Berlin Heidelberg. 48, 119, XX

[Kohonen, 1998] Kohonen, T. (1998). The self-organizing map. *Neurocomputing*, 21(1):1 – 6. 16

[Kowalska and Peel, 2012] Kowalska, K. and Peel, L. (2012). Maritime anomaly detection using Gaussian Process active learning. In *2012 15th International Conference on Information Fusion*, pages 1164–1171. 18

[Le Guillarme and Lerouvreur, 2013] Le Guillarme, N. and Lerouvreur, X. (2013). Unsupervised Extraction of Knowledge from S-AIS Data for Maritime Situational Awareness. *FUSION*, page 8. 11, 12, 16

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

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. 67

[Leetaru and Schrodt, 2013] Leetaru, K. and Schrodt, P. A. (2013). GDELT: Global Data on Events, Location and Tone,. *ISA Annual Convention*, page 51. 25, 44, 119, XX

[Li et al., 2018] Li, J., Cheng, K., Wu, L., and Liu, H. (2018). Streaming Link Prediction on Dynamic Attributed Networks. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining - WSDM '18*, pages 369–377, Marina Del Rey, CA, USA. ACM Press. 51, 52, 92, X

[Li et al., 2017] Li, J., Dani, H., Hu, X., Tang, J., Chang, Y., and Liu, H. (2017). Attributed Network Embedding for Learning in a Dynamic Environment. *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 387–396. arXiv: 1706.01860. 49, 50, X

[Lin et al., 2015a] Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., and Liu, S. (2015a). Modeling Relation Paths for Representation Learning of Knowledge Bases. pages 705–714. Association for Computational Linguistics. 34

[Lin et al., 2016] Lin, Y., Liu, Z., and Sun, M. (2016). Knowledge Representation Learning with Entities, Attributes and Relations. page 7. 40, 41, IX

[Lin et al., 2015b] Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015b). Learning Entity and Relation Embeddings for Knowledge Graph Completion. page 7. 34, 35, 36, 40, VIII

[Liu et al., 2019] Liu, H., Kou, H., Yan, C., and Qi, L. (2019). Link prediction in paper citation network to construct paper correlation graph. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):233. 92

[Liu et al., 2017] Liu, H., Wu, Y., and Yang, Y. (2017). Analogical Inference for Multi-relational Embeddings. *ICML*, page 11. 36, 37, VIII

[Ma et al., 2020] Ma, Y., Guo, Z., Ren, Z., Tang, J., and Yin, D. (2020). Streaming Graph Neural Networks. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 719–728, New York, NY, USA. Association for Computing Machinery. 48, 67

[Mao et al., 2016] Mao, S., Tu, E., Zhang, G., Rachmawati, L., Rajabally, E., and Huang, G. (2016). An automatic identification system (AIS) database for maritime trajectory prediction and data mining. *CoRR*, abs/1607.03306. 15, 16

[Mascaro et al., 2014] Mascaro, S., Nicholso, A. E., and Korb, K. B. (2014). Anomaly detection in vessel tracks using Bayesian networks. *International Journal of Approximate Reasoning*, 55(1):84–98. 17

[Mazzarella et al., 2017] Mazzarella, F., Vespe, M., Alessandrini, A., Tarchi, D., Aulicino, G., and Vollero, A. (2017). A novel anomaly detection approach to identify intentional AIS on-off switching. *Expert Systems with Applications*, 78:110–123. 18

[Mei and Eisner, 2017] Mei, H. and Eisner, J. (2017). The neural hawkes process: a neurally self-modulating multivariate point process. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6757–6767, Red Hook, NY, USA. Curran Associates Inc. 80, 84, XIV

[Meng et al., 2019] Meng, Z., Liang, S., Bao, H., and Zhang, X. (2019). Co-Embedding Attributed Networks. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 393–401, New York, NY, USA. Association for Computing Machinery. 44, 92, IX

[Meng et al., 2020] Meng, Z., Liang, S., Zhang, X., McCreadie, R., and Ounis, I. (2020). Jointly Learning Representations of Nodes and Attributes for Attributed Networks. *ACM Transactions on Information Systems*, 38(2):16:1–16:32. 51, 92, X

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*. arXiv: 1301.3781. 26, 38, 40

[Mikolov et al., 2014] Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., and Ranzato, M. (2014). Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753. 68

[Murata et al., 2017] Murata, T., Onuki, Y., Nukui, S., Inagi, S., Qiu, X., Watanabe, M., and Okamoto, H. (2017). Predicting Relations Between RDF Entities by Deep Neural Network. In Blomqvist, E., Hose, K., Paulheim, H., Ławrynowicz, A., Ciravegna, F., and Hartig, O., editors, *The Semantic Web: ESWC 2017 Satellite Events*, volume 10577, pages 343–354. Springer International Publishing, Cham. 37, IX

[Nguyen et al., 2018] Nguyen, G. H., Lee, J. B., Rossi, R. A., Ahmed, N. K., Koh, E., and Kim, S. (2018). Continuous-Time Dynamic Network Embeddings. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pages 969–976, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee. event-place: Lyon, France. 47, 96, IX, XVI

[Nickel et al., 2016] Nickel, M., Murphy, K., Tresp, V., and Gabrilovich, E. (2016). A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1):11–33. arXiv: 1503.00759. 23, 24

[Nickel et al., 2015] Nickel, M., Rosasco, L., and Poggio, T. (2015). Holographic Embeddings of Knowledge Graphs. *arXiv:1510.04935 [cs, stat]*. arXiv: 1510.04935. 34, 36, 37, VIII

[Nickel et al., 2011] Nickel, M., Tresp, V., and Kriegel, H.-P. (2011). A Three-Way Model for Collective Learning on Multi-Relational Data. page 8. 35, 36, VIII

[Nickel et al., 2012] Nickel, M., Tresp, V., and Kriegel, H.-P. (2012). Factorizing YAGO: scalable machine learning for linked data. page 271. ACM Press. 35, 53

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. 40

[Perera and Soares, 2010] Perera, L. P. and Soares, C. G. (2010). Ocean Vessel Trajectory Estimation and Prediction Based on Extended Kalman Filter. *2nd International Conference on Adaptive and Self-adaptive Systems and Applications*, page 7. 18

[Perozzi et al., 2014] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652. 44, IX

[Pitsikalis and Artikis, 2019] Pitsikalis, M. and Artikis, A. (2019). Composite maritime events. 89, XV

[Portnoy et al., 2001] Portnoy, L., Eskin, E., and Stolfo, S. (2001). Intrusion detection with unlabeled data using clustering. In *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001*, pages 5–8. 15

[Pujara et al., 2015] Pujara, J., Miao, H., Getoor, L., and Cohen, W. W. (2015). Using Semantics and Statistics to Turn Data into Knowledge. *AI Magazine*, 36(1):65. 53

[Ray et al., 2019] Ray, C., Dréo, R., Camossi, E., Jousselme, A.-L., and Iphar, C. (2019). Heterogeneous integrated dataset for Maritime Intelligence, surveillance, and reconnaissance. *Data in Brief*, 25:104141. 89, XV

[Rhodes et al., 2005] Rhodes, B., Bomberger, N., Seibert, M., and Waxman, A. (2005). Maritime situation monitoring and awareness using learning mechanisms. pages 646–652. 17

[Richardson et al., 2003] Richardson, M., Agrawal, R., and Domingos, P. (2003). Trust management for the semantic web. In *International semantic Web conference*, pages 351–368. Springer. 119, XX

[Richardson and Domingos, 2006] Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62(1-2):107–136. 53

[Riveiro et al., 2008] Riveiro, M., Johansson, F., Falkman, G., and Ziemke, T. (2008). Supporting Maritime Situation Awareness Using Self Organizing Maps and Gaussian Mixture Models. In *Proceedings of the 2008 conference on Tenth Scandinavian Conference on Artificial Intelligence: SCAI 2008*, pages 84–91, NLD. IOS Press. 16

[Riveiro et al., 2018] Riveiro, M., Pallotta, G., and Vespe, M. (2018). Maritime anomaly detection: A review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(5):e1266. 13, 15, 16

[Roweis and Saul, 2001] Roweis, S. and Saul, L. (2001). Nonlinear dimensionality reduction by locally linear embedding. *Science (New York, N.Y.)*, 290:2323–6. 53

[Roy, 2008] Roy, J. (2008). Anomaly detection in the maritime domain. page 69450W, Orlando, FL. 13, 14

[Salehinejad et al., 2017] Salehinejad, H., Sankar, S., Barfett, J., Colak, E., and Valaee, S. (2017). Recent Advances in Recurrent Neural Networks. *arXiv:1801.01078 [cs]*. arXiv: 1801.01078. 70

[Sankar et al., 2018] Sankar, A., Wu, Y., Gou, L., Zhang, W., and Yang, H. (2018). Dynamic Graph Representation Learning via Self-Attention Networks. *arXiv:1812.09430 [cs, stat]*. arXiv: 1812.09430. IX

[Schlichtkrull et al., 2017] Schlichtkrull, M., Kipf, T. N., and Bloem, P. (2017). Modeling Relational Data with Graph Convolutional Networks. page 9. 38

[Serry and Lévêque, 2015] Serry, A. and Lévêque, L. (2015). Le système d'identification automatique (AIS). Une source de données pour étudier la circulation maritime. *Netcom. Réseaux, communication et territoires*, (29-1/2):177–202. Number: 29-1/2 Publisher: Netcom Association. 12

[Socher et al., 2013] Socher, R., Chen, D., Manning, C. D., and Ng, A. (2013). Reasoning With Neural Tensor Networks for Knowledge Base Completion. page 10. 37, IX

[Souza et al., 2016] Souza, E. N. d., Boerder, K., Matwin, S., and Worm, B. (2016). Improving Fishing Pattern Detection from Satellite AIS Using Data Mining and Machine Learning. *PLOS ONE*, 11(7):e0158248. Publisher: Public Library of Science. 15

[Tang et al., 2015] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). LINE: Large-scale Information Network Embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE. 39, IX

[Tang et al., 2008] Tang, J., Zhang, J., Yao, L., Li, J., Zhang, l., and Su, Z. (2008). Arnetminer: Extraction and mining of academic social networks. pages 990–998. 90, XV

[Tay et al., 2017] Tay, Y., Tuan, L. A., Phan, M. C., and Hui, S. C. (2017). Multi-Task Neural Network for Non-discrete Attribute Prediction in Knowledge Graphs. pages 1029–1038. ACM Press. 42, 44, IX

[Trivedi et al., 2017] Trivedi, R., Dai, H., Wang, Y., and Song, L. (2017). Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. *arXiv:1705.05742 [cs]*. arXiv: 1705.05742. 45, 47, 48, 67, 70, 74, 78, IX

[Trivedi et al., 2019] Trivedi, R., Farajtabar, M., Biswal, P., and Zha, H. (2019). DyRep: Learning Representations over Dynamic Graphs. In *ICLR*. 48, 67, 84, XIV

[Trouillon et al., 2016] Trouillon, T., Welbl, J., Riedel, S., Gaussier, , and Bouchard, G. (2016). Complex Embeddings for Simple Link Prediction. *arXiv:1606.06357 [cs, stat]*. arXiv: 1606.06357. 37

[Vouros et al., 2019] Vouros, G., Santipantakis, G., Doulkeridis, C., Vlachou, A., Andrienko, G., Andrienko, N., Fuchs, G., Cordero Garcia, J., and García, M. (2019). The datacron ontology for the specification of semantic trajectories. *Journal on Data Semantics*, 8:1–28. 22

[Wang et al., 2017] Wang, Q., Mao, Z., Wang, B., and Guo, L. (2017). Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743. 35, 36, 53, VIII

[Wang et al., 2015] Wang, Q., Wang, B., and Guo, L. (2015). Knowledge Base Completion Using Embeddings and Rules. page 7. 53

[Wang et al., 2014] Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014). Knowledge Graph Embedding by Translating on Hyperplanes. page 8. 34, 35, 36, 40, VIII

[Xiao et al., 2015] Xiao, H., Huang, M., Yu, H., and Zhu, X. (2015). TransA: An Adaptive Approach for Knowledge Graph Embedding. page 8. 36

[Xiao et al., 2016] Xiao, H., Huang, M., and Zhu, X. (2016). From One Point to a Manifold: Knowledge Graph Embedding for Precise Link Prediction. page 7. 36

[Xie et al., 2016] Xie, R., Liu, Z., Jia, J., Luan, H., and Sun, M. (2016). Representation Learning of Knowledge Graphs with Entity Descriptions. page 7. 34, 53

[Yang et al., 2014] Yang, B., Yih, W.-t., He, X., Gao, J., and Deng, L. (2014). Embedding Entities and Relations for Learning and Inference in Knowledge Bases. *arXiv:1412.6575 [cs]*. arXiv: 1412.6575. 37

[Yu, 2007] Yu, L. (2007). *Semantic Web and Semantic Web Services*. 21

[Zhang et al., 2019] Zhang, D., Yin, J., Zhu, X., and Zhang, C. (2019). Attributed network embedding via subspace discovery. *Data Mining and Knowledge Discovery*, 33(6):1953–1980. 44, IX

[Zhao et al., 2020] Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., and Li, H. (2020). T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858. 67

[Zhou et al., 2018] Zhou, L.-k., Yang, Y., Ren, X., Wu, F., and Zhuang, Y. (2018). Dynamic Network Embedding by Modeling Triadic Closure Process. In *AAAI*. 48

[Zissis et al., 2016] Zissis, D., Xidias, E. K., and Lekkas, D. (2016). Real-time vessel behavior prediction. *Evolving Systems*, 7(1):29–40. 18

