Honours Project Dissertation

Event Driven Causal Information Extraction from Text

Nathan Scott – 18913101

19 May 2025

https://github.com/nathanpaulscott/CausalRE/tree/main/CRE%20Model

1 Introduction

Extracting causal pairs from text is a challenging NLP task, primarily due to the myriad ways causal information and relationships can be encoded in natural language. Consider two examples:

1. > The earthquake caused destruction in the city.

   This sentence contains a causal pair: **"earthquake"** => **"destruction in the city"**. Detecting and extracting this causal relationship is achievable with discriminative (non-generative) Information Extraction (IE) models. This is largely because the spans of text are short, clearly defined, and the relationship is explicitly indicated by the verb *"caused"* located between them.

2. > 'Belliqueux' rapidly outran Landolphe's flagship 'Concorde', leaving Landolphe with no option but to surrender without any serious resistance.

   Here, the causal pair is: **"'Belliqueux' rapidly outran Landolphe's flagship 'Concorde'"** => **"surrender without any serious resistance"**. Detecting and extracting this pair poses greater challenges for discriminative models, due to the longer spans, the increased complexity of the concepts involved, and the more intricate sentence structure. As spans grow longer, span boundaries become harder to define precisely, and conceptual encodings become more abstract. Although an explicit causal signal is provided between the spans, it is not a common simple form as in the first example.

As illustrated above, extracting causal relationships as in example 2 is more challenging. This dissertation focuses specifically on detecting and extracting such causal pairs using discriminative IE models.

1.1 Motivation

The central motivation of causal relation extraction, as with other branches of IE, lies in converting unstructured textual data into structured representations, such as knowledge graphs. Incorporating causal relations into knowledge graphs enables more precise causal reasoning and analysis. Specifically, this project was motivated by the need to extract causal information from geological survey reports, which are characterized by formal language and complex, domain-specific terminology.

Why employ discriminative models for this task? Historically, discriminative models have offered superior performance for IE tasks; however, generative models have recently made rapid advances, offering increased flexibility and adaptability [1, 2]. Given that causal re-

lation extraction differs substantially from standard Named Entity Recognition and Relation Extraction (NER-RE) or Event Extraction (EE) tasks, it is not immediately clear whether discriminative models retain their historical advantage. Nonetheless, discriminative models currently remain appealing due to their smaller size, computational efficiency, and feasibility for deployment on-premises.

1.2 Causal RE and the Long Span Problem

Causal pairs typically differ from standard NER-RE pairs because they are not always short entities. Often, one or both elements of a causal pair encompass complex ideas, described by longer spans of text. Detecting and clearly delineating each span within a causal pair is essential for correctly identifying their causal relationship and thus poses a significant challenge for causal IE tasks.

Longer text spans describe more complex ideas and thus inherently introduce several complicating factors:

- **Ambiguous boundaries:** Span boundaries become increasingly uncertain as textual structures become less clearly defined.
- **Contextual dilution:** Important contextual information tends to become sparse within longer spans.
- **Increased noise:** Irrelevant linguistic elements and noise become more prevalent within longer spans.
- **Co-reference ambiguity:** Co-referential phrases and associated ambiguity become more common and problematic in longer spans.
- **Annotation complexity:** Longer spans significantly complicate annotation tasks for both humans and automated annotation systems.

1.2.1 How Do Humans Identify Causal Pairs?

Human readers typically do not process text in terms of clearly defined long spans. Instead, they read, identify, and mentally combine smaller informational chunks to construct complex ideas. Consequently, explicitly translating this innate cognitive process into exact span boundaries suitable for computational models can be challenging.

1.2.2 Why Not Break the Problem into Smaller Informational Chunks?

One possible strategy involves initially detecting smaller semantic chunks and then linking these chunks through relational structures to build more complex events or states. This approach parallels Event Extraction (EE) methodologies, in which event triggers are first identified and subsequently enriched by identifying event arguments. After such extraction, inter-event causal relationships can be classified. However, employing this approach requires

specifically and intricately annotated datasets, which were not available for this project.

1.2.3 Why Not Use a Large Language Model (LLM) to Manage Complexity?

Leveraging large language models (LLMs) to address the complexity inherent in long-span causal extraction appears promising, as LLMs excel at capturing nested patterns and complex linguistic structures. However, utilizing LLMs was outside the scope of this research, which instead focused specifically on exploring and assessing discriminative models for this task.

1.2.4 Why Not Approach Causal Extraction as an Extension of NER-RE?

This project explicitly aimed to simplify the causal extraction task into fundamental machine learning terms—specifically, identifying two spans of text, regardless of their length, which together form a causal relationship. Conceptually, this resembles traditional NER-RE tasks, which perform well on short-span extraction scenarios. However, identifying longer spans is considerably more difficult if one treats them as monolithic, clearly delineated structures. This difficulty arises because long spans rarely possess explicitly clear boundaries and instead tend to encode information contextually. Moreover, despite the simplicity of the desired annotation format, there currently exist no freely available causal datasets annotated with explicitly defined span boundaries suitable for this task.

1.3 Research Questions

This dissertation seeks to answer the following primary questions:

1. To what extent can existing discriminative NER-RE models be adapted effectively for long-span causal relation extraction?
2. What are the primary failure modes exhibited by discriminative IE models when applied to extracting long-span causal relations?
3. Which alternative modeling or methodological strategies show the greatest potential for overcoming identified limitations and should be prioritized for future research?

2 Related Work

Causal relation extraction (CRE) has traditionally been approached via pattern-based and syntactic methods, including rule-based systems, dependency parsers, and semantic role labeling. These methods are limited by their reliance on surface syntax and their poor generalization to implicit causality. Neural approaches have primarily focused on the simpler task of short-span, entity-based NER-RE [3, 4]. Some CRE-specific models have been developed [5, 6], aiming to identify events via event triggers and their arguments, but not the relations between events

Recent work has extended span-based models to multi-task settings incorporating event extraction and co-reference resolution [7], but this also stops short of modeling inter-event causal relations. Generative approaches using large language models (LLMs) have also been proposed that could handle causal extraction implicitly [1, 8], but such methods are computationally intensive, not inherently privacy-preserving, and less suited to lightweight, interpretable extraction pipelines.

In contrast, this work investigates the capacity of lightweight, span-based discriminative models to capture event-level causality directly from raw text. It also introduces auxiliary mechanisms, such as pruning, consistency enforcement, and custom span representations, to mitigate structural prediction errors and improve robustness under noisy supervision.

3 Methodology

3.1 Model Architecture

This work explores how span and span-pair triplet extraction architectures, inspired by models such as [3] and GraphER [4], can be adapted for long-span causal relation extraction (CRE). The goal is to evaluate whether the structural simplicity of these models remains effective when applied to CRE's more complex span dynamics. As shown in fig 1 the overall model is a pipeline structure.

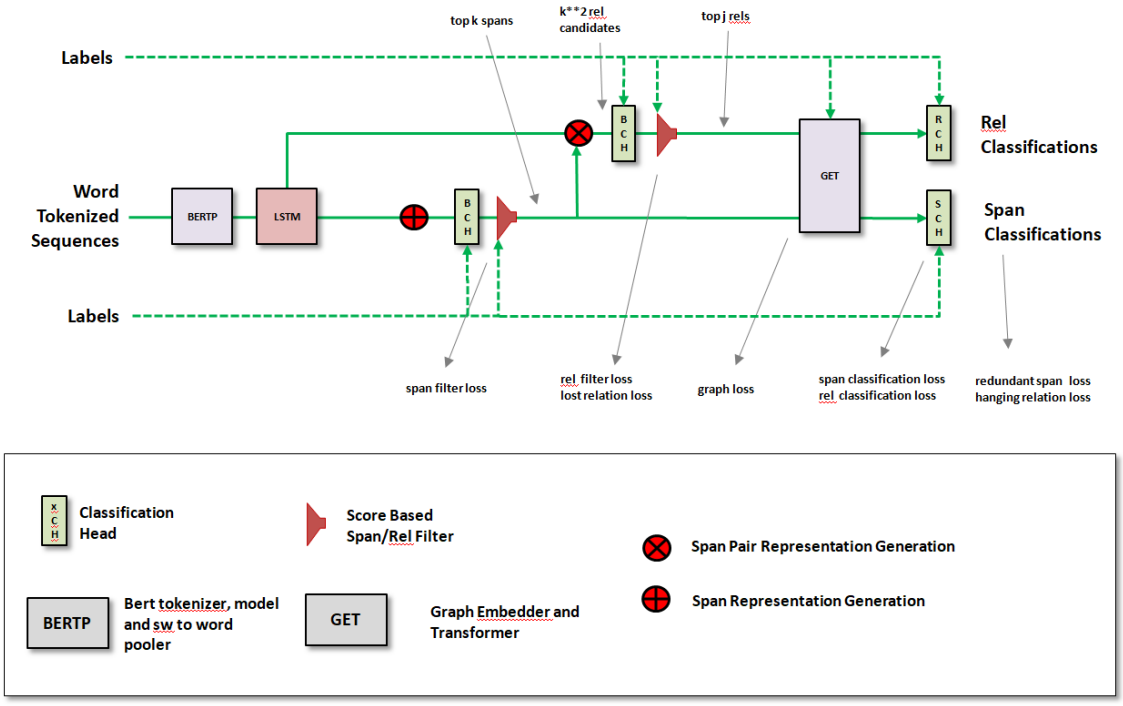

Figure 1: Model, Original Variant

The model contains the following components in order:

- **Pretrained Transformer Encoder Layer**
- **LSTM Layer**
- **Span Generator and Filter:**
  - *Either:* **Token Tagging Layer + Span Generator + Span Filter**
  - *Or:* **Span Generator + Binary Filter Head + Span Filter**
- **Relation Generator + Binary Filter Head + Relation Filter**
- **Graph Embedder + Graph Transformer**
- **Span Classification Head + Relation Classification Head**

These components are described in more details in the following sections.

3.1.1 Pretrained Transformer Encoder Layer

The model uses a pretrained BERT encoder [9] from HuggingFace Transformers [10] to encode input text at the subword level. Since BERT operates on subword tokenization (e.g., WordPiece), a conversion step is required to obtain word-level token representations. This is achieved via max-pooling over the hidden states of each word's subword pieces:

$$\mathbf{h}_i^{(\text{word})} = \max_{j \in \mathcal{S}(i)} \mathbf{h}_j^{(\text{sub})}$$

where $\mathcal{S}(i)$ is the set of subword token indices corresponding to word token i, and $\mathbf{h}_j^{(\text{sub})}$ is the hidden state from BERT for subword token j.

This pooling strategy was selected to simplify downstream processing and eliminate the need to adapt all downstream modules to subword boundaries. Other pooling methods, such as first-subword or first-last averaging (as implemented in Flair [11]), were also considered, but empirical results showed that max-pooling performed best in this context. Similar pooling techniques have been used successfully in prior work such as [4, 12].

3.1.2 LSTM Layer

An optional BiLSTM layer [13] is applied on top of the word token embeddings to further enrich their contextual representation. This design choice was inspired by its successful use in relation extraction pipelines such as [4, 14].

Given a sequence of word-level token embeddings $\{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n\}$, the BiLSTM outputs:

$$\mathbf{h}_i^{(\text{BiLSTM})} = [\overrightarrow{\mathbf{h}}_i;\ \overleftarrow{\mathbf{h}}_i]$$

where $\overrightarrow{\mathbf{h}}_i$ and $\overleftarrow{\mathbf{h}}_i$ are the hidden states from the forward and backward LSTM respectively at position i, and $[\cdot;\cdot]$ denotes vector concatenation.

The output of the BiLSTM is further processed via layer normalization and dropout. A residual connection is optionally applied between the input and output of the BiLSTM layer.

3.1.3 Token Tagger Layer + Filter

This layer provides an alternative to brute-force span search by predicting candidate spans directly from token-level boundary labels. It is significantly more efficient for longer sequences and offers comparable performance. The BIO tagging scheme was considered to be too restrictive, hence two custom tagging strategies were implemented, both designed to predict span boundaries and deal with overlaps:

- **BE Multiclass Token Tagging (Begin-End):** A binary vector of length 2 is predicted

for each token t_i, representing `[Begin, End]` probabilities:

$$\hat{\mathbf{y}}_i = [\hat{y}_i^{(B)},\ \hat{y}_i^{(E)}] = \sigma(W \cdot h_i + b)$$

All valid pairs (i, j) such that $\hat{y}_i^{(B)} > 0$, $\hat{y}_j^{(E)} > 0$, and $j > i$, with span width $\leq w_{\text{max}}$, are used to construct candidate spans. The span filter score is:

$$\text{score}_{i,j} = \frac{1}{2}\left(\hat{y}_i^{(B)} + \hat{y}_j^{(E)}\right)$$

- **BECO Uniclass Token Tagging (Begin-End-Combined-Other):** Each token is assigned one of 4 mutually exclusive classes using softmax:

$$\hat{y}_i = \text{softmax}(W \cdot h_i + b), \quad \hat{y}_i \in \{\text{B}, \text{E}, \text{C}, \text{O}\}$$

Candidate spans are constructed by pairing predicted B and E tags, or directly using C (Combined) tags as single-token spans. Valid span conditions remain the same.

Finally, the top-K candidate spans, ranked by their filter scores, are selected for downstream processing.

The token-level classification loss is:

- **Binary Cross Entropy (BCE)** for BE tagging:

$$\mathcal{L}_{\text{BCE}} = -\sum_i \left[y_i^{(B)} \log \hat{y}_i^{(B)} + y_i^{(E)} \log \hat{y}_i^{(E)}\right]$$

- **Cross Entropy (CE)** for BECO tagging:

$$\mathcal{L}_{\text{CE}} = -\sum_i \log \hat{y}_i^{(c_i)} \quad \text{where } c_i \in \{\text{B}, \text{E}, \text{C}, \text{O}\}$$

Note: Teacher forcing was optionally applied during training, ensuring that gold spans were retained regardless of filter score, without affecting loss computation.

3.1.4 Span Representation Generator

Spans are of variable token lengths, but for modeling, span representations must be converted into fixed-size hidden vectors. Span representations are constructed from candidate spans—either the top-K spans selected by a token tagger, or all spans (up to a width limit) if no tagger is used. Several construction strategies were implemented:

- **First-Last:** Concatenate the word token representations for the start and end of the span, then re-project to the hidden dimension:

$$\text{span\_rep} = \text{FFN}\left([h_{\text{start}};\ h_{\text{end}}]\right)$$

- **Maxpool-CLS-Width:** Concatenate the max-pooled span representation, the BERT `[CLS]` embedding, and a learnable span width embedding:

$$\text{span\_rep} = \text{FFN}\left([\text{maxpool}(h_{\text{span}});\ h_{[\text{CLS}]};\ \text{width\_emb}]\right)$$

- **First-MaxpoolBetween-Last-CLS-Width:** Concatenate the pooled start, end, and inner span segments, plus `[CLS]` and width embeddings:

$$\text{span\_rep} = \text{FFN}\left([\text{pool}_{\text{start}};\ \text{pool}_{\text{end}};\ \text{pool}_{\text{inner}};\ \text{width\_emb};\ h_{[\text{CLS}]}\right)$$

  Where:

  - $\text{pool}_{\text{start}}$ is max-pooled over the first w tokens of the span, $w = \min(1, \alpha \cdot \text{span\_width})$
  - pool_{end} is max-pooled over the last w tokens of the span
  - $\text{pool}_{\text{inner}}$ is max-pooled over remaining span tokens (or $\text{pool}_{\text{start}}$ if none)

- **AttentionPooling-CLS-Width:** Apply self-attention pooling over the span tokens, then concatenate with `[CLS]` and width embedding:

$$\text{span\_rep} = \text{FFN}\left([\text{attnpool}(h_{\text{span}});\ h_{[\text{CLS}]};\ \text{width\_emb}]\right)$$

In all cases, $\text{FFN}(\cdot)$ is a two-layer feedforward network with an intermediate expansion, ReLU, dropout, and final projection to the model hidden dimension.

3.1.5 Binary Span Filter Layer

This layer applies a configurable binary classification head to each span representation to produce a filtering score. Two configurations were supported:

- **Single-logit mode:** A single scalar logit is produced for each span representation $\mathbf{SR}_i$:

$$s_i = \mathbf{w}^\top \mathbf{SR}_i + b$$

  where s_i is used directly as the filtering score. The label $y_i \in \{0, 1\}$ is the binarised gold label for the span (1 = keep, 0 = discard). The loss is Binary Cross Entropy (BCE):

$$\mathcal{L}_{\text{BCE}} = -\left[y_i \cdot \log \sigma(s_i) + (1 - y_i) \cdot \log(1 - \sigma(s_i))\right]$$

- **Double-logit mode:** A 2-class classifier outputs two logits for each span:

$$\mathbf{s}_i = W^\top \mathbf{SR}_i + \mathbf{b} \quad \text{where} \quad \mathbf{s}_i \in \mathbb{R}^2$$

  The filtering score is computed as the logit delta:

$$\text{score}_i = s_i^{(1)} - s_i^{(0)}$$

  The binarised label $y_i \in \{0, 1\}$ is used with Cross Entropy (CE) loss:

$$\mathcal{L}_{\text{CE}} = -\log\left(\frac{e^{s_i^{(y_i)}}}{e^{s_i^{(0)}} + e^{s_i^{(1)}}}\right)$$

In both cases, the top-K scoring spans are selected for downstream processing. If a token-level tagger is used prior to this layer, the top-K here may be set lower—but this setup did not show consistent benefit and was typically not used in combination.

Note: Teacher forcing was optionally applied during training, in which case gold spans were always passed through the filter without affecting the loss.

3.1.6 Relation Representation Generator

After the top K spans are generated, all possible directed relations are formed via the Cartesian product of the span set, producing K^2 candidate relations. This stage introduces a quadratic scaling bottleneck relative to the span count.

Each candidate relation is represented using a combination of its head and tail span representations and optional context. The following configurations were implemented:

- **No Context:** Concatenate the head and tail span representations, then re-project to the hidden dimension in the hope that all required signal is contained within the head and tail:

$$\text{rel\_rep} = \text{FFN}([\mathbf{SR}_h;\ \mathbf{SR}_t])$$

- **Between Context:** Concatenate the head and tail span representations with a pooled representation of the tokens between the two spans. This is commonly used for NER-RE models:

$$\text{rel\_rep} = \text{FFN}([\mathbf{SR}_h;\ \mathbf{SR}_t;\ \text{pool}(T_{\text{between}})])$$

- **Window Context:** Concatenate the head and tail span representations with pooled representations from windows before and after each, this was an adaptation to account for longer spans and more varied context:

$$\text{rel\_rep} = \text{FFN}([\mathbf{SR}_h;\ \mathbf{SR}_t;\ \text{pool}(T_{\text{pre}});\ \text{pool}(T_{\text{post}})])$$

- **Between + Window Context:** Combine both the between-span and windowed context tokens:

$$\text{rel\_rep} = \text{FFN}([\mathbf{SR}_h;\ \mathbf{SR}_t;\ \text{pool}(T_{\text{between}});\ \text{pool}(T_{\text{pre}});\ \text{pool}(T_{\text{post}})])$$

Note: The function $\text{pool}(\cdot)$ refers to a configurable pooling mechanism applied over a token set. Variants tested included max pooling, self-attention pooling, and cross-attention pooling with the head or tail span representations as queries.

3.1.7 Binary Relation Filter Layer

This layer applies a configurable binary classification head to each relation representation to produce a filtering score. Two configurations were supported:

- **Single-logit mode:** A single scalar logit is produced for each relation representation $\mathbf{RR}_i$:

$$s_i = \mathbf{w}^\top \mathbf{RR}_i + b$$

where s_i is used directly as the filter score. The binarized label $y_i \in \{0, 1\}$ indicates whether the relation is positive (1) or negative (0). The loss is Binary Cross Entropy (BCE):

$$\mathcal{L}_{\text{BCE}} = -\left[y_i \cdot \log \sigma(s_i) + (1 - y_i) \cdot \log(1 - \sigma(s_i))\right]$$

- **Double-logit mode:** A 2-class classifier outputs two logits:

$$\mathbf{s}_i = W^\top \mathbf{RR}_i + \mathbf{b} \quad \text{where} \quad \mathbf{s}_i \in \mathbb{R}^2$$

The filter score is calculated as the logit difference:

$$\text{score}_i = s_i^{(1)} - s_i^{(0)}$$

With the binarized label $y_i \in \{0, 1\}$, the classification loss is Cross Entropy:

$$\mathcal{L}_{\text{CE}} = -\log\left(\frac{e^{s_i^{(y_i)}}}{e^{s_i^{(0)}} + e^{s_i^{(1)}}}\right)$$

In both modes, the top-K relations (by score) are retained for downstream processing.

Note: Teacher forcing was optionally applied during training, ensuring that gold relations bypassed filtering without contributing to the loss.

3.1.8 Graph Embedder

The graph module is optional. When enabled, the top-K span representations and top-K relation representations are used to form the initial graph structure. Each span representation $\mathbf{SR}_i$ is stamped with a learned span node embedding $\mathbf{e}_{\text{span}}$, and each relation representation $\mathbf{RR}_j$ is stamped with a learned edge embedding $\mathbf{e}_{\text{rel}}$:

$$\tilde{\mathbf{SR}}_i = \mathbf{SR}_i + \mathbf{e}_{\text{span}}, \quad \tilde{\mathbf{RR}}_j = \mathbf{RR}_j + \mathbf{e}_{\text{rel}}$$

This addition allows the transformer to differentiate span nodes from relation edges. This typed graph embedding approach is adapted from [4].

3.1.9 Graph Transformer

The stamped span and relation representations are stacked along the graph axis (i.e., treated as distinct graph elements in a flat sequence) to form the input to the graph transformer:

$$\mathbf{G} = \text{concat}(\tilde{\mathbf{SR}}_{1:K_s},\ \tilde{\mathbf{RR}}_{1:K_r}) \in \mathbb{R}^{(K_s+K_r)\times d}$$

where d is the hidden dimension, K_s is the number of span candidates, and K_r is the number of relation candidates.

This combined representation is passed through a multi-layer transformer encoder with multi-head self-attention (MHA):

$$\mathbf{G}' = \text{Transformer}(\mathbf{G})$$

After encoding, the output tensor $\mathbf{G}'$ is split back into updated node (span) and edge (relation) representations:

$$[\mathbf{SR}'_1, \ldots, \mathbf{SR}'_{K_s}],\ [\mathbf{RR}'_1, \ldots, \mathbf{RR}'_{K_r}] = \text{split}(\mathbf{G}')$$

3.1.10 Span Classification Head

Each node (span) representation is passed through a linear classification head to produce logits over N_s span types:

$$\mathbf{z}_i^{\text{span}} = W_{\text{span}} \cdot \mathbf{SR}'_i + b_{\text{span}} \in \mathbb{R}^{N_s}$$

A Cross Entropy loss is applied, assuming each span belongs to exactly one class (uniclass setup):

$$\mathcal{L}_{\text{span}} = -\log\left(\frac{e^{z^{(y_i)}}}{\sum_{j=1}^{N_s} e^{z^{(j)}}}\right)$$

3.1.11 Relation Classification Head

Each edge (relation) representation is passed through a linear classification head to produce logits over N_r relation types:

$$\mathbf{z}_j^{\text{rel}} = W_{\text{rel}} \cdot \mathbf{RR}'_j + b_{\text{rel}} \in \mathbb{R}^{N_r}$$

As multiple relation types may apply to each relation (multilabel setup), a sigmoid activation is applied element-wise, and Binary Cross Entropy is used:

$$\mathcal{L}_{\text{rel}} = -\sum_{k=1}^{N_r} [y_k \log \sigma(z_k) + (1 - y_k) \log(1 - \sigma(z_k))]$$

3.2 Model Variants

Various model variants were coded and tested, these have been broken down into 2 main variants.

3.2.1 Original Variant

This is the original configuration 1 which uses brute force span searching in the initial span filtering stage. This works well for smaller span widths and sequence lengths. The resource

limitation comes from the initial span representation generation algorithm. The number of possible spans to search being proportional to $sequence.length \times max.span.width$. This was loosely inspired by the model presented in [4] which essentially uses intermediate binary filter layers for both spans and relations as a form of smart negative sampling to shortlist the spans/relations to smaller sets based on the likelihood of the span/relation being significant. The graph transformer and graph filter layer are optional.

The thoughts behind these filter layers as described by [4] was to form an initial graph with spans being nodes and relations being edges. The graph transformer then being a final step to score the graph structure wholistically vs the labels. The pipeline for the original variant is thus:

- shared BERT pre-trained encoder with post encoder sub-token pooling followed by an optional LSTM layer
- span representation generator and optional random negative case sampler
- binary span filter layer which produces both a span filtering loss and a filtering score for each candidate span. This is used for smart filtering (negative sampling) down to the best Ks spans for downstream processing
- relation representation generator and optional random negative case sampler
- binary relation filter layer which produces both a relation filtering loss and a filtering score for each candidate relation. This is used for smart filtering (negative sampling) down to the best Kr relations for downstream processing.
- optional graph transformer and binary graph filter layer which produces a graph structure filtering loss vs the span/relation labels.
  No logit based score filtering/pruning is done in the graph stage.
- output span and relation classification heads

Sub variations of this model were:

- use a separate BERT instance for the span generator and relation generator.
- disabling of teacher forcing after a certain point in the training process and a lost relation loss to be enabled as a replacement.
- enabling and disabling different combinations of the lstm and graph layers
- use training span/relation negative sampling in addition to the binary filtering.
  NOTE: random negative sampling can't just replace the smart negative sampling as it is not used during inference mode, leading to the full size tensors of possible spans/relations being passed downstream, which can quickly cause memory issues. Smart negative sampling, however, trains a binary filter head which will be used during inference, thus reducing memory use explosion after the initial warm-up stage.

3.2.2 Long span Variant

This variation swapped out the binary span filter layer with the a token tagging layer, see fig 2. This is primarily to deal with the explosion in memory usage as the sequence length and span widths get longer. There was minimal performance variation from the original variant quantitatively or qualitatively. The token tagging scheme used also allowed for overlapping spans. For longer sequences and span widths, token tagging (depending on the tagging scheme) would typically produce < 1000 span candidates where the brute force method for the same configuration would produce > 10000 candidates.

Ablations on the token tagger specifically revolved around the tagging scheme with two variants being tested (BE and BECO). Standard BIO was not reviewed as it did not fit the usage case.

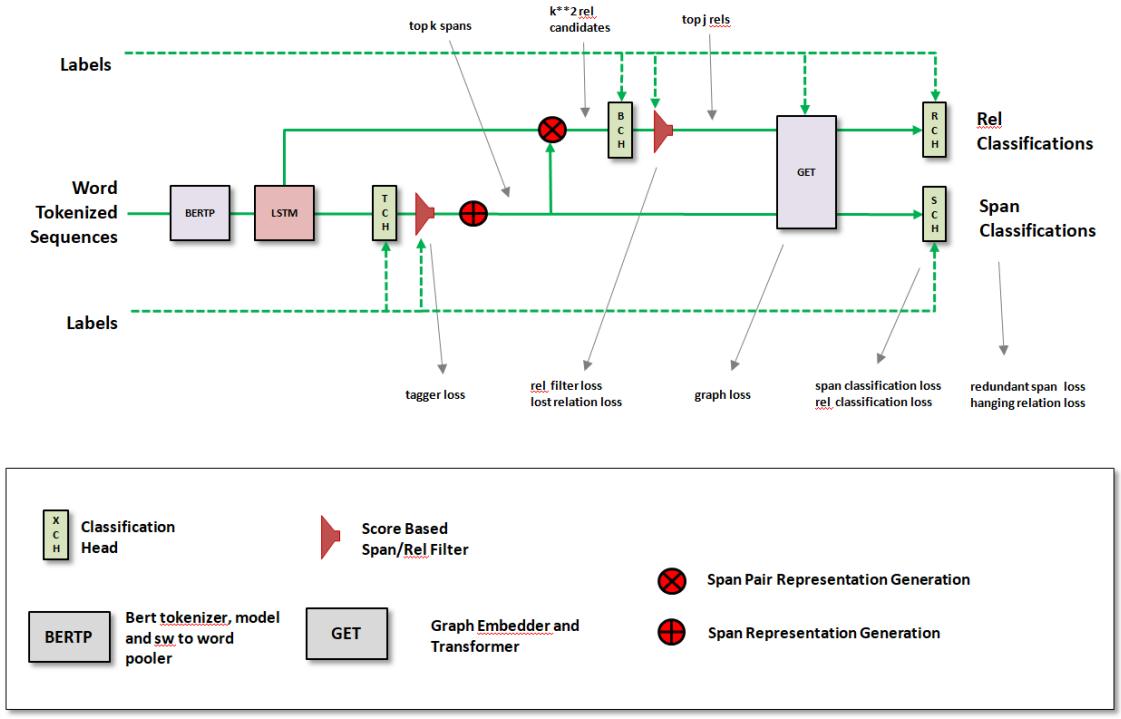

Figure 2: Model, Tagging Variant

3.2.3 Other variants

Some other variants were trialed such as marking spans and relations. This involved marking the top-K spans and relations in the original sequence with tags and passing the modified sequence through an additional pre-trained encoder transformer for each span/relation. More specifically, the top K spans produced after the token tagging layer were marked in the original sequences and run through a separate BERT instance and the pooler token (CLS)

embeddings collected as the span representations. These span representations were then used as inputs to the relation representation generator, which then passed through the binary relation filter layer to produce the top K relations. There again, each relation (pair of spans) was marked on the original sequence and passed through a separate BERT instance. The pooler (CLS) embeddings collected to form the new representation for each of the top K relations. The idea here was to test whether pre-marking spans/relations and leveraging BERT could improve the span/relation representations in any way as good relation F1 results were shown in [15]. However, it became apparent that the numbers in that particular paper were misleading as they effectively gave the model the label span-pairs for marking, thus only testing the ability of the marking scheme to determine causality given the labels, this is very different to the whole end to end pipeline as was being done with this model. As this was extremely resource inefficient, it was only tested as a proof of concept with results indicating that it offered no benefit to performance metrics in the full end to end pipeline. Aside from these modifications the pipeline and optional components remained similar to previous variants.

4 Experimental Setup

4.1 Evaluation Metrics

Model performance was primarily measured using strict (in boundary and type) micro F1 score. A more relaxed boundary F1 metric was also collected for informational purposes for both spans and relations.

4.2 Datasets

Datasets were a significant issue for this research as there simply do not exist a selection of free quality datasets for causal relation extraction annotated in a compatible way. This actually became one of the major stumbling blocks of the project as the model and the annotations it is designed for are tightly coupled. Some of the key points regarding datasets are:

- No compatible causal relation datasets were available. The closest found were curated by the Unicausal project [15], however each of the datasets from this project had their own issues including excessive annotation noise (spans too long, too short, irrational spans, including the causal phrases). These problems were severe enough to render each and every dataset effectively unusable. Several other known datasets were paid [16, 17, 18], so no analysis was attempted on these.
- Overlap with target domain. The target domain is geology reports which formal language with a very specific geology-centric vocabulary. No dataset had anything related to this kind of text. Attempts were made in the initial phase of the project to develop code to extract pdf text from geological reports, chunk it and prepare it for annotations. However time constraints limited the depth of this investigation and it was not developed further.
- Self-Annotation and Time Constraints: some minor self annotations were attempted to at least form a core sample dataset for causal data. A lightweight annotation tool was developed to enable rapid labeling under time constraints. A small test dataset was annotated for testing purposes. It is clear that with more time and resources, a higher quality and larger dataset could have been curated.

4.2.1 Altlex

Altlex has a single causal pair annotated per observation. The sequence is effectively cut in two parts with each part making up the spans. Thus many cases do not have any real rational behind the span boundaries, making it less than ideal for use with the model as these would be counted as positive span cases and cause confusion during model training. To be useful, this dataset would need re-annotation of the span boundaries.

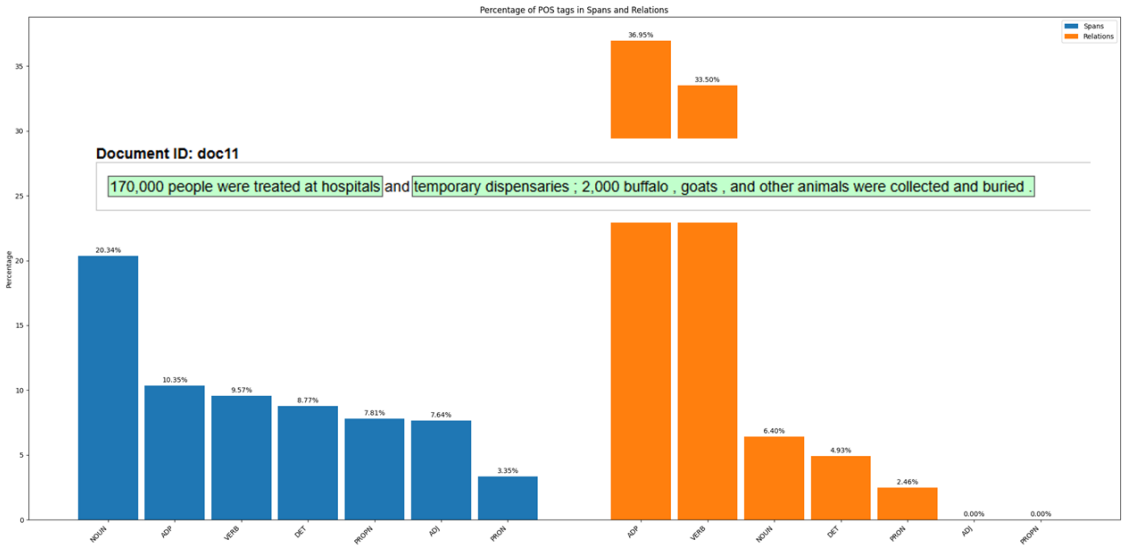

Figure 3: Altlex POS Breakdown

4.2.2 Semeval

Semeval has many samples, but has very short spans (up to 4 words) annotated causal pairs. As can be seen from the POS breakdown, the annotated spans are almost all entities. This dataset has approximately 90% negative cases, where the negative cases are just unrelated and irrational spans, i.e. not forming an event/state, just randomly chosen entities. It is only for the few positive cases the annotated spans typically are the entity form of events or states or entity proxies for events and states. Additionally all causal pairs are explicitly linked. Thus this dataset again has limited usage for the model and would need significant revision.

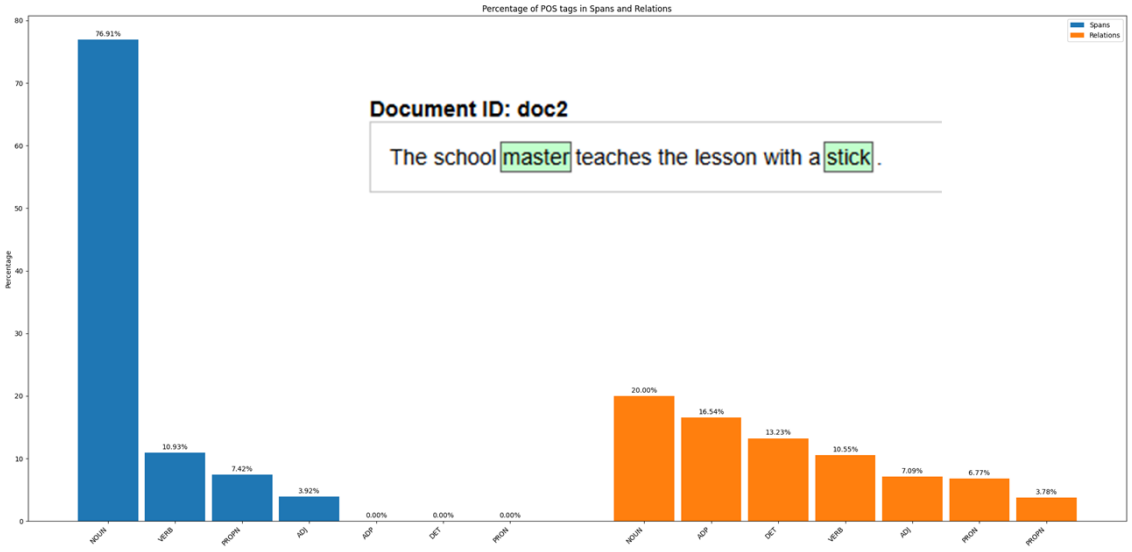

Figure 4: Semeval POS Breakdown

4.2.3 Because

Somewhat similar in form to Altlex with the annotated spans a little closer to the desired format with spans representing events/states. However this dataset still has excessive annotation noise to be usable and woudl need re-annotation.

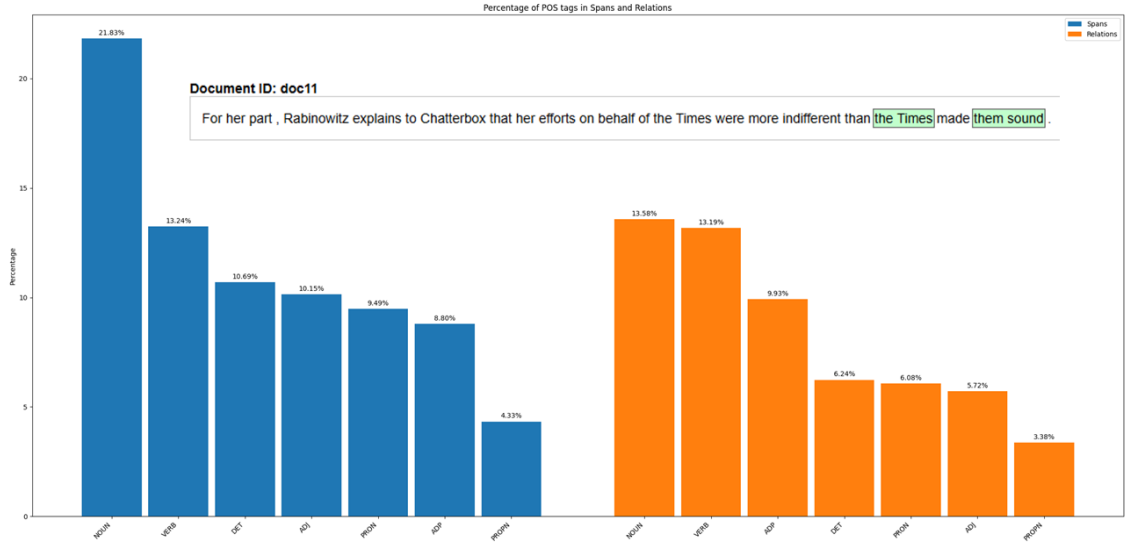

Figure 5: Because POS Breakdown

4.2.4 CTB

CTB seems to have short span annotations with a mix of verbs and nouns, but the annotations do not make a lot of sense when viewed as representing events or states.

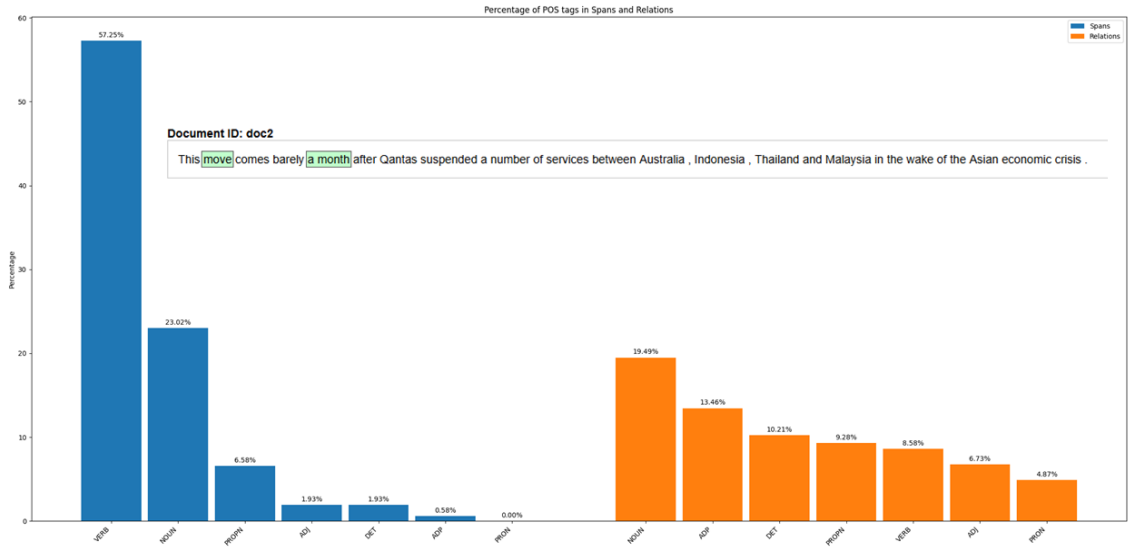

Figure 6: CTB POS Breakdown

4.3 Annotating a Custom Dataset

It was decided that in order to demonstrate the ability of the designed model to detect causal triplets, it would be best to simply annotate a custom dataset in-line with described model architecture. Chunks of observations from Altlex, Maven, Cassie, Semeval were gathered. Additionally some synthetic sequences produced by LLM prompting were added. The total observation count was approximately 900 observations. Primary issues with this dataset are the small size and the annotation noise. There is noise due to the single annotator and the short time taken to annotate. However there is also noise inherent to the ambiguous nature of annotating flat long spans representing events and states. It became clear during the annotation process that span boundaries become less defined the longer and more complex the text becomes. The maximum sequence length for this dataset was 200 words and the maximum span width was 80 words.

The annotations were done in an initial pass then used to train the model and which was then used to predict the correct annotations. These predictions were then used to review the human annotators inputs. While this was done for efficiency, it showed how the model could be useful as a first pass annotation guide.

Fig 7 shows the POS breakdown of the spans as well as the relation context tokens.

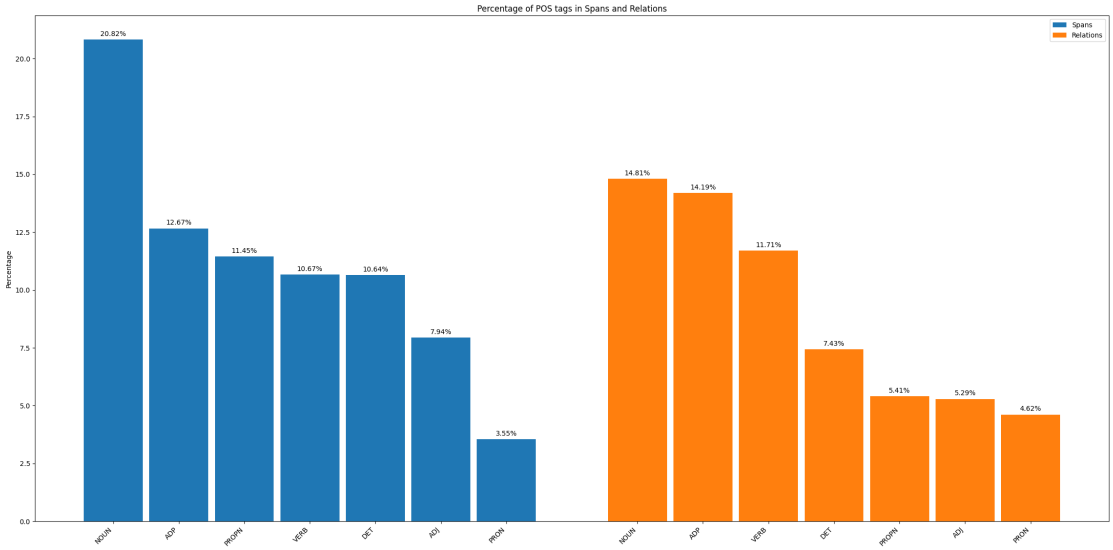

Figure 7: Custom Dataset POS Breakdown

Fig 8 gives some examples of the annotated observations from this dataset.

Document ID: doc0

An English textile industry was established in the 15th century , providing the basis for rapid English capital accumulation .

Document ID: doc1

In November 2006 , Lorin Maazel submitted Barenboim 's name as his nominee to succeed him as the New York Philharmonic 's music director .

Document ID: doc2

Many of these deaths were caused by war crimes committed by German and Japanese forces in occupied territories .

Document ID: doc3

Von Neumann acknowledged that the central concept of the modern computer was due to this paper .

Document ID: doc4

Consequently , the U.S. could find itself bombing operational missiles were the blockade to fail to force Khrushchev to remove the missiles already on the island .

Document ID: doc5

Low sea-levels meant that Britain was still attached to the continent for much of this earliest period of history , and varying temperatures over tens of thousands of years meant that it was not always inhabited at all .

Figure 8: Custom Dataset Examples

4.3.1 Annotation Guidelines

These annotation guidelines define a flat span-based approach for identifying events and states in text. This schema was chosen because it aligned well with the model architecture—specifically, a span classification framework that predicts full event spans directly. It also proved to be far more practical and intuitive than complex hierarchical trigger-argument-role schemes, especially for solo annotation. By selecting a single contiguous span that captures the event or state along with its relevant arguments, this method simplifies both the annotation process and the model's learning task.

That said, this approach is not without challenges. In sentences with convoluted struc-

ture—such as causal triggers surrounded by noisy or scattered arguments—it can be difficult to decide on clear span boundaries. These edge cases highlight some limitations of the flat span method, but in the majority of situations, it remains a practical and scalable solution that balances expressiveness with simplicity.

Flat Schema

- Each span represents a **single coherent event or state**.
- No nested structures or trigger/argument decomposition.
- Annotate the **best contiguous span** that expresses the event/state.

Span Scope: Include Relevant Arguments

- Include the event trigger and all relevant arguments (e.g., agent, theme, time, cause).
- Minor interleaved noise is acceptable if it improves span coherence.
- **Example:** *"A few months after the hotel's bombing the Government of Pakistan had reconstructed it"* is a valid single event span.

Span Boundary Preference

- Prefer the **widest natural span** that fully describes the event or state.
- Avoid fragmenting semantically integrated components (e.g., time or agent).

Annotating Embedded Events Separately Annotate embedded events as separate spans if:

- They are **independently eventive**.
- They could be the **head of a causal relation**.
- They are **referable** elsewhere in the text.
- They convey meaning **beyond acting as a modifier**.

Overlapping Spans Are Allowed

- Overlap is permitted if multiple distinct events share part of the same text.
- Flat schema means no hierarchy, but does not require disjoint spans.

What Not to Annotate

- Do **not** annotate attributional or discourse-level elements (e.g., *"according to Xinhua"*).
- Do **not** annotate standalone time expressions unless eventive.
  - **Yes:** *"the explosion on 5 May"*
  - **No:** *"in 2008"*

Causal Relations

- Annotate binary causal links between spans only if:

- One event **logically or directly causes** the other.

- Avoid speculative, metaphorical, or weakly implied causality.

Heuristic for Ambiguous Cases *"Would this text span make sense as a row in a knowledge graph?"*
If yes — annotate it.

4.4 Model Configuration Space

Rather than defining distinct model variants, a single modular pipeline was developed to support configurable architectural components. Most experiments were conducted by toggling or swapping individual modules, allowing controlled analysis of their impact.

Key configurable components included:

- **Backbone Model:** Swappable HuggingFace transformer (e.g., BERT base cased, SpanBERT base cased)
- **Subtoken Pooling Strategy:** Max-pooling (default), but framework allows alternate strategies
- **Span Generator and Filter:** Brute-force generation with binary filtering vs. token-level boundary tagging
- **Span Representation Strategy:** how the span representations were constructed from the word token embeddings, this was configurable and modular
- **Relation Representation Strategy:** how the relation representations were constructed from the word token embeddings, this was configurable and modular
- **BERT Sharing:** Shared vs. separate encoders for span and relation modules
- **BiLSTM Module:** Optional 3-layer BiLSTM over contextualized token representations
- **Graph Module:** Optional multi-head self-attention (transformer-style)
- **Projection Head:** FFN with expansion layer + ReLU, dropout vs. single linear layer
- **Top-K Spans:** the number of shortlisted spans from the span filter
- **Top-K Relations:** the number of shortlisted relations from the relation filter
- **Span/Rel Marking:** Briefly tested marked-sequence re-encoding through BERT; found too slow and ineffective

Common configurations across most variants:

- **Backbone:** BERT base cased or SpanBERT base cased
- **Subtoken Pooling:** Max-pooling
- **Backbone Sharing (Spans and Relations):** Enabled
- **Dropout:** 10%
- **Max Sequence Length:** Up to 200 word tokens
- **Max Span Width:** 80 word tokens
- **Projection Layer Type:** FFN with an intermediate expansion layer and nonlinearity, used instead of a single linear layer
- **BiLSTM:** 3 layers
- **Span Types:** Uniclass (each span assigned one of N types)

- **Relation Types:** Multilabel (each relation may take multiple of M types)
- **Span Representation Strategy:** `start + end + maxpool(inner) + width emb + [CLS]`, followed by FFN re-projection
- **Relation Representation Strategy:** `head + tail + cross-attn(head,context) + cross-attn(tail,context)`, followed by FFN re-projection
- **Span Width Embedding Size:** 100
- **Span Start/End Context Window Size:** 20% of span width
- **Redundant Span Pruning Threshold:** 80% overlap
- **Fallback for Missing Context Tokens:** Learned embedding
- **Relation Context Window (before/after):** 30 tokens
- **Top-K Spans:** 30
- **Top-K Relations:** 50
- **Graph MHA Transformer:** 3 layers, 8 heads
- **Relation Head Prediction Threshold:** 0.3

4.5 Training Configuration

The model was trained using the AdamW optimiser [19] with a learning rate of 1×10^{-5} for the pre-trained transformer encoder and 5×10^{-5} for all other parameters. Weight decay was set at 0.01.

A linear scheduler was used with a total of 20,000 steps (batches), including a warm-up phase of 2,000 steps.

Floating-point precision was maintained at full. Gradient accumulation was not employed. Gradient clipping was applied with an initial threshold of 10, linearly reduced to a minimum threshold of 1 halfway through training (at step 10,000).

The training batch size was set to 2, while inference utilized a batch size of 4. The terms 'batch' and 'step' are used interchangeably. The training set was shuffled, and the loader continuously cycled through batches until explicitly stopped. Training progress was controlled by specifying the number of steps rather than epochs.

The evaluation loop consisted of a single pass through the validation split. During training, evaluation was typically performed every 100 steps: training paused, and the model ran a full evaluation cycle over the validation split in inference mode, computing performance metrics and validation loss. These validation metrics were used to determine when to halt training. Additionally, a randomised evaluation interval was introduced to prevent periodic evaluation biases (see Section 5.3.2 on early stopping).

4.6 Loss Structure

While the model could be trained using only the span and relation classification losses, performance improved significantly when additional auxiliary losses were introduced, providing the model with greater flexibility to learn. The total loss was defined as the weighted sum of the following components:

- Token tagger or span filter loss
- Lost relation (from missing span) penalty
- Relation filter loss
- Graph structure Loss
- Span classification loss
- Relation classification loss
- Redundant Span Penalty, Hanging Relation Penalty

4.6.1 Warm-up Teacher Forcing

In all model variants, teacher forcing was implemented in the span and relation filtering stages. Specifically, logits (scores) for positive cases were artificially set to a large positive number, ensuring these cases always ranked at the top of the shortlisted K spans or relations. The remaining shortlisted cases thus served as negative samples that the model predicted as the most likely false positives. No other forms of teacher forcing were employed in the model pipeline.

It was observed that disabling this teacher forcing strategy after an initial warm-up period was beneficial for model performance, provided that an auxiliary penalty was introduced. This penalty, termed the *lost relation penalty*, penalised the model whenever a missed span prevented a positive relation from entering the final set of relation candidates. This effect can be clearly observed in Figure 15, where the teacher forcing is disabled at step 3000, triggering an immediate rise in the lost relation penalty. Subsequently, the penalty steadily decreases as the model improves its ability to identify correct spans.

4.7 Implementation and Hardware

The models were implemented in PyTorch using the HuggingFace Transformers library. Training was performed on T4 or L4 GPUs, using full-precision. Logging included evaluation every 50 steps, with best checkpoint saved based on a validation F1 related metric.

5 Results and Discussion

5.1 Quantitative Performance

Model performance is summarized in Table 1, 4 and Figure 9, 10. The following are some general trends observed from the data:

NOTE: performance metrics shown are for the validation set, not the test set. This was a result of resource, time and financial constraints. As will be shown later, the correlation between validation set and test set save scores was very high so it was not helpful to redo the experiments again. However, note that there is some variation so the test number could typically vary 5-10% from the validation set numbers

- the model results variance increases for the custom dataset, very likely due to the longer spans and more ambiguous boundaries. Additionally the custom dataset was not cleaned in any way.
- relation F1 is always below span F1, this is not suprising for this class of model or data as relations are constructed from spans, thus to identify an relation you need to identify two spans.
- conll04 and semeval08 had far better numbers than the custom dataset, again, most likely due to the shorter spans and less associated noise.
- spanbert cased worked better on the custom dataset, while bert cased worked better on the short span datasets (conll04 and semeval08). Again this is unsuprising given that the custom dataset consists of much longer spans. Additionally bert cased worked better than bert uncased, which is expected as no cleaning was done on the datsets.
- using separate transformer encoders for the spans and relations paths, did not have any benefit with a shared backbone giving better results.
- Token Tagging with a BE (Begin, End) scheme was as performant as the brute force method. The BECO (Begin, End, Combined, Other) token tagging scheme was not as good as the BE scheme. The Token tagging methods used far fewer compute resources also.
- attention pooling to form the span representations was inferior to the more simple method of concatenating max-pooled representations from the start window, inner window and end window along with the width embedding and the cls token from bert/spanbert. The attention pooling used excessive resources also.
- relation representation structure appeared to prefer the concatenation of the head, tail and context representations. With the head and tail being similar to the span representation minus the width embedding and cls embedding.
- for each relation, the context tokens were either the tokens between the spans and optionally the tokens in a window before and after the head and tail spans respectively. The pooling of these tokens was either max-pooling or cross attention pooling using

the head/tail representation. The cross attention pooling over the between and window context tokens performed best for the custom dataset. Max-pooling between only worked best for the short span datasets. This makes sense considering the more complex signals in the custom dataset, thus the cross attention over a wider area may have had an easier time finding it.

- using a lower span top K (30) for the span filter seemed to work better than a larger top K. The span top K greatly affects the resource utilisation as the number of candidate relations that need a representation are the square of the span top K. The span filter stage seemed to be good at finding the best spans and allowed the use of lower top K numbers. Potentially lower numbers could have been tested, the main limit is the maximum number of spans per observation. The relation top K was tested at 200 and 50 with no noticeable difference, so it was left at 200. The relation top K has less impact on resource usage.
- Disabling of both the LSTM after the transformer encoder and the graph transformer after the relation filter. The performance appeared to favour using the LSTM and graph transformer, although the differences were not great. Both of these structures worked by further enhancing the incoming representations (token representations for the LSTM, span/relation representations for the graph transformer). They did not add that much extra resource usage, so they were just left in for most configurations.
  Note: No targeted analysis was conducted to isolate which types of examples benefited from the graph transformer (e.g., complex chains or overlapping relations); evaluation was based solely on overall F1 metrics and training dynamics.
- warm-up teacher forcing appeared lead to better results as opposed to always on teacher forcing. The warm-up period was anywhere from 1000-3000 steps. Reasons for this could be that running the data through the model in training without labels allowed the model to learn under similar conditions to inference.
- Three penalties were used:
  1. lost relations (caused by the span filter missing spans)
  2. redundant spans (caused by the span classification head classifying very similar spans
  3. hanging relations (caused by the relation classification head classifying relations between one or both spans that did not make it through the span classification head

  Each one of these penalties was tied to the logit of the offending source span/relation so it would nudge the model. The penalties were enabled when teacher forcing was disabled. The penalties appeared to improve the results somewhat, however, it should be stated that the model already has quite a few avenues to learn from and it works

fine without these penalties, it may just take longer to learn.

- other aspects of the model not mentioned in these experimental results due to time constraints and uninteresting performance were:
  - marking spans and relations and re-running through bert showed no noticeable benefit to performance. Additionally it was extremely resource intensive.
  - cosine similarity loss as opposed to regular strict losses was not a viable training option, giving no usable signal to the model to learn from.

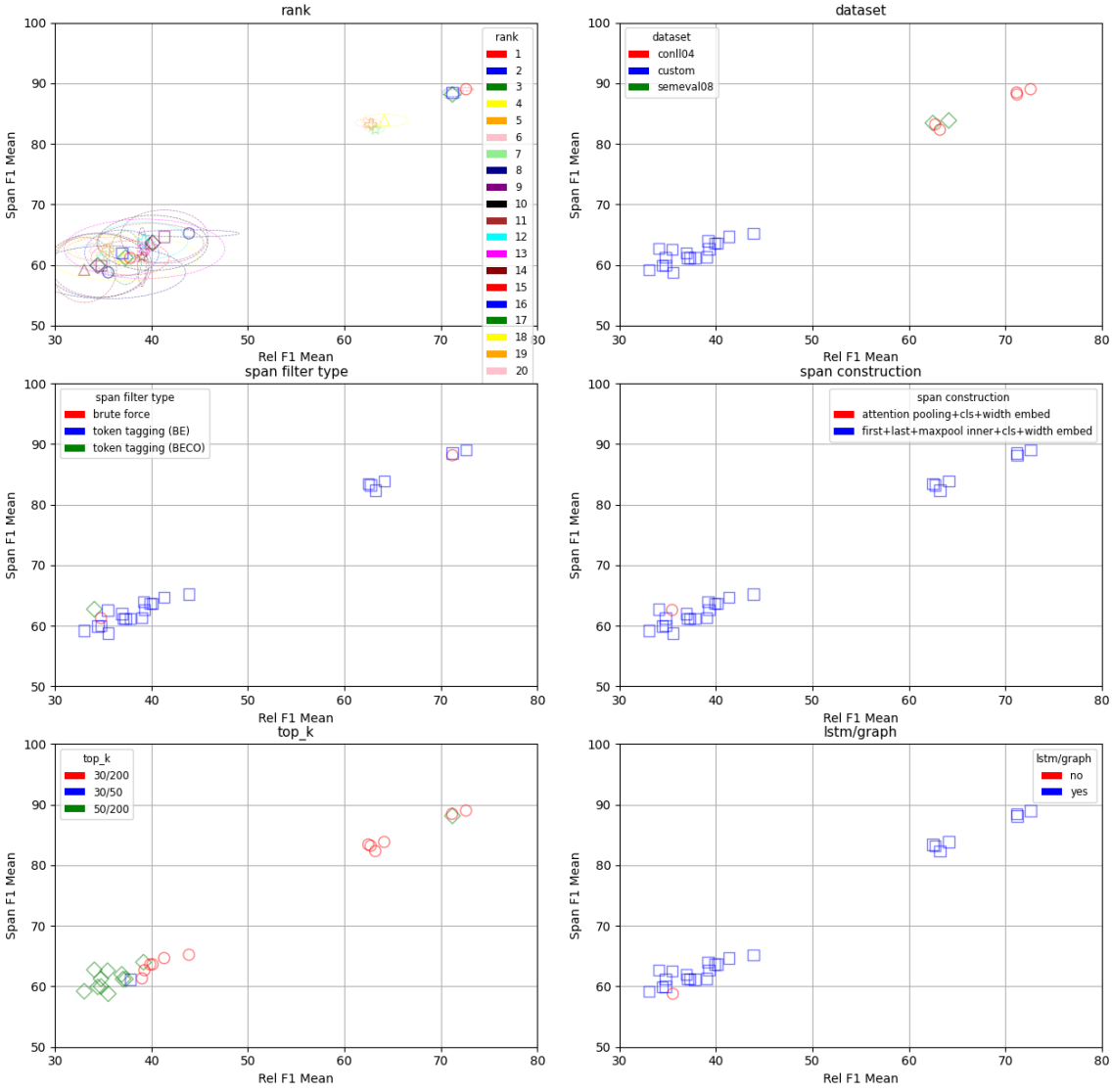

Figure 9: Experiment Results Part 1

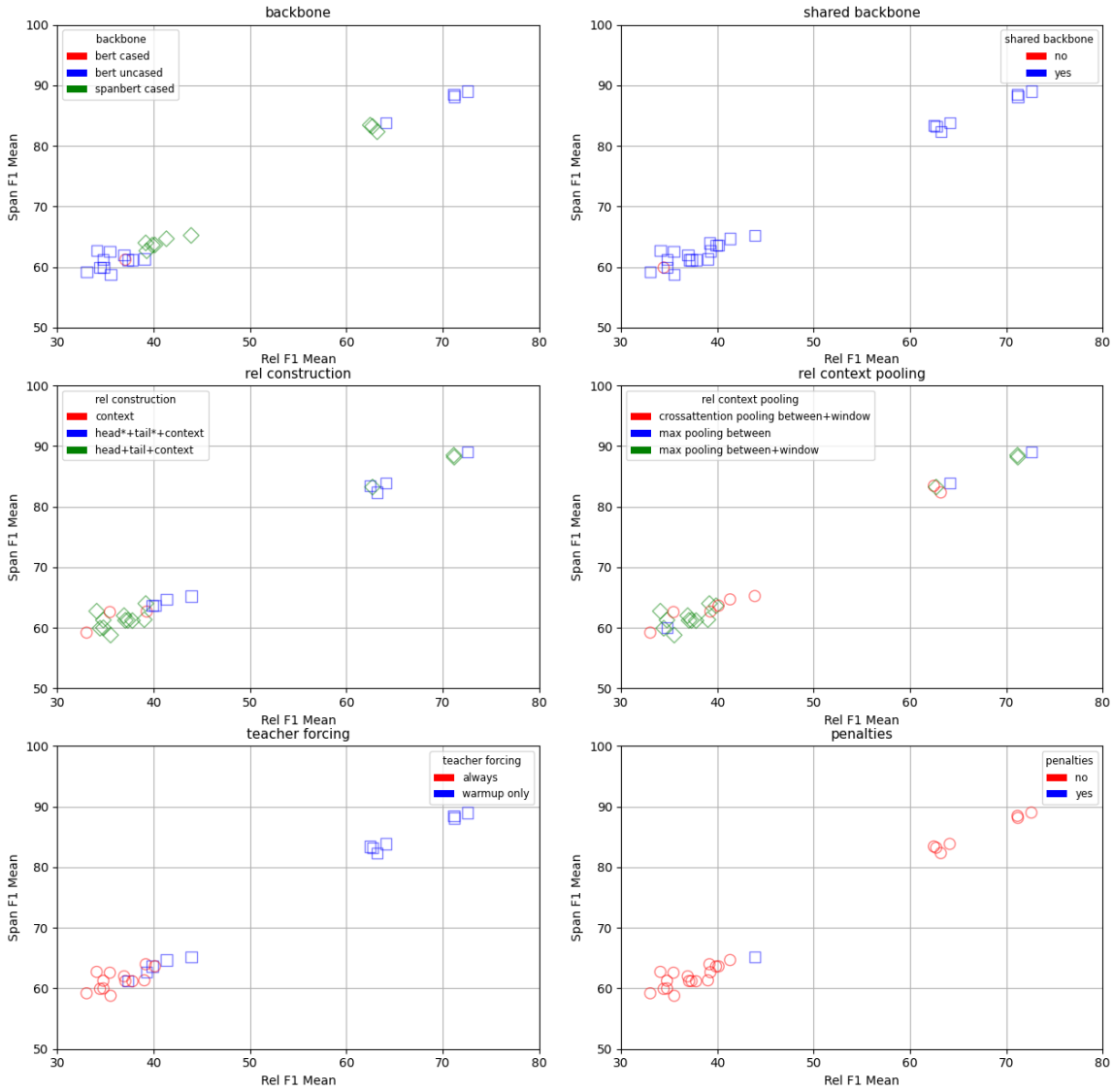

Figure 10: Experiment Results Part 2

Model	Span P	Span R	Span F1	Rel P	Rel R	Rel F1
conll04_1	89.11 ± 0.96	87.24 ± 1.26	88.15 ± 0.83	72.64 ± 2.61	69.87 ± 1.61	71.19 ± 0.92
conll04_2	89.36 ± 0.50	88.69 ± 1.01	89.02 ± 0.26	73.10 ± 0.57	72.11 ± 1.35	72.60 ± 0.82
conll04_3	89.23 ± 0.89	87.76 ± 0.83	88.48 ± 0.68	70.52 ± 0.73	71.82 ± 1.66	71.16 ± 1.03
conll04_4	82.79 ± 2.09	81.93 ± 1.18	82.34 ± 0.46	65.48 ± 3.38	61.23 ± 2.54	63.20 ± 0.94
conll04_5	84.71 ± 2.04	81.82 ± 0.81	83.23 ± 0.63	62.77 ± 1.09	62.68 ± 1.05	62.71 ± 0.21
semeval08_1	83.61 ± 2.87	84.15 ± 1.08	83.85 ± 1.08	65.43 ± 1.76	62.90 ± 3.51	64.12 ± 2.44
semeval08_2	83.43 ± 2.11	83.48 ± 0.81	83.43 ± 0.69	62.81 ± 1.68	62.20 ± 2.44	62.47 ± 1.37
custom_1	60.66 ± 3.42	61.78 ± 4.22	61.21 ± 3.80	41.23 ± 11.41	33.89 ± 5.79	37.06 ± 7.78
custom_2	58.91 ± 5.96	64.10 ± 2.18	61.26 ± 2.24	34.30 ± 0.21	35.61 ± 7.33	34.77 ± 3.66
custom_3	62.00 ± 2.39	63.20 ± 1.60	62.59 ± 2.01	34.33 ± 4.44	36.70 ± 4.24	35.45 ± 4.22
custom_4	58.32 ± 5.79	60.13 ± 5.08	59.20 ± 5.39	37.55 ± 5.55	29.55 ± 2.14	33.03 ± 3.33
custom_5	62.10 ± 3.93	61.94 ± 4.11	61.99 ± 3.79	38.31 ± 7.73	36.60 ± 4.07	36.92 ± 3.85
custom_6	57.52 ± 5.44	60.38 ± 5.48	58.78 ± 4.36	35.94 ± 8.58	35.17 ± 7.59	35.52 ± 8.01
custom_7	57.74 ± 4.20	62.27 ± 6.53	59.91 ± 5.29	35.83 ± 3.70	33.24 ± 6.63	34.43 ± 5.32
custom_8	60.56 ± 5.28	61.92 ± 3.85	61.21 ± 4.43	37.39 ± 3.44	37.35 ± 0.88	37.31 ± 1.69
custom_9	60.24 ± 5.84	59.74 ± 4.83	59.98 ± 5.29	35.54 ± 6.60	34.10 ± 3.94	34.77 ± 5.24
custom_10	61.51 ± 6.35	61.29 ± 4.43	61.33 ± 4.89	41.38 ± 0.35	36.92 ± 0.97	39.02 ± 0.48
custom_11	61.79 ± 6.29	60.85 ± 1.13	61.18 ± 2.85	37.86 ± 7.67	38.29 ± 1.74	37.79 ± 4.04
custom_12	61.77 ± 3.38	63.80 ± 1.85	62.73 ± 2.01	34.49 ± 2.25	33.74 ± 2.60	34.08 ± 1.98
custom_13	67.25 ± 1.70	63.34 ± 0.42	65.22 ± 0.59	45.87 ± 7.99	42.19 ± 2.79	43.88 ± 5.18
custom_14	61.27 ± 3.69	64.22 ± 6.42	62.68 ± 4.90	40.78 ± 9.73	38.23 ± 8.21	39.27 ± 8.32
custom_15	64.87 ± 2.90	62.52 ± 6.06	63.64 ± 4.55	45.76 ± 7.00	35.84 ± 5.27	40.12 ± 5.72
custom_16	63.77 ± 6.89	65.91 ± 3.45	64.68 ± 4.41	44.15 ± 6.06	39.02 ± 4.09	41.32 ± 4.53
custom_17	62.72 ± 2.75	64.58 ± 3.87	63.62 ± 3.27	42.55 ± 9.50	37.94 ± 3.00	39.87 ± 5.54
custom_18	64.28 ± 3.52	63.80 ± 3.72	63.99 ± 3.10	41.05 ± 6.97	37.72 ± 3.28	39.17 ± 4.60

Table 1: Micro-averaged strict precision (P), recall (R), and F1 scores for span and relation extraction.

5.2 Ablation Studies

For investigation into some of the key features of the model, four aspects were selected for ablation studies, namely:

- Token Tagging vs Brute Force Span Search
- Relation Context Pooling Method
- Use a Graph Transformer
- Use of Penalties

The Ablation results were collected from the test set metrics on the custom long span dataset using 3 random seeds and 2 runs per seed. The seed refers to the seed used to make the train-val-test splits. The random seed used to initialise the model was always 42. The model initialisation random seed was not varied as there were so many other sources of variation in the training process, it made no difference. Repeated runs with identical seeds still resulted in different outcomes due to the unstable nature of the training process. This variance complicated both early stopping and model comparison

The baseline had the following configuration:

- backbone = spanbert base cased
- BiLSTM with 3 layers
- max span length = 200, max span width = 80 (50 for the brute force comparison)
- span representation method = start window maxpool, inner maxpool, end window maxpool, width embedding, cls embedding
- span filtering = token tagging with BE scheme
- relation representation construction = head*, tail*, context. Where * denotes span representations without width and cls embeddings.
- relation context pooling = cross attention with the head and tail spans
- top K spans = 30, top K relations = 200
- graph transformer with 8 heads and 3 layers
- using lost relation, redundant span and hanging relation penalties

One overall observation is consistent with the other experimental results is that the relation metrics have a much higher variance than the span metrics. This is not suprising due to the pipeline architecture and the relations being dependent on spans. Lastly, these ablation results highlight the sensitivity of relation extraction performance under noisy settings, and show that several architectural and training decisions, though not significant in isolation, can compound into more pronounced effects.

Model	Span P	Span R	Span F1	Rel P	Rel R	Rel F1
baseline (sw 50)	59.06 ± 1.53	59.88 ± 1.48	59.46 ± 1.25	37.84 ± 5.06	36.25 ± 7.47	36.64 ± 5.36
brute force spans	59.58 ± 1.58	61.31 ± 2.11	60.41 ± 1.39	40.29 ± 5.85	37.7 ± 5.06	38.89 ± 5.16
baseline	60.25 ± 1.97	59.74 ± 1.74	59.98 ± 1.58	38.76 ± 7.58	37.79 ± 7.01	38.15 ± 7.03
no graph	58.13 ± 1.28	59.89 ± 1.59	58.98 ± 1.09	35.65 ± 6.91	36.07 ± 6.73	35.68 ± 6.18
graph 12h-6l	56.69 ± 2.29	58.77 ± 2.57	57.64 ± 0.88	35.89 ± 8.22	36.93 ± 5.06	36.27 ± 6.56
graph 16h-8l	58.13 ± 1.41	58.34 ± 2.48	58.22 ± 1.69	32.98 ± 6.32	35.21 ± 7.67	34.01 ± 6.86
graph 32h-16l	58.06 ± 2.56	58.89 ± 1.99	58.45 ± 2.0	35.36 ± 5.7	38.56 ± 4.87	36.87 ± 5.25
baseline	60.25 ± 1.97	59.74 ± 1.74	59.98 ± 1.58	38.76 ± 7.58	37.79 ± 7.01	38.15 ± 7.03
no consist penalties	57.12 ± 2.23	59.28 ± 1.54	58.16 ± 1.61	35.88 ± 8.06	38.14 ± 8.27	36.96 ± 8.15
no lr penalties	56.99 ± 3.67	59.25 ± 3.33	58.02 ± 2.63	37.2 ± 6.68	36.22 ± 5.68	36.54 ± 5.68
no penalties	57.08 ± 3.39	58.28 ± 1.68	57.64 ± 2.29	31.95 ± 3.27	34.88 ± 6.82	33.25 ± 4.6
baseline	60.25 ± 1.97	59.74 ± 1.74	59.98 ± 1.58	38.76 ± 7.58	37.79 ± 7.01	38.15 ± 7.03
rel cxt crossattn 16h	60.39 ± 3.22	61.17 ± 2.26	60.76 ± 2.59	35.67 ± 8.09	37.12 ± 7.59	36.17 ± 7.35
rel cxt maxpool	57.56 ± 2.13	60.22 ± 2.89	58.84 ± 2.21	35.96 ± 6.12	37.41 ± 3.99	36.44 ± 4.34

Table 2: Micro-averaged strict precision (P), recall (R), and F1 scores for span and relation extraction across ablation groups.

5.2.1 Token Tagging vs Brute force Span Search

Two competing span filtering methods where evaluated, token tagging vs brute force span search. The advantage of token tagging is that is could be scaled to much larger sequence lengths and span widths as was far more resource efficient than the brute force method. In this comparison the tagging scheme used in the baseline was BE (Begin-End). For the brute force setup, span representations were made for all possible spans given the batch max sequence length and max span width. An A100 processor was required given the GPU requirements of the span representations tensor and even then it could only support a max span width of 50. The span representations tensor is passed through a binary filter head and the top K spans are chosen based on the logit score. The primary advantage of the brute force method over the token tagging is that it is less likely to miss spans.

Figure 11 shows the comparison. The brute force method achieved slightly better test set metrics than the baseline, although the range of values was broadly similar across both. In short, brute force is preferable when feasible (e.g., NER-RE tasks), but impractical for longer spans typical of EE-RE tasks.

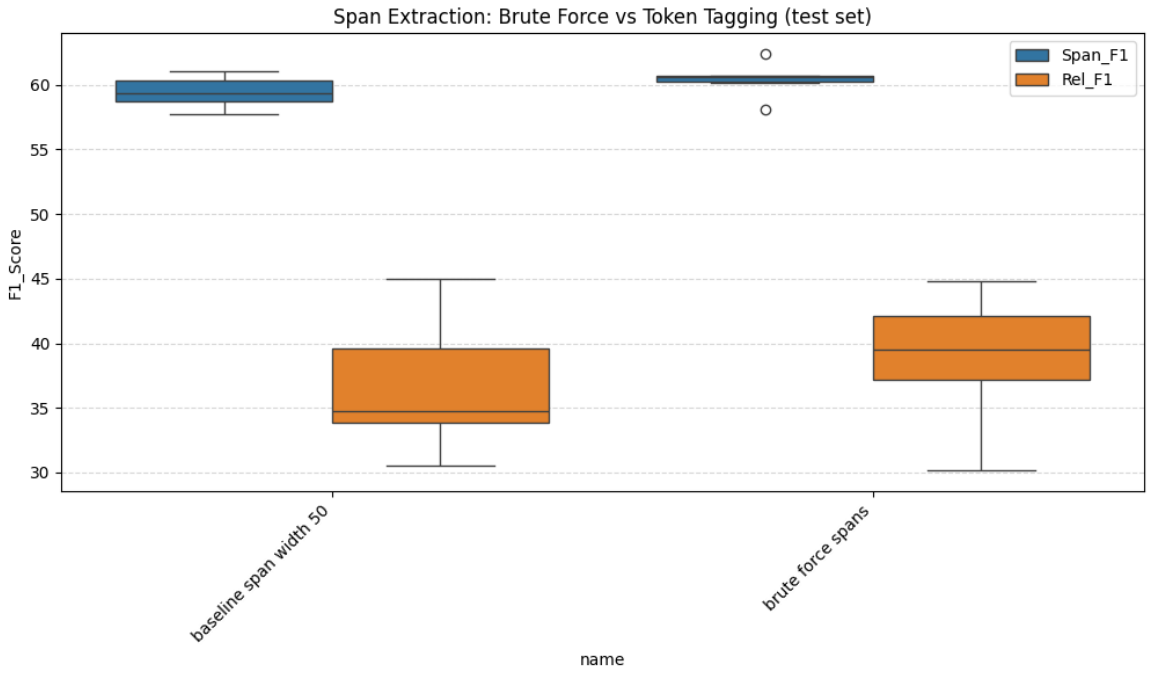

Figure 11: Token Tagging vs Brute Force Span Search

5.2.2 Relation Context Pooling Method

This compared the method for pooling the variable number of context token embeddings into a context representation used to construct the relation representation. The baseline uses the concatenation of the cross-attention with the head and tail span representations. Each cross-attention block uses the head or tail span representation as query and the context token embeddings as key, value. The first variant tested was using 16 heads for this MHA block as opposed to the baseline of 8 heads. The second variant tested used much simpler max-pooling over the context tokens.

Figure 12 shows minimal performance differences between methods. The baseline appears marginally stronger overall. Max-pooling tended to underperform in relation F1, though occasional poor runs were also observed for the baseline. This is consistent with earlier observations relating to noise in the annotation process, span boundaries, and causality signals.

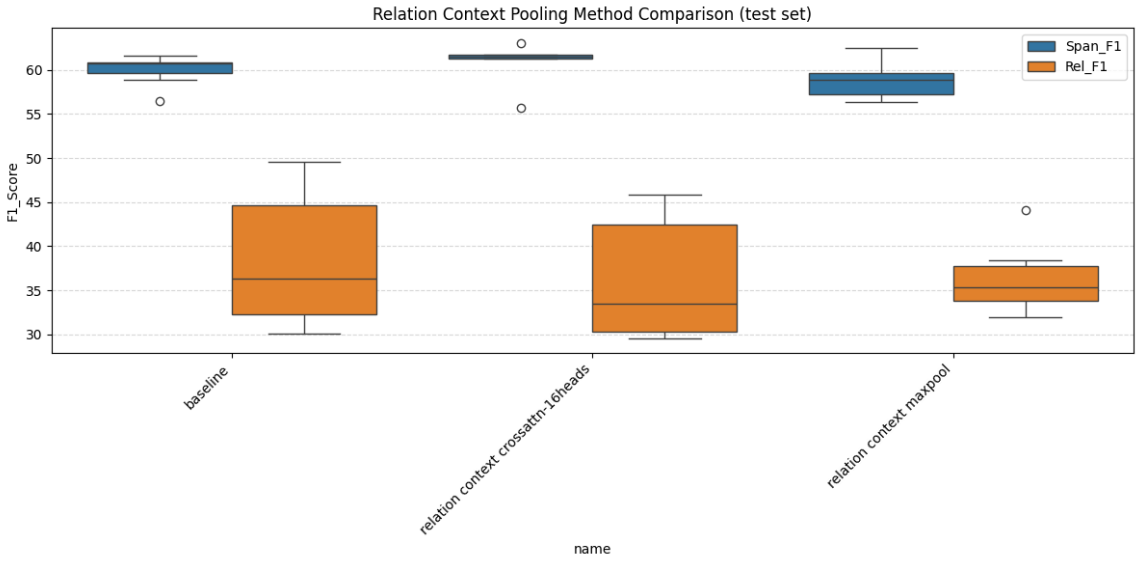

Figure 12: Relation Context Pooling Method

5.2.3 Graph Transformer

The graph transformer acts as an optional enrichment module for span (node) and relation (edge) representations before final classification. Validation results indicated a possible small benefit, and resource usage was low enough to justify its inclusion.

The ablation compared the baseline configuration (8 heads, 3 layers) against both a no-graph setup and larger graph variants. As shown in Figure 13, using a graph transformer had a marginal benefit over not using one. However, increasing its size beyond the baseline did not yield further gains and sometimes degraded performance.

In short, a lightweight graph transformer may offer small improvements at low cost, but larger configurations do not appear beneficial.

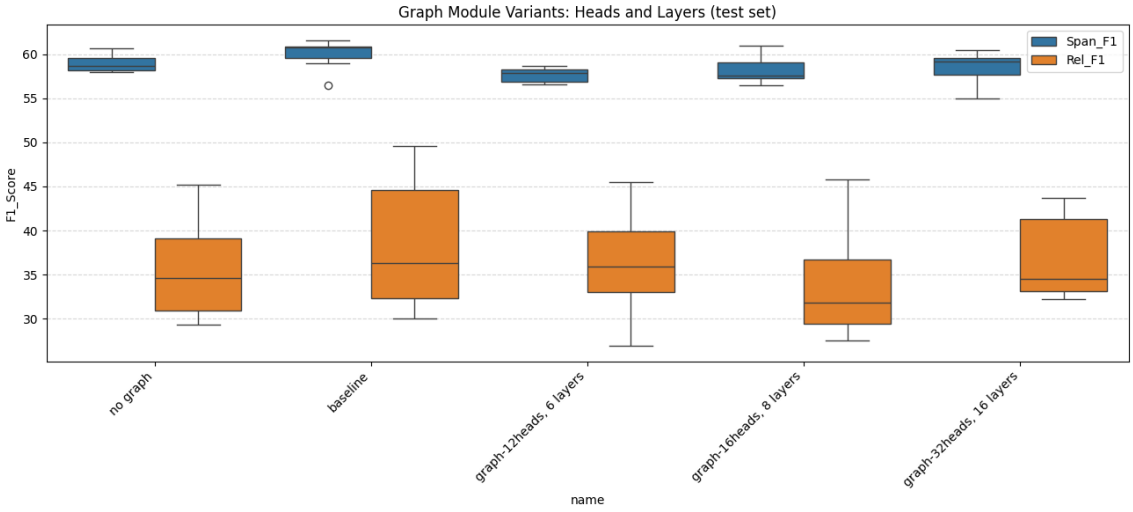

Figure 13: Graph Transformer

5.2.4 Use of Penalties

Learnable penalties were compared against the baseline which used both lost relation penalties as well as consistency penalties (redundant span, hanging relation). The results would appear to favour using both types of penalties which effectively discourage the model from making common span-relation consistency errors and span filtering errors.

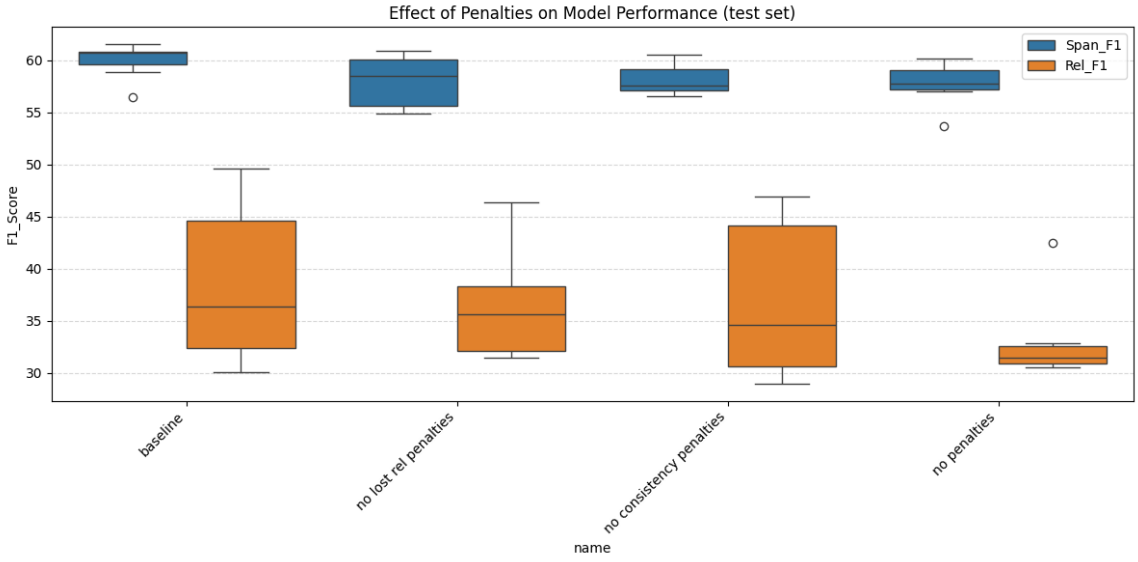

Figure 14: Penalties

5.3 Training Dynamics

5.3.1 Train Loss

As is shown in the model diagrams, 1,2 multiple losses are generated along the model pipeline in order to give the opportunity for different aspects to send training signal to the gradients. The following figure (15) shows the various training loss components changing during a training run.

NOTE: the discontinuity at step 3000 is due to the disabling of teacher forcing and the enabling of penalties

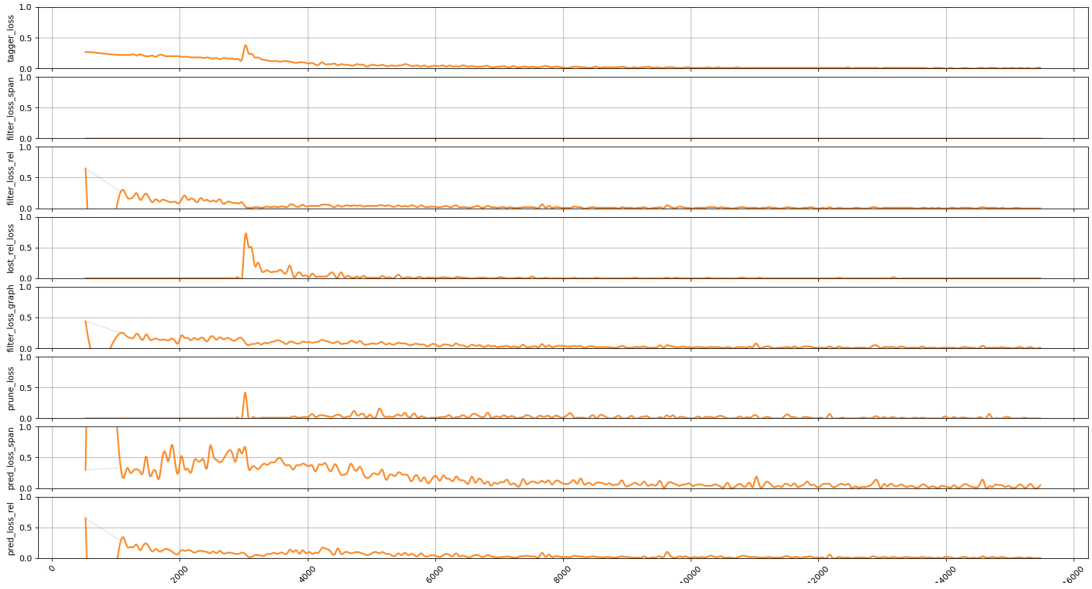

Figure 15: Train Loss Breakdown vs Train Step

5.3.2 Early Stopping

This particular model being a more complex joint loss model with multiple interacting stages appeared to have some decoupling between validation set loss and validation set metrics (precision, recall, f1). As shown in fig 16 and 17 the train loss decreases throughout the training, while the validation set loss hits a minima while still warming up and before the model effectively starts predicting relations. As a result the training was ended not by looking for a minima in the validation set loss, but instead, looking for a maxima in a performance metric. This metric (save score) is simply the addition of the span and relation F1 with each one moderated by a balance factor as show below:

$$\textbf{Save Score} = \textbf{SpanF1} \cdot \left(\frac{\min(\textbf{SpanP}, \textbf{SpanR})}{\max(\textbf{SpanP}, \textbf{SpanR})} \right)^x + \textbf{RelF1} \cdot \left(\frac{\min(\textbf{RelP}, \textbf{RelR})}{\max(\textbf{RelP}, \textbf{RelR})} \right)^x$$

where x penalizes P/R imbalance. ($x = 2$ was used)

Additionally, the period at which the trainer checks the validation set save score was randomised around a given period (typically 50 steps). This helped to find a good stopping point.

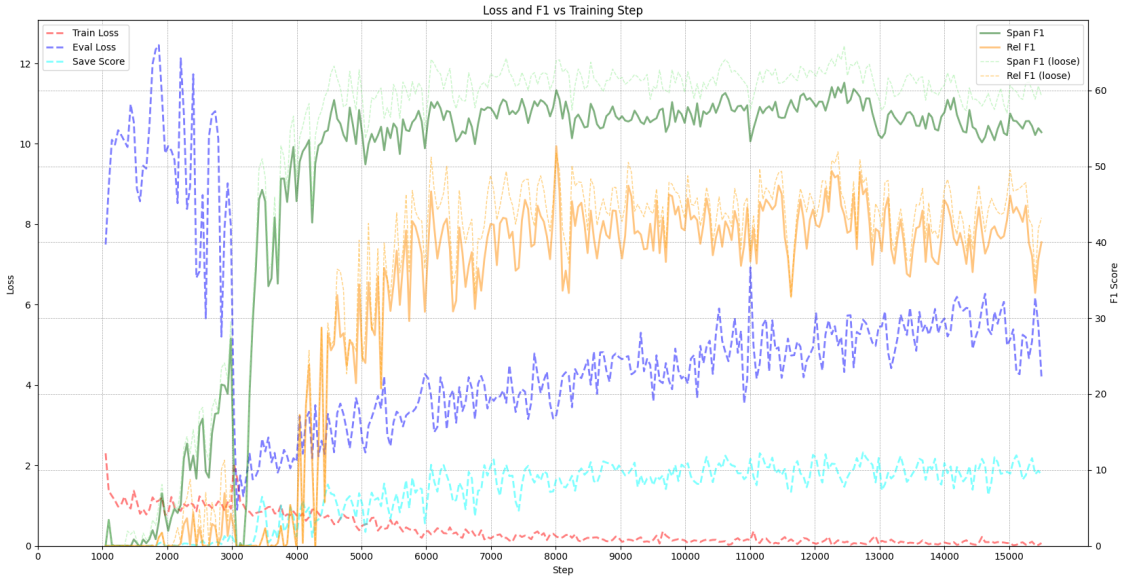

Figure 16: Training Results

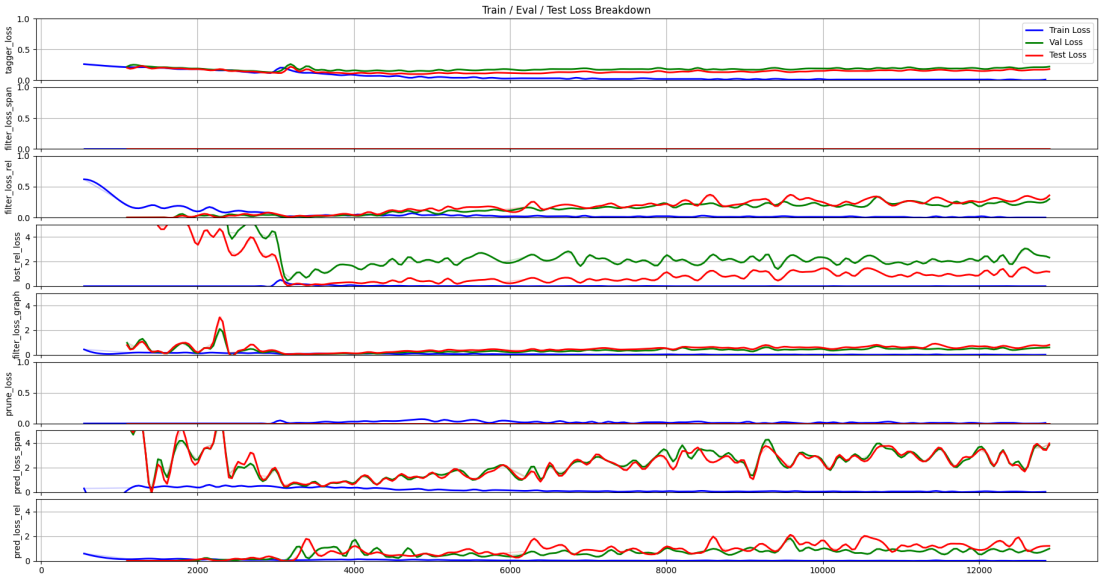

Figure 17: Train/Val/Test Loss Breakdown vs Train Step

There is concern about using validation set metric peaks to choose the stopping point of the training, this is not standard practice and there is no guarantee that a spike in validation metrics will be the same for test set metrics and broader model generalisation. But this was

the only real way of finding a stopping point. The main issue was that the model would start to identify spans and the span metrics would plateau, the relation metrics however, would only start appearing after the span F1 breached 40-50%, then the relation F1 would rise and plateau more slowly with more variance. Thus the ideal stopping point would be soon after the relation F1 plateaus. Perhaps with a cleaner, larger dataset, some of these patterns may have been different and a modified early stopping algorithm developed.

To quantify potential issues with divergence of validation and test set performance, they were analysed. The scatter plot in Fig 18 and line plot 19 confirm this relationship, showing a strong positive correlation (Pearson r = 0.974) between the validation and test save scores, despite modest variance at higher performance levels.

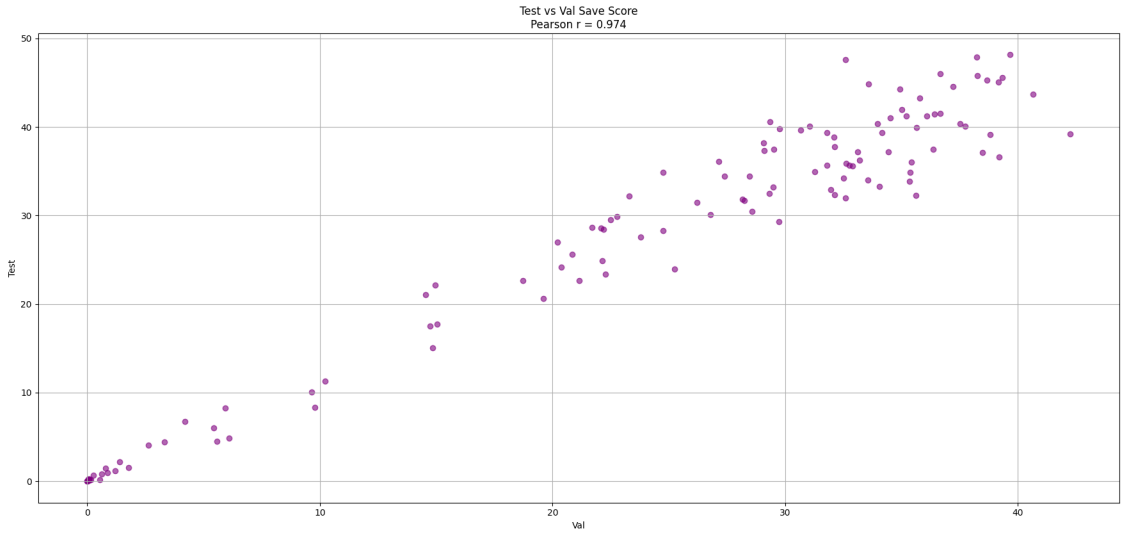

Figure 18: Test Set vs Validation Set Save Score Correlation

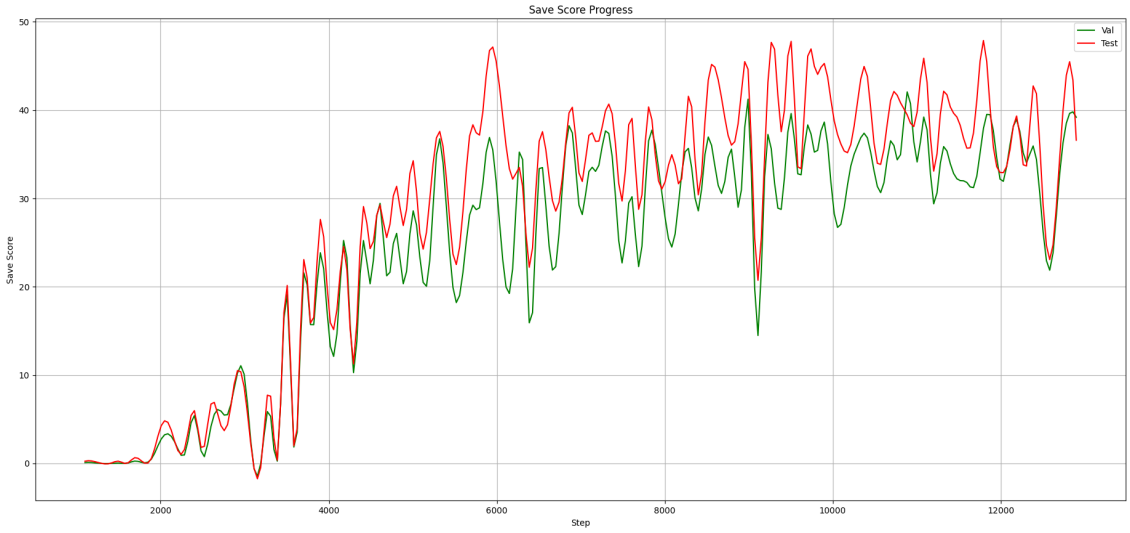

Figure 19: Test Set vs Validation Set Save Score Line Plot

5.4 Qualitative Evaluation and Error Analysis

Some human and LLM aided reviews of the model predictions vs labels were undertaken with some general outcomes indicating that the model's performance was not as poor as the F1 metrics for several reasons. The most common being poor quality annotations that did not actually abide by the annotation guidelines or were just plain wrong. Another reason is that F1 metrics can be very strict in the sense of span boundaries, while human and LLM review can be more flexible. The following is an output from one such review. Please note that due to the variability of the model and stopping point, every model run will give slightly different results.

What stands out is that in this particular model and review, for 67% of cases, the predictions were preferable or as good as the labels. The other key point is that almost 30% of cases had annotation problems.

Category	Percentage
Winner: predictions	28%
Winner: labels	32%
Winner: both	39%
Winner: neither	4%
Annotation issues flagged	28%

Table 3: Summary of evaluation outcomes.

Model Strengths:

- **Improved Span Boundary Adherence:** The model often avoids overly broad or speculative spans present in labels. It demonstrates better granularity and argument inclusion.
- **Better Causal Link Detection:** Several times, the model detects plausible causal links that labels miss due to noisy or incomplete annotations.
- **Avoids Common Label Errors:** For example, the model avoids labeling hypothetical/non-eventive spans or attributive phrases, A common mistake in labels.
- **Embedded Events:** It successfully segments embedded or compound events more accurately than the labels, in some cases.

Model Weaknesses:

- **Missed Relations:** The model occasionally fails to predict causal relations even when spans are correct. This was the most frequent reason for preferring labels.
- **Over-fragmentation or Incomplete Spans:** Some spans are too short or fragmented, excluding critical arguments or temporal markers.
- **Misinterpretation of Causality Structure:** In a few instances, causal chains are mislinked (e.g., linking cause directly to final result, skipping intermediate steps).

In summary, despite imperfections, the model exhibited useful generalization and meaningful output on ambiguous or complex examples, occasionally surpassing the reference annotations.

5.5 Example Cases

Here some of the test set cases are shown with a quick explanation of what is happening. Four examples have been chosen that demonstrate failures of both the model and the human annotator.

In Fig 20, is an example of a case with a quote, which has confused the model and potentially the annotator. The quote could have been left out entirely from the annotation with just 2 two events A) the church not granting his favour and B) Henry citing the passage in the Book of Leviticus. The model has gone for “They shall be childless”, which is not incorrect, but why is it causally related to the church event? In regards to the quote, should it be left in, should its events and states be separate spans. This shows some of the ways language can create convoluted situations that make both annotation and training simplistic models difficult.

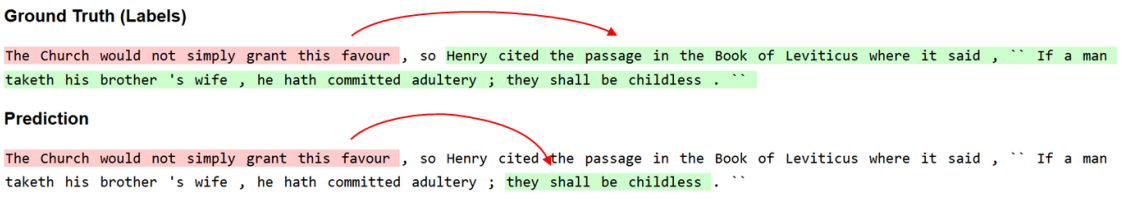

Figure 20: Example 1

Fig 21, is a generally straight forward case. Possibly the models two spans are cleaner than the annotator who has included an additional span which is really just a time attribute of the crash event, so potentially it should have been included with the crash event span, but then why did the model not include it. However, both the annotator and model got the simple causal relation.

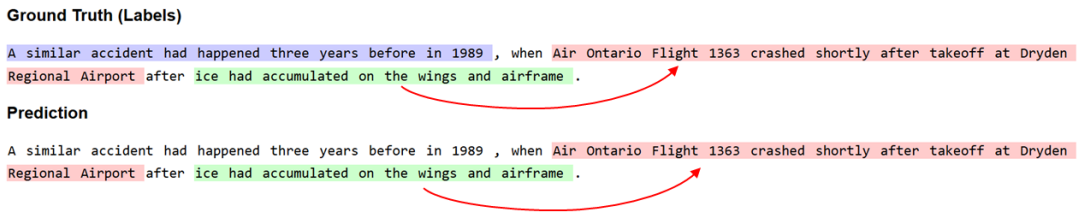

Figure 21: Example 2

Fig 22, is a longer text, which comes from a news report which includes quotes. There are many events and states that could be annotated in this text thus making it a good example of the conundrum of event and state annotation. The annotator has gone for a minimalist annotation here picking the two most obvious key causal event pairs and just ignoring anything else. The model has gone and annotated more spans, some make sense, some questionable. It has additionally missed the cyber-attack event. It did pick one good causal relation. However, neither the predictions, nor the human annotations are comprehensive in this example.

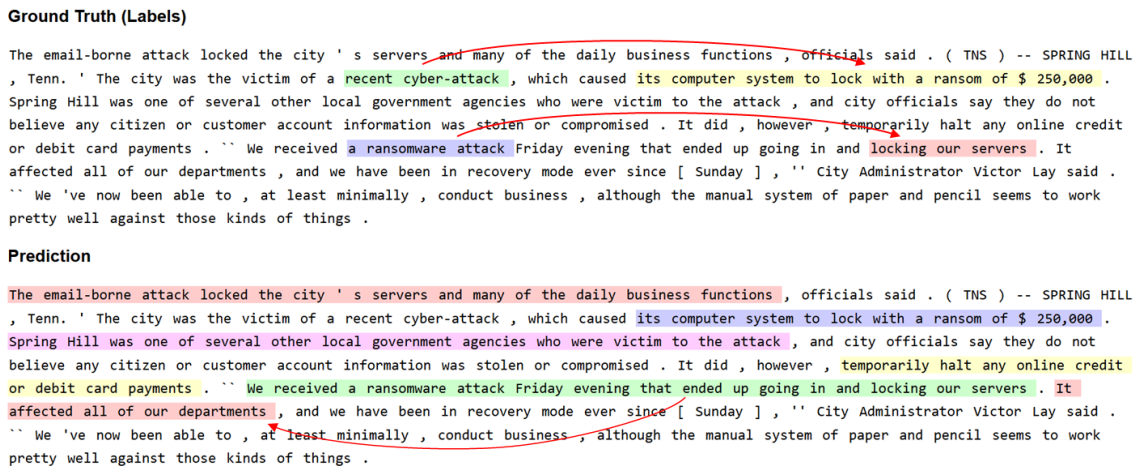

Figure 22: Example 3

Fig 23, is a moderate length sequence with much causality. The human has gone for all the contributing factors as causing “the disaster” as well as linking the worker related states to the last worker actions span. The model has come up with several overlapping spans and linked most of them back to “the disaster” as well as the relation between the worker training and the worker actions. Overall, the model had issues with overlapping spans and some questionable spans and missed relations.

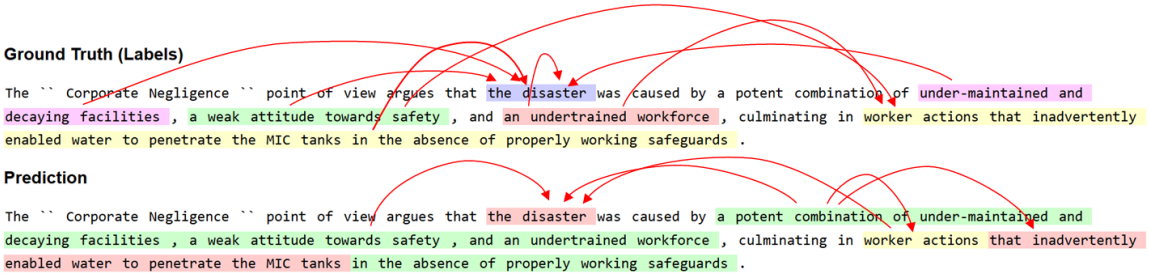

Figure 23: Example 4

6 Conclusion and Future Work

6.1 Key Findings

- Flat span based discriminative models are indeed capable of extracting useful causal structure. However, performance declines on longer spans, implicit causal signal, complex causal chains and ambiguous and convoluted language structures. To stay within the safe working zone for this class of discriminative models would give significantly better results but would require limiting the input text to shorter spans with explicit causal indicators and keeping convoluted and complex linguistic structure and semantics to a minimum as these just confuse the model and are arguably beyond the it's ability.
- The quality of training labels is a critical bottleneck — rushed or noisy spans yield poor supervision. A good metric as to how appropriate the annotation scheme is (and model architecture), would be to review the annotators feelings on the task. If the annotation guidelines are confusing, the annotators will be confused and frustrated indicating that the whole annotation strategy may be not the best way of breaking down the causal structures in the text.
- Qualitative outputs often exceeded label quality, indicating the model learned valid causal abstractions beyond strict token boundaries. Thus softer metrics or review is beneficial, this job can be done quite well with the help of LLMs.

6.2 Limitations

- The small dataset with inconsistent annotations had a performance hit for the model.
- Training and Evaluation is limited to strict span matching, missing nuanced improvements.
- This class of model appears to be limited in its ability to deal with complex language, which is expected. This is a key limitation though and is important to understand when choosing this kind of model for training.

6.3 Future Directions

- Improve annotation quality and quantity via collaborative multi-pass review and perhaps spending more resources on the topic of weakly supervised annotators which holds a lot of promise [20].
- Potentially look at short span trigger-argument style event extraction or using LLMs to normalise the incoming text into more simplistic code style language that this class of model could digest.
- Test generative models End-to-End (e.g., [1, 8, 2]) that are robust to boundary variation.

6.4 Re-framing the Problem

Ultimately, causal information extraction from raw natural text may be ill-posed for smaller, task-specific discriminative models. A more effective pipeline may instead involve first transforming unstructured text, potentially via LLMs, into simplified, causally explicit textual representations. Span and relation extraction could then operate more reliably on this structured input. This re-frames the problem as one of layered causal reasoning, where upstream semantic abstraction becomes a necessary precondition for effective downstream extraction, rather than relying on direct parsing of raw linguistic complexity.

Appendix A: Experiment Configuration Details

Model	Features
conll04_1	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: brute force, **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
conll04_2	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head*+tail*+context, **rel context pooling**: max pooling between, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
conll04_3	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
conll04_4	**backbone**: spanbert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head*+tail*+context, **rel context pooling**: crossattention pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
conll04_5	**backbone**: spanbert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
semeval08_1	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head*+tail*+context, **rel context pooling**: max pooling between, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no

Model	Features
semeval08_2	**backbone**: spanbert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head*+tail*+context, **rel context pooling**: crossattention pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
custom_1	**backbone**: bert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no
custom_2	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: brute force, **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no
custom_3	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: attention pooling+cls+width embed, **rel construction**: context, **rel context pooling**: crossattention pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no
custom_4	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: context, **rel context pooling**: crossattention pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no
custom_5	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no

Model	Features
custom_6	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 50/200, **lstm/graph**: no, **teacher forcing**: always, **penalties**: no
custom_7	**backbone**: bert uncased, **shared backbone**: no, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no
custom_8	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
custom_9	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no
custom_10	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no
custom_11	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 30/50, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no
custom_12	**backbone**: bert uncased, **shared backbone**: yes, **span filter type**: token tagging (BECO), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no

Model	Features
custom_13	**backbone**: spanbert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head*+tail*+context, **rel context pooling**: crossattention pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: yes
custom_14	**backbone**: spanbert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: context, **rel context pooling**: crossattention pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
custom_15	**backbone**: spanbert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head*+tail*+context, **rel context pooling**: crossattention pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no
custom_16	**backbone**: spanbert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head*+tail*+context, **rel context pooling**: crossattention pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
custom_17	**backbone**: spanbert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head*+tail*+context, **rel context pooling**: max pooling between+window, **top_k**: 30/200, **lstm/graph**: yes, **teacher forcing**: warmup only, **penalties**: no
custom_18	**backbone**: spanbert cased, **shared backbone**: yes, **span filter type**: token tagging (BE), **span construction**: first+last+maxpool inner+cls+width embed, **rel construction**: head+tail+context, **rel context pooling**: max pooling between+window, **top_k**: 50/200, **lstm/graph**: yes, **teacher forcing**: always, **penalties**: no

Table 4: Config Details for Experiments

References

[1] Y. Lu, Q. Liu, D. Dai, X. Xiao, H. Lin, X. Han, L. Sun, and H. Wu, "Unified structure generation for universal information extraction," 2022.

[2] X. Wang, W. Zhou, C. Zu, H. Xia, T. Chen, Y. Zhang, R. Zheng, J. Ye, Q. Zhang, T. Gui, J. Kang, J. Yang, S. Li, and C. Du, "Instructuie: Multi-task instruction tuning for unified information extraction," 2023.

[3] M. Eberts *et al.*, "Span-based joint entity and relation extraction with transformer pre-training," in *proceedings of ECAI 2020*, 2020.

[4] U. Zaratiana, N. Tomeh, N. E. Khbir, P. Holat, and T. Charnois, "Grapher: A structure-aware text-to-graph model for entity and relation extraction," 2024.

[5] X. Liu, Z. Luo, and H. Huang, "Jointly multiple events extraction via attention-based graph information aggregation," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2018.

[6] S. Yang, D. Feng, L. Qiao, Z. Kan, and D. Li, "Exploring pre-trained language models for event extraction and generation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 5284–5294, Association for Computational Linguistics, July 2019.

[7] D. Wadden, U. Wennberg, Y. Luan, and H. Hajishirzi, "Entity, relation, and event extraction with contextualized span representations," 2019.

[8] O. Sainz, I. García-Ferrero, R. Agerri, O. L. de Lacalle, G. Rigau, and E. Agirre, "Gollie: Annotation guidelines improve zero-shot information-extraction," 2024.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[10] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Online), pp. 38–45, Association for Computational Linguistics, 2020.

[11] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, and R. Vollgraf, "Flair: An easy-to-use framework for state-of-the-art nlp," in *Proceedings of the 2019 Conference of the*

North American Chapter of the Association for Computational Linguistics (Demonstrations), pp. 54–59, Association for Computational Linguistics, 2019.

[12] B. Ji, S. Li, H. Xu, J. Yu, J. Ma, H. Liu, and J. Yang, "Span-based joint entity and relation extraction augmented with sequence tagging mechanism," *arXiv preprint arXiv:2210.12720*, 2022.

[13] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2524–2527, IEEE, 1997.

[14] U. Zaratiana, P. Holat, N. Tomeh, and T. Charnois, "Hierarchical transformer model for scientific named entity recognition," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

[15] F. A. Tan, X. Zuo, and S. Ng, "Unicausal: Unified benchmark and repository for causal text mining," in *Big Data Analytics and Knowledge Discovery - 25th International Conference, DaWaK 2023, Penang, Malaysia, August 28-30, 2023, Proceedings* (R. Wrembel, J. Gamper, G. Kotsis, A. M. Tjoa, and I. Khalil, eds.), vol. 14148 of *Lecture Notes in Computer Science*, pp. 248–262, Springer, 2023.

[16] D. Mariko, H. Abi-Akl, K. Trottier, and M. El-Haj, "The financial causality extraction shared task (fincausal 2022)," in *Proceedings of the 4th Financial Narrative Processing Workshop @LREC2022*, (Marseille, France), pp. 105–107, European Language Resources Association, June 2022.

[17] R. Prasad, N. Dinesh, A. Lee, A. Joshi, and B. Webber, "Attribution and its annotation in the penn discourse treebank," in *Proceedings of the Linguistic Annotation Workshop*, (Prague, Czech Republic), pp. 31–38, Association for Computational Linguistics, 2007.

[18] R. Prasad, B. Webber, A. Lee, and A. Joshi, "Penn discourse treebank version 3.0," 2019.

[19] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *International Conference on Learning Representations (ICLR)*, 2019.

[20] R. Zhang, Y. Li, Y. Ma, M. Zhou, and L. Zou, "Llmaaa: Making large language models as active annotators," 2023.