Imperial College
London

Department of Computing

Imperial College of Science, Technology and Medicine

Using Graph Neural Networks and LLMs to Extract Knowledge from HTML files Using BERT-Based Models for the Purpose of Creating a Fact Database

Author:
Michael Hodgins

Supervisor:
Ovidiu Serban

September 10, 2025

Submitted in partial fulfillment of the requirements for the Masters of Science of Imperial College London

Abstract

This work presents a pipeline designed to provide a cheaper and more practical alternative to expensive LLMs like ChatGPT to extract facts from HTML sources. This pipeline is intended to be used to curate a large fact database for future use. The pipeline introduces a graph transformer based model to predict edges between HTML nodes that might provide richer context than summarising the individual elements on their own. It then introduces filtering and clustering techniques to extract the final relational edges between HTML nodes. This pipeline takes approximately 1 minute 15 seconds to run per average news article webpage, and costs appropriately $0.002US. A comparative evaluation against state-of-the-art open information extraction models on the SWDE expanded dataset shows that the Graph Transformer with filtering and clustering achieves the strongest balance of precision (0.83), recall (0.87), and F1 score (0.85). Unlike semantics-dependent models such as DOM-LM and GraphScholarBERT, this method relies mostly on structural filtering and clustering, enabling strong zero-shot generalisation across domains. These results highlight the effectiveness of graph clustering as a lightweight yet robust strategy for HTML relational extraction.

Acknowledgments

The project was supervised by Dr Ovidiu Serban from Imperial College London and co-supervised by Beth James from Ridgeway Information and Stehphen Francis from the IAEA. I would like to thank Ovidiu for his excellent supervision throughout this project. His guidance and support have been instrumental to its success. My sincere thanks go to Beth and Steve, whose thoughtful feedback was vital in shaping and strengthening this project.

Contents

Chapter 1

Introduction

1.1 IAEA Nuclear Non-Proliferation

The International Atomic Energy Agency (IAEA) (1) "is the world's intergovernmental forum for scientific and technical cooperation in the nuclear field". One of the IAEA's purposes is to ensure the use of nuclear material is for peaceful reasons, such as energy production within power plants. The IAEA (2) classifies nuclear material that fuels nuclear power plants as either "direct use" or "indirect use", determining whether it can be applied directly to create nuclear weapons. Due to the consequences of misuse, it is necessary to prevent access of nuclear material only to those that wish to use it for peaceful purposes. Within the International Atomic Energy Agency (IAEA), the IAEA Safeguards Department (3) exists "to deter the spread of nuclear weapons by early detection of the misuse of nuclear material or technology".

IAEA member states are committed to upholding three general safeguards for non-proliferation (2):

1. To detect any diversion of declared material at nuclear facilities or locations out of facilities.

2. To detect any undeclared production or processing of nuclear material at nuclear facilities or locations out of facilities.

3. To detect any undeclared nuclear material or activities in the state as a whole.

The IAEA ensures the first objective is achieved through nuclear material accounting (2). This is the activity of strictly monitoring the flows of nuclear material in and out of a facility. The second objective is partially ensured through the verification of information about features and characteristics of a facility. This verification involves (2):

- On-site inspections

- Nuclear material accountancy, including the reviewing of facility records and supporting documents

- Measurements of nuclear material (e.g. weight, gamma radiation, neutron measurements)
- Unique identifiers for nuclear material items
- Surveillance (cameras), containment (seals) and monitoring (nuclear material flows)
- Collection and analysis of environemnt and nuclear material samples
- Verification of facility design

The IAEA also ensures the second object through the collection of open-source data. This is usually sourced through news articles on the web, but is also acquired through other websites on the internet.

The IAEA collects both numerical and written report data. While numerical data is easy to analyse and uncover inconsistencies, it is harder to do this between multiple written reports. Algorithms cannot be applied to words as easily, and it takes slow manual labour to look through and compare reports. The IAEA would like to speed this process up, therefore it is necessary to create a pipeline that can read open source data from the web, and output facts. These facts could then be stored into a fact database to compare other open source data against.

1.2 Objectives

This project aims to create a pipeline that can be used by analysts to quickly extract a large number of facts from open-source data. This would allow for the creation of a fact database, useful to the IAEA to help point an analyst in directions that need further investigation. This needs to be in an easy to understand format. The aims are as follows:

1. **Extract facts from websites**. This pipeline will be able to extract a list of facts found on a website when provided its URL
2. **Extract facts from prose**. This pipeline will be able to extract a list of facts found within written reports
3. **Explain-ability**. All facts at every stage in the pipeline should be able to be backtracked to where it was found. This allows users to know where a fact came from, and can understand why it exists.

1.3 Current Methods

Large Language Models have advanced substantially over the past few years. Chat GPT 5 can now take an HTML webpage as input and output a very accurate and detailed summary of that webpage. It is possible to extract very high quality facts

using this method. However, the very large drawback to these LLMs is their cost. A much cheaper alternative that performs well for fact extraction is needed if a large fact database is to be curated.

The solution is to create a pipeline that does not require must input from advanced LLMs such as ChatGPT. This pipeline takes a website url as an input. The pipeline then outputs all the semi-structured relationships, and uses that knowledge to extract facts within that website. This includes all the facts within paragraphs (like in news reports), as well as short HTML snippets (e.g. " In 100 countries " is extracted alongside other nodes that provide context, so that nuanced facts can be found). The whole pipeline will retain the information on where those facts came from so that the IAEA can look into it further if they wanted. This pipeline is a generalisable pipeline, and can be applied to the IAEA's use-case of nuclear power articles and nuclear power companies. This pipeline can be used to curate a large set of facts from various sources, where future work can verify and flag contradictions in the database.

Chapter 2

Background

2.1 Language models

Language models give machines the ability to understand the semantics of text, enabling them to read, interpret, and reason with it. They have evolved from statistical approaches to the sophisticated neural architectures that now dominate natural language processing. The foundational concept behind modern language models lies in their ability to learn distributed representations of words and their contextual relationships.

Early approaches such as Word2Vec (4) and GloVe (5) provided static word embeddings, where each word had a fixed representation regardless of context. However, these models were limited in their ability to capture contextual nuances that are fundamental to understanding language.

The introduction of the Transformer (6) represented a breakthrough in language modelling. Using self-attention mechanisms, Transformers could capture long-range dependencies and contextual relationships more effectively than previous recurrent architectures. This paved the way for the development of powerful pre-trained language models that could be fine-tuned for downstream tasks.

BERT (7) (Bidirectional Encoder Representations from Transformers) was a significant advancement in contextual language modelling by using a bidirectional approach to text understanding. Unlike previous models that processed text unidirectionally, BERT considered both left and right context when generating representations for each token. This produces rich, context-sensitive token embeddings that capture both syntactic and semantic information.

2.1.1 FLAN-T5 and DeBERTa

BERT's success was followed by numerous variants and improvements, including DeBERTaV3 (8). DeBERTa improves on BERT in two ways: it contains a disentangled attention mechanism, and has an enhanced mask decoder. The disentan-

gled attention mechanism separates content and positional information, where each word is represented using two distinct vectors that encode its semantic content and structural position. This separation allows for more precise modelling of syntactic relationships and positional dependencies within sentences. The enhanced mask decoder incorporates absolute positional information in the decoding layer, providing better understanding of sentence structure and grammatical relationships. Together, these innovations improve the model's ability to capture complex linguistic structures, making DeBERTaV3 particularly effective for natural language understanding tasks.

FLAN-T5 (9) is a large language model that builds upon Google's T5 architecture; LLMs trained on instructional tasks. Unlike traditional language models that require task-specific fine-tuning, FLAN-T5 is designed to understand and execute instructions provided in natural language. The model uses an encoder-decoder architecture where the encoder processes input instructions and context, while the decoder generates responses. FLAN-T5 comes in a variety of sizes. This paper uses the FLAN-T5-Large model with approximately 770 million parameters. This provides good language understanding and generation capabilities while remaining small enough to be run on mid-powered GPUs. The model excels at tasks such as text summarization, question answering, classification, and text generation when provided with appropriate prompts and examples.

2.1.2 LLMS

While encoder-only models, such as BERT and DeBERTa, have transformed text understanding, recent advances in large language models (LLMs) have far surpassed the capabilities of BERT. These generative models can engage in complex reasoning, write code, perform mathematical calculations, and exhibit emergent abilities not explicitly programmed during training. Unlike BERT-style models that rely on bidirectional encoders to process existing text, modern LLMs are typically built on either decoder-only architectures that generate text sequentially, or encoder–decoder architectures that first encode input before producing output. They go through a much more intensive training process, and are must larger, scaling from millions to hundreds of billions of parameters. This has progressed their capacity to follow generalise to new tasks with few or no examples. Through widely available APIs, researchers can now combine the strengths of both models, such as encoder models like DeBERTa, while employing generative models for instruction following and text generation.

2.2 Graph based techniques

2.2.1 LapPE and random walk positional encoding

Using positional encodings as features can help graph-based models understand a graph's structure, allowing them to create more meaningful embeddings of the nodes. Global encoding can also provide more sophisticated representations beyond local neighbourhood features.

Laplacian eigenvector positional encoding (LapPE) captures global structural information by utilising the spectral properties of the graph Laplacian matrix L = D - A, where D represents the degree matrix and A is the adjacency matrix (10). The eigenvectors of this Laplacian serve as a coordinate system that embeds nodes in a meaningful geometric space, where the Euclidean distance between node embeddings reflects their structural relationship within the graph. By selecting the k-smallest non-zero eigenvectors, LapPE provides each node with k-dimensional coordinates that encode its global position. This allows models to distinguish between structurally different nodes even when they have similar local neighbourhoods.

Random Walk Positional Encoding (RWPE) focuses on local structural patterns by encoding the probability that a random walker starting from a node returns to itself after exactly k steps, for k = 1, 2, ..., K (10). This approach captures the local connectivity patterns around each node.

While LapPE excels at capturing long-range structural relationships and global graph topology, RWPE provides robust local structural information that is particularly effective for distinguishing nodes based on their immediate neighbourhood characteristics.

2.2.2 Traditional Graph Techniques

The Disparity Filter

In complex weighted networks, one can extract the most significant connections and output the backbone of a graph. This is especially true when edge weights span multiple orders of magnitude. The disparity filter, developed by *Serrano et al.* (11), addresses this challenge with a local statistical approach. It identifies edges carrying a statistically significant weight relative to a null model of uniform random weight distribution. For each node, the method calculates normalized edge weights and applies a statistical test. This determines whether observed weight variations exceed what would be expected from random fluctuations. As a result, it only preserves edges that represent structural importance.

When applying statistical significance tests across many edges, the Benjamini-Hochberg (BH) procedure (12) can control for multiple hypothesis testing. The BH method ranks p-values from all edge significance tests and applies an adaptive threshold.

This accounts for the number of simultaneous comparisons, ensuring that the proportion of falsely identified significant edges remains below a specified threshold. Combining local statistical significance testing with multiple comparison correction provides a framework for backbone extraction.

These methods offer an alternative to simple weight thresholds when extracting the backbone of a graph, and can better preserve the hierarchical structure present in real-world networks.

Louvain and Leiden clustering

Cluster detection algorithms identify densely connected groups within networks, such as social circles or functional units. The Louvain algorithm (13) can efficiently maximize modularity, a measure of community separation. Louvain works in two phases: moving nodes between communities for higher modularity and then grouping communities into super-nodes. This approach made it effective for large-scale network analysis. However, Louvain can produce disconnected communities, especially in later iterations, affecting up to 16% of cases. The Leiden algorithm corrects this with a refinement step that ensures all communities remain connected, while also producing faster and higher-quality results, making it a more reliable method for community detection. (13)

2.2.3 GNNs and Graph transformers

GNNs and Graph Attention

Graph Neural Networks (GNNs) represent the predominant framework for machine learning tasks on graph-structured data, utilising message-passing algorithms allowing for nodes to aggregate representations from their local neighbourhoods. Despite their widespread adoption, architectural limitations can constrain their effectiveness for tasks that require long-range dependency modelling.

Alon et al. (14) identified a critical information bottleneck in GNN architectures called ”over-squashing,” where exponentially increasing volumes of information from distant nodes must be compressed into fixed-dimensional vector representations as network depth increases. Although Graph Attention Networks (GAT) demonstrate improved resilience to over-squashing through selective attention-based message weighting, they remain susceptible to information degradation across extended graph paths. Over-squashing is particularly detrimental for applications necessitating long-range node interactions, as GNNs fail to capture patterns dependent on distant relationships, resulting in suboptimal performance despite the presence of relevant training signals.

Graph Transformers

These constraints motivated the development of Graph Transformer architectures, which circumvent the over-squashing issues in traditional GNN message-passing. They do this through global self-attention mechanisms that enable direct pairwise node interactions across arbitrary graph distances. Early graph transformer implementations such as TransformerConv (15) attempted to adapt standard transformer attention to graphs but suffered from quadratic $O(N^2)$ computational complexity, limited incorporation of graph structural information, and inadequate handling of edge features, severely restricting their scalability to graphs beyond a few hundred nodes. The GPS (General, Powerful, Scalable) framework (16) addresses these fundamental limitations of TransformerConv through several key innovations: achieving linear $O(N + E)$ complexity by decoupling local message-passing from global attention, systematically incorporating positional and structural encodings categorized as local, global, or relative features, and providing a modular architecture that combines efficient MPNN layers with linear attention mechanisms such as Performer or BigBird. Unlike TransformerConv's monolithic approach that forces all interactions through expensive quadratic attention, GPS preserves edge information through its MPNN component while enabling global connectivity through scalable attention mechanisms, effectively solving the scalability and structural encoding limitations of earlier Graph Transformer approaches. Empirical results demonstrate that GPS achieves state-of-the-art performance across diverse benchmarks while scaling to graphs with thousands of nodes, representing a crucial advancement from proof-of-concept implementations like TransformerConv to practical, scalable solutions for addressing the over-squashing problem.

2.3 DOM and HTML

When a browser receives an HTML file, it creates a hierarchical, tree-like structure whose nodes correspond with the elements, attributes and text content declared in the markup. Each node is typed (e.g., element, text, comment) and linked bidirectionally to its parent and children. This outputs a graph: the DOM tree. This tree provides us with machine-interpretable semantics, as headings sit at higher structural levels than paragraphs, list items are grouped under ordered or unordered lists, and so on. Acting as a bridge between static text and interactive behaviour, the DOM encapsulates both structure and meaning in a form that is uniformly addressable across platforms and programming languages. Training a machine learning model to understand the semantics hidden within the DOM tree specifically can yield better results than a pure language model that only understands text semantics.

2.4 Fact Extraction for HTML

Language models are great at understanding the semantics of a token given the surrounding context. This works well for written prose, where each sentence is co-

herent, and the next sentence follows on from the previous one. However, if we take an HTML file and flatten all the text into a paragraph, we are given some incoherent sentences and stand-alone words, which loses the important context held within the HTML structure. To extract facts held within the HTML structure, we must use a model that incorporates the HTML structure when outputting representations of our token. These representations will include the important context that our HTML provides.

Current models successfully use the HTML tree-like structure to improve upon base language models that ignore the HTML structure [(17), (18), (19), (20), (21)]. All these models focus on triplet extraction for either the OpenIE or ClosedIE task. However, these models all attempt to extract triplet relations directly from the token representations. No model attempts to output a phrase or sentence from every piece of text and then extract perform OpenIE or ClosedIE from this new text. I predict this extra step will improve upon current IE methods, as it will force the model to consider each text instance more carefully. Then more advanced OpenIE models designed specifically for coherent sentences can be used instead of these models.

2.4.1 Datasets and benchmarks

Web information extraction (IE) usually aims to achieve three types of task. The first is Closed Information Extraction (ClosedIE). This benchmark expects the model to retrieve fixed number of provided categories (e.g. <director, price, ISBN-13...>) and the model must extract the attribute. The second, Open Information Extraction (OpenIE), does not have a schema. The model must extract complete subject–relation–object triplets for any phrase that appears on the page. The final is Web question answering (QA), which judges the model on whether it can return the location (or yes/no) of the answer to a natural-language question about the content. There are a few datasets that most HTML language models have placed benchmarks against: SWDE (22), Extended SWDE (23), Common Crawl (24) and WebSRC (25).

SWDE

The Structured Web Data Extraction (SWDE) (22) corpus holds 124 291 HTML "detail" pages from 80 sites across eight types of webpages, called verticals (auto, book, camera, job, movie, NBA-player, restaurant, university). Each vertical declares 3-5 attributes per page, and the data is simple tuples <doc-id, attribute, value>, making SWDE the most common ClosedIE benchmark. To study OpenIE, OpenCeres re-annotated 21 of those sites (movie, university, NBA) with $\approx$ 856 k subject–relation–object triples over 27 641 pages, releasing the Expanded-SWDE.

Common Crawl

The Common Crawl (24) dataset contains 250 billion+ pages ($\approx$ 3–5 billion new each month) and serves two distinct roles. First, models (like DOM-LM (17) and MarkupLM (18)) pre-train on raw HTML drawn from a filtered slice (MarkupLM

keeps 24 million English pages after language and tag filtering) to teach their encoders DOM structure at scale. Second, several evaluation sets are extracted from Common Crawl. The WebFormer and MarkupLM papers create IE subsets for the domains events, products, movies. They mine pages that carry a schema.org markup, then test attribute extraction exactly like SWDE (22). Common Crawl QA turns Schema.org Question and Answer annotations into $\approx$ 130 million natural QA pairs; models are assessed with standard extractive-QA metrics (EM/F1) in zero-shot or fine-tuned settings.
Schema.org microdata is a blueprint for website developers to label their HTML to help search engines understand the page contents. And example can be seen in Listing 2.1 (26) showing how Schema.org can be added to the HTML to label the tags. This shows that all "itemprop" tags under the same "itemscope" tag are related.

Listing 2.1: Example of HTML with and without Schema.org

```
# Without Schema.org
    <div>
        <h1>Avatar</h1>
        <span>Director: James Cameron (born August 16, 1954)</span>
        <span>Science fiction</span>
        <a href="../movies/avatar-theatrical-trailer.html">Trailer</a>
    </div>
# With Schema.org
    <div itemscope itemtype ="https://schema.org/Movie">
        <h1 itemprop="name">Avatar</h1>
        <span>Director: <span itemprop="director">James Cameron</span> (born
            August 16, 1954)</span>
        <span itemprop="genre">Science fiction</span>
        <a href="../movies/avatar-theatrical-trailer.html"
            itemprop="trailer">Trailer</a>
    </div>
```

WebSRC

WebSRC contains 440 k question–answer pairs over 6.5 k real (therefore changing) webpages, each delivered with full HTML, a screenshot and node-level metadata. Questions require reasoning over the DOM hierarchy to answer (e.g., "What is the discounted price of the third product?"). The benchmark uses Exact Match, F1 and the Path Overlap Score, rewarding the model for correctly locating the answer in the DOM.

2.4.2 SOTA HTML Language Models

DOM-LM

DOM-LM (17) attempts to encode the semantics of the HTML DOM tree. Similar to BERT, a transformer encoder is used to create a representation of tokens, however

DOM-LM adds extra positional embeddings to encode the semantics of the DOM. DOM-LM has two main parts. The first part consists of creating an encoder that encodes the tree-level context. The second is to pre-train this encoder to capture the semantics of the DOM nodes with DOM related context. The output is a representation of the DOM nodes that can be fine-tuned for downstream tasks.
First, irrelevant nodes, such as <script>, are removed. The DOM is then broken into subtrees. This is because DOM trees can get long, and the whole tree would not be able to fit as an input into a transformer. This model therefore allows a large website to be processed. Once these subtrees have been created, the tree semantic data is extracted, and the tree is linearised before being inputted into the transformer.

When pre-training, a similar technique to pre-training BERT is used, where tokens are masked. However the whole flattened tree (including HTML tags) are tokenised, so the model predicts tags too. To prevent the model from only picking up local cues, entire nodes are also masked. DOM-LM used the SWDE dataset for pre-training. DOM-LM has a RoBERTa_Base backbone, and trains on the 120,000 SWDE websites.

The results show that after fine-tuning for downstream tasks, DOM-LM achieves:

- an F1 score of 54.1 and 55.1 for zero-shot OpenIE on the Movie and University categories of the expanded SWDE dataset
- an F1 score of 87.5 and 77.5 for few-shot OpenIE on the Movie and University categories of the expanded SWDE dataset
- an F1 score of 69.9 and 77.5 for attribute-level zero-shot ClosedIE on seed 2 and 5 of the SWDE dataset
- an average F1 score of 94.2 for attribute-level few-shot ClosedIE on the SWDE dataset
- an Exact match (EM) and F1 score of 69.7 and 73.9 for question answering on the WebSRC development set

Note that for few-shot learning, 0.5% of the data is used for training.

DOM-LM achieves significantly better results than baseline models (FreeDOM, SimpDOM...) while being much more lightweight. The authors directly mention MarkupLM (18) and theorise that DOM-LM is more generalisable.

MarkupLM

Like DOM-LM, MarkupLM attempts to encode the semantics of the HTML DOM tree. MarkupLM also uses a transformer encoder to create a representation of tokens, like BERT. MarkupLM flattens the entire text sequentially, then for each token, encodes it's string XPath along with its positional embedding. The XPath is string is embedded using a FFN that they trained. MarkupLM pre-trains the encoder so that it

understands the semantics of the embedded XPaths by asking it to predict the relation between different XPaths. The output is a representation of the DOM nodes that can be fine-tuned for downstream tasks. MarkupLM can therefore predict the next token, understand if the <title>matches the <body>and predict the relation between nodes.

Both a base and a large model were trained, each trained on 24M websites from the Common Crawl (24) dataset. This represents whether RoBERTa_base or RoBERTa_Large were behind each model. The results show that after fine-tuning for downstream tasks, MarkupLM achieves:

1. Base
    - an F1 score of 91.29 and 95.89 for page-level ClosedIE on seed 2 and 5 of the SWDE dataset
    - an EM, F1 and Path Overlap Score (POS) of 68.39, 74.47 and 87.93 for question answering on the WebSRC development set
2. Large
    - an F1 score of 93.57 and 97.37 for page-level ClosedIE on seed 2 and 5 of the SWDE dataset
    - an F1 score of 71.73 for attribute-level ClosedIE on the WebSRC development dataset (19)
    - an EM, F1 and Path Overlap Score (POS) of 74.43, 80.58 and 90.15 for question answering on the WebSRC development set
    - an EM and F1 of 85.33 and 89.84, 85.93 and 91.12, 78.67 and 82.28 for question answering on the WebSRC development set in the movie, events and products category respectively (19)

The paper also describes an ablation study, and shows that predicting the <title>matches the <body>improves the text prediction model more than the XPath relation prediction, however both improve the model.

MUSTIE

MUSTIE improves upon MarkupLM and DOM-LM by including images on the page, as well as the text extracted from those images. This does significantly increase the computational cost, however it performs better. It is specifically designed for ClosedIE; it requires as an input the attribute field to extract. MUSTIE is a multimodal transformer that learns the relationships between the modes. The modes it integrates are raw text, HTML and Images.

In the pre-processing step, MUSTIE creates a DOM HTML tree, where all the leaves are text or images. If text is detected inside the image, it is given the tag <OCR>and is added to the tree below the <img>tag. Then the leaves are indexed in the order

that they appear in the DOM.

MUSTIE can be broken down into three parts: the embedding layer, the MUST encoder and the extraction layer.

- The embedding layer embeds each DOM node, the text and the images. The leaves of the DOM tree (text or images) are embedded using BERT-base or ResNet101, with the addition of the type embedding (text, image or HTML node). The DOM node embeddings (non-leaves) are a summarisation of the embeddings beneath it in the tree, with the addition of the type embedding and tag embedding.
- The MUST encoder has twelve layers each with four attention patterns: structural attention between DOM nodes, bottom up attention from text/image tokens to DOM, top down attention from DOM to text/image tokens and local attention for contextual embedding of text/image tokens in the same leaf.
  - Structural attention: attention is between a DOM tag, its parent, siblings and children
  - Bottom up: each text/image token is attending to its direct parent DOM tag
  - Top down: each text/image token receives information from all DOM tags
  - Local: Each text/image token receives information from all text/image token in the same DOM leaf tag.

  All four attention patterns are weighted and summed to get the final attention
- The extractor uses a transformer decoder to look for the provided attribute field, and generates a response.

The results show that MUSTIE achieves the following:

- an F1 score of 73.42 for attribute-level ClosedIE on the WebSRC development dataset
- an EM and F1 of 75.68 and 81.13 for question answering on the WebSRC development set
- an EM and F1 of 87.79 and 92.32, 87.67 and 93.37, 82.3 and 85.41 for question answering on the WebSRC development set in the movie, events and products category respectively

MUSTIE performed an ablation study and found that the bottom-up attention pattern is the most effective attention pattern (other than the necessary local attention on which LMs are built). This is because it is the pattern that connects the text/image information with the DOM. They also note that the HTML mode increases performance more than the image mode.

MUSTIE acknowledge two limitations. The first is they only assume one field attribute per page, and that DOM nodes are not masked, therefore the model does not understand the relations between the DOM nodes. They also only include attention between local DOM nodes, and therefore long range DOM connections may be missed.

GraphScholarBERT

GraphScholarBERT uses graph attention and the scholar-BERT language model for ClosedIE. Given a target field and short description of the relation, GraphScholarBERT extracts the relevant information. It is aimed to be zero-shot generalisable to all domains.

GraphScholarBERT's architecture can be split into two parallel flows. The first flow inputs the HTML tree into a GAT edge classifier, understanding whether two HTML nodes are related. Here, two GAT convolutional layers are followed by an edge feature construction layer. The input DOM tree has the node features: tag, index and text frequency. The edges of the DOM tree are all the direct connections, and virtual edges are added between nodes that have other relations. The second flow is a language model that inputs the query attribute field and description, along with a possible relation and value. The model then guesses whether the (query, relation, value) triplet is a valid triplet. The output of the GAT flow and the LM flow is inputted into a final classification head, which decides if the (relation, value) pair follows the query prompt.

GraphScholarBERT trained and intra-vertical and inter-vertical model. This denotes whether the model was shown a similar webpage's vertical during training. The inter-vertical model is more generalisable, whereas the intra-vertical model is trained for a specific vertical. The results for the inter-vertical model are as follows:

1. Inter-vertical
    - an F1 score of 75.0, 51.0 and 60.0 for zero-shot OpenIE on the Movie, University and NBA categories of the expanded SWDE dataset.
2. Intra-vertical
    - an F1 score of 80.0, 59.0 and 74.0 for zero-shot OpenIE on the Movie, University and NBA categories of the expanded SWDE dataset.

2.4.3 Model Comparison

DOM-LM and MarkupLM are very similar models, both attempting to create a BERT-like model that takes into account the DOM tree. The differences lie in how the models are structured. In DOM tree semantics, MarkupLM learns the relationships, whereas DOM-LM directly encodes the relationships from the start. This

means DOM-LM does not have to learn (potentially incorrectly) the relations between nodes. However, by nature of how MarkupLM is structured, it can also predict if an HTML title belongs to the body, as well as predicting the relationship between distant nodes; something DOM-LM cannot. The only similar results that the two papers both report and can be compared on is the QA experiment on the WebSRC dataset. Here, we can see that DOM-LM performs comparably to MarkupLM_Base, whereas the larger MarkupLM_Large allows it to outperform DOM-LM. DOM-LM however has about 13 million less parameters than MarkupLM_Base (as it does not need a model to embed XPaths), and is trained on 120,000 websites as opposed to MarkupLM's 24M. A better base model would also improve both models.

MUSTIE performs better than MarkupLM_Large in all aspects. MUSTIE attributes their superior performance over MarkupLM_Large as MarkupLM encodes text and HTML separately and concatenates the embeddings before inputting into the transformer, instead of using multi-modal cross-attention. They also include information about images within their model, which MarkupLM ignores. MUSTIE is a much larger model, and taking into account multimodal inputs makes it slower if applied in an IE pipeline.

The issue with DOM-LM, MarkupLM and MUSTIE is that they are all supervised techniques that are not that generalisable to webpages, or verticals, that they have not seen before. DOM-LM is the only model that reports on zero-shot OpenIE, which is an objective of this paper. GraphScholarBERT attempts to improve upon zero-shot OpenIE, and achieves promising results in some categories. It improves upon DOM-LM's F1 score of 54.1 to 75 in the movie category in zero-shot learning, but does 4 points worse (55.1 to 51.0) in the university category. This implies that this model is much more capable in reading HTML with less complicated HTML, however when sentences are complex, the models perform comparably.

Chapter 3

Contribution

3.1 Overview

The most promising method in literature for zero-shot webpage information extraction is GraphScholarBERT (27)(22). This is the model from which this project is based. GraphScholarBERT has three major drawbacks:

1. The architecture uses graph attention to get an embedding for a node - GAT is known to struggle with capturing long-range dependencies (14), which are common in webpages.

2. It requires a search keyword and description to classify a pair of nodes as "related" - this makes the model more of a closedIE than openIE model.

3. The concatenated GAT edge and BERT embeddings are placed through a FFN to identify a relation. - This allows the model to rely on text content to define edges. This works for simplistic domains where text nodes are short phrases, but it does not work for complicated nuclear paragraphs. ScholarBERT would struggle to tell if two text nodes are similar or not, limiting the generalisablilty of the model.

This paper makes two architecture changes:

1. It uses Graph Transformers to enable the model to better capture long range structural patterns in webpage layouts.

2. It removes the parallel BERT model, and instead uses both traditional graph techniques and BERT-based models to filter edges down. This prevents the edge prediction from relying on text, and removes the search keyword requirement.

Explain-ability

Explain-ability is important to be able to support all facts and claims that are outputted by the pipeline. It instils confidence that the pipeline is not outputting incorrect statements, and if it does, explain-ability helps rectify mistakes. Without

explain-ability of the facts, the user would need to go through a time consuming process to justify the claim.

This pipeline uses XPaths to explain where the facts originated from. With each fact outputted, supporting XPaths are outputted, describing where the claim originated from. All the XPaths that went into making a claim are outputted, so that all the context is also traceable on the page.

3.2 Data

Two different data sources were considered: the SWDE expanded dataset (22) and Schema.org websites from Web Data Commons (28).

3.2.1 SWDE dataset

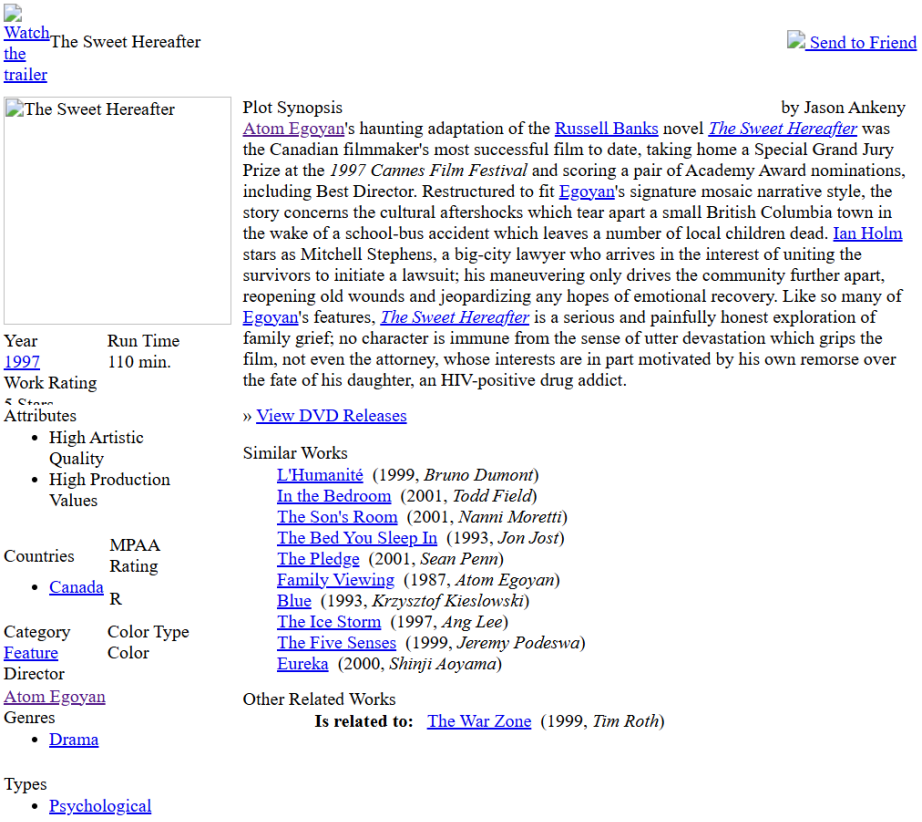
SWDE expanded dataset

The expanded SWDE dataset (22) was used to train the graph transformer model. The origin of the 27 641 .htm files comes from 21 different websites domains (e.g. 2000 pages from IMDB). The domains came from three different categories: Movie, NBA player profiles, and University. Each webpage has corresponding JSON entries, similar to Listing 3.1, where key-value pairs are the expected text relations to be extracted from each page.

Listing 3.1: SWDE Expanded dataset snippet. *Age* and *Birthplace* are expected to be extracted from *0000.htm* from the *nbaplayer-fanhouse(446)* domain.

```
{
    "0000.htm": {
        "Admission Difficulty:": [
            "Very Easy"
        ],
        "Comparable Schools": [
            "Aaron's Academy of Beauty",
            "Abdill Career College Inc",
            "Dixon Nursing",
            "Abraham Baldwin"
        ],
        "Undergrad Student Body | Full-Time:": [
            "109"
        ],
        ...
```

Listing 3.1 shows three examples of relations. One heading to one value, one heading to many values, and a hierarchy of headings (split by '|') to one value. All keys and values match the text of an HTML node. Relational pairs were extracted between a value and its direct parent only. Keys with direct parents were also treated


Watch the trailer The Sweet Hereafter

Send to Friend

The Sweet Hereafter

Plot Synopsis by Jason Ankeny

Atom Egoyan's haunting adaptation of the Russell Banks novel *The Sweet Hereafter* was the Canadian filmmaker's most successful film to date, taking home a Special Grand Jury Prize at the *1997 Cannes Film Festival* and scoring a pair of Academy Award nominations, including Best Director. Restructured to fit Egoyan's signature mosaic narrative style, the story concerns the cultural aftershocks which tear apart a small British Columbia town in the wake of a school-bus accident which leaves a number of local children dead. Ian Holm stars as Mitchell Stephens, a big-city lawyer who arrives in the interest of uniting the survivors to initiate a lawsuit; his maneuvering only drives the community further apart, reopening old wounds and jeopardizing any hopes of emotional recovery. Like so many of Egoyan's features, *The Sweet Hereafter* is a serious and painfully honest exploration of family grief; no character is immune from the sense of utter devastation which grips the film, not even the attorney, whose interests are in part motivated by his own remorse over the fate of his daughter, an HIV-positive drug addict.

» View DVD Releases

Year 1997

Run Time 110 min.

Work Rating

Attributes
- High Artistic Quality
- High Production Values

Countries
- Canada

MPAA Rating R

Category Feature

Color Type Color

Director Atom Egoyan

Genres
- Drama

Types
- Psychological

Similar Works
- L'Humanité (1999, *Bruno Dumont*)
- In the Bedroom (2001, *Todd Field*)
- The Son's Room (2001, *Nanni Moretti*)
- The Bed You Sleep In (1993, *Jon Jost*)
- The Pledge (2001, *Sean Penn*)
- Family Viewing (1987, *Atom Egoyan*)
- Blue (1993, *Krzysztof Kieslowski*)
- The Ice Storm (1997, *Ang Lee*)
- The Five Senses (1999, *Jeremy Podeswa*)
- Eureka (2000, *Shinji Aoyama*)

Other Related Works

Is related to: The War Zone (1999, *Tim Roth*)

Figure 3.1: Example website from the SWDE dataset (22)

as values of a relational pair. Relational pairs are provided only between text that can be found in different HTML nodes. This means that the data provides a way to link HTML nodes together. The relations do not provide labels for relational values found in the same paragraph, meaning the JSON files to not provide all the openIE facts within a website (say, within a paragraph). The dataset is designed to inform the user of inter-html-node relationships.

For each webpage, there is a special entry in the SWDE json to denote the title of the webpage. This label is the title's text, and it lies under the key *"topic_entity_name"*.

The 2000 pages from the *allmovie* domain was held aside for testing. This domain was chosen to keep the training set balanced, as the movie domain-type contained the highest number of pages. Secondly, this domain type contained an expansive set of relational pairs to test on.

Only the .htm files were provided. A lack of JavaScript or images leads to the websites looking bare. Figure 3.1 is an example layout. This semi-structured pages layout are the types of pages that the model is trained on. This is a drawback to this dataset, as the interest of the IAEA is to extract information from news articles.

Converting .htm Files to Graphs

Each .htm file was converted into an XML tree. The html contained many errors and needed to be cleaned before converting. The parser from the Python package "html5lib" (29) cleaned most errors, however, it did not handle incorrect hex character codes and literal C0 bytes. These were removed from the text. A list of "approved" HTML tags was constructed, and any tag not found in this list was removed in its entirety. Tags like <script>and <noscript>makes the DOM long and noisy. Removing them normalises the graph, keeping only those nodes that can display elements on the screen. The list can be found in Listing 6.1.

The XML tree extracted from each .htm file was processed to extract node and edge features. The node features' purpose is to help the model understand the structure of the HTML. Therefore, the chosen features were:

- *Tag* - one-hot encoded vector describing the HTML tag.
- *Sibling Number* - The index of the node between its siblings.
- $k = 8$ *LapPE* - Adds a global structure feature, as described in section 2.2.1.
- *Depth degree* - Adds a global structure feature
- $k = 6$ *RWPE* - Adds a local structure feature, as described in section 2.2.1.

The edge features' purpose is to help the model understand the relationship between HTML nodes on the webpage. Therefore, the chosen features were:

- *Tag and Sibling Number of both nodes* - This allows the edge embedding to recognise the relationship between nodes
- *Hops* - the number of hops between the nodes. This helps gauge distance within the graph
- *Webpage relative position* - this helps the edge understand the relationship the nodes have on the webpage. All relative position are normalised by the viewport width and height. These positional features are:
  - *Difference between the X and Y coordinate of the nodes.*
  - *Angle between the centres of the nodes.*
  - *Difference between the widths and the heights of the nodes.*
  - *Ratio of the widths and the heights of the nodes.*

These features could all be extracted from the XML graph, except for the *webpage relative position*. Selenium (30) was used to extract this information. To ensure a quick, error-free extraction, JavaScript was disabled.

Both positive and negative edges were extracted between text nodes to create the label edges. The positive edges were extracted using normalised string matching.

The exact text of a node often comes with noise, either too many spaces, a special character, or a miss-capitalisation. Using Listing 3.1 as an example, both “admission-difficulty” and “veryeasy” were searched for on the webpage. Due to the abundance of text, often these appear more than once. The pair of nodes that were the least number of hops away from each other are selected as the correct nodes to join.

The negative edges are randomly picked between two nodes that have been seen in the positive edge labels, and that are fewer hops than the largest positive hop. 2% of the edges were allowed to be any number of hops. This is done so that we teach the model to label edges between only relevant nodes (text nodes), and because we know which edges are negative in this set. Randomly assigning any edge as negative might accidentally return an unlabelled positive edge, and would provide many irrelevant edges. The positive and negative edge list contained the same number of edges to create a balanced edge-label set.

Due to the highly sparse nature of the embeddings (almost all values in the matrix are 0), the features were stored as Numpy’s Compressed Sparse Row format (31).

3.2.2 Web Data Commons

Common Crawl (24) provides the largest publically available database of websites. Web Data Commons (WDC) (28) extracts a subset of Common Crawl that contains Schema.org annotations. Schema.org microdata can be used to link nodes together through sibling “itemprop” nodes under an “itemscope” tag, as seen in Listing 2.1.

This dataset was explored, but eventually rejected as a source. There is no quality control on the HTML when crawling the internet. This leaves the database full of poorly annotated Schema.org. Most websites only contain one “itemscope”, and those that contain more are often annotating webpage navigational buttons, or products. It was found that the Schema.org real-world usage had a narrow scope. It was also evident that two sibling “itemprop” tags occasionally did not actually semantically relate to each other, which would lead to noisy data. The lack of quality control also lead to some pages not loading properly.

This paper decided to train a graph transformer model with the well annotated SWDE dataset, and supplement it with some WDC pages only if the SWDE dataset proved to poorly generalise.

3.3 HTML Aware GNN

3.3.1 Graph Transformer

HTML graphs were batched together before being fed through the network. To form a batch, N graphs were combined to form one graph of N disjoint graphs.

Architecture

The model's task is edge prediction. It predicts whether a prospective edge exists or not. This is done by passing the HTML graph through a graph transformer to output the node embeddings. A concatenation of the two node embeddings of the edge, and the edge embedding itself, is passed through a two layered FFN.

The graph transformer is built from six GPSConv layers (16). Each layer combines message passing with transformer-style attention. This allows the model to capture local neighbourhood structure, through the GINEConv component, as well as long-range dependencies through the global attention mechanism. Therefore, the model learns node embeddings that encode both fine-grained edge features and higher-level structural context. This makes the resulting node embeddings richer and more expressive for the downstream edge prediction task.

The FFN consists of a ReLU-activated linear layer to output a total edge embedding from the node and edge embeddings. This embedding is then classified by a second linear layer, which outputs a logit, defining the probability of the edge existing.

A second FFN designed for node title prediction was constructed in parallel to the first FFN. The FFN was designed to extract the webpage title node using the node embeddings. Adding this title prediction was theorised to teach the model some absolute features about the graph, providing more accurate node embeddings.

Loss function

Three different loss functions were tested. As this is a binary classification task, binary cross-entropy (BCE) was attempted. An alternative to this loss function is the focal loss (32), defined as

$$\mathcal{L}_{focal} = -\alpha(1 - p_t)^{\gamma} \log(p_t) \tag{3.1}$$

where p_t is the predicted probability for the true class, γ is the focusing parameter, and α balances positive vs. negative classes. This prevents the loss from being dominated by easy, correctly classified examples. Instead, it down-weights those easy cases and puts more emphasis on the hard, misclassified ones. This loss function worked better as it is easy to classify a negative edge as negative, however positive edges are much harder to classify.

To introduce a ranking comparison between edges, a small pairwise AUC loss was added, defined as

$$\mathcal{L}_{AUC} = \max\left(0,\, 1 - (s^{+} - s^{-})\right) \tag{3.2}$$

where s^{+} and s^{-} are the predicted scores for a positive and negative edge, respectively. This allowed the model to rank edges correctly, even when it was unsure of the absolute probability. This works well, as we care about the top ranked edges

when adding context, not the raw probabilities. The total final loss was a combination of 90% focal loss and 10% AUC loss.

Model Parameters

The model was trained using the AdamW optimiser, which uses adaptive learning rates and can provide a more generalisable model. To decide those learning rates, the OneCycleLR learning rate scheduler was used (33). This schedule gradually warms up the learning rate to a maximum value before annealing it down using a cosine strategy. This helps the model escape sharp minima early and settle into flatter regions of the loss landscape, often leading to better generalisation (33).

The parameters used for training can be seen in Table 3.1

Parameter	Value
Epochs	400
Batch size	512
Beginning LR	$1 \cdot 10^{-5}$
Max LR	$4 \cdot 10^{-4}$
Final LR	$4 \cdot 10^{-8}$
GPS Layer size	6 Layers: 112 Channels
Edge embedding size	32
Heads	4

Table 3.1: Model Parameters

Metrics used

As edge prediction is a binary class prediction, the evaluation metric was the F1 score. At first the probabilities were rounded to classify an edge. This set a hard threshold of 0.5. Any probability above yields a positive edge and any probability below yields a negative edge. This was found to be a bad way at predicting positive and negative edges. Most of the time, the model was low in confidence on an edge existing. Almost all the edges had a probability less than 0.5.

Since it was known that half the edges were positive, and half were negative, the threshold for a positive and negative class was set as the median probability of all the predictions. This allowed the F1 score to reflect how well the model was doing at ranking edges. This is a better threshold, as the model needs to predict the best edges, and how they rank against each other, rather than provide accurate probabilities. Note that this forces $precision = recall = F1$.

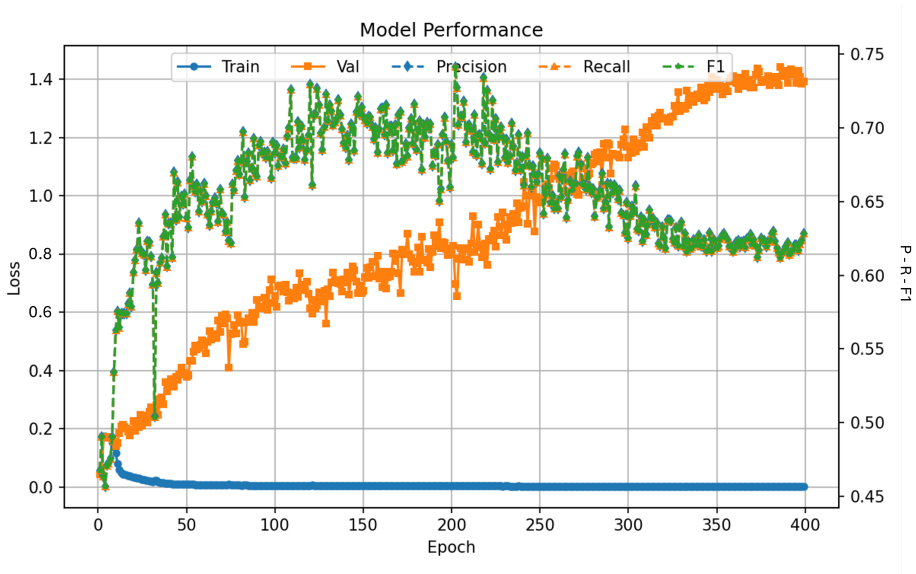


Figure 3.2: The training loss of the model

Edge dropout and warmup

Early on it was found that the FFN would ignore the node embeddings, and make a prediction solely on the candidate edge's features. This meant that the graph transformer would not learn, and the model would collapse to only a two layered FFN. Two techniques were tried to combat this. First, an edge feature dropout was introduced before concatenating node and candidate edge features. This forced the final FFN to make a prediction based off the two node embeddings it was being fed, rather than the candidate edge features. The second was a warm-up epoch, where all candidate edge features were zeroes for the first epoch. This forced the model to only use the node embeddings to make a prediction. This was designed to kickstart the graph transformer learning.

It was found that the best solution to this issue was parameter tuning. Too high a learning rate would stop the model from learning the node embeddings and lead it to only predicting on the simpler loss landscape of the edge features.

Loss curve

Figure 3.2 shows the training loss, validation loss and the F1 score during training. It should be noted that an F1 score of around 0.5 means the model is no better than random. The model took 31h to train on one NVIDIA A100 80GB GPU, however the optimal model was found after 20h.

3.4 Extracting Edges from Webpages

Once the edge prediction model was trained, the next stage in the pipeline was to use the predicted probabilities to refine the edge selection, and create mini-graphs

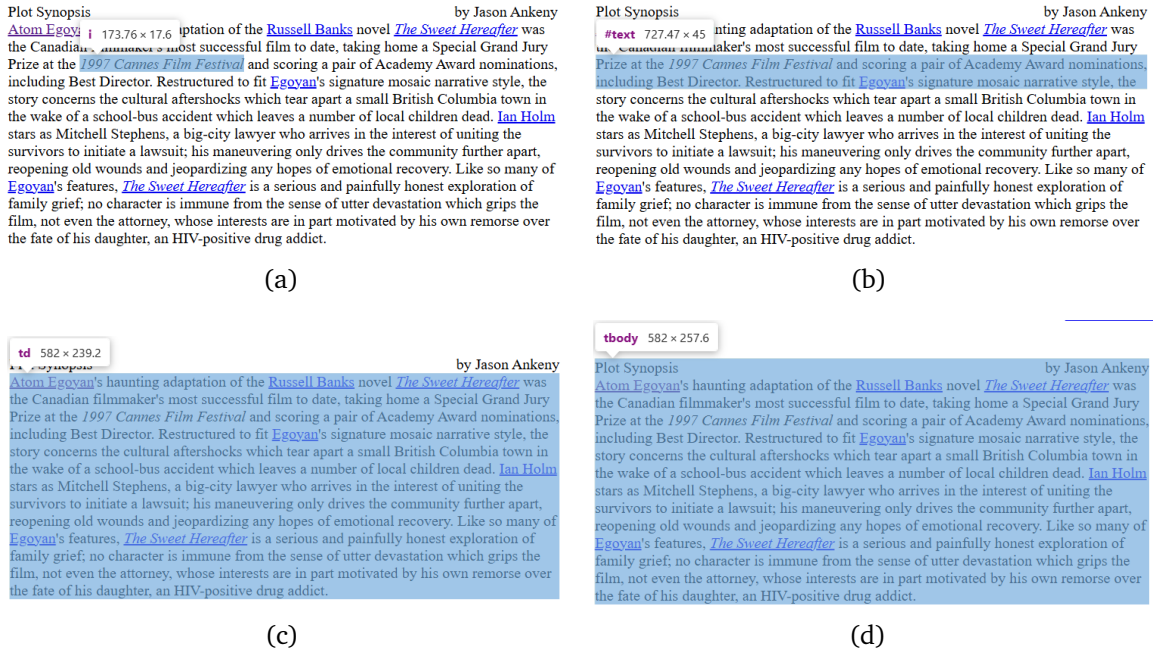


Figure 3.3: Shows all the different HTML tags that can be found to make up one chunk of text in an HTML paragraph.

where all the nodes are related to each other.

3.4.1 Extracting All Candidate Edges

Extracting all text nodes

We want to find all the possible edges that provide context within a website. To do this, we need to extract all the text nodes from the graph and examine the edges between them.

Extracting the correct text nodes from an HTML graph is not a trivial task. We cannot just extract all the HTML nodes that contain text. Figure 3.3 illustrates this challenge; which text node best represents this paragraph?

Most well built websites place well defined chunks of text inside the listed tags found in Listing 3.2. The websites of interest are news articles and large company websites. Therefore this is a fair assumption to make.

Listing 3.2: Common HTML text tags

```
# Tags text is found in
    p, li, dt, dd, blockquote, pre, h1, h2, h3, h4, h5, h6, figcaption
# Tags to skip over
    script, style, noscript, template, button, nav, footer, header
```

Every tag of these types are extracted, with a few exceptions.

- Any tag hidden by inline style
- Any empty tag
- Any tag within the a classname containing *hidden*, *footer* or *navbar*
- Any text tag inside a tag to skip over (Listing 6.1)

To pick up forgotten text, three more tag types were extracted only if their text had not been extracted: *td*, *a*, *div* and *span*. These nodes had the added condition that they had to be the deepest element of its type that contained text. This avoids the extraction of a large chunk of the website that is all encapsulated by one *td* tag.

This method still extracts duplicates, i.e. two different HTML node that refer to the same text chunk. These duplicates are always an ancestor or child of the other. These duplicates were kept, and deduplicated after edge probabilities were found. This allowed us to keep the highest edges per text chunk, and remove any influence of the HTML node chosen to represent a text chunk. Deduplication was directed, i.e. two edges always survived between two text nodes; one in each direction.

Predicting edges

A directed edge was assigned between every pair of text nodes extracted (two edges per pair of text chunks). The model outputs predicted probabilities per edge. The issue now lies in how to select which edges are the true edges. While training our edge prediction model, we knew half the outputted edges were positive and half were negative. This allowed us to assign the top half edges as positive, and the bottom half as negative. We have no such assurances now.

Arbitrarily picking a probability threshold is a bad choice. This is because the range of probabilities differs between websites. More complicated HTML leads to lower probabilities. Equally, ranking the edges per node and only keeping the top k edges is a bad choice. The results of the top k hits are shown in Section 4.1. It was found that hits were often found in the top 10 edges, however, occasionally they were not. It also assigns k edges to a node that only has one true connection, and k edges to a node that has many more valid edges. The following sections outline the filters and traditional graph techniques that were used.

3.4.2 LLM and Graph-based Edge Filters

Up until now, edges have been predicted using HTML structure only. Text semantics and traditional graph techniques are now introduced to further refine the edges. To filter on semantics, the FLAN-T5 (9) model was used. FLAN-T5 was chosen as it understands semantics, and has been fine-tuned to follow instructions.

Buttons and Navigational Links

The first filter is for removing all text nodes that appear to be a button or navigational link. This is designed to remove nodes that contain no important information or facts. A list of examples can be seen from the prompt in Listing 3.3. FLAN-T5 (9) was chosen as this filter due to its great performance at few-shot learning.

Listing 3.3: FLAN-T5 button and navigational link prompt

```
    '''Decide if the text looks like an website button or navigation label.
Return 1 for button/navigation, or 0 otherwise. Output only a single digit.

Examples of 1: "Read more", "Learn more", "Explore", "Get started", "Try
    free", "Watch video", "Sign in", "Sign up", "My account", "Download",
    "Contact", "Home", "About", "Privacy Policy", "Terms", "Fr/En", "Menu",
    "Next", "Previous", "Back", "Dashboard", "Add to cart".

Examples of 0: descriptive copy, names of people or products in context,
    long summaries, and info labels such as "Best seller", "Made in USA" or
    "Free shipping on orders over $50"

TEXT: {text}

Answer with 1 or 0 only:'''
```

Low Probability Edges

Serrano et al. (11) describe a method to extract the backbone of a graph with edge probabilities. It tests whether each edge probability is significantly larger than expected. Since there are so many edges to prune, it is unwise to only compare each edge against a null hypothesis. A 0.05 significance level would still allow 5% false edges through. The Benjamini–Hochberg (BH) procedure described in section 2.2.2 is applied to counteract this.

An edge successfully survives pruning if either the outgoing or incoming edge passes the pruning. It is possible for a node to have all of its edges pruned using this technique. Therefore, if this were to happen to a node, its two highest outgoing and incoming edges are restored. This ensures that the backbone retains only statistically significant edges, while not isolating nodes.

This results in a pruned graph, where all statistically insignificant edges are pruned. Since the model does not output the true probabilities, a generous significance level of 10% is applied.

This pruning step is done after the button pruning step as it would not be sensible for these edges to affect the null probabilities, since they will be removed anyway. This is performed before the semantic filtering step as that is a more intensive process.

Reducing the candidate edges allows the pipeline to run quicker.

Semantically Unrelated Edges

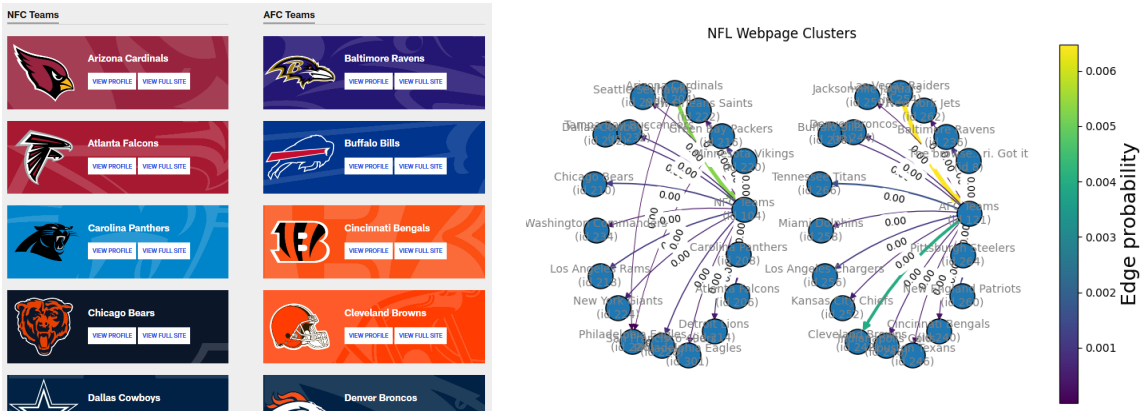
The last filter removes edges where the text nodes are semantically different, and have no added context to share with the other. This removes all edges that do not help with information extraction. FLAN-T5 (9) was chosen as this task requires semantic understanding between two phrases, which FLAN-T5 performs well at.

Classifying whether two entries could provide context to one and other can be a challenging task. At first, the model was asked to classify whether a pair of nodes could provide context that the other was missing. However FLAN-T5 would usually return a positive value if any information could be added from one to the other. Therefore, the prompting was split into two stages.

The first pass through FLAN-T5 asked if the pair of nodes could be relevant to each other. The second pass asked whether one contains any contextual information that the other is missing. This extra first pass allowed the rejection of completely unrelated edges, even when information from one could be added to the other. The prompts used are found in Listing 3.4.

Listing 3.4: FLAN-T5 Semantic link prompts

```
# First prompt
    '''Classify the contextual relation between the L and R text:

Choose only one classification:
    2: The meaning of L is unclear OR the meaning of R is unclear;
    1: The meaning of L could be related to the meaning of R;
    0: The meaning of L is definitely completely irrelevant to the meaning
       of R;

L: {left}

R: {right}

Answer with the number only:'''

# Second prompt
    '''For the purpose of fact extraction, classify the relation between
       the L and R text:

Choose only one classification:
    0: L only contains contextual information that R already has;
    1: L contains any key contextual information that R is missing;

L: {left}
```



(a) (b)

Figure 3.4: Example cluster of an NFL teams webpage (34)

```
R: {right}

Answer with the number only:'''
```

3.4.3 Extracting mini-graphs

Now that the graph has been pruned, only the most probable edges exist. This allows us to extract text graphs that contain useful information with substantially less noise. The best way is through the Leiden clustering method (13) described in section 2.2.2. The method is applied on the current text graph, and it uses the edge probabilities to output the most likely clusters. These clusters are then turned into mini text-graphs. An example can be seen in figure 3.4. In preparation for the next stage, the Leiden clustering was recursively applied until each cluster contained a maximum of 30 words, or Leiden clustering could not partition the graph further. 30 words was chosen as any more, and FLAN-T5 would become overwhelmed and miss relations.

3.5 Fact Extraction

To extract facts from the text graph, two different methods were used. For the short, semi-structured text, FLAN-T5 was prompted on the mini-text clusters. For the more complicated and long form text, LLMs were prompted per edge to to see if the connection provided extra context for a target node. Since this required nuanced understanding of the next, an OpenAI API was used to target larger models than FLAN-T5.

The use of OpenAI means that there needs to be a more selective approach to extracting information to keep costs low. Since the webpages of interest are news articles, long form text should be targeted, and context from its edges should help

the complete missing or ambiguous information. This means splitting the nodes into those that contain phrases, and those that do not; i.e. nodes that receive context and those that give context. A prompt is also used to classify if the long form text needs context, or if the sentence is unambiguous and fully self-contained. Another reason for doing this is that the facts live in the long form text. Short phrases do not hold facts, they only provide context.

3.5.1 Prompting Flan-T5 on the Semi-Structured Text Clusters

The mini-text clusters were prompted through FLAN-T5. The prompt asked the model to extract information from the collective "knowledge base" provided. As a side task, it was implied that if not enough information is provided to create a fact, that it should match up the relational pairs it finds inside the cluster. The prompt used can be found in Listing 3.5.

Listing 3.5: FLAN-T5 prompt for summarising clusters

```
    '''Instruction:
The different INPUTS are related to each other.
Group the similar INPUTS together to create a list of facts.
Output the relations and the list of facts.
Add no external information.

Example 1:

INPUTS:
- Doctor
- Address
- 43 Palace Gardens, Newcastle
- The date is 1968
- Alexander Evans
- March

OUTPUT (4 facts):
1. The Doctor is Alexander Evans.
2. The Address is 43 Palace Gardens, Newcastle.
3. The date is March, 1968.

Example 2:

INPUTS:
- Challenging
- Thoughts:
- Rewarding
- Acceptable

OUTPUT (3 facts):
1. The Thought is Challenging.
```

```
2. Thoughts: Rewarding.
3. Thoughts: Acceptable.

Now do the same for the following INPUTS:

INPUTS:
- {INPUTS}
OUTPUT (~{N} facts):'''
```

3.5.2 Classifying Nodes as 'Phrases' and 'Non-Phrases'

DeBERTa-v3 (8) was used to classify whether a node was a complete phrase or not. This is preferable to simplistic processing (for example, *"Are there more than 7 words"*) as *"Leeds vs Bournemouth score prediction"* is a phrase but *"At 04 November 2022 15:35"* is not a phrase. Context should be added to one, and extracted form the other. DeBERTa-v3 was found to perform better than FLAN-T5 at this task. This is because it is not a generative mode, and can utilise a bidirectional transformer. Bidirectional context encoding allows for a better understanding of sentence structure and the roles of individual words within a sentence, and therefore is better and distinguishing complete phrases.

DeBERTa-v3 was used to classify whether a statement entails or does not entail (is neutral or contradicts) a hypothesis. The hypothesis used was *"This text contains at least a sentence (ignoring punctuation)"*. This did well enough to classify *"Leeds vs Bournemouth score prediction"*. The word *"phrase"* was replaced for *"sentence"*, however the model began to classify everything with more than a few words as a *"phrase"*.

3.5.3 Classifying Context Requirement

For those nodes that are sentences, another classification must be done to see if the sentences need context to fill in a gap in knowledge. Prompting an LLM to classify this is difficult. It requires reasoning over the text to understand when something is missing, rather than just generating the next most likely token. This means the OpenAI API is needed to do this classification.

This is a minimal cost prompt, since it only requires one output token per edge, *"0"* or *"1"*. Therefore the gpt-4.1-mini model was used. This is because its reasoning capabilities are high, which is required. Cost is a consideration, however with only one output token per edge, it is not as an important factor as reasoning capability. The model temperature was set to 0, so the highest probability token is outputted every time.

The prompt for this task can be seen in Listing 3.6. It was found that, depending on the length of the node's text, this model could accurately handle two nodes at once.

Figure 3.5: A screenshot of `https://westinghousenuclear.com/`. It shows three descriptions that need context from their title.

Listing 3.6: Open AI prompt to classify a sentence as needing context or not

```
    '''TASK:
    You are a fact-finding classifier. Read the QUERY text and assign it
        one classification. Output the classification number for each pair

CLASSES:
    0: All subjects are known. The text does not reference or allude to
        something unknown;
    1: There are key unknown subjects referenced;

Classify each, one by one:

{INPUT}

Output {N} space separated integers ONLY:
'''
```

3.5.4 Classifying entity category

Those nodes that are not phrases give context to those that are phrases. There is an issue, however, where two edges can give conflicting context. Figure 3.5 demonstrates this problem. After all the pruning, the text *"The world's first proven Generation III+ pressurized water reactor and passive safety plant available."* is connected to the title *"AP1000 PWR"*, however it has a weaker edge connected to *"eVinci Microreactor"* that survived the pruning. Only *"AP1000 PWR"* should be added as context, therefore edges that connect a text node to context should be scanned to see if two different context is of the same category, and therefore conflicting. This is a safe assumption to make in all cases, as the nodes extracted are usually only a few sentences long, including from news articles. This means that only one contextual node can fill the gap. We don't take just the top edge no matter what, as the top edge can be structurally relevant but semantically irrelevant. These 'false' edges are tackled when extracting context.
OpenAI's 4.1-mini model for this. This is because a more powerful model is needed to understand the meaning of the two candidate context nodes. To compare short phrases, the model needs to reason whether the candidates will give conflicting or complementing context to a sentence. This can be difficult for a BERT-based model,

especially since the use case is for the nuclear domain. Listing **??** details the prompt used to categorise the conflicting context. If two nodes were conflicting, the lower probability edge was removed.

Listing 3.7: Open AI prompt to classify context as being conflicting or not

```
    '''You are a classifier.
Decide the relation between INPUT 1 and INPUT 2.

CLASSES:
    0 = Both belong in the same category (e.g., both are a product, both
        are a feeling, both are dates).
    1 = Both belong in seperate categories

CLASSIFY one by one:

INPUT 1: {INPUT1}
INPUT 2: {INPUT2}

...
...

CLASSIFICATION (output {N} space separated numbers only):'''
```

3.5.5 Extracting facts from sentences

OpenAI 4.1-nano was used to extract facts from sentences. Summarising nodes with OpenAI was found to be far superior to summarising with FLAN-T5. FLAN-T5 struggled to understand how context could be added to the input, and often completely ignored the context. 4.1-nano is the smallest of the OpenAI language models. This model was chosen for two reasons. First, outputting summaries require more tokens, and therefore this is an expensive API call. Second, extracting facts from a sentence, and using context to fill in the blank, is not as tough a reasoning task. A human could extract facts from a sentence of a highly complex domain, and still do well regardless of their understanding of the domain itself. This is because more than meaning can be used to complete this task, sentence structure is also a large indicator.

Two different prompts were required to send to Open AI for fact extraction: one for context-requiring nodes, and one for stand-alone nodes.

Summarising Context-Requiring Nodes

It was found that instructional prompts worked well. The higher up the instruction, the more the model followed the instruction. The prompt to add context to a node can be seen in Listing 3.8.

Listing 3.8: Open AI prompt to add context to a node

```
    '''Summarise the INPUT provided into facts. Do not remove important
       facts. Do not add facts.
If the CONTEXT can make a subject or premise within the INPUT more
   specific, replace it.
If there is missing information within the INPUT (e.g. pronouns, alluding
   to something) use the CONTEXT to enrich it.
Do not use the CONTEXT if the fact is complete.
Do not ouput a fact directly from the CONTEXT. Output only the facts from
   the INPUT. Be concise.
If there are no fact, output "NO FACTS"

CONTEXT: {CTX}
INPUT: {INP}

Summary:'''
```

Summarising tags that don't need Context

To summarise a node's sentences, the prompt in Listing **??** was used. This instructional style worked well, and correctly could distinguish if facts existed in the sentence or not. The definition of what a fact is helped the model know not to extract everything it saw.

```
    '''A fact is a declarative, verifiable claim with a concrete subject
       and predicate that can be true or false.
Summarise the INPUT provided into a minimal list of self-contained facts.
If the INPUT contains no facts, output "NO FACTS". Ignore sentences that
   contain no facts.
Within each fact, NEVER use pronouns (e.g., him, these, it).
The previous fact must not imply something in the next fact.
Explicitly state everything, even if it means repeating words.
Be concise. Seperate the facts with "\\n".

INPUT: {INP}

Summary:'''
```

Chapter 4

Experimental Results

4.1 Results of the Graph Transformer

Training

The graph transformer model shows strong results on the *movie-allmovie(2000)* validation dataset. Figure 3.2 shows us that the training loss quickly dropped to very low levels, on the scale of $1e-3$. This contrasts with the validation loss, which continuously increased to large values. This contrasts with an improving model, as seen with the F1 score increasing to a high of 0.741. The rising validation loss alongside an improving model should not raise red flags. Focal loss down-weights easy, correctly classified labels and increases the weighting of misclassified labels. As the model improves, it quickly learns the easy labels, leaving the more difficult labels to dominate. This is supported by zooming into a range of a few epochs. At this range, a decrease in validation loss equates to an improvement of the F1 score. A rising loss here does not correlate to an improving or worsening model. It should also be noted that the F1 score is calculated by classifying the top half of the predicted edges as positive. Knowing that half the labels are positive tell us that an F1 score of 0.75 tells us that the model is ranking 75% of the positive edges in the top half assigned probabilities. After around 200 epochs, the model started over-fitting to the training data.

Parameters

The choice of parameters was essential for a successfully trained model. HTML graphs are a very challenging graph-type to generalise to. Each developer has a different way of writing HTML, leading to significantly different HTML structures to display the same information. HTML is also inherently noisy and full of errors, depending on the skill of the developer. This made training very volatile, and over-fitting to the quirks of the training HTML easy.

The most important parameter was the learning rate. Using a loss-curve-plateau-based scheduler lead to no generalisation of the model; by the time a plateau was reached, over-fitting had already occurred. OneCycleLR (33) was chosen to get

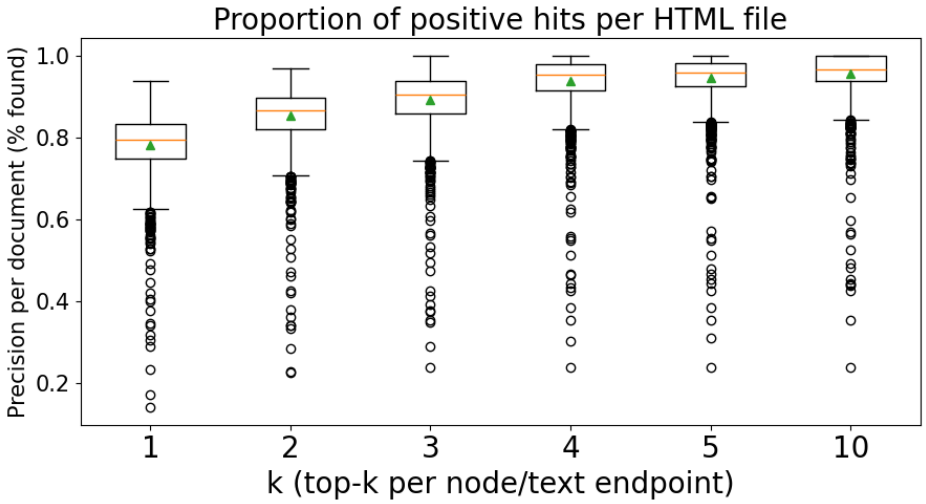


Figure 4.1: The distribution of precision over the 2000 validation webpages. 6 distributions shown for different values of k

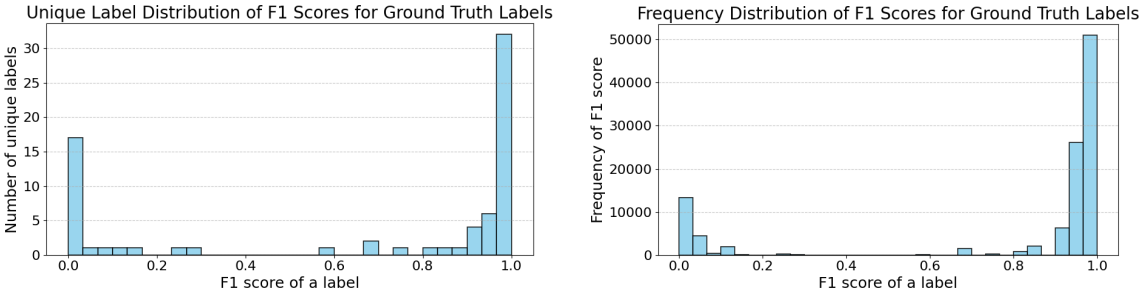
ahead of the over-fitting by changing the it before any plateau. The size of the learning rate also greatly affected the generalisable capabilities of the model. Too large a learning rate seemed to encourage the model to ignore the difficult graph embeddings, short-cutting to only judging an edge by its features only. This initially seemed to show great results, with the model getting an F1 score above 0.8. It was very clear that this model was not generalisable; classifying other websites showed it learned "close proximity horizontal elements are edges". Too small a learning rate lead to the model not learning anything meaningful within 50 epochs.

The number of graph transformer layers was varied, with a sweet-spot being found at 6. Too many lead to the model over-fitting the training data, and fewer than 6 layers lead to the model reaching lower maximum F1 scores.

Top k global metrics

Fixing half the training outputs to be the positive class gave a good metric to track how the model was learning, however, with such an arbitrary threshold, this F1 score is not informative on how the model generally performs. It is more telling to look at the top k edges that the model predicts for a node. Then we can see the top k-hit rate of the model, and how successful it is at classifying specific edges. Figure 4.1 shows the distributions of the proportion of hits per HTML file. We can see that the median HTML file finds the correct edge **80%** of the time. It also shows that the correct edge is usually found within the top four edges, or not at all, since only small improvements are seen after $k = 4$. Note that the validation data is from the *movie-allmovie(2000)* pages, whose nodes usually only have one connection.

Using a $k = 1$ hit rate, we can calculate a true destination-node F1 score. This means a list of edges are iterated through, and the edge is judged by whether a



(a) Distribution of unique labels. Each unique label is only counted once.

(b) Distribution of frequency of labels. Each unique label is counted by frequency.

Figure 4.2: The F1 score distribution of the ground truth labels. $k = 1$: classification is based on the first hit of a label.

"value node" (e.g. *Atom Eyogan*) has the correct "label node" (e.g. *Director*). A $k = 1$ labelling assigns the top edge per "value node" as positive, and the other of its edges as negative.

- True Positives - the number of correct $k = 1$ predictions.
- False Positives - the number of incorrect $k = 1$ predictions.
- True Negatives - the rest of a nodes' edges where the key is incorrect
  An overwhelmingly large class, prior to edge pruning
- False Negatives - the number of times a correct edge was predicted in more than $k = 1$ edges.
  Note that if only one correct key exists, this is equal to the false positive count. However this is not always the case.

Top k local metrics

The global metrics for the model have reported the performance of the model on average. Now let us examine individual edges.

Figure 4.2(a) shows the F1 score of each unique ground-truth label, calculated over all 2000 websites. Note that not all ground-truth labels are found on all websites, i.e. the label *"Includes:"* is only present on 7 of the 2000 websites. Low frequency labels can skew the distribution to be more binary, as low occurrence allows for more extreme F1 scores (a label occurring only once can only achieve an F1 score of 0 or 1). Figure 4.2(b) is provided to debunk this; it shows that this binary pattern exists. Each webpage is displayed very similarly, meaning that the model has consistent performance across similar webpages, and always struggles with the same labels. Table 4.1 ranks the edges to show which do well and which do not.

Observe that the precision is usually 1, even for one of the worst performing labels. This trend is followed by all labels. Here, false positives are classified by any HTML

Search Keyword	**Precision**	**Recall**	**F1**
MPAA Reasons	1.0	1.0	1.0
AMG Work ID	1.0	1.0	1.0
Genres	1.0	0.99	0.99
⋮	⋮	⋮	⋮
Is part of the series:	0.5	0.006	0.01
Year	1.0	$5e-4$	$1e-4$
Run Time	0.0	0.0	0.0

Table 4.1: The best and worst performing ground truth labels

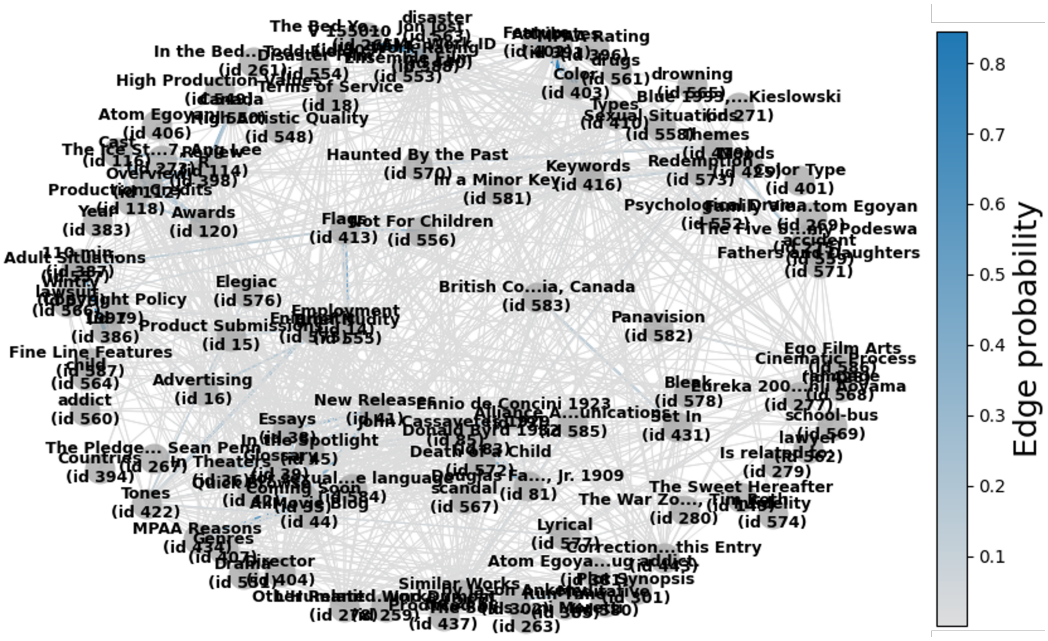


Figure 4.3: SWDE's *Allmovie(2000)/0000.htm* webpage converted to a text graph using the graph transformer model. The displayed graph hides edges all edges below $p = 1e{-}6$

text incorrectly choosing *"Year"* as its label. If the model gets a relation wrong, it usually picks a random HTML tag, not one of the expected labels. Therefore, when an HTML text node chooses *"Year"* as its label, it is usually correct, leading to a precision of 1. Recall is more informative. It tells us how many of *"Year"*'s true values actually chose it for its label. When investigating why *"Year"* did so poorly, it is because it almost always chose *"Run Time"*'s value as its label, and vice versa. *"Year"* would usually be the second choice. Looking at Figure 3.1 it is clear why. Structurally, it makes sense that *"1997"* could be the title of *"110 min."*. The model accepts no text as input, and even a human would struggle to know which node to pick if they could not see the text.

The graph transformer model now gives us a list of edges with sudo-probabilities. These probabilities should not be interpreted as well calibrated. Figure 4.3 demonstrates this. Of the 12882 edges between the 114 nodes, 15 edges hold a weighting

above 0.5. The graph is void of any colour, suggesting a majority of the edges are below 0.1. This means that relative probabilities are more telling on edge existence than the probabilities themselves.

Title selection

The model was also trained with the task of finding the title of the page.

k	1	2	3	4	5	10	25
Proportion	0.14	0.29	0.35	0.38	0.41	0.55	0.90

Table 4.2: Proportion of title nodes found in top k-hits

This task proved to be successful, with the model picking out the correct text node 14% of the time, and correctly ranking it in the top three a third of the time. This shows that the model can understand absolute structure of the HTML graph, not just relative structure. It should be noted again that the model has seen no text up to this point, and is working on HTML structure alone.

4.1.1 Another Notable Attempt - TransformerConv

The GPSConv (16) six-layer graph transformer model was replaced with a three-layer TransformerConv (15) graph transfomer model to compare the expressivity of the learned node embeddings. The motivation was to find out whether focusing on local message passing only outputs more relevant node embeddings for edge prediction. Three layers were chosen, as adding more layers lead the model to over-fit the training data. The loss curve for the TransformerConv model can be seen in Figure 6.1 in the appendix.

This variant performed worse on the edge prediction task. TransformerConv (15) implements transformers differently to GPSConv (16). It uses attention based mechanisms inspired by transformers to pass messages. However, it is fundamentally still a neighbourhood-based message-passing layer. Each node computes attention-weighted sums only over its immediate neighbours. Therefore, it behaves more like a graph-aware version of GAT (Graph Attention Network) than a full graph transformer. In contrast, GPSConv combines local message passing with a global attention component. This enables it to capture both local edge-aware structure and long-range dependencies. TransformerConv's architecture limits its ability to learn long-range dependencies, and since a significant number of node relations are over 6 hops away, a three-layer TransformerConv is unlikely to learn expressive node embeddings.

4.2 The Edge Filters

Buttons and Navigational Filter

This filter was the most accurate of the language model filters. FLAN-T5-large did an excellent job at recognising which text nodes were buttons/navigational text. This is likely because the prompt included many examples for FLAN-T5 to draw inspiration from. The model was very good at using those examples to extrapolated to other buttons. Listing 3.3 shows that *"Explore"* is listed as an example button. Therefore FLAN-T5 knows to classify *"Explore the training videos"* as a webpage button, but not classify *"Explore the mountains in the summer to avoid the cold"*. Even though the word "Explore" begins each three statement, FLAN-T5 recognises the first two are instructions, and the last is a fact.

The edge cases that the model struggles on, humans might struggle on too. For example, while *"Explore the mountains in the summer to avoid the cold"* is negative, *"Explore the mountains"* is marked as positive. Depending on the webpage, this might be a navigational link, or a title that could provide context to a statement. The model is just as unsure as us, and in those cases, it usually opts to claim it as a button.

Navigational links already tend to be avoided when extracting text chunks from an HTML. Only specific stand-alone links are selected in case they contain important information. This leads to a few number of buttons leaking through, which FLAN-T5 picks up well. With the SWDE expanded dataset, example buttons that the model catches are: *-View DVD Releases -Send to Friend -Category -About Us -Privacy Policy -Contact*. There is rarely a false positive that occurs, and if there is, it is an ambiguous entry to classify.

Over the 2000 websites, an average of 6.23 buttons were found per page. After removing the buttons, an average of 1292 edges were deleted. This edge to node ratio is large as the text graphs prior to this pruning were fully connected, often with over 100 text nodes.

Low Probability Filter

This filter used the approach outlined by *Serrano et al.* (11) to extract the backbone of the graph. It was a successful filter that removed a majority of the incorrect edges. This was despite the fact that the probabilities of the edges are best guesses, rather than absolute truth. The relative sizes of probabilities were still valid enough to correctly remove a substantial number of edges. However, a lenient threshold of 10% significance was applied, allowing all edges with a high to medium significance to remain un-pruned. Even still, this resulted in pruning an average of 9961 edges per page, down from 10229 to 268 edges. This is a much more manageable size, and easier to analyse.

This worked much better than the alternative, which was to only take the top k edges

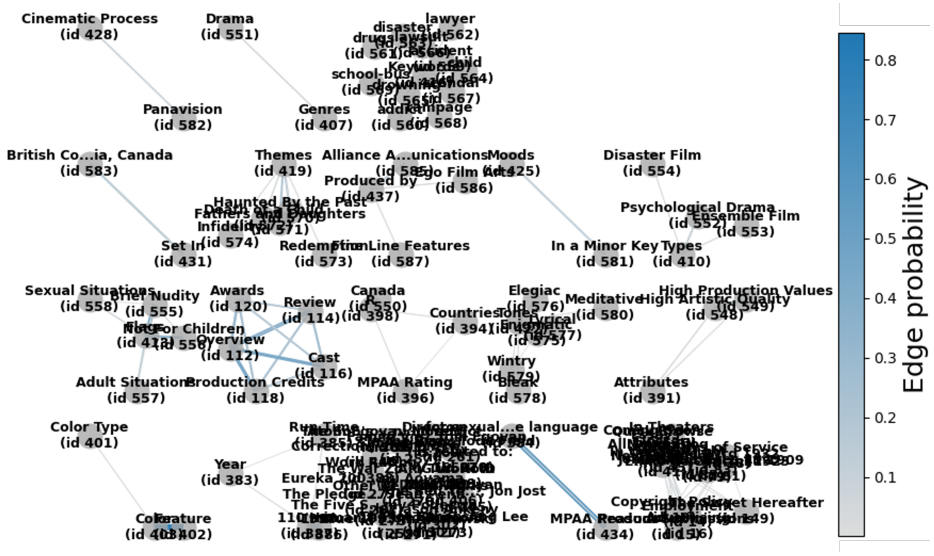


Figure 4.4: SWDE's Allmovie(2000)/0000.htm webpage after the Low Probability filter

per node. The superiority of this method can be seen in Figure 4.4. We can see clusters are starting to form, and some correct relational pairs are already extracted. For example, we can see *"Genres"* and *"Drama"* already clustered. However, this is not a final stage. There are still clusters that are too big, and need to be filtered down.

This filter worked very well for semi-structured data, where one HTML tag has a clear partner, and only one partner. When a node begins to have many valid partners, or it has weak (but still useful) relations with other text nodes, this filter did not help cut down on those edges, and occasionally it cut some of those edges. While this is not ideal, the benefits of this filter vastly outweigh the side-affect of removing some useful edges. This negative consequence is not too dire though, as edges can be restored in the clustering phase.

Semantically Unrelated Edges

This filter is designed to remove edges that make no sense semantically, even if they are strong structurally. This filter must be sure of this semantic difference, as only strong candidate edges remain.

This filter works well on short, well defined entries. For example, it successfully removes the edge *['Year' '110 min.']* that the structural transformer model was very confident about. This leaves *'110 min.'*s strongest edge with it's correct parent, *'Run Time'*.

This filter works poorly on text that is longer than a phrase. For example, it removes the edge *['Atom Egoyan's...' 'by Jason Ankeny']*, where the first entry is the plot synopsis: written by Jason Ankeny. It also removes *['Sexual Situations' 'Flags']*, which is a correct pairing. With the context that the webpage is about a movie, we could

be aware that a 'flag' of the movie is 'Sexual Situations', however just the pair alone with no context means the edge appears to link two semantically different nodes.

This filter removes a lot of junk edges, however it also removes a lot of positive, good edges. The reason this filter remains is that in theory, two closely related nodes that faced an incorrect removal are still clustered together. On average, this filter works more than it doesn't, and therefore removes noisy edges before clustering. Any valid edge that was removed has a greater chance of being restored after clustering than an invalid edge.

This removed on average 12 edges over the 2000 websites. This conservative amount was done to avoid the filter having too much of an adverse affect on the graph, and to be used to only remove clear violations.

4.3 Results after Clustering

The clustering did a good job at recovering edges, and grouping the similar nodes together. We can see the final clusters in figure 4.5. All edges between clusters are removed so that only the edges within clusters remain. A full list of the clusters in figure 4.5 can be seen in Listing 6.2.

Figure 4.5: The clusters found after the all filters and clustering was complete.

We see that the clusters are almost perfectly picking out the labels. Compare the clusters found in Listing 6.2 and Listing 6.3, and we see very similar results. It should be noted that Cluster 4 contains some text that is not found in the ground truth. These are text chunks that were not filtered out by the end of the filtering process.

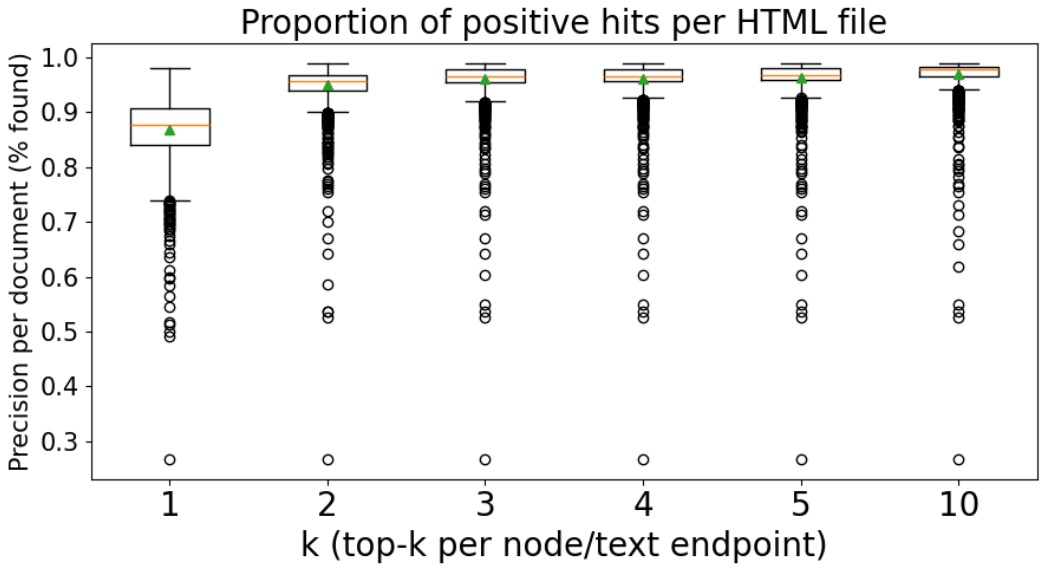


Figure 4.6: The precision of each of the 2000 validation webpages. This is the hit-rate post filter and clustering

We can see how the filters and clustering had an effect on the hit-rate in Figure 4.6. The precision of every webpage was pushed upwards, showing a great positive effect of the filtering and clustering of the edges. By k=2, the correct edges were either selected, or had been filtered out and therefore could never be selected. That is why you do not see much change in the poorly performing extremities as k rises above 2.

Method	**Precision**	**Recall**	**F1 Score**
DOM-LM zero-shot	-	-	0.54
DOM-LM few-shot	-	-	**0.88**
GraphScholarBERT	0.65	**0.88**	0.75
Graph Transformer pre-filters	0.36 *(*0.82)*	0.78	0.80
Graph Transformer post-clustering	**0.83**	0.87	0.85

Table 4.3: Comparison of methods for Open IE on the SWDE expanded *Movie* dataset. The value of 0.36 the Precision when taking into account all the edges returned by the model. It returns many edges irrelevant to the ground truth in Listing 6.3. If we only focus on edges connecting nodes in the ground truth, we get an artificial precision of 0.82

Table 4.3 compares the performance of the pipeline with other state-of-the-art HTML relational extraction methods. We can see that the Graph Transformer performs the best in a zero shot environment. This means that it is the most generalisable out

of the models shown. This generalisability is further reinforced by the fact that semantics is not needed to extract edges, only filter them down. Since the dataset is focused on movie, sports and university language, the other models might suffer from over-fitting to the language and learning which words often pair together.
It should be noted that in calculating the F1 score, GraphScholarBERT and DOM-LM performed a K-fold style analysis. They trained one model per 21 verticals, and averaged the results per vertical type. Table 4.3 displays the averaged movie-type vertical. Due to time constraints, only one vertical could be tested - AllMovie. To do a K fold analysis on the movie vertical, eight models would have been trained. With a training time of 30h per model (plus waiting for NVIDIA A100 availability), training time would have taken too long to do this analysis.

4.4 Results of the summarisation

The LLM filters above work well with simple, semi-structured categorical data. However, when it comes to long-form text, the BERT-based models cannot understand the text content well enough. This leads to poor performance on long-form text. The filters are designed for semi-structured webpages, and the models are not powerful enough at semantic reasoning.

DeBERTa and FLAN can achieve good results at fact extraction and summarisation, but only when the inputs are not noisy; when the correct text nodes and context have been extracted. Using these models as filters on short-form text is proven to be useful above, but the models are no better than random filters on the long-form text. The bottleneck here is using language to filter the text.

Prompting OpenAI to create summaries on the long-form text, we get the list of facts seen in Listing 4.1. This does a good job at extracting information from this paragraph. No fact is ambiguous, and it is possible to trace back where these facts came from using the XPath associated with the input

Listing 4.1: Open AI output after prompting a summary on the long form text. Source node: '/html/body/div/div(3)/...'

```
    Atom Egoyan directed the film adaptation of the Russell Banks novel The
        Sweet Hereafter.
The Sweet Hereafter won a Special Grand Jury Prize at the 1997 Cannes Film
    Festival.
The Sweet Hereafter received two Academy Award nominations, including Best
    Director.
The Sweet Hereafter is set in a small British Columbia town affected by a
    school-bus accident that killed several local children.
Ian Holm stars as Mitchell Stephens, a lawyer who arrives to unite
    survivors for a lawsuit in The Sweet Hereafter.
Mitchell Stephens' actions in The Sweet Hereafter cause further division in
    the community.
```

```
The Sweet Hereafter explores themes of family grief and emotional
    devastation.
Mitchell Stephens in The Sweet Hereafter is motivated partly by remorse
    over his HIV-positive drug addict daughter.
```

Prompting FLAN-T5 to summarise the clusters had limited success. We can see that the model usually regurgitates the input, or doesn't quite form the correct fact. One such example is "['1. Run Time is 110 min. 2. Year is 1997 3. Run Time is 1997']" on the "Run Time, 110 min., Year, 1997" cluster. This did extract to correct facts, however this is a simple cluster. More complex clusters did not form as useful facts.

Listing 4.2: FLAN-T5 output after prompting a summary on the clusters in Listing 6.2. Source node: '/html/body/div/div(3)/...'

```
["1. The Son's Room 2001, Nanni Moretti - The Bed You Sleep In 1993, Jon
    Jost - The Pledge 2001, Sean Penn - Family Viewing 1987, Atom Egoyan -
    In the Bedroom 2001, Todd Field - The Five Senses 1999, Jeremy Podeswa
    - The Ice Storm 1997, Ang Lee - Blue 1993, Krzysztof Kieslowski -
    L'Humanit 1999, Bruno Dumont - Eureka 2000, Shinji Aoyama - Corrections
    to this Entry"]
['1. The War Zone 1999, Tim Roth is related to Other Related Works. 2.']
["1. Atom Egoyan's haunting adaptation of the Russell Banks novel The Sweet
    Hereafter was the Canadian filmmaker's most successful film to date,
    taking home a Special Grand Jury Prize at the 1997 Cannes Film Festival
    and scoring a pair of Academy Award nominations, including Best
    Director. 2. Ian Holm stars as Mitchell Stephens, a bigcity lawyer who
    arrives in the interest of uniting the survivors to initiate a lawsuit
    his maneuvering only drives the community further apart, reopening old
    wounds and jeopardizing any hopes of emotional recovery. 3. No
    character is immune from the sense of utter devastation which grips the
    film, not even the attorney, whose interests are in part motivated by
    his own remorse over the fate of his daughter, an HIV-positive drug
    addict."]
['1. Atom Egoyan is a director. 2.']
['1. The Sweet Hereafter - Terms of Service - Copyright Policy -
    Advertising - Employment - Product Submissions - New Releases -
    AllMovie Blog - Essays - Quick Browse - Coming Soon - Glossary']
['1. Drugs are a type of substance. Drugs can be used to treat many
    diseases. Drugs are dangerous. Accidents can be caused by drugs.']
['1. Lyrical is a type of writing. 2. Elegiac is a kind of writing. 3.
    Tones is a sort of writing. 4. Enigmatic is a form of writing.']
['1. Fathers and Daughters is a film. 2. Haunted By the Past is a movie. 3.
    Infidelity is a character. 4. Redemption is a theme.']
['1. Sexual Situations is a list of sexual situations. 2. Adult Situations
    is another list of adult situations. 3. Not For Children is a category
    of not for children. 4. Brief Nudity is a collection of brief nudity.']
['1. Cast is unknown. 2. Production Credits is unknown. 3. Overview is
    unknown. 4. Review is unknown.']
['1. John Cassavetes died in 1929. 2. Ennio de Concini died in 1923. 3.
```

```
   Donald Byrd died in 1932. 4. Douglas Fairbanks, Jr. was born in 1909.']
['1. Run Time is 110 min. 2. Year is 1997 3. Run Time is 1997']
['1. Canada has a MPAA Rating of R.']
['1. Fine Line Features was produced by Alliance Atlantis Communications.
   2. Ego Film Arts was produced by Fine Line Features.']
['1. Disaster Film is a type of film. 2. The following films are types of
   films: Disaster Film. 3. Psychological Drama is a genre of films.']
['1. Color Type is a color. 2. Color is a feature. 3.']
['1. High Production Values High Artistic Quality are the same.']
['1. V 155010 - AMG Work ID']
['1. MPAA Reasons - for sexuality and some language']
['1. Set In is British Columbia, Canada. 2.']
['1. Moods 2. In a Minor Key 3.']
['1. Cinematic Process is related to Panavision.']
['1. Drama is a genre. 2. Genres is Drama. 3.']
```

The useful facts mostly came from the OpenAI API. OpenAI was good at adding important context to facts when provided with it, and summarising long-form text. This is useful for news articles, which is the use-case for the IAEA. The facts from the smaller FLAN-T5 model did not yield as successful results, however if these clusters were fed into OpenAI's API, more successful (and expensive) results would be outputted.

Chapter 5

Conclusion

This pipeline shows that graph transformers work very well at learning HTML embeddings. Superior performance against other models that do not utilise graph transformers prove that the long-range dependencies that the graph transformer captures well allows the model to understand the HTML better. This model can be taken a step further by training on more broad HTML data. This would make the graph much more generalisable. Currently it has only been trained on 21 different website structures, and 3 different website types (movie, NBA player and university). If this architecture was extended and trained on more than three types of websites, it should show very promising results.

This pipeline also provides a good way to approach edge classification. Creating a fully connected network between nodes of interest, and filtering down those edges by applying traditional graph-based techniques works very well. In future work, other techniques can be compared and investigated. One particular type should be an investigation into Edge-to-Node graph transformations. This turns the edge prediction model into a node prediction model. This can provide better predictions, as node prediction methods use more graph-based methods to make their prediction. Edge-based prediction methods require a linear layer to process two embeddings. This is not as expressive as node prediction methods that utilise the whole graph to make a prediction.

In comparison to ChatGPT and other state of the art LLMs, ChatGPT can perform exceptional results for zero-shot information extraction on HTML. However, this has a large downside. ChatGPT is exceptionally expensive and time consuming. Manually extracting facts from a large number of news articles is impractical. The purpose of this pipeline is to produce a quick and cheap way to create a fact database from many different websites. After importing the dependencies, this pipeline takes around 1 minute 15 seconds to run per average news article webpage (e.g. a Nucnet.org article (35)). The usage of OpenAI comes with costs. The average cost of extracting facts from a webpage is usually around $0.002USD, depending on the number of text elements and edges found. If a fact database from 40,000 news articles was to be created, this pipeline, if run in a realistic deployment of 8 parallel pipelines, would take 4 and a half days. The OpenAI cost would be in the range of about

$100USD. This is a vastly quicker and cheaper method for creating a fact database than using ChatGPT 5 to extract information from 40,000 websites. This makes it far more practical than ChatGPT for large fact extraction.

Bibliography

[1] International Atomic Energy Agency. Overview; 2025. Accessed: 2025-05-20. https://www.iaea.org/about/overview. pages 1

[2] International Atomic Energy Agency. International Safeguards in the Design of Fuel Fabrication Plants. IAEA; 2017. 978-92-0-103315-4. https://www.iaea.org/publications/10746/international-safeguards-in-the-design-of-fuel-fabrication-plants. pages 1

[3] IAEA Safeguards. Basics of IAEA Safeguards; 2025. Accessed: 2025-05-20. https://www.iaea.org/topics/basics-of-iaea-safeguards. pages 1

[4] Mikolov T, Chen K, Corrado G, Dean J. Efficient Estimation of Word Representations in Vector Space. In: Proceedings of the International Conference on Learning Representations (ICLR) Workshop; 2013. . pages 4

[5] Pennington J, Socher R, Manning CD. GloVe: Global Vectors for Word Representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); 2014. p. 1532-43. pages 4

[6] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is All you Need. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 30; 2017. . pages 4

[7] Devlin J, Chang MW, Lee K, Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics; 2019. p. 4171-86. Available from: https://aclanthology.org/N19-1423/. pages 4

[8] He P, Gao J, Chen W. DeBERTaV3: Improving DeBERTa using ELECTRA-Style Pre-Training with Gradient-Disentangled Embedding Sharing. 2021. pages 4, 30

[9] Chung HW, Hou L, Longpre S, Zoph B, Tay Y, Fedus W, et al.. Scaling Instruction-Finetuned Language Models. arXiv; 2022. pages 5, 25, 26, 27

[10] Dwivedi VP, Luu AT, Laurent T, Bengio Y, Bresson X. Graph Neural Networks with Learnable Structural and Positional Representations. In: International Conference on Learning Representations (ICLR); 2022. ArXiv preprint arXiv:2110.07875. pages 6

[11] Serrano M Boguñá M, Vespignani A. Extracting the Multiscale Backbone of Complex Weighted Networks. arXiv preprint arXiv:09042389. 2009. pages 6, 26, 39

[12] Benjamini Y, Hochberg Y. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. Journal of the Royal Statistical Society: Series B (Methodological). 1995;57(1):289-300. pages 6

[13] Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Scientific Reports. 2019;9(1):5233. pages 7, 28

[14] Alon U, Yahav E. On the Bottleneck of Graph Neural Networks and its Practical Implications. In: NeurIPS (Neural Information Processing Systems). Curran Associates, Inc.; 2020. Highlights the over-squashing bottleneck in GNNs, including GAT, which limits propagation of long-range information. pages 7, 16

[15] Shi Y, Huang Z, Feng S, Zhong H, Wang W, Sun Y. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. In: Proceedings of the International Conference on Learning Representations (ICLR); 2021. ArXiv preprint arXiv:2009.03509. pages 8, 38

[16] Rampášek L, Galkin M, Dwivedi VP, Luu AT, Wolf G, Bengio Y, et al. Recipe for a General, Powerful, Scalable Graph Transformer. In: Advances in Neural Information Processing Systems (NeurIPS); 2022. . pages 8, 21, 38

[17] Deng X, Shiralkar P, Lockard C, Huang B, Sun H. DOM-LM:Learning Generalizable Representations for HTMLDocuments; 2022. `https://arxiv.org/abs/2201.10608`. pages 9, 10

[18] Li J, Xu Y, Cui L, Wei F. MarkupLM: Pre-training of Text and Markup Language for Visually Rich Document Understanding. In: Muresan S, Nakov P, Villavicencio A, editors. Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics; 2022. p. 6078-87. `https://aclanthology.org/2022.acl-long.420/`. pages 9, 11

[19] Wang Q, Wang J, Quan X, Feng F, Xu Z, Nie S, et al. MUSTIE: Multimodal Structural Transformer for Web Information Extraction. In: Rogers A, Boyd-Graber J, Okazaki N, editors. Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics; 2023. p. 2405-20. `https://aclanthology.org/2023.acl-long.135/`. pages 9, 12

[20] Hong Z, Chard K, Foster I. Combining Language and Graph Models for Semi-structured Information Extraction on the Web; 2024. https://arxiv.org/abs/2402.14129. pages 9

[21] Bohra A, Saroyan M, Melkozerov D, Karufanyan V, Maher G, Weinberger P, et al.. WebLists: Extracting Structured Information From Complex Interactive Websites Using Executable LLM Agents; 2025. https://arxiv.org/abs/2504.12682. pages 9

[22] Hao Q, Cai R, Pang Y, Zhang L. From one tree to a forest: a unified solution for structured web data extraction. In: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '11. New York, NY, USA: Association for Computing Machinery; 2011. p. 775–784. https://codeplexarchive.org/codeplex/project/swde. Available from: https://doi.org/10.1145/2009916.2010020. pages 9, 10, 16, 17, 18

[23] Lockard C, Shiralkar P, Dong XL. OpenCeres: When Open Information Extraction Meets the Semi-Structured Web. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics; 2019. p. 3047-56. https://aclanthology.org/N19-1309/. pages 9

[24] Common Crawl Foundation. Common Crawl: Open Repository of Web Crawl Data; 2025. Accessed: 2025-06-05. Available from: https://commoncrawl.org/. pages 9, 12, 20

[25] Chen X, Zhao Z, Chen L, Zhang D, Ji J, Luo A, et al.. WebSRC: A Dataset for Web-Based Structural Reading Comprehension; 2021. Available from: https://arxiv.org/abs/2101.09465. pages 9

[26] org S. Getting Started with Schema.org Using Microdata; 2025. Accessed: 7 September 2025. https://schema.org/docs/gs.html#microdata_itemscope_itemtype. pages 10

[27] Hong Z, Ajith A, Pauloski J, Duede E, Chard K, Foster I. The Diminishing Returns of Masked Language Models to Science. In: Findings of the Association for Computational Linguistics: ACL 2023; 2023. p. 1270-83. Available from: https://arxiv.org/abs/2205.11342. pages 16

[28] Web Data Commons Project. Web Data Commons: Extracting Structured Data from the Common Crawl;. Accessed: 6 September 2025. https://webdatacommons.org/. pages 17, 20

[29] Graham J, contributors. html5lib: a Python library for parsing HTML (conforming to the WHATWG HTML specification); 2020. Version 1.1, MIT License; accessed 6 September 2025. https://pypi.org/project/html5lib/. pages 19

[30] SeleniumHQ. selenium: Python language bindings for Selenium WebDriver; 2025. Latest version 4.35.0 (released August 12, 2025); accessed 7 September 2025. `https://pypi.org/project/selenium/`. pages 19

[31] Developers S. scipy.sparse.csr_matrix — SciPy v1.11.0 Manual; 2023. Accessed: 2025-09-06. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.csr_matrix.html`. pages 20

[32] Lin TY, Goyal P, Girshick R, He K, Dollár P. Focal Loss for Dense Object Detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2017. p. 2980-8. pages 21

[33] Smith LN, Topin N. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. arXiv preprint arXiv:170807120. 2017. pages 22, 34

[34] NFL Football Teams – Official Sites of all 32 NFL Teams; 2025. Accessed: 2025-09-08. `https://www.nfl.com/teams/`. pages 28

[35] Kraev K. Estonia's Parliament Resolution Paves Way for Establishing Nuclear Energy Legislation; 2024. Accessed: 2025-09-08. pages 46

Chapter 6

Appendix

Listing 6.1: List of HTML nodes kept when converting the HTML into a graph. All others were deleted.

```
a, abbr, address, article, aside, audio, b, bdi, bdo, blockquote, body,
    br, button, canvas, caption, cite, code, data, dd, del, details,
    dfn, dialog, div, dl, dt, em, embed, fieldset, figcaption, figure,
    footer, form, h1, h2, h3, h4, h5, h6, header, hr, html, i, iframe,
    img, input, ins, kbd, label, legend, li, main, mark, menu, meter,
    nav, object, ol, optgroup, option, output, p, pre, progress, q, rp,
    rt, ruby, s, samp, section, select, small, span, strong, sub,
    summary, sup, svg, table, tbody, td, textarea, tfoot, th, thead,
    time, tr, u, ul, var, video, wbr, UNK, center, font, basefont, big,
    tt, strike
```

Listing 6.2: Results of the clustering of 0000.html

```
    Cluster 0 -> nodes: [388, 263, 265, 267, 269, 261, 275, 273, 271,
        259, 277, 443, 257]
"Work Rating",
"The Son's Room 2001, Nanni Moretti",
"The Bed You Sleep In 1993, Jon Jost",
"The Pledge 2001, Sean Penn",
"Family Viewing 1987, Atom Egoyan",
"In the Bedroom 2001, Todd Field",
"The Five Senses 1999, Jeremy Podeswa",
"The Ice Storm 1997, Ang Lee",
"Blue 1993, Krzysztof Kieslowski",
"L'Humanit 1999, Bruno Dumont",
"Eureka 2000, Shinji Aoyama",
"Corrections to this Entry",
"Similar Works",
  Cluster 1 -> nodes: [279, 278, 280]
"Is related to:",
"Other Related Works",
"The War Zone 1999, Tim Roth",
```

```
  Cluster 2 -> nodes: [381, 302, 301]
"Atom Egoyan's haunting adaptation of the Russell Banks novel The Sweet
    Hereafter was the Canadian filmmaker's most successful film to date,
    taking home a Special Grand Jury Prize at the 1997 Cannes Film Festival
    and scoring a pair of Academy Award nominations, including Best
    Director. Restructured to fit Egoyan's signature mosaic narrative
    style, the story concerns the cultural aftershocks which tear apart a
    small British Columbia town in the wake of a school-bus accident which
    leaves a number of local children dead. Ian Holm stars as Mitchell
    Stephens, a big-city lawyer who arrives in the interest of uniting the
    survivors to initiate a lawsuit his maneuvering only drives the
    community further apart, reopening old wounds and jeopardizing any
    hopes of emotional recovery. Like so many of Egoyan's features, The
    Sweet Hereafter is a serious and painfully honest exploration of family
    grief no character is immune from the sense of utter devastation which
    grips the film, not even the attorney, whose interests are in part
    motivated by his own remorse over the fate of his daughter, an
    HIV-positive drug addict.",
"by Jason Ankeny",
"Plot Synopsis",
  Cluster 3 -> nodes: [404, 406]
"Director",
"Atom Egoyan",
  Cluster 4 -> nodes: [149, 18, 19, 16, 14, 15, 36, 45, 41, 44, 38, 42, 35,
      39]
"The Sweet Hereafter",
"Terms of Service",
"Copyright Policy",
"Advertising",
"Employment",
"Product Submissions",
"In Theaters",
"In the Spotlight",
"New Releases",
"AllMovie Blog",
"Essays",
"Quick Browse",
"Coming Soon",
"Glossary",
  Cluster 5 -> nodes: [561, 569, 562, 560, 568, 563, 564, 567, 566, 565,
      416, 559]
"drugs",
"school-bus",
"lawyer",
"addict",
"rampage",
"disaster",
"child",
```

```
"scandal",
"lawsuit",
"drowning",
"Keywords",
"accident",
  Cluster 6 -> nodes: [580, 579, 578, 577, 576, 422, 575]
"Meditative",
"Wintry",
"Bleak",
"Lyrical",
"Elegiac",
"Tones",
"Enigmatic",
  Cluster 7 -> nodes: [574, 573, 572, 571, 419, 570]
"Infidelity",
"Redemption",
"Death of a Child",
"Fathers and Daughters",
"Themes",
"Haunted By the Past",
  Cluster 8 -> nodes: [558, 557, 556, 413, 555]
"Sexual Situations",
"Adult Situations",
"Not For Children",
"Flags",
"Brief Nudity",
  Cluster 9 -> nodes: [120, 116, 118, 112, 114]
"Awards",
"Cast",
"Production Credits",
"Overview",
"Review",
  Cluster 10 -> nodes: [85, 79, 83, 81]
"John Cassavetes 1929",
"Ennio de Concini 1923",
"Donald Byrd 1932",
"Douglas Fairbanks, Jr. 1909",
  Cluster 11 -> nodes: [383, 385, 387, 386]
"Year",
"Run Time",
"110 min.",
"1997",
  Cluster 12 -> nodes: [394, 396, 398, 550]
"Countries",
"MPAA Rating",
"R",
"Canada",
  Cluster 13 -> nodes: [587, 586, 437, 585]
```

```
"Fine Line Features",
"Ego Film Arts",
"Produced by",
"Alliance Atlantis Communications",
  Cluster 14 -> nodes: [554, 553, 410, 552]
"Disaster Film",
"Ensemble Film",
"Types",
"Psychological Drama",
  Cluster 15 -> nodes: [401, 403, 402]
"Color Type",
"Color",
"Feature",
  Cluster 16 -> nodes: [391, 549, 548]
"Attributes",
"High Production Values",
"High Artistic Quality",
  Cluster 17 -> nodes: [509, 440]
"V 155010",
"AMG Work ID",
  Cluster 18 -> nodes: [434, 584]
"MPAA Reasons",
"for sexuality and some language",
  Cluster 19 -> nodes: [431, 583]
"Set In",
"British Columbia, Canada",
  Cluster 20 -> nodes: [425, 581]
"Moods",
"In a Minor Key",
  Cluster 21 -> nodes: [428, 582]
"Cinematic Process",
"Panavision",
  Cluster 22 -> nodes: [407, 551]
"Genres",
"Drama",
```

Listing 6.3: Ground truth for 0000.htm

```
    "0000.htm": {
        "AMG Work ID": [
            "V   155010"
        ],
        "Attributes": [
            "High Artistic Quality",
            "High Production Values"
        ],
        "Category": [
            "Feature"
```

```
],
"Cinematic Process": [
    "Panavision"
],
"Color Type": [
    "Color"
],
"Countries": [
    "Canada"
],
"Director": [
    "Atom Egoyan"
],
"Flags": [
    "Brief Nudity",
    "Not For Children",
    "Adult Situations",
    "Sexual Situations"
],
"Genres": [
    "Drama"
],
"Keywords": [
    "accident",
    "addict",
    "drugs",
    "lawyer",
    "disaster",
    "child",
    "drowning",
    "lawsuit",
    "scandal",
    "rampage",
    "school-bus"
],
"MPAA Rating": [
    "R"
],
"MPAA Reasons": [
    "for sexuality and some language"
],
"Moods": [
    "In a Minor Key"
],
"Other Related Works | Is related to:": [
    "The War Zone"
],
"Plot Synopsis": [
```

```
        "Atom Egoyan's haunting adaptation of the Russell Banks novel
            The Sweet Hereafter was the Canadian filmmaker's most
            successful film to date, taking home a Special Grand Jury
            Prize at the 1997 Cannes Film Festival and scoring a pair of
            Academy Award nominations, including Best Director.
            Restructured to fit Egoyan's signature mosaic narrative
            style, the story concerns the cultural aftershocks which
            tear apart a small British Columbia town in the wake of a
            school-bus accident which leaves a number of local children
            dead. Ian Holm stars as Mitchell Stephens, a big-city lawyer
            who arrives in the interest of uniting the survivors to
            initiate a lawsuit; his maneuvering only drives the
            community further apart, reopening old wounds and
            jeopardizing any hopes of emotional recovery. Like so many
            of Egoyan's features, The Sweet Hereafter is a serious and
            painfully honest exploration of family grief; no character
            is immune from the sense of utter devastation which grips
            the film, not even the attorney, whose interests are in part
            motivated by his own remorse over the fate of his daughter,
            an HIV-positive drug addict."
    ],
    "Produced by": [
        "Alliance Atlantis Communications",
        "Ego Film Arts",
        "Fine Line Features"
    ],
    "Run Time": [
        "110 min."
    ],
    "Set In": [
        "British Columbia, Canada"
    ],
    "Similar Works": [
        "L' Humanit ",
        "In the Bedroom",
        "The Son's Room",
        "The Bed You Sleep In",
        "The Pledge",
        "Family Viewing",
        "Blue",
        "The Ice Storm",
        "The Five Senses",
        "Eureka"
    ],
    "Themes": [
        "Haunted By the Past",
        "Fathers and Daughters",
        "Death of a Child",
```

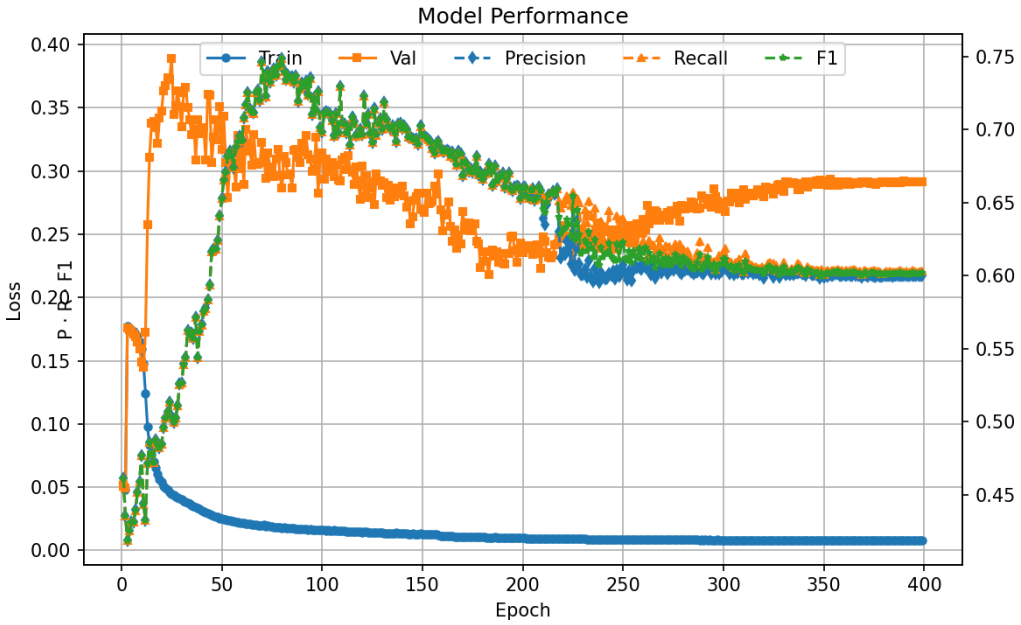


Figure 6.1: Loss curve of the three-layered TransformerConv Model. It peaked with an F1 score of 75 at epoch 80.


```
            "Redemption",
            "Infidelity"
        ],
        "Tones": [
            "Enigmatic",
            "Elegiac",
            "Lyrical",
            "Bleak",
            "Wintry",
            "Meditative"
        ],
        "Types": [
            "Psychological Drama",
            "Ensemble Film",
            "Disaster Film"
        ],
        "Year": [
            "1997"
        ],
        "topic_entity_name": [
            "The Sweet Hereafter"
        ]
    },
```