# Paths to Relation Extraction through Semantic Structures

**Jonathan Yellin** **Omri Abend**
School of Computer Science and Engineering
The Hebrew University of Jerusalem
{jonathan.yellin,omri.abend}@mail.huji.ac.il

## Abstract

Syntactic and semantic structure directly reflect relations expressed by the text at hand and are thus very useful for the relation extraction (RE) task. Their symbolic nature allows increased interpretability for end-users and developers, which is particularly appealing in RE. Although they have been somewhat overshadowed recently by the use of end-to-end neural network models and contextualized word embeddings, we show that they may be leveraged as input for neural networks to positive effect. We present two methods for integrating broad-coverage semantic structure (specifically, UCCA) into supervised neural RE models, demonstrating benefits over the use of exclusively syntactic integrations. The first method involves reduction of UCCA into a bilexical structure, while the second leverages a novel technique for encoding semantic DAG structures. Our approach is general and can be used for integrating a wide range of graph-based semantic structures.[1]

## 1 Introduction

Early work on RE focused on pattern-based rules for capturing the structure of relation-evoking words and phrases. These rules are applied over text to identify entity relations in much the same way a regular expression would be applied to discover matching text. The pattern machinery spans from simple, regular-expression like, surface patterns (Brin, 1999; Agichtein and Gravano, 2000), through systems that integrate both lexical features and syntactic dependencies into the pattern construct (Mintz et al., 2009). The PredPatt and pyBART frameworks (Zhang et al., 2017a; Tiktinsky et al., 2020) are examples of syntactic dependency based systems that leverage a set of rules defined over Universal Dependencies (UD; Nivre et al., 2016) to extract predicate-argument structures.

In supervised RE, a multi-class classifier is trained to determine whether a relation between entities is evoked by a text. With the increased predominance of end-to-end neural network architectures in NLP practice, it is not surprising that recent work on supervised relation extraction has focused on adapting end-to-end neural systems for the task (Peters et al., 2019; Yamada et al., 2020). However, end-to-end neural models pose interpretability and customization challenges (Conneau et al., 2018; Belinkov et al., 2020), motivating the study of hybrid models, in which the neural architecture is fed explicit structure representations. The Contextualized Graph Convolution Network (C-GCN; Zhang et al., 2018) represents a method for exposing structure representation to the machinery of a deep neural network. C-GCN uses a sentence's UD structure as explicit input to the neural network, resulting in a model whose results are both competitive and interpretable.

In this paper we explore whether we can adopt broad coverage semantic structure to the same effect, and whether we are able to observe improved performance in comparison to the baseline model, based on syntactic structure. We focus on the Universal Conceptual Cognitive Annotation (UCCA; Abend and Rappoport, 2013) framework as a test case, but note that our method can be easily extended to other semantic representations. Our results demonstrate that broad coverage semantic structures, including those that, like UCCA, require representation by directed acyclic graphs (DAGs), can be integrated effectively in neural networks for relation extraction.

[1]Code can be found on GitHub at https://github.com/yyellin/gcn-over-semantic-representations.

## 2 Background

We introduce the representation approaches and datasets we employ in our study.

### 2.1 Universal Dependencies

Since the late 1990s, various NLP motivated dependency grammar representations have been proposed. Carroll et al. (1999); King et al. (2003); De Marneffe and Manning (2008) are examples of such systems, each one accompanied with corpora of accordingly annotated sentences, used for training supervised models for dependency parsing. The UD (Nivre et al., 2016) project, based on De Marneffe and Manning (2008), is a more recent dependency grammar representation, that emphasizes cross-linguistic consistency and has over 300 contributors producing more than 150 treebanks in 90 languages.[2] UD dependency grammar representations come in three forms: *basic*, *enhanced* and *enhanced++* (Schuster and Manning, 2016); in *basic* the sentence is represented as a tree structure, with every word in the sentence assigned a single head word, while *enhanced* and *enhanced++* are DAG representations, in which a word may have multiple heads, and where otherwise implicit relations between content words are captured explicitly. In this paper, we use UD v1 to maintain consistency with the setup used by Zhang et al. (2018).

### 2.2 UCCA

The Universal Conceptual Cognitive Annotation framework (UCCA; Abend and Rappoport, 2013) is a multi-layered system for semantic representation that seeks to capture the semantic, rather than syntactic patterns, expressed through linguistic utterances. The UCCA scheme maps sentences to DAGs that embody these semantic structures. In contrast to graphs formed by dependency grammars, whose nodes all represent lexical entities, a UCCA graph contains both nodes that represent word terminals, which are leaves in the DAG, and non-leaf nodes that represent entities according to some semantic consideration. The foundational layer of UCCA covers the semantics of predicate-argument structure evoked by predicates of all grammatical categories (verbal, nominal, adjectival and others). The layer's primary construct is a *Scene*, which captures a temporally persistent state or an evolving event. A Scene contains one or more Participants, and may also contain secondary scenes, known as Adverbials. Scene, Participant and Adverbial manifest as units in the DAG. In appendix A we provide an example to highlight UCCA's semantic dexterity.

TUPA (Transition-based UCCA Parser) is a transition-based parsing model that can be trained to map sentences to their UCCA scene-based foundational layer (Hershcovich et al., 2017). Pre-trained TUPA models are available online;[3] the BERT based pre-trained model is used extensively in this work.

### 2.3 TACRED

TACRED was designed by Zhang et al. (2017b) to address the dearth of annotated data required for supervised learning for RE. It is based on examples from the corpus used in the yearly TAC Knowledge Base Population (TAC KBP) challenges, conducted from 2009 to 2015 (McNamee and Dang, 2009). In each annual challenge, 100 entities, people, and organizations, were provided to competing systems for them to identify relations between those given entities (referred to as *subjects*), and other *objects* mentioned in the text. The TAC KBP relation extraction task was formulated in terms of slot filling: a person entity is assigned 26 attribute types while an organization is assigned 16 — the challenge posed to competing systems being the extraction of the values of these attributes based on the given corpus (or in other words, the object and relation are given; the challenge is to find the subject). TACRED leverages the results of these challenges to form a set of 106,264 example sentences, each one containing an object and a subject.

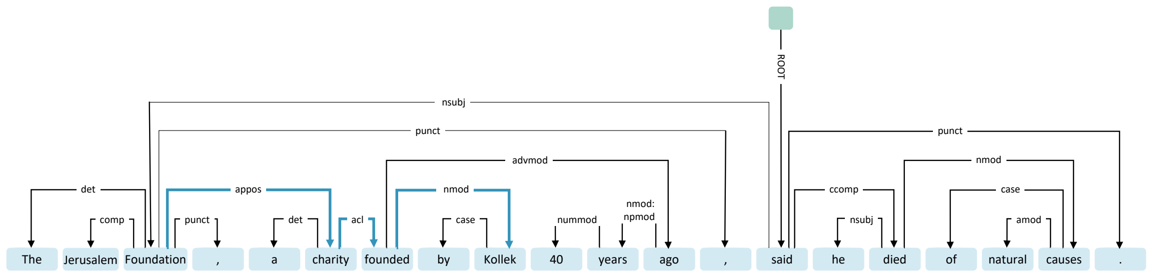
Annotation is implemented through crowdsourcing, with each sentence annotation containing the spans of both object and subject, one of the 42 entity relation attributes, or a *no_relation* classification if no relation exists. In addition to the human created annotations for subject and object spans and relation type, the TACRED dataset contains POS tags, named entities and UD parses by the Standford CORENLP parser (Manning et al., 2014). TACRED's dependency representation annotation corresponds to *basic* UD v1.

## 3 Motivating Experiment: A Rule-based Approach

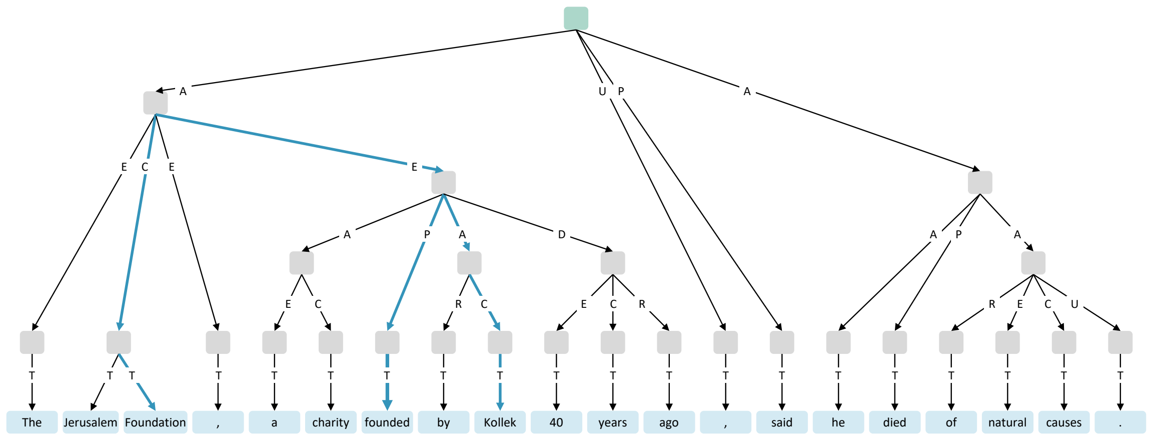
Syntactic sentence representation is used in both rule-based and supervised methods for relation ex-

[2] https://universaldependencies.org

[3] https://github.com/danielhers/tupa



(a) UD Parse Graph



(b) UCCA Parse Graph

Figure 1: UD v1 (basic) and UCCA parse graphs for the sentence 'The Jerusalem Foundation, a charity founded by Kollek 40 years ago, said he died of natural causes.' The diagrams highlight paths from the subject 'Jerusalem Foundation, through the trigger word 'founded' to the object 'Kollek' (sentence 61b3a5c8c9272ce895a6 in the TACRED training dataset). For UD the resulting path pattern is **`<sub>`**↓**`appos`**↓**`acl`** **`<trigger>`**↓**`nmod`** **`<obj>`**, and for UCCA it is **`<sub>`**↑**`C`**↓**`E`**↓**`P`** **`<trigger>`**↑**`P`**↓**`A`**↓**`C`** **`<obj>`**.

traction. Our hypothesis is that semantic sentence representation can provide further benefit for this task. We begin the examination of this conjecture by comparing the relation extraction performance of two simple rule-based methods, one using UD and the other using UCCA, both applied to sentences from the TACRED dataset.

**Method.** Asserting that a relation between two entities is evoked by a text is often contingent on the presence of a "trigger word", or phrase, with semantic application to both entities. We use pattern-based rules that denote a path from the relation subject to the trigger word, and from the trigger word to the relation object. Figure 1 illustrates this by highlighting the path for both UD and UCCA for the sentence "The Jerusalem Foundation, a charity founded by Kollek 40 years ago, said he died of natural causes".

Our procedure comprises three phases: UCCA tagging, pattern extraction for both UD and UCCA, and pattern evaluation.

**UCCA Tagging.** While the TACRED dataset contains the UD representation for each sentence, it does not contain the corresponding UCCA representation. To begin our experiment, we use TUPA with the BERT based pre-trained model, to create UCCA annotations for all sentences in the training set, producing TACRED sentences that have both UD and UCCA representations. We do the same for the sentences in the test set, which we use for evaluation.

**Pattern Extraction.** Pattern extraction requires a manual step in which trigger words for the relation in each sentence are identified. With the trigger words identified, automatic pattern construction follows for both UCCA and UD, per relation type. The caption of Figure 1 presents an example. The end result is a list of paths and trigger words for each relation.

We declare a path match to occur when the target sentence contains a subject and object entity pair, any one of the trigger words in our list for the relation, and a path from the subject entity to the trigger word and from it to the object entity, that matches one of our extracted patterns.

**Evaluation.** We evaluate our system using the *org:founded_by* relation. For precision we count the number of no_relation sentences in the test set incorrectly identified as containing an org:founded_by pair, as a fraction of the total number of no_relation and org:founded_by sentences, to produce a precision value. We ignore sentences containing other relations. For recall, we count the number of correctly identified org:founded_by relations, from the total number in the test set.

**Results.** For org:founded_by, recall is 0.38 for the UD patterns and 0.2 for UCCA. Interestingly, when applying either UD or UCCA patterns, recall improves to 0.47. The precision result is 0.99 for both UD and UCCA. We note that since the UCCA representation system contains non-terminal nodes, we can expect more variation in a UCCA path than in the terminal-only UD representation-based path, contributing to the inferior recall score of UCCA versus UD.

While our experiment considers entity pairs with an org:founded_by relation or no relation at all, and thus represents a binary classification problem, rather than the multi label classification problem posed by the 41 relations of the TACRED ontology, it nonetheless highlights the potential for leveraging trigger words and sentence graph representations (syntactic as in UD v1 *basic* and semantic as in UCCA) for construction of relation matching systems. Indeed, comprehensive trigger word selection per relation is not scalable if done manually, however we could utilize word embedding techniques, so that trigger word matching could be performed using a vector distance threshold, which would capture similar trigger word terms (Batista et al., 2015).

In the next sections, we describe more sophisticated machinery, which realizes a softer notion of matching between the paths of the training data, and those of the test set.

## 4 C-GCN for Semantic Representation

We propose a supervised deep-neural-network model that explicitly utilizes sentence graph representations, so that we may compare the utility of UCCA and UD paths (and their combination). The model receives a sentence and two entity spans (subject and object) as input and gives preference to the word representations corresponding to the syntactic or semantic path between subject and object, for processing by the deep neural network. Contextualized Graph Convolution Networks (Zhang et al., 2018) fulfil this requirement, and we select them as the framework for our study.

### 4.1 The C-GCN Model

At the heart of the graph convolution network method presented by Zhang et al. is the notion that a sentence's bi-lexical dependency structure can be captured as an adjacency matrix and used efficiently to fuse the representation of each token on the path with the representations of its dependency induced neighbors. Zhang et al. experimented with various implementations and hyper-parameters; we reference the C-GCN implementation for which their published results are achieved, in which only words on the path or one dependency edge away from the path are considered ("distance-one-pruning"), and which uses two convolution layers.[4]

In the case of UD, where all edges in the dependency graph representation are between terminals, this adjacency matrix stems from the graph representation directly (this is not the case with UCCA that contains non-terminal nodes; see 4.2). The adjacency matrix captures the dependencies between words that are either on the shortest path from subject to object, or are one dependency edge away from a word on the shortest path ("near-path" tokens); all other dependencies are ignored.

**C-GCN Architecture.** Figure 2 presents the six building blocks of the C-GCN implementation. The nexus of the C-GCN model is in the graph convolution layer (Block 3): it allows the deep neural network apparatus to focus on tokens that are on-path or near-path. Best results are reached by Zhang et al. with two instances of graph convolution, as depicted in the diagram. The first GCN instance receives the $[\texttt{sentence-length} \times 400]$ matrix input from the LSTM and outputs a $[\texttt{sentence-length} \times 200]$ matrix. Subsequent GCN instances all accept as input, and produce as output, $[\texttt{sentence-length} \times 400]$ matrices.

To explain the core GCN mechanism we employ some notation. Let A represent our pruned

[4]This is also the model implemented by Zhang et al. and available on GitHub at https://github.com/qipeng/gcn-over-pruned-trees.

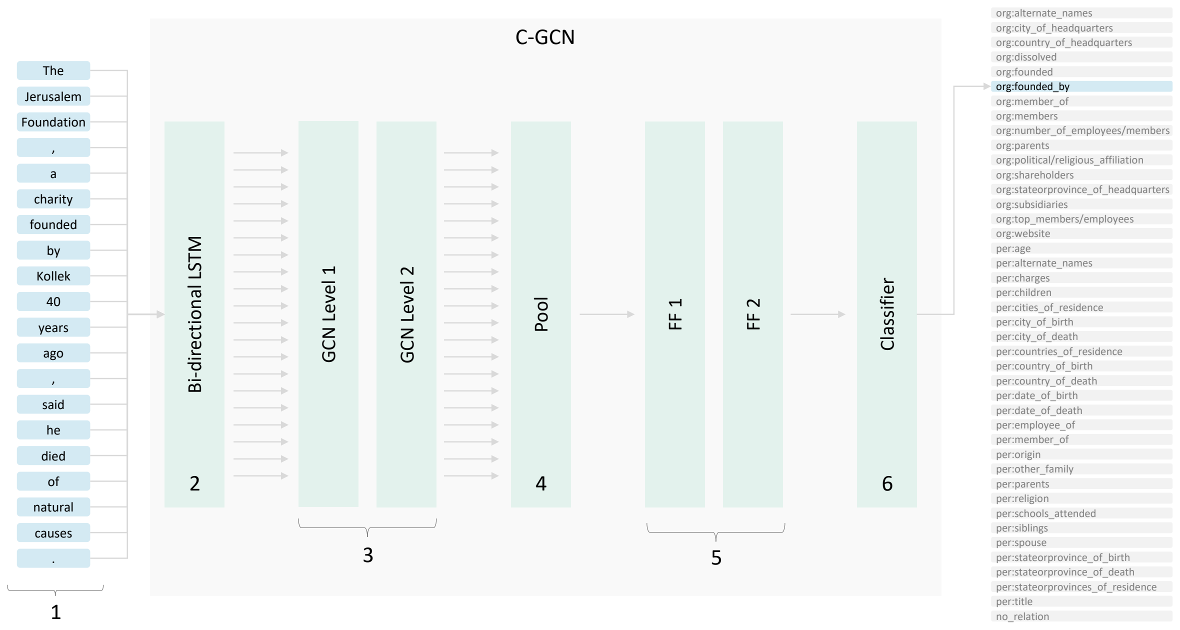


Figure 2: High level overview of the six blocks comprising the contextualized graph neural network, with the sentence from figure 1 as input.

adjacency matrix, where $A_{i,j} = 1$ if both $i$ and $j$ are on the path between subject and object, or one edge away from it, and there exists a dependency between tokens $i$ and $j$ of any kind. The direction of the dependency is ignored by the model, resulting in a symmetric matrix. Each GCN instance is assigned its own weight matrix; the first, of dimensions $[400, 200]$, all others of dimensions $[200, 200]$. Let $\mathrm{W}^{(l)}$ represent the weight matrix corresponding to instance $l$. Let $d_i$ denote the degree of token $i$, where a token's degree corresponds to the number of tokens it is connected to directly, according to the dependency graph representation. Finally, let $\mathrm{h}^{(l)}$ and $\mathrm{h}^{(l+1)}$ represent the input to GCN instance $l$ and its output respectively. Then we have:

$$h_i^{(l+1)} = \sigma\left(\frac{\sum_{j=1}^{n} (\mathrm{A}+\mathrm{I})_{i,j} \mathrm{W}^{(l+1)} h_j^{(l)}}{d_i} + b^{(l+1)}\right) \tag{1}$$

In other words, $\mathrm{h}^{(l+1)}$ is computed by applying a non-linearity function to the a linear combination of both its immediate, unpruned, neighbors in the graph, and itself, normalized by the degree of token $i$. Intuitively, all on, or near-path, tokens are fused with their on or near-path neighbors in the dependency graph, up to a dependency distance that corresponds to the number of GCN instances.

We describe the other network blocks in appendix C.

**GCN vs C-GCN** In their paper, Zhang et al. analyze two variants of the graph convolution network: C-GCN, and GCN. These models are identical except with regard to an LSTM block, which is only used in C-GCN. The 'C' of C-GCN refers to the contextualization achieved by the LSTM apparatus.

Modification of the C-GCN model to support experimentation with other dependency representations involves multiple stages. In the following sections we review these modifications and the resulting models.

### 4.2 Converting UCCA Structures to Bi-Lexical Graph

A goal of this study is to apply UCCA's semantic representations to the task of RE, by leveraging UCCA's parse graph in the graph convolutional network machinery of the C-GCN model. As in our motivating experiment (section 3), the preliminary step is to produce UCCA representations for each of TACRED's 106,264 sentences. We again employ TUPA's BERT-based pre-trained model to this end, and generate parse-graph representations for each sentence. We are confronted however with a challenge: C-GCN expects a *tree* structure, where all nodes correspond to tokens in the sentence and have a single parent; UCCA produces a *graph* structure with nodes that correspond to non-terminal units and that have multiple parents.

| Sentence | Path to minimal sub-tree root | Embedding (first 4 bytes from 10 in total) | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| The | ↑E | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Jerusalem | ↑E | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Foundation | ↑C | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| , | ↑U | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| a | ↑E ↑A ↑E | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| charity | ↑C ↑A ↑E | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| founded | ↑P ↑E | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| by | ↑R ↑A ↑E | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Kollek | ↑C ↑A ↑E | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 40 | ↑E ↑D ↑E | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| years | ↑C ↑D ↑E | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ago | ↑R ↑D ↑E | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 3: Representation of the first 4 bytes in the initial UCCA embedding (prior to tuning that occurs during training), for the sentence ”The Jerusalem Foundation, a charity founded by Kollek 40 years ago, said he died of natural causes”. The first 12 terminals in this sentence are contained within the DAG comprised of the lowest common ancestor of both the *subject* (The Jerusalem Foundation) and *object* (Kollek), and the first 20 dimensions of the UCCA embeddings for these are depicted in this illustration. The embedding for all other terminals will be the a $\vec{0}$ vector.

The first step in bridging this gap involves conversion of the UCCA graph into a bilexical structure, where all nodes correspond to a token in the sentence. We employ the method described by Hershcovich et al. (2017), which heuristically selects a head terminal for each non-terminal node and attaches all terminal descendants to the head terminal.[5] Using this conversion procedure, we produce a UCCA-based bi-lexical graph representation for each sentence in the TACRED dataset. Appendix B provides an example of a bi-lexical reduction.

In the bi-lexical representation result, a terminal token may have multiple parents, deeming the structure a non-tree DAG. Indeed, the notion of multiple parents may be viewed as a core feature of semantic representations, reflecting the fact that a word in a sentence or text may have multiple semantic roles (Oepen et al., 2020). We thus modify the C-GCN model implementation, so that it can produce the adjacency matrix using a bi-lexical DAG rather than tree.

### 4.3 Extending C-GCN for UCCA DAG Representation

Conversion of a UCCA sentence representation into a bi-lexical graph clearly loses extensive semantic information captured in the original structure. We pursue an embedding-based mechanism that will allow us to exploit the full UCCA structure representation in our model.

Our novel approach seeks to utilize a lossless representation of the UCCA DAG structure as training input for an **additional embedding vector**. To this end, we map each token in a sentence to its path to the root of the sentence-representing UCCA DAG. With a set of fourteen different edge labels in UCCA's foundational level, we can encode each label with four 0/1 bits. Given a maximum distance of 18 steps from terminal to root for all sentences in the TACRED dataset, we can encode all paths with up to 72 zero/one bits in total. In a new pre-training step, we collect all distinct paths from terminal to root as derived from TACRED training dataset (for a total of 22,922 distinct paths), assigning each one to an initial 80-dimensional vector, where the value of dimension $i$ corresponds to bit $i$ in the corresponding path (we assign 0 to all beyond-path dimensions).[6] We add a $\vec{0}$ vector for representing out-of-vocabulary paths (necessary for evaluation). Finally, we extend the 360-dimensional representation of each token to 440 dimensions, by concatenating the 80-dimensional representation of the

[5]Details of the algorithm in Hershcovich et al. supplementary material `https://www.aclweb.org/anthology/attachments/P17-1104.Notes.pdf`.

[6]This preparatory procedure is somewhat similar to the pre-training step required for word embeddings, in which we collect the set of word embeddings that our training data vocabulary corresponds with for training initialization of word embeddings.

token's path to root. Like word embeddings, the UCCA path-to-root representation embeddings are tuned during training.

While we can assign a UCCA-embedding for each token in the sentence, we would like to focus our network's attention on tokens that relate closely to *subject* and *object*. We achieve this with the help of a ***minimal sub-DAG***, which we define as the DAG comprised of the lowest common ancestor of both *subject* and *object*, and all their descendants. Where there is more than a single lowest common ancestor, we choose the one that induces the smallest number of terminals. All tokens that are outside the *minimal sub-DAG* are assigned the $\vec{0}$ embedding. In figure 3 we provide an example of the embedding vector values before they are fine-tuned during training.

## 5 Experiments

We perform twenty independent cycles of training-and-test for all our experiments, computing an average of the recall, precision and $F_1$ results across these runs. This technique differs from the common evaluation protocol of selecting the model with the median dev $F_1$ from five independent runs and reporting its $F_1$ result; we frame our approach in terms of statistical significance in Appendix F. We use TACRED's standard train/dev/test splits. Results are reported using the standard $F_1$ score. We use a fork of the source code made available by Zhang et al.[7] In general we maintain the default parameters of Zhang et al., including the default pruning factor of one, by which tokens that are up to one edge away from the path are also considered by the convolution and pruning mechanisms.

As noted, C-GCN is trained on the TACRED dataset.[8] For each of the 106,264 sentences, the dataset contains attributes including the relation type, subject and object span and UD dependency information. The process of model extension begins with the enrichment of the TACRED dataset with representation for additional semantic structures. As additional representation needs to be expressed in terms of the attributes of **tokens** for the enrichment to be valid, alignment on **tokenization** must be reached. In appendix D we describe how this alignment is achieved and reproduce Zhang et al.'s results using our updated tokenization.

[7]https://github.com/yuhaozhang/tacred-relation.

[8]Available for download from https://catalog.ldc.upenn.edu/LDC2018T24

| **System** | **Adjacency Combination** | | | **$F_1$** |
|---|---|---|---|---|
| | **cons** | **UD** | **UCCA** | |
| C-GCN$_{ud}$ | ✗ | ✓ | ✗ | 66.27 |
| C-GCN$_{ucca}$ | ✗ | ✗ | ✓ | 66.44 |
| C-GCN$_{ud \cup ucca}$ | ✗ | ✓ | ✓ | 66.67 |
| C-GCN$_{all}$ | ✓ | ✓ | ✓ | <u>66.94</u> |

Table 1: Results for different adjacency matrix combinations. C-GCN$_{ud}$ is the baseline model described in Zhang et al. (2018). C-GCN$_{ucca}$ replaces the UD representation with bi-lexical UCCA representation. Best performance is achieved when supplementing the combined UD and UCCA path representations with all tokens between subject and object (C-GCN$_{all}$).

| **System** | **$F_1$** |
|---|---|
| C-GCN$_{ud+emb}$ | 66.66 |
| C-GCN$_{ucca+emb}$ | 66.60 |
| C-GCN$_{all+emb}$ | <u>67.32</u> |
| C-GCN$_{all+emb}$ ensemble | **<u>68.41</u>** |

Table 2: The models listed in table 1 are further enriched with UCCA DAG representation, and their resulting scores are listed here. We demonstrate an increase of one $F_1$ point for C-GCN$_{all+emb}$ over baseline model C-GCN$_{ud}$.

The graph-convolution network uses each sentence's UD dependency tree to determine the set of tokens (and their intra-connections) that it should focus on in the convolution and pooling phases of the network operation. As we presented in section 4.1, this is achieved by representing the UD dependency tree as adjacency matrix $A_{ud}$. The bi-lexical UCCA graph projection we described in section 4.2 is represented by $A_{ucca}$.

In addition to UD and UCCA structures, we employ a synthetic bi-lexical tree, where, for an $n$-token sentence, the only dependencies are between *token$_i$* and *token$_{i+1}$* for all $i < n$. This artificial dependency tree will cause the convolution network to focus on all tokens that appear between *subject* and *object* in the surface structure of the sentence (including *subject* and *object* themselves), and, in consideration of path pruning, two additional tokens adjacent to either ends of the span. We denote the adjacency matrix for this artificial structure by $A_{cons}$ ("cons" for consecutive).

We establish a performance baseline by testing different combinations of these adjacency matrices: $A_{ud}$ alone, $A_{ucca}$ alone, $[A_{ud} \cup A_{ucca}]$, and finally $[A_{ud} \cup A_{ucca} \cup A_{cons}]$. Our results are presented in table 1.

The UCCA-based adjacency matrix slightly outperforms the UD-based one, providing experiment-based evidence that tokens on the path between *subject* and *object* on the UCCA bi-lexical tree reduction, provide more linguistic cues for the RE task than those on the equivalent UD path. The superiority of the UCCA model is despite some loss of semantic content that occurs when transforming a full UCCA DAG to a bi-lexical one. Interestingly, the addition of $A_{cons}$ further improves the result.

Armed with baseline performance, we employ our novel UCCA terminal-path-to-root embedding method described in section 4.3. We first apply UCCA embeddings to the C-GCN$_{ud}$ model; we observe an $F_1$ score improvement of close to 0.4 points. We observe a similar improvement when we apply UCCA embeddings to C-GCN$_{all}$: the $F_1$ score rises from 66.94 to 67.32.

Our headline result is an improvement of 1.05 $F_1$ points between the C-GCN$_{ud}$ system described by Zhang et al. with a mean score of 66.27, and our C-GCN$_{all+emb}$ system, with a mean score of 67.32. We confirm the statistical significance of these results by conducting Welch's t-tests and Mann-Whitney U-tests. For further discussion, see Appendix F.

We follow the practice of performing ensemble testing (Zhang et al., 2017b; Zhou et al., 2020), by applying a soft-max function on the sum of the classifier's logits from five different C-GCN$_{all+emb}$ models. We repeat the ensemble experiment for all combinations of five models, from the set of C-GCN$_{all+emb}$ models we produced in separate runs of the experiment. We average the $F_1$ scores for a final score of 68.41. Table 2 summarizes these results.

Finally, we perform a relation-based comparison of C-GCN$_{ucca}$ and C-GCN$_{ud}$, to determine whether the performance improvements are a product of improved results in specific relation categories. Table 3 lists the results for all the relations for which a difference of over 5% was demonstrated. We discuss these results in the next section.

## 6 Discussion

Our goal is to measure how UCCA performs in comparison with UD based C-GCN model. The results of our baseline experiment provide evidence that the graph convolution network produces a slightly stronger model when the UD sentence representation is substituted for a UCCA bi-lexical representation, and a significantly stronger model when UD and UCCA are used in tandem.

| Relation | Avg. Improvement |
|---|---|
| per:country_of_birth | 71.7% |
| per:city_of_birth | 29.4% |
| per:city_of_death | 10.5% |
| per:date_of_death | 9.4% |
| org:country_of_headquarters | 8.3% |
| per:stateorprovinces_of_residence | 7.2% |
| per:stateorprovince_of_death | 6.4% |
| per:schools_attended | -6.2% |
| org:parents | -8.3% |
| org:subsidiaries | -8.9% |
| org:shareholders | -9.4% |
| per:children | -9.6% |
| org:political/religious_affiliation | -23.0% |
| per:other_family | -55.4% |

Table 3: Relations with average improvement/decline of above 5% for C-GCN$_{ucca}$ in comparison to C-GCN$_{ud}$.

In our main line of experimentation, we attempted to utilize the entire UCCA representation, rather than its bi-lexical reduction. We tested a system that uses embedding representation of the path of each token, within the minimal sub-DAG that contains both *subject* and *object*, to the root of that sub-DAG. We found that integration of this additional input produces superior results for all the models we tested: C-GCN$_{ud}$, C-GCN$_{ucca}$ and C-GCN$_{all}$; interestingly, the improvement is the smallest for C-GCN$_{ucca}$ itself. We postulate that the results demonstrate the efficacy of this novel approach for embedding semantic representation and believe that further analysis of this method in the context of other systems for semantic representation could be helpful to the NLP community.

As detailed by Zhang et al., the C-GCN pooling mechanism allows for identification of which tokens contribute significantly to the pooling results (block 4 of the network). Thus, it is possible to directly analyze sentences classified correctly by, for example, C-GCN$_{all+emb}$, but incorrectly by C-GCN$_{ud}$, to ascertain which tokens contribute significantly to the pooling stage output in the former but not the latter. We leave this analysis for future work as well.

To complete our study, we compared UCCA and UD performance, relation by relation. All the relations in which UCCA exhibits improved results are location-oriented. Contrarily, UD does considerably better than UCCA for family relations. Why should UCCA do better for location relations at the expense of its performance for family relations? We note that location relations associate a subject, person, or organization, to the object, a location, via an explicit event such as birth, death or residence. This is different from a family relation, for which there is detachment between the event that evoked the relation, and the reality of the relation at the present time. Indeed, for family relations the relation-evoking event may vary considerably for the same relation (consider adoption versus birth). We hypothesize that UCCA, with its improved semantic awareness, is more adept at sensing an event that binds object to subject, as opposed to UD, which performs better for family relations that may emerge more readily from syntax. This finding could lead to improved classification performance by constructing an ensemble of UCCA and UD models, and weighting the softmax function on the sum of the classifiers to favor UCCA results for location relations and UD results for family ones. This experimentation is also left for future work.

## 7 Conclusion

Much recent work on RE focuses on improving the contextualized representations of words and entities, including by means of injecting knowledge into large pretrained models such as BERT, to be used for a final step of supervised training using comparatively small training datasets like TACRED. This approach is successful in overcoming the dearth of annotated data required for supervised learning for RE and is producing ever-improving bottom-line results. However, it further intensifies the interpretability challenges posed by end to end models and deepens the chasm between the linguistic domain and the practice of NLP.

Our approach also leverages pretrained models that are dependent on extensive prior training data (via the TUPA parser); however, the intention is linguistically explicit: generation of UCCA representations for the sentences in the TACRED dataset. The C-GCN architecture then provides visibility into which words actually contribute to a relation classification, providing a level of transparency devoid in end-to-end models.

Our results indicate that representation of explicit semantic structure is indeed beneficial for the RE task, providing linguistically-explicit means for leveraging prior training to the task of relation extraction.

Because our approach to embedding semantic structure is not specific to UCCA, future work could analyze the effects of integrating other semantic representations, as well as the possibility of embedding semantic structure representation in the latest cutting-edge RE models.

## Acknowledgments

This work was supported by a grant from the Allen Institute for Artificial Intelligence. We thank Yoav Goldberg for his advice and support. We thank Or Honovich and Shachar Don-Yehiya for their contribution to the rule-based pattern matching work that inspired this study.

## References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–238.

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the fifth ACM conference on Digital libraries - DL '00*. ACM Press.

David S. Batista, Bruno Martins, and Mário J. Silva. 2015. Semi-supervised bootstrapping of relationship extractors with distributional semantics. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yonatan Belinkov, Sebastian Gehrmann, and Ellie Pavlick. 2020. Interpretability and analysis in neural NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. Association for Computational Linguistics.

Sergey Brin. 1999. Extracting patterns and relations from the world wide web. In *Lecture Notes in Computer Science*, pages 172–183. Springer Berlin Heidelberg.

John Carroll, Guido Minnen, and Ted Briscoe. 1999. Corpus annotation for parser evaluation. *arXiv preprint cs/9907013*.

Alexis Conneau, Germán Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Marie-Catherine De Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: proceedings of the work-*

*shop on cross-framework and cross-domain parser evaluation*, pages 1–8.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for UCCA. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Tracy Holloway King, Richard Crouch, Stefan Riezler, Mary Dalrymple, and Ronald M Kaplan. 2003. The parc 700 dependency bank. In *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora (LINC-03) at EACL 2003*.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics.

Paul McNamee and Hoa Trang Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Text Analysis Conference (TAC)*, volume 17, pages 111–113.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - ACL-IJCNLP '09*. Association for Computational Linguistics.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666.

Stephan Oepen, Omri Abend, Lasha Abzianidze, Johan Bos, Jan Hajic, Daniel Hershcovich, Bin Li, Tim O'Gorman, Nianwen Xue, and Daniel Zeman. 2020. MRP 2020: The second shared task on cross-framework and cross-lingual meaning representation parsing. In *Proceedings of the CoNLL 2020 Shared Task: Cross-Framework Meaning Representation Parsing*. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2018. Why comparing single performance scores does not allow to draw conclusions about machine learning approaches. *arXiv preprint arXiv:1803.09578*.

Sebastian Schuster and Christopher D Manning. 2016. Enhanced english universal dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 2371–2378.

Aryeh Tiktinsky, Yoav Goldberg, and Reut Tsarfaty. 2020. pyBART: Evidence-based syntactic transformations for IE. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics.

Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. 2020. LUKE: Deep contextualized entity representations with entity-aware self-attention. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Sheng Zhang, Rachel Rudinger, and Benjamin Van Durme. 2017a. An evaluation of predpatt and open ie via stage 1 semantic role labeling. In *IWCS 2017—12th International Conference on Computational Semantics—Short papers*.

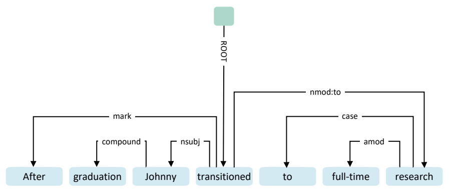
Yuhao Zhang, Peng Qi, and Christopher D. Manning. 2018. Graph convolution over pruned dependency trees improves relation extraction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yuhao Zhang, Victor Zhong, Danqi Chen, Gabor Angeli, and Christopher D. Manning. 2017b. Position-aware attention and supervised data improve slot filling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

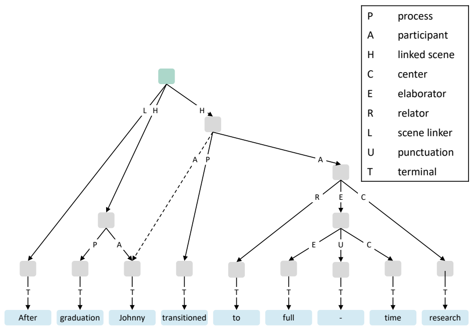
Li Zhou, Tingyu Wang, Hong Qu, Li Huang, and Yuguo Liu. 2020. A weighted gcn with logical adjacency matrix for relation extraction. In *European Conference on Artificial Intelligence 2020 (ECAI2020)*.

## A Example of UCCA Advantage

Figure 4 provides an example of UCCA's advantage over UD in capturing scenes and their participants by representing non-verbal predicates ("graduation") like verbal ones ("transition").



(a) UD Parse Graph



(b) UCCA Parse Graph

Figure 4: UD v1 (enhanced++) and UCCA representation graphs for the sentence 'After graduation Johnny transitioned to full-time research'. The UCCA parse directly captures Johnny's participation in both the occurrences described in the sentence, graduation, and transition. This semantic connection does not arise from the UD representation (not even the enhanced++ flavor).

## B Example of Bi-Lexical Reduction for UCCA

Figure 5 provides a graphic illustration for the results of bi-lexical reduction of UCCA representation, for the sentence depicted in figure 4. The product of this bi-lexical reduction is a representation containing terminal nodes only.

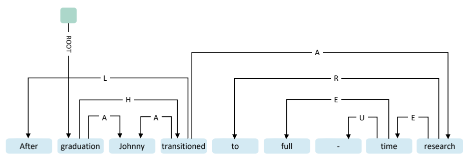


Figure 5: Bilexical UCCA representation of the sentence from 1.

As this example illustrates, the bi-lexical reduction constitutes a DAG; in this example "Johnny" is the child of both the "graduation" and the "transitioned" terminals.

## C C-GCN Architecture

We describe the six blocks of the C-GCN model as depicted in figure 2.

**Block 1 – Word Embeddings:** Zhang et al. use 300-dimensional GloVe vectors to initialize word embeddings corresponding to each token. Each token's initial corresponding GloVe based vectors is further extended with 30-dimensional embedding for the token's part-of-speech, and another 30-dimensional embedding for its entity type. Part-of-speech and entity-type are both provided as input in the TACRED dataset (POS and NER attributes respectively) and are initialized in training with random values for each of the unique classes in the part-of-speech of entity-type sets. All the embeddings undergo fine-tuning during training and are therefore variables of the model.

**Block 2 – LSTM:** The input vector embeddings do not contain information about word order or contextual cues required for disambiguation. A bi-directional long short-term memory network addresses this, by taking the list of 360-dimensional vectors corresponding to the sentence's tokens and producing a corresponding list of 400-dimensional vectors.

**Block 3 – Graph Convolution:** We cover the third block in section 4.1

**Block 4 – Pool:** The fourth block is responsible for converting the two-dimensional sentence representation into a single-dimensional vector. We resort to formalism to describe precisely how this is achieved. Let $\mathcal{H}_p = [h_{p_1}, \ldots, h_{p_n}]$ denote the output representations of the final GCN layer for all on-or-near-path tokens, and let $\mathcal{H}_s = [h_{s_1}, \ldots, h_{s_n}]$ and $\mathcal{H}_o = [h_{o_1}, \ldots, h_{o_n}]$ denote the vector lists corresponding to the *subject* and *object* spans respectably. A simple max pooling function is applied to $\mathcal{H}_p$, $\mathcal{H}_s$ and $\mathcal{H}_o$, resulting in three single vectors, $h_{p_{max}}$, $h_{s_{max}}$ and $h_{o_{max}}$, representing the on-or-near-path tokens, the *subject* and the *object* respectably. The output of block four is a simple concatenation of these vectors resulting in a single 600-dimensional vector.

**Block 5 – Feedforward Network** Block five is a simple stacked feed-forward network, with each FF layer comprising a single linear transformation followed by a RELU non-linearity. Zhang et al. use two layers as depicted in the diagram. The first layer receives a 600-dimensional vector and produces a 200-dimensional vector as output; all other layers receive and produce 200-dimensional vectors.

**Block 6 – Classifier** The sixth and final block of the network is a simple feed-forward linearity, which receives the 200-dimensional vector from block five, and outputs a 42-dimensional vector, one dimension per relation category. This final vector is used in both training with a cross-entropy-loss function, and in evaluation, where a soft-max function is applied to produce a probability vector. It is possible to control the recall/precision balance by applying a thresholding mechanism such that a positive classification (i.e. any label other than *no_relation*) must surpass a threshold greater than 0.5, however the model chooses to treat a negative classification symmetrically.

## D Token Alignment

TUPA, the UCCA parser we briefly mentioned in the introduction, uses the the SpaCy NLP pipeline for basic NLP tasks including tokenization. SpaCy's default tokenization results vary considerably from TACRED's given tokenization. Indeed, in 30% of train sentences, 28% of dev sentences, and 25% of test sentences tokenization is different. For example, differences arise in the case of intra-word-hyphens[9]. Additionally, the TACRED dataset contains some obvious tokenization errors; for example, there are over 130 entries in which two sentences have been merged into one, by erroneously fusing the last token of the first sentence, it's period punctuation mark, and the first one of the next sentence into a single token.

To address these tokenization concerns, we re-parse the entire TACRED dataset with the Standford CORENLP parser, configuring it to adhere to the given tokenization produced by the TUPA parser[10]. This results in a tokenization-aligned standford_pos, stanford_ner, stanford_head and stanford_deprel attribute set (see table 4). For subj_start, subj_end, obj_start and obj_end, which are human annotated properties, we employ a simple algorithm to reassign their index values to correspond to TUPA's tokenization. The end result is a transformed TACRED input dataset where all sentence attributes that require adherence to the sentence's token breakdown conform with TUPA/SpaCy tokenization.

As a baseline, we reproduce the experimental setup of Zhang et al. (2018), using the given sentence attributes as they appear in the TACRED dataset. We compare these results with results we attain when using TACRED sentences **re-annotated** by the CORENLP engine, using SpaCy/TUPA sentence tokenization, as described above. Table 5 captures this comparison. We hy-

| Attribute | Description |
|---|---|
| **id** | UUID for the sentence |
| **relation** | The relation that exists between subject and object (or no_relation) if no relation exists |
| **token** | Array of the sentence's tokens, the token at each array position corresponding to the token at the same |
| **subj_start** | start zero-based index of the subject token span |
| **subj_end** | end zero-based index of the subject token span |
| **obj_start** | start zero-based index of the object token span |
| **obj_end** | end zero-based index of the object token span |
| **subj_type** | Entity type of subject (either organization or person) |
| **obj_type** | Entity type of object (not limited to organization and person; 18 different classes of object type, including, for example, location, duration and ideology) |
| **stanford_pos** | Array of the sentence tokens' parts-of-speech, the part-of-speech at each array position corresponding to the token at the same |
| **stanford_ner** | Array of the sentence tokens' entity types, the entity-type at each array position corresponding to the token at the same (with 'O' used to indicate a non-entity token) |
| **stanford_head** | Array of the sentence tokens' head word one-based indices, the index at each array position corresponding to the head of the token at the same |
| **stanford_deprel** | Array of the sentence tokens' incoming dependency type, the index at each array position corresponding to the incoming dependency type of the token at the same |

Table 4: Main attributes associated with each sentence in the TACRED dataset

[9]Discussion on this tokenization divergence on Stack Overflow: https://stackoverflow.com/questions/52293874/why-does-spacy-not-preserve-intra-word-hyphens-during-tokenization-like-stanford

[10]We achieve this by reconstructing a sentence where all TUPA generated tokens are separated by whitespace, and then setting CORE NLP's 'tokenize.whitespace' flag to True

| System | Note | $F_1$ |
|---|---|---|
| GCN | reported result | 64.0 |
| C-GCN | reported result | 66.4 |
| C-GCN | our result with given dataset | 66.68 |
| C-GCN | our results with TACRED input re-annotated with CORE NLP engine, using SpaCy tokenization | 66.27 |

Table 5: Our results for the unmodified C-GCN model, using given and re-annotated TACRED datasets.

pothesize that the decrease in score when using re-annotated data may stem from errors introduced by our procedure for reassignment of index values for the human annotated subj_start, subj_end, obj_start and obj_end properties.

## E Variations of DAG encoding

We test two variations of our UCCA terminal-path-to-root embedding method: the first adds non-zero embedding representations for the token-to-root path of *all* tokens in the sentence; the second uses non-zero embeddings for tokens in the *minimal sub-DAG* only, using zero vector embedding for all other tokens. We also perform an ablation test by using random values for each unique path embedding, rather than values that encode the actual paths.

| System | Path Encoding | Min Sub-DAG | $F_1$ |
|---|---|---|---|
| C-GCN$_{ucca}$† | - | - | 66.44 |
| C-GCN$_{ucca+emb}$ | ✗ | ✓ | 65.63 |
| C-GCN$_{ucca+emb}$ | ✓ | ✗ | 66.22 |
| C-GCN$_{ucca+emb}$ | ✓ | ✓ | 66.60 |

Table 6: Measuring impact of UCCA embeddings under different settings. † marks the baseline C-GCN$_{ucca}$ model score as reported in table 1. An ✗ in the **Path Encoding** column indicates the use of random values.

The results in table 6 indicate that using path encodings for tokens in the minimal sub-DAG produce improved results when compared to the plain C-GCN$_{ucca}$ model. While the improvement is marginal – 0.2 $F_1$ points – we do see a more marked impact, to the negative, when we use random initialization. When we consider all sentence tokens and not just those in the minimal sub-DAG evoked by *subject* and *object* we also see a negative impact, albeit more mild.

## F Statistical Significance

We call attention to Reimers and Gurevych (2018), who critique the common evaluation practice of selecting the model with the median dev $F_1$ from five independent runs and reporting its test $F_1$ result. As noted, the $F_1$ scores reported in our experiments are the mean scores on the test set for 20 multiple models, trained according to the parameters of the system at hand (e.g. GCN$_{ud}$, C-GCN$_{ucca}$), using the full TACRED train and dev datasets, and tested with the test dataset.

Our headline result is an improvement of 1.05 $F_1$ points between the C-GCN$_{ud}$ system described by Zhang et al., with a mean score of 66.27, and our C-GCN$_{all+emb}$ system, with a mean score of 67.32. We conduct two sets of significance tests to validate this result:

**Welch's t-test:** Welch's t-test assumes that the compared distributions are approximately normally distributed. We apply Welch's t-test to a pair of 20 sample pairs, each pair using the same random seed, with the first set corresponding to C-GCN$_{ud}$, and the second to C-GCN$_{all+emb}$, with a null hypothesis asserting that both systems produce models with similar $F_1$ scores on the test set. The resulting $p$-value is 3.015e-09. We subtract 0.75 $F_1$ from all the samples corresponding to C-GCN$_{all+emb}$ and reapply Welch's t-test. The $p$-value for this new null hypothesis is 0.019. In other words, we have a statistically significant improvement of 0.75 $F_1$ points with $p < 0.02$.

**Mann-Whitney U-test:** We abandon the assumption that our systems produce models with normally distributed $F_1$ on the test set and apply the Mann-Whitney U test after once again subtracting 0.75 $F_1$ from the C-GCN$_{all+emb}$ samples. The $p$-value for this new null hypothesis is 0.027. Restated, we have demonstrated a statistically significant improvement of 0.75 $F_1$ points with $p < 0.03$.