Toward a New History of the Recording Industry: Extracting relationships between entities from crowdsourced free-text data

[redacted]

Submitted as part of the degree of BSc Computer Science to the Board of Examiners in the Department of Computer Sciences, Durham University

Abstract—Within the recording industry, there is a need to be able to visualise and conceptualise different labels and the relationships between them. Using free-text data from Discogs, a major crowdsourced online music encyclopedia, we create a temporal network graph to visualise these relationships over time via a graphical user interface, using natural language processing, named entity recognition and graph construction algorithms. We propose a simple pattern matching solution to build these graphs, which identifies these relationships with 64.2% accuracy.

Index Terms—Graphs & Networks, Langauge Parsing & Recognition, Natural Language Processing, Visualisation Systems & Software

1 Introduction

The recording industry, as well as the music industry as a whole, tend to have many layers of complexity with regards to business relationships. Up until the late 1990s, there were six major record labels that had a majority share in the music market, those being Warner Music Group, EMI, Sony Music, BMG, Universal Music Group and PolyGram. Within only 15 years, this "Big Six" shrunk into the "Big Three" in 2012, with Polygram merging into Universal, BMG merging into Sony, and EMI merging into Universal by this point [2].

In an article by Alison Wenham [21], these labels were estimated to have around 65 to 70% of the market share in the field of record labels, with this "Big Three" owning the majority of labels under their wing. However, this article was written at a time when streaming services, such as Apple Music, became mainstream platforms, making it substantially easier for independent labels to publish their music without much need for large financial backing since it was much easier to pay the fee for the song to go onto the streaming service than having to pay for physical media to be created and shipped out. As the age of digital music distribution continues, smaller and newer artists have an opportunity to share their unique creative vision faster and easier than ever. [20]

Discogs, a privately owned company founded by Kevin Lewandowski in the year 2000, was designed as an online website in which users could share discographic information about electronic music [5], but has since grown significantly to become one of the largest online music databases with almost half a million contributors [15] who voluntarily catalogue all of their known infor-

mation regarding various music releases, artists and, most importantly, record labels. Each label on the Discogs website features a 'profile', acting as a description of the history of each record label, including events that happened within that label's history, such as acquisitions, mergers and dissolution events.

With such a high number of labels and complicated business relationships between them, there is a need to understand the nature and timing of relationships between businesses. This inspired the creation of a program that would parse the crowdsourced free-text data found in the Discogs profile of a label, clean the text by removing irrelevant information that may hinder accurate parsing; then using natural language processing to pick out events, entities and times, extracting these into a file format used to construct a temporal network graph, in order to represent each of these business events over time visually. The use of a temporal network graph allows for a user to not only visually analyse, but also interact with a graph and be able to see these relationships between various labels, in addition to seeing how the graph changes over time by using a slider to adjust the year displayed in the graph.

The main research question to answer was as follows: "Given a database, filled with 'messy' free-text profiles, can the relationships between entities spoken about in the free-text be extracted accurately?" Other questions also arise from this, as the main question itself is very broad. One such question is "how can these relationships be visually represented?" – these business relations change over time due to various business reasons. There then becomes a need to represent these relationships over time, as yearly "slices" or snapshots. This then makes the information extraction more complicated as the year must be extracted, giving rise to the question of "if the year cannot be extracted successfully from an event, how can we estimate the time of a given event?" As there are also some labels with the same names, such as Seasons Recordings, there then becomes a need for disambiguation when constructing these graphs: how can we determine which "Seasons Recordings" entity is being referenced when the graph is being constructed?

Following on from these research questions, we propose 2 deliverables. First, a program that takes in the Discogs data, taken from the public dump CSV file, pre-processes the data, uses various natural language processing techniques to extract the relationships from the text, then outputs them to a JSON file. This JSON file is read by the second deliverable, which uses the JSON to construct a temporal network graph using various Python packages, namely Dash, NetworkX and Plotly.

2 Related Work

This section presents a survey of existing work on the problems that this project addresses.

This research spans multiple areas – including information extraction, natural language processing, temporal network graphs and crowdsourced data analysis. While there is limited work that directly addresses the full scope of this project, there are many relevant contributions within each subdomain. Zuo et al. (2017) [22] propose a semi-supervised method to extract relationship types based on the surrounding context of a company pair, enabling the construction of directed business relationship graphs, which aligns closely with the objectives of this research. This approach informs aspects of our own method, particularly in how relationship directionality is inferred, as they propose a candidate pattern extraction algorithm to determine the "direction" of a relationship, which is a similar technique used when drawing the graph. Khaldi et al.'s 2022 paper [10] provides a fully annotated corpus known as BizRel, resulting in over 25,000 sentences across 4 languages representing various business relations. Although the multilingual BizRel corpus is comprehensive, this project solely focuses on English data. Furthermore, several annotation types within BizRel corpus fall outside of the scope of this project, such as competition, legal proceedings and supply.

With regard to extracting relationships from free text, there are many papers that have covered a similar task in other fields; Alfattni et al.'s 2020 paper [1] covers extracting temporal relations from free text data within a clinical

setting, using various free-text descriptors within medical records to query which TLINK tasks can be applied to clinical data. Nassirtoussi et al.'s 2014 paper [13] reviews the use of mining free-text data across various platforms such as social media and online news outlets in order to predict patterns in the financial market, taking a 3-step approach of selecting, pre-processing and feeding the data into a neural network to output "predictions". However, while both of these papers are firmly rooted within distinct domains, certain preprocessing strategies, such as selective data curation, are relevant to this research.

Jossé et al.'s 2017 paper [9] covers the challenges of working with crowdsourced data, claiming that plain textual mentions may be ambiguous and might not map to a unique location. While this paper focuses on the use of crowdsourced data for mapping out road networks, an important takeaway of this paper is, as Discogs can have multiple labels with the same name, a plain textual mentions of identically named labels can cause some issues, such as mapping to the incorrect label upon construction of the graph. This issue was researched by Töpper et al. in their 2012 paper [19], which focuses on DBpedia. The paper itself calls DBpedia "error prone due to its automatic nature", proposing the use of an enriched ontology to base the process of extraction upon, allowing for inconsistencies to be detected during the extraction process. While they focus on deriving missing information from the DBpedia dataset as there exists no domain and range restriction for a number of properties, a similar example cannot apply to Discogs' data, as the actual relationships purely stem from the profile, though the idea of the paper, generally creating a higher quality knowledge base, can still be applied within this field. These papers highlight the challenge of ambiguity within crowdsourced datasets, which is a key consideration for accurate label disambiguation in our graph construction process.

Other papers have also researched the ability to transform between text and graph, with Koncel-Kiedzorski et al.'s 2019 paper [11] uses graph-text pairs to train a novel attention-based encoder-decoder model for knowledge graph-to-text generation. While our work focuses on extracting relationships *into* a temporal graph from free text, these efforts in generating *from* graphs inform our understanding of structural mappings between text and network representations. Heindorf et al. expanded upon this idea in 2020 [6], creating a causality graph extracted from the web, primarily utilising Wikipedia. Causality graphs, unlike knowledge graphs, focus on events where "A causes B". Heindorf et al.'s work is particularly relevant as it leverages Wikipedia, which, much like Discogs, is an online encyclopedia fuelled by volunteers who aid with filling out information in a free-text format. To our knowledge, there is no research that focuses specifically on Discogs in terms of taking free-text data from the web and constructing a temporal knowledge graph after extracting relationships from the database.

As mentioned in Section 1, Sicilia's 2020 paper [15] and Hartnett's 2015 article [5] detail the nature of Discogs, which can both be used to provide a brief justification for why Discogs was chosen as the dataset for this research question; it contains extensive amounts of cross-referenced information about record labels, publishers, release dates, formats and more. With nearly half a million contributors via archival contribution and more than 1.5 million unique record labels, over 300,000 of which have a profile with a sufficiently high enough data correctness rating, Discogs presents a rich, structured and community-verified dataset. This rating is determined by "correctness" of the data, voted on by members who have contributed a lot to the Discogs dataset, according to the Discogs support guidelines [17], which also provides definitions for various data quality labels found within the dataset.

3 METHODOLOGY

This section presents the solutions to the problems in detail. As the problem has many steps, we have broken them down into different subsections so as to make it easier to navigate.

3.1 Data pre-processing

Prior to applying downstream processing steps, the data must first be pre-processed. The data itself comes from a public dump of all available data from Discogs all compiled into a CSV file with multiple headings for different characteristics of each label, including its name, its ID and, most importantly, the label "profile", which is the free-text data which we want to extract the information from.

These profiles often contain noisy or inconsistently formatted data, as they feature Discogs' own hyperlink system. These hyperlink annotations follow a semi-structured markup system specific to Discogs, where links to other entities are embedded using bracketed expressions. The first (A) involves square brackets, with an 'a' or 'l' (for Artist or Label), followed by an equals sign and the name of the artist or label. The second (B) involves a similar format, but instead of an equals sign and a name, it instead uses the ID of the label or artist. Due to the constraints of the dataset given, we cannot use this to replace artist IDs with names, but we can use the IDs of labels to get the name of a label and replace it within the text, keeping the hyperlink itself to one side for later disambiguation. As these hyperlinks tend to follow a given format, a simple regular expression is used to capture and remove these.

(A): `\[\s*a\s*=\s*([^\]]+)\s*\]`

(B): `\[\s*(a\d+)\s*\]`

Both of these regular expressions create a capture group to pick up on both formats, removing the square brackets and equal signs. This means that, in any case where the artist or label are directly mentioned in the hyperlink, the hyperlink itself is essentially removed, whereas any in the format of "l12345" still remains, making it obvious that there is a hyperlink that needs replacing, which can be done by looking up the label (if there is an l) by its ID (the number after the l) in a separate column within the outputted CSV file. Since the Discogs profile data is in their own version of Markdown with some slight changes, it becomes somewhat easy to ignore escape characters, such as new line and tab characters.

There are varying levels of data quality in the Discogs dataset, as voted on by trusted members of the community, which was previously mentioned in Section 2, and can be found in the Discogs guidelines [17], which shows the various data quality labels and their meanings. For the sake of having a dataset that can have relationships properly extracted, we ignore entries where the data quality rating is Needs Major Changes or Entirely Incorrect, which reduces the number of labels in the dataset significantly. We also ignore any labels which have a profile with no text in it, i.e. a profile that is null; if we attempt to extract relationships from a null profile, we run into errors further down the line. After filtering for data quality, over 300,000 labels need pre-processing before being utilised downstream.

Once this simple pre-processing step is complete, the data is saved into a CSV file using Pandas and is then passed on to the next part.

3.2 Part of speech tagging

Within part of speech tagging, there are two main Python libraries that can be used, namely NLTK [3] and spaCy [7]. Both seem to do a similar task, in that they can take in text data and tag each of them as parts of speech, where every word is given a tag depending on the type of word that it is. Both were empirically evaluated to determine which was the more suitable library for the task. After some experimentation and scoring the outputs of each, we decided that spaCy was more robust and accurate compared to NLTK, which we will cover in more detail in Section 3.5.

This part of speech tagging processes each word in a cleaned sentence, using its own system to determine what type of word it is. For example, in a sentence like "Label Alpha was founded by Mark Stephenson in 1998", a part of speech tagging algorithm will look at each of these words separately and tag it. "Label" and "Mark" may be extracted as proper nouns, as they are capitalized, with "founded" being extracted as a verb, and "1998" being extracted as a year. This then becomes the later foundation for the pattern matching approach covered in Section 3.4.

However, this token-level part of speech tagging approach can cause some problems. By assuming each word has the same meaning as the

word at face value, the algorithm could potentially miss out on some minute details. For example, the band name ”Bullet for my Valentine” may not be recognised as an entity, as the PoS tagging may only recognise it by parsing each word separately, correctly tagging ”bullet” and ”valentine” as nouns, but the ”for” and ”my” may be tagged incorrectly. These limitations necessitate the use of Named Entity Recognition (NER), which can identify multi-token entities such as brand names or organisations that are otherwise missed by token level parsing.

3.3 Named entity recognition

As these label profiles contain the names of people, labels and other such entities, which are crucial for discovering the relationships between various labels, part of speech tagging alone is insufficient for resolving named entities as it takes every single word at face value. As a human reading the text, we can understand whether a certain string of text corresponds to what the words actually mean, or if they are the name of a band. This emphasises the importance of named entity recognition to capture multi-token entities.

For example, if there was a band with the name ”Hello There”, the part of speech tagging would only be able to pick out each word individually. However, named entity recognition approaches consider broader context windows to identify multi-token spans, tagging them accordingly with labels such as ”ORG” for organisation, or ”PERSON” for person.

As mentioned by Sun et al. in their 2018 survey [16], there are many approaches to named entity recognition, in which the more modern approaches tend to use machine learning methods, such as deep learning or supervised learning, in order to train the model on what counts as a named entity. spaCy uses a variety of pre-trained models and pipelines in order to naturally process the language and, more importantly, to recognise entities.

As shown in Figure 1 above, the Named Entity Recognition (NER) occurs at the end of the pipeline, and these named entities are saved as ”ents” when the document (in this case, each

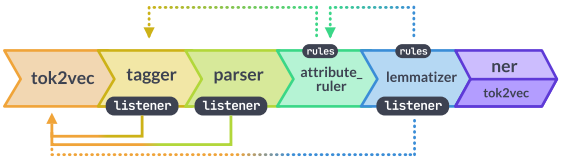

Fig. 1. Overview of the spaCy CNN-based NLP pipeline

label’s individual profile) is put through spaCy’s natural language processing algorithm. By using this, the model can extract, to some degree of success, the names of various artists and labels. As these models are pre-trained, there is a limit to the effectiveness of this approach, which will be covered later in Section 5.

Many of spaCy’s core components, such as the tagger, parser and text categoriser are built on statistical models. Based on this pre-trained model’s weight values, a prediction is made as to what type of word is found in the current context of the sentence. As this training data all comes from Discogs, and the named entities that we are interested in are almost always record labels, there is a higher chance that the pre-trained models that spaCy has are unaware of various label and band names, which can cause some confusion when the model is being executed. This would result in having to train our own model to recognise these labels, bands and people, which, due to the time constraints of the project, was not possible.

3.4 Business relationships: pattern matching approach

Within the English language, there are many ways of conveying one message by varying the sentence structure, which could confuse a model. Within this research, we consider ”directed” relationships, as we want to construct a graph later on (which we will also cover in Section 3.6).

These ”directed relationships” can convey in which direction a verb corresponds to. As an example, let us propose three sentences, which all mean the same thing at a basic level.

(A): Label Gamma acquired Label Delta in 2010.

(B): Label Delta was acquired by Label Gamma in 2010.

(C): In 2010, Label Delta underwent an acquisition from Label Gamma.

From a human perspective, we may extract four key components from each of these sentences. Firstly, that Label Gamma is the ”subject” (i.e. the label that acquires the other); secondly, Label Delta is the ”object” (the label being acquired); thirdly, that the verb is ’to acquire’, meaning one label has bought out another; and finally, that the year this acquisition happened was 2010.

However, from a computer’s perspective, all of these sentences look different, as the words are in a different order. This then becomes a key point of contention, as we want to be able to pick up on each of these nuances within the sentences to be able to extract the necessary information. A rule-based pattern matching approach can serve as a practical baseline.

By using NER, as previously mentioned in Section 3.3, as well as using part of speech tagging to extract the desired verb from the sentence, which was covered in Section 3.2, we can construct a pattern matching approach that can successfully extract the desired information from the text and save it into a file. We can create a simple pattern for each of the common ways of phrasing these expressions, for example:

(A): ENT VB ENT YEAR ($\rightarrow$)

(B): ENT VB VB ENT YEAR ($\leftarrow$)

(C): YEAR ENT VB VB ENT ($\leftarrow$)

As shown above, by using a basic pattern matching approach, the entity (ENT) and year (YEAR) can be extracted, while preserving the directionality of the relationship represented by the verb (VB), shown as the arrow in the brackets after the pattern, meaning the object and subject can be preserved no matter which way the sentence is structured. This yields a relatively comprehensive solution by being able to extract each event, alongside a timestamp for each event. Each sentence is parsed to extract named entities and verbs. Using spaCy’s tagging software and a token-based pattern matching approach, we can define these token patterns as shown above which correspond to directed relationships. The relative order of entities and verbs within the sentence determine the directionality of the relationship. This rule-based method, however, may fail to capture more complex or passive constructions that are not covered by predefined patterns.

Once extracted using our token-based pattern matching approach, each relationship is saved in a structured JSON file containing the object label, subject label, the type of relationship (e.g. ”acquisition”) and the year of occurrence. This structured representation facilitates the construction of a temporal knowledge graph, capturing the directionality and chronology of inter-label relationships derived from free-text descriptions.

3.5 Scoring relationship recognition

Once relationships are extracted from the text, it becomes necessary to evaluate the accuracy of these extractions against a ground truth. While humans can intuitively extract relationships from natural language, a computational model relies solely on the features it is programmed to detect, without any inherent understanding of the language.

In Section 3.2, we briefly mentioned an experiment comparing the effectiveness of NLTK to spaCy’s pipeline in transforming the plaintext data into a structured JSON file format. This experiment involved 10 manually constructed sentences, each individually fed through the identical pipeline of both NLTK and spaCy, including part of speech tagging and token-based pattern matching. Each of these sentences varied in format and focussed on the ”founded” verb at the time, covering the varied syntactic formats as described in Section 3.4. This allowed for a more rigorous test, as the desired algorithm should be able to pick up on as many different phrasing methods as possible in order to extract as much information from the dataset as it could.

For each of these 10 hand-written test cases, we also wrote a ”ground truth” statement for each of them. Using Python’s built-in Difflib library and the Sequence Matcher within the library, we were able to take both the generated output from NLTK and spaCy’s pipelines, using this sequence matcher to quantify character-level overlap between the expected and extracted JSON outputs as a percentage.

NLTK failed to extract relationships from several test cases, resulting in a low average similarity score of 14%, whereas spaCy processed all sentences and achieved a mean similarity score of 64.2%, demonstrating superior performance in relationship recognition tasks.

Whilst this initial experiment did not make use of named entity recognition, we later altered the code to add this functionality, repeatedly running the pipeline and applying this similarity scoring method to evaluate whether modifications made to the code improved the accuracy, which could be inferred from either an increase or a decrease in the similarity score. The higher the similarity score, the more accurate the data produced by the algorithm, meaning the graph construction can be a more confident representation of the data shown in the Discogs dataset.

3.6 Creating a temporal network graph from recognisable relationships

The final step in this pipeline involves constructing a temporal network graph to visualise the relationships extracted from the free-text data. As researched by Prasad and Ojha in their 2012 paper [14], based on cognitive experiments, graphical representations are around 25% faster to understand than textual mentions of the same information, making this an effective medium for representing these relationships and summarising complex interactions.

There are two main types of relationships that can be represented on the graph; one as a Node Creation or Destruction (NCD) relation, such as a founding event, which is a relationship either involving one label and itself or a label and an artist (which we omit from the graph for ease of visualisation); and the second as a Node To Node (NTN) relation, such as an acquisition, which is a relationship between two labels, usually directional, which can be drawn as an edge in the graph.

As well as extracting the labels and the relationship type each time, the program aims to extract a timestamp wherever it can. If the year of the event is mentioned explicitly within the text, the node or edge can be assigned a higher confidence level and be timestamped. If the year is not plainly mentioned for a given event within the text, a time frame can be given for the event. This is represented on the graph with a distinct visual encoding, until we know the event definitely happened before a certain point.

As an example, consider the following profile in which Label Epsilon has the following profile: ”Epsilon was founded in 1987. After this, it acquired Label Phi, before Label Alpha acquired Label Epsilon in 2002.” The middle acquisition has no plain textual mention, but given that this relationship occurs between two plainly stated years, we can assume that the acquisition of label Phi happened at some point between 1987 and 2002.

When labels are created or destroyed, in a ”founding” event or a ”dissolving” event, for example, the label will appear or disappear from the graph respectively. Much like detailing the relationships, we can use different colours to represent whether we are certain the founding event happened within a certain time frame. If a label’s founding year is explicitly mentioned, we may draw this on the graph with some degree of confidence. However, if this is not explicitly mentioned, we may assume that it happened at some point before the first year mentioned in the text.

As the graph itself becomes very cluttered by the amount of different labels and their relationships between each other. The graph includes a filtering system, allowing the user to isolate specific labels. If a user wants to see everything that happened within the history of Label Gamma, for example, the graph filters to only display the labels directly connected to Label Gamma – its parents, children and their respective ancestors and descendants, effectively giving each label profile its own tree structure. This results in a much more ”clean” and friendly user experience as they are not overwhelmed with unnecessary information.

3.7 Implementation remarks

Due to the complexity of the solution, this section outlines various technical components involved

in implementation, which may support future adaptation of the framework in other domains.

At first, the data is loaded and cleaned using Pandas [18], a scientific Python tool for reading, analysing and cleaning data, as covered by McKinney's 2010 paper [12]. Discogs' data releases as a tarball file monthly, and can be downloaded from their public dump, as linked here.

Next, the data is passed through to spaCy. Not only did spaCy exceed the performance of NLTK in the earlier experiment, but due to its flexibility, clear documentation and state of the art natural language processing capabilities, it became the preferred choice for this project. It allows for the development of a pattern matching approach involving named entities in a more effective way than NLTK can, since NLTK's older architecture and approach can result in encountering some performance issues, unlike spaCy's more up-to-date methods of named entity recognition.

Graph visualisation is handled using NetworkX [4], Dash and Plotly [8]. This integration enables the graph to be rendered and viewed interactively via a local web server. Customisation of node appearances can be achieved by modifying the structure and content of the JSON input files.

Although this pipeline was developed for analysing record label relationships within the Discogs dataset, its modular design and use of pattern-based natural language processing suggest potential applicability within other domains requiring basic relationship extraction from free-text, with only a few adjustments being necessary.

4 Results

A number of experiments were conducted throughout the project to inform key implementation decisions. The first of these was the decision to use spaCy over NLTK, due to its more rigorous detection of correct words and labels.

The two pipelines of NLTK and spaCy are ever so slightly different with the way language processing is handled, but regarding the experimentation process, we devised 10 test sentences with varied phrasings to simulate different syntactic structures found within the Founding events – specifically, instances where a record label is created. The NLP algorithm should ideally extract the label name, founder(s) and founding year, if explicitly mentioned within the text. We then manually created 10 expected results we wanted from both NLTK and spaCy, if they were able to parse the text with complete accuracy. These test cases were designed to evaluate the boundaries of each model's parsing capabilities. As mentioned earlier in Section 3.5, we scored both NLTK and spaCy's effectiveness and opted to side with spaCy, as it had an accuracy score of 64.2% compared to NLTK's 14%.

In order to pick up on events that were the most relevant within the text, we also ran an experiment to determine the most common verbs within the dataset, allowing for us to decide which relationships were worth modelling and would be worthwhile to use in order to demonstrate the various relationship types discussed in Section 3.6. We focussed on one representative verb for each relationship type (NCD and NTN) to demonstrate the system's functionality. For NCD, we decided on "founded", as within the entire Discogs dataset, it appears over 30,000 times within the profiles of various labels, which makes it relatively representative within this dataset. Alongside 'found', we also picked the words 'create' (6777) and 'establish' (7894), which we took to be synonymous with "founded" in most contexts within the Discogs dataset.

For a NTN (node to node) relation, we decided on an acquisition. This is a common business relation that can have a directionality between the label that performs the acquisition and the label being acquired. 'Acquire' had 1746 entries in the dataset, and we also identified other contextually similar verbs to "acquire" that are used in a similar context, with 'purchase' (857) and 'merge' (1114) both appearing frequently enough to appear within multiple sets of nodes.

Figure 2 shows the graph of Universal Music in the year 2020. The search box narrows down the number of nodes to only those that contain "Universal Music" in their name. At the top of the interface, an interactive slider allows users to filter the graph by year to only show labels that exist in that year and before. Naturally, as the year

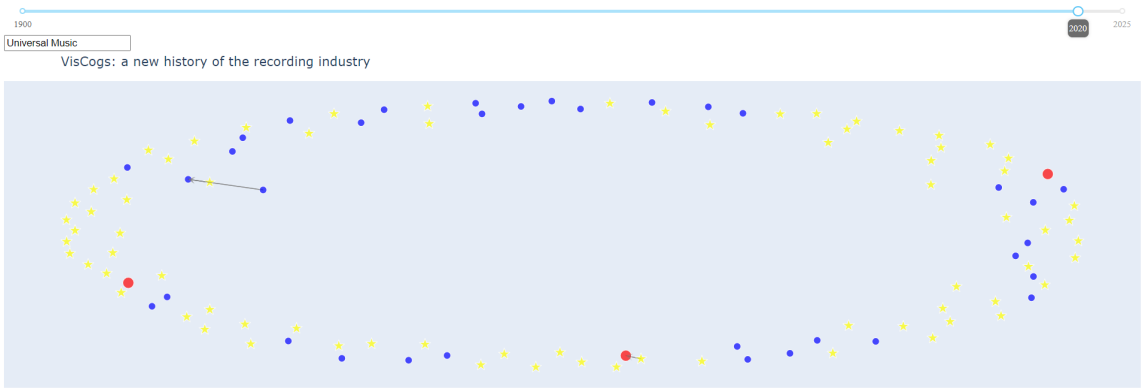

Fig. 2. An example graph of Universal Music in 2020

increases, more nodes appear since more record labels exist in the modern day. The background of the graph is a light grey colour to ensure contrast and visual clarity of nodes and edges.

Unfortunately, due to the sparse number of acquisitions within the graph, there were not many label filters that would be able to highlight every feature of this graph, but we decided that Universal Music in 2020 was suitable to show every single part of the program at one time.

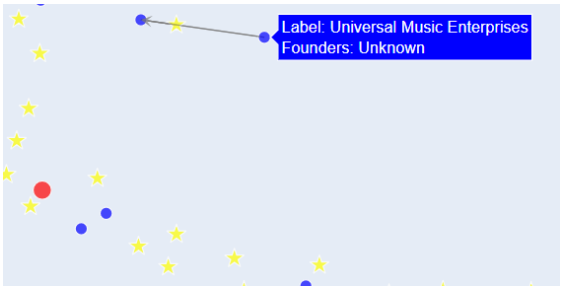

Fig. 3. A closer look at the Universal Music graph

Figure 3 provides a closer snapshot of the label, showing every single feature. The arrow between the two blue nodes indicates an acquisition event, which provides a tooltip when the user hovers over it, to indicate when the acquisition happened. Due to limitations in Plotly and NetworkX, the edges cannot be given colours like the nodes, resulting in the need for these tooltips. Hovering over a node gives a tooltip with the label name and the names of the founders. In addition, since Plotly does not feature native support for directed graphs, the arrow was made by using a Plotly annotation.

Yellow star-shaped nodes represent labels with an unknown or unspecified founding year. These are still displayed as their presence provides valuable context. The larger and red nodes are brand new nodes, as shown in the bottom left of Figure 3, which means that the node was founded that year. Blue nodes are nodes with a definite founding year before the year shown in the sliding bar.

Naturally, the results exhibit certain limitations due to the simplified assumptions and the heuristic-based approach. We assume that any time a label is created, the founders are explicitly mentioned and that within the sentence that introduces the founding only contains the names of the label and the founders; the dataset has shown this to be not the case. We must also mention that, in some cases, it picks up on a full name, as well as all parts of the same name. For example, within the dataset, Planet E has ”Carl Craig” and ”Carl Craig’s” listed as founders. This is due to how spaCy tokenises each word, with possessive forms like ”Carl Craig’s” being misinterpreted as a distinct entity due to how spaCy tokenises possessives.

5 EVALUATION

Finally, we evaluate the effectiveness of this application on the initially proposed use case. It is understood that humans, as a whole, will understand the graphical information laid out before them and feel more inclined to explore and search these data, as opposed to laying out ”walls of text”, as many people would commonly refer to these online. By presenting data extracted from CSV files (e.g. from Discogs), we make otherwise complex information more legible and accessible.

As previously mentioned in Section 3.6, Prasad and Ojha’s 2012 paper [14], graphs are around 25% faster to understand than textual mentions of the same information, which gives some credibility to this method of laying out the information in an interactive, filterable temporal network graph. The tooltips are helpful for giving extra information about the nodes and the relationships between them when hovering over the edges and nodes.

As well as this, with the rise in applications based on artificial intelligence, we attempted to base this application from a purely standard programmatic approach not involving AI whatsoever, which, while the accuracy is not perfect, for a very simple approach, the results produced by the simple pattern matching that we originally proposed can be seen as impressive, though somewhat limiting, due to the variety of the English languages and the variety in the Discogs dataset.

Since there are so many different contributors registered to the Discogs website, almost 750,000 as of April 2025, there will be many different writing styles that each of these people will have when writing the profiles for various record labels. Some of these contributors are not native English speakers, even though Discogs is an entirely English website, which means that some of the profiles that are potentially written by non-native English speakers would not have perfect grammar or would just have different styles of writing the same thing. Small inconsistencies, often unnoticed by human readers, can significantly impact the pattern matching approach, lowering its accuracy.

However, the approach is not completely rigorous. The extraction method often retrieves too much information, as the pattern matching approach tends to pick up every noun in the sentence. This results in overly long founder lists for some nodes where unrelated label names or duplicate names are mistakenly identified as founders, as illustrated within Section 4 with the Planet E label in the dataset.

As well as this, due to the amount of labels within the dataset, the program itself runs slowly. The label filtering system was added, to display a limited number of nodes at one time to improve performance. However, as the filter updates with every letter typed, it results in unnecessary processing and slows down the program. Unfortunately, Plotly and NetworkX cannot be easily tweaked to account for this, so we have to settle for processing which can be quite slow, resulting in a wait time of around 5 minutes while the program filters down to the labels with the desired name and year.

6 CONCLUSION

In conclusion, we present a pipeline that processes a CSV dataset, tags profiles using spaCy’s natural language processing algorithms and creates a temporal network graph from the relationships extracted from the CSV file.

By employing a simple token-based pattern matching approach, we achieved a reasonable degree of accuracy despite the inherent limitations of this method. By analysing word types in a certain order, we are able to recognise this data with roughly 64% accuracy and represent it effectively. However, with the large amount of data, varying writing styles and levels of information which differ between each profile, developing rules to match every pattern across all label profiles would be a highly complex and resource-intensive task. Also, some information that we originally assumed is commonplace across all label profiles does not show in every single profile, resulting in some anomalous results.

FUTURE WORK

Building upon the concepts discussed in Section 3.7, we anticipate this software pipeline to be

adapted for use within various other applications, as the only parts that would need changing would be the database itself, how it is read and the patterns used when approaching. spaCy supports the creation of custom models to be made for more accurate named entity recognition. By training it on specific data, such as recognising band and label names, the accuracy for the task at hand can be increased.

spaCy also features support for other languages, which would allow for this pipeline to work on multilingual datasets by changing the spaCy model that is being used, as well as changing the patterns to be recognised, as grammatical structure can vary between languages. For example, in Japanese, pronouns are not often used in sentences, which can leave some level of ambiguity, like in translation software.

As well as this, since the approach itself purely gets these relationships by pattern matching, it could be fed as the input to a machine learning algorithm which can accurately determine these relationships from text. We anticipate an approach involving machine learning and artificial intelligence to be a more effective approach to the given problem as, if trained properly, they can be made to effectively pick up every nuance within a piece of text, and can be specifically trained for a different use case.

REFERENCES

[1] Ghada Alfattni, Niels Peek, and Goran Nenadic. "Extraction of temporal relations from clinical free text: A systematic review of current approaches". In: *Journal of biomedical informatics* 108 (2020), p. 103488.

[2] Mike D (Arkatech Beatz). *The Rise and Fall of Major Record Labels*. https://web.archive.org/web/20211022194113/https://www.arkatechbeatz.com/the-rise-and-fall-of-major-record-labels. Accessed Mar. 4, 2025, Archived Nov. 22, 2021, Originally published in 2012.

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[4] Aric Hagberg, Pieter J Swart, and Daniel A Schult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Laboratory (LANL), Los Alamos, NM (United States), 2008.

[5] Joseph Hartnett. "Discogs.com". In: *The Charleston Advisor* 16.4 (2015), pp. 26–33.

[6] Stefan Heindorf et al. "Causenet: Towards a causality graph extracted from the web". In: *Proceedings of the 29th ACM international conference on information & knowledge management*. 2020, pp. 3023–3030.

[7] Matthew Honnibal et al. *spaCy: Industrial-strength Natural Language Processing in Python*. 2020. DOI: 10.5281/zenodo.1212303.

[8] Shammamah Hossain. "Visualization of Bioinformatics Data with Dash Bio". In: *Proceedings of the 18th Python in Science Conference*. Ed. by Chris Calloway et al. 2019, pp. 126–133. DOI: 10.25080/Majora-7ddc1dd1-012.

[9] Gregor Jossé et al. "Knowledge extraction from crowdsourced data for the enrichment of road networks". In: *Geoinformatica* 21 (2017), pp. 763–795.

[10] Hadjer Khaldi et al. "How's Business Going Worldwide? A Multilingual Annotated Corpus for Business Relation Extraction". In: *13th Conference on Language Resources and Evaluation (LREC 2022)*. European Lan-

guage Resources Association (ELRA). 2022, pp. 3696–3705.

[11] Rik Koncel-Kedziorski et al. "Text generation from knowledge graphs with graph transformers". In: *arXiv preprint arXiv:1904.02342* (2019).

[12] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[13] Arman Khadjeh Nassirtoussi et al. "Text mining for market prediction: A systematic review". In: *Expert Systems with Applications* 41.16 (2014), pp. 7653–7670.

[14] Gollapudi VRJ Sai Prasad and Amitash Ojha. "Text, table and graph–which is faster and more accurate to understand?" In: *2012 IEEE Fourth International Conference on Technology for Education*. IEEE. 2012, pp. 126–131.

[15] Maria Sicilia. *Knowledge Coproduction in Discogs Music Database: A study of the motivations behind a crowdsourced online discography*. 2020.

[16] Peng Sun et al. "An Overview of Named Entity Recognition". In: *2018 International Conference on Asian Language Processing (IALP)*. 2018, pp. 273–278. DOI: 10.1109/IALP.2018.8629225.

[17] Discogs Community Support. *Database Guidelines 20. Voting Guidelines — web.archive.org*. https://web.archive.org/web/20220504160555/https://support.discogs.com/hc/en-us/articles/360005055593-Database-Guidelines-20-Voting-Guidelines. [Accessed Mar. 10, 2025, Archived May. 4, 2022].

[18] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[19] Gerald Töpper, Magnus Knuth, and Harald Sack. "DBpedia ontology enrichment for inconsistency detection". In: *Proceedings of the 8th International Conference on Semantic Systems*. 2012, pp. 33–40.

[20] Gregory Walfish. *Artists vs. the industry: 5 notorious battles in music: Xposure music*. Accessed Mar. 5, 2025, Archived Mar. 5, 2025, Published Dec. 12, 2024. URL: https://web.archive.org/web/20250305150921/https://info.xposuremusic.com/article/artists-vs-the-industry-5-big-battles-in-music.

[21] Alison Wenham. *Independent Music is now a growing force in the global market*. https://web.archive.org/web/20190223054625/http://www.musicindie.com/independent-music-now-growing-force-global-market/. Accessed Mar. 4, 2025, Archived Feb. 23, 2019, Originally published Feb 1, 2014.

[22] Zhe Zuo et al. "Uncovering Business Relationships: Context-sensitive Relationship Extraction for Difficult Relationship Types." In: *LWDA*. 2017, p. 271.