DERE: A Task and Domain-Independent Slot Filling Framework for Declarative Relation Extraction

Heike Adel, Laura Bostan, Sean Papay, Sebastian Padó, and **Roman Klinger***
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart, Germany
{firstname.lastname}@ims.uni-stuttgart.de
dereproject@ims.uni-stuttgart.de

Abstract

Most machine learning systems for natural language processing are tailored to specific tasks. As a result, comparability of models across tasks is missing and their applicability to new tasks is limited. This affects end users without machine learning experience as well as model developers. To address these limitations, we present DERE, a novel framework for declarative specification and compilation of template-based information extraction. It uses a generic specification language for the task and for data annotations in terms of spans and frames. This formalism enables the representation of a large variety of natural language processing challenges. The backend can be instantiated by different models, following different paradigms. The clear separation of frame specification and model backend will ease the implementation of new models and the evaluation of different models across different tasks. Furthermore, it simplifies transfer learning, joint learning across tasks and/or domains as well as the assessment of model generalizability. DERE is available as open-source software.

1 Introduction

A large number of tasks in natural language processing (NLP) are information extraction (IE) tasks, such as n-ary relation extraction (Doddington et al., 2004; Mintz et al., 2009; Hendrickx et al., 2010), semantic role labeling (Das et al., 2014) and event extraction (Kim et al., 2009; Doddington et al., 2004). Researchers address these tasks with a variety of different model paradigms, such as support vector machines (Rink and Harabagiu, 2010), convolutional neural networks (Collobert et al., 2011; Zeng et al., 2014) and recurrent neural networks (Tang et al., 2015; Nguyen et al., 2016).

This landscape of different tasks and models gives rise to four challenges: (C1) *Lack of generalizability*: Most models are tailored to a specific task or setup, making it hard to transfer lessons learned between tasks; (C2) *Lack of comparability*: Although benchmark datasets are available for most tasks, end-to-end evaluation typically includes peripheral aspects, such as preprocessing components – thus, it is unclear to what extent reported improvements mark actual advances in the core models or model components; (C3) *Difficulty of reusability*: Given task-specific models inside complex systems, it is hard to reuse specific code or models; (C4) *Difficulty of usage*: Users typically have limited areas of expertise, but IE systems span a range of such areas. Thus, developers of IE tools may have trouble properly (re)training complex machine learning models, and end users without ML or CS background might even be unable to use existing tools.

To tackle these challenges, we develop the general framework DERE (Declarative Relation Extraction). It enables users to (i) specify (novel or established) IE tasks, (ii) compile models and transfer them across tasks without additional development effort, (iii) develop and evaluate models across tasks, (iv) formulate and address research questions, such as the investigation of model generalizability across tasks, transfer learning, or joint learning across tasks and/or domains, and (v) verify the generalizability of models by applying them to a large variety of tasks.

DERE achieves this by providing (a) a general mechanism to declaratively specify IE tasks and (b) a shared processing framework that decouples frontend and backend. This provides an attractive shared basis for modeling tasks which are typically perceived as being very different. In this paper, we use *BioNLP event extraction* and *aspect-based sentiment analysis (ABSA)* as examples. At the same time, the decoupling exposes accessible interfaces for different user groups (*cf.* Section 3).

*All authors contributed equally.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (System Demonstrations), pages 42–47
Brussels, Belgium, October 31–November 4, 2018.

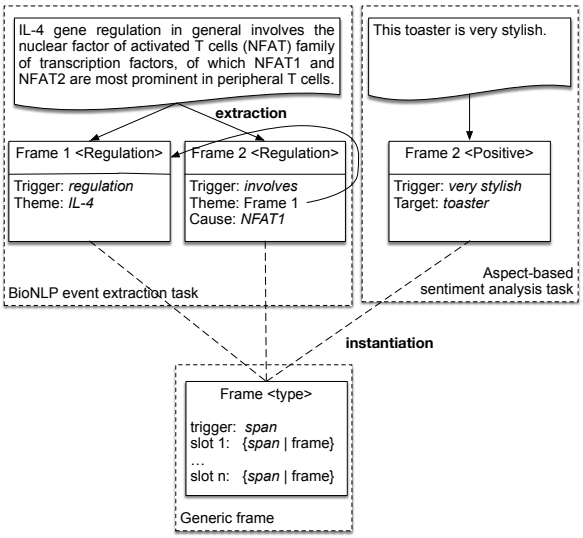

Figure 1: Example formalizations of two different tasks in terms of frames, slots, and spans.

The declarative specification of a task (which we call a *schema*) builds upon *spans* and *relations between spans* as basic concepts which are used by essentially all IE tasks. To model n-ary relations, we propose a *slot-filling* scheme in which *frames* model n-ary relations and their arguments. Figure 1 shows the general structure of frames (below) and two concrete instantiations for ABSA and BioNLP (above). Each frame is triggered (anchored) by a span, *e.g.,* a subjective evaluating phrase like "very stylish" or a BioNLP event trigger, such as "regulation" or "involves".

Frames hold a task-specific number of typed slots, filled by relation arguments. The frames for ABSA have a slot filled by the target (aspect) of the sentiment while the frames for the BioNLP regulation event hold a Theme slot and an optional Cause slot. While triggers are always textual spans, slots can be filled by either spans or frames, depending on the task specification. We argue that this simple setup can model most IE tasks. Note that the framework poses no theoretical restrictions to the window from which frames are extracted. Thus, it can model sentence-level, document-level as well as multi-document tasks.

2 Related Work

Several applications require the joint extraction of spans and relations between spans, such as the BioNLP shared task (Kim et al., 2009), semantic role labeling (Das et al., 2014) or (temporal) slot filling (Surdeanu, 2013). However, all systems we are aware of for solving these tasks are tailored to specific scenarios (Angeli et al., 2016; Adel et al., 2016, *i.a.*). As a result, it is not straightforward to apply them to other use cases. In contrast, our framework is designed to be task- and domain-independent.

Clarke et al. (2012) develop an NLP component manager which combines several existing NLP tools in a pipeline. Similarly, Curran (2003) aims at a general NLP infrastructure but only reports implementations of non-relational sequence-tagging tasks. Examples of the few available toolkits which are intended to provide users with the possibility of automatically extracting information from text data are Jet (Java Extraction Toolkit), GATE (General Architecture for Text Engineering, Cunningham et al., 2013), UIMA (Unstructured Information Management Architecture, Ferrucci and Lally, 2004), FACTORIE (McCallum et al., 2009) and Stanbol which integrates other NLP frameworks, *e.g.,* OpenNLP (Morton et al., 2005).

Stanbol and OpenNLP, however, focus on tagging tasks and do not provide tools for relation extraction. FACTORIE is a general approach to formulate factor graphs for arbitrary tasks. Our framework takes arbitrary model paradigms as a backend and is focused on IE, which enables the abstraction layers introduced earlier. Jet, on the other hand, is an IE engine developed specifically for the ACE task specification.

GATE is most similar to our framework in scope. It offers both a framework for programmers and an environment for language engineers and computational linguists. However, it is a very general framework and working with it requires both domain and machine learning knowledge. In contrast, our framework provides end users with an interface for training models on new tasks without requiring any specific knowledge.

3 Framework Design

Use Cases. We address the needs of the following three user groups with associated use cases: (1) Researchers/Model developers: Our framework helps researchers to formulate their models in a task-independent manner, such that they can be tested and compared across tasks. This addresses challenges C1, C2 and C3 mentioned in Section 1. (2) Developers of IE tools for a new use case: Our framework provides a common interface to models previously developed for other tasks. Those mod-

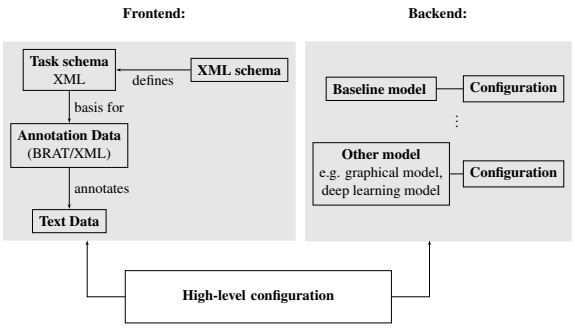

Figure 2: Structure of the DERE framework.

```
<deREschema name="BioNLP-ST 2009" ver="0.01" auth="Klinger">
  <spantypes>
    <span name="Protein" predict="False"/>
    <span name="Gene_expression" anchors="Gene_expression"
          predict="True"/>
    <span name="Binding" anchors="Binding" predict="True"/>
  </spantypes>
  <frames>
    <frame name="Gene_expression">
      <slot name="Theme" types="Protein" cardinality="1"/>
    </frame>
    <frame name="Binding">
      <slot name="Theme" types="Protein" mincardinality="0"/>
    </frame>
  </frames>
</deREschema>
```

Figure 3: A small but complete task schema for part of the BioNLP shared task. Three span types are specified: `Protein`, `Gene_expression`, and `Binding`. The latter two anchor frames of the same name. Both frames possess a single slot `Theme` which can be filled by `Protein` spans. `Gene_expression` frames always have exactly one `Theme`, while `Binding` frames may have zero or more `Themes`.

```
T1 Protein 1650 1655 IP-10
T2 Protein 951 955 PU.1
T3 Protein 1665 1670 ISG54
T4 Protein 978 992 CSF receptor
T5 Binding 932 937 binds
T6 Gene_expression 1634 1644 expression
E1 Binding:T5 Theme:T2 Theme2:T4
E2 Gene_expression:T6 Theme:T1
E3 Gene_expression:T6 Theme:T3
```

Figure 4: An example annotation in BRAT format, following the task specification from Figure 3. The text-bound annotations `T` are the span annotations, the event annotations `E` define our frames.

els can be integrated and interchanged for new use cases. This addresses challenges C3 and C4 from Section 1. (3) Users of IE tools: With the common interface our framework provides, end-users do not need to know theoretical details about models but can still use different models for their use case. This addresses challenge C4 from Section 1.

Framework Structure. Figure 2 illustrates the structure of the DERE framework. It is composed of two main components: The *frontend* comprises the user specification of the task ("task schema"), including the types of spans and entities to be identified, and the possible relations that can exist between them. It manages reading corpora and annotation files and provides an interface for users. The *backend* hosts the models that make actual predictions for spans, frames, and slots, given the task schema, and their configurations. DERE backends follow a modular design, wherein different backends, using different methods for prediction, can be used interchangeably with minimal changes to the frontend.

Task Schemata. DERE represents all relations $r \in \mathcal{R}$ in terms of two types of entities: *spans* and *frames*. A span $s \in \mathcal{S}$ is a contiguous span of text from the input corpus. Each span has a type $t \in \mathcal{T}_\mathcal{S}$, corresponding to the kind of entity that that span represents. The set of possible span types $\mathcal{T}_\mathcal{S}$ is specified by the user for the task. A frame $f \in \mathcal{F}$ represents a relationship between multiple spans or other frames. Each frame contains a number of named *slots* $l \in \mathcal{L}$. These slots can each be filled by zero or more other spans or frames. The set of frame types $\mathcal{T}_\mathcal{F}$, like span types, is task-specific. For each frame type, the user specifies a set of slots, and for each slot, what types of frames or spans can fill it, plus optional cardinality constraints.

We represent a task schema as an XML file. Figure 3 gives an example task schema file, for a subset of the BioNLP shared task (Kim et al., 2009). Note that this specification defines a directed graph with spans and frames as vertices $\mathcal{V} = \mathcal{S} \cup \mathcal{F}$ and relations as edges: $\mathcal{E} = \mathcal{R}$.

Data Files. Annotated data, needed for training models, are provided to DERE as annotation files. We currently support annotations in the BRAT (Stenetorp et al., 2012) format, *cf.* Figure 4.

4 Proof-of-Concept System

As a proof of concept, we present the following system consisting of a pipeline of traditional NLP formalizations: First, spans relevant for the task are identified. Then, a classifier decides for each pair of relevant spans which slots of which frame they are likely to fill. Finally, a heuristic decoding step compiles the results into frames. Figure 5 illustrates this pipeline. The proof-of-concept system only supports non-recursive structures: slots of frames cannot be filled by other frames, but must be filled by spans – *i.e.*, the right-hand BioNLP frame from Figure 1 could not be predicted in this

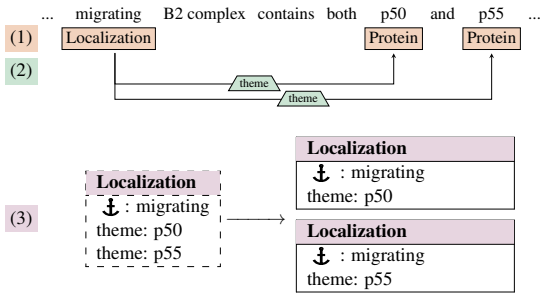

Figure 5: Proof-of-concept pipeline: span identification (1), slot classification (2), and decoding into frames (3). ⚓ : frame anchors (triggers)

implementation. Note that this is only a proof-of-concept baseline but the framework is not limited to pipeline models. In the future, we will develop joint models that can cope with recursive structures.

Span Identification. We cast the span identification problem as a BIO-style sequence-labeling task that predicts the span boundaries. To model overlapping spans, we train one model per span type which outputs all spans of that type. Our proof-of-concept system uses conditional random fields (CRF, Lafferty et al., 2001). The feature set consists of the lower-cased words, their stems, their shape (orthographic case, digits, punctuation), and a flag indicating whether the word is included in a task-specific gazetteer. All features (except the last one) are applicable to any NLP task. The gazetteer feature is based on a simple lexicon of label-specific words (*e.g.,* positive words for detecting positive spans for sentiment analysis) and can be instantiated without any technical knowledge.

Slot Classification. Once the spans are identified, the slot classifier is used to predict which slots of which frame they are likely to fill. We break this question down to a classification task at the level of span pairs – one anchor span representing a frame, and another span representing a potential argument. The search space is restricted to those pairs with compatible types according to the schema.

Formally, the classifier takes as input the set $\mathcal{S}$ of all spans identified previously, along with a task schema. For each pair $(s_i, s_j) \in \mathcal{S}^2$ of spans following the task schema, our classifier produces as output either a single relation label r_{ij}, or NR (no relation)[1] if the two spans are unrelated. Conceptually, two spans s_i and s_j are related iff s_i anchors a frame, and s_j fills a slot in that same frame. Relation labels r_{ij} are pairs $(f_i, l_j) \in \mathcal{T}_\mathcal{F} \times \mathcal{L}$, where f_i is the frame type anchored by s_i and l_j is the slot type in f_i that s_j fills. This enables us to model, *e.g.,* in the task schema in Figure 3, BINDING.THEME and GENE_EXPRESSION.THEME as separate relations. A linear support vector machine is used to predict the most likely relation label (or NR). Users can enable subsampling of negative examples.

As outlined in the introduction, the features we take into account are included with the aim of being task-agnostic. Intra-span features are types of identified spans and the bag of words in both spans. Inter-span features take into account context. We use the bag of words of tokens between the spans, and of the tokens on the shortest path connecting the spans in a parsed dependency tree, which we assume to accurately capture the relationship expressed by the slot that links the two spans. Since spans can contain multiple tokens, there can be several shortest paths between tokens from the two spans. Under the assumption that tokens in a span are closely related to each other, we select the shortest of these paths. In addition, we also use a bag of bigrams of alternating label-token sequence on that same path. Finally, we measure the length of the shortest path and the token distance.

Decoding. Once the slot classifier identifies all related span pairs, the decoding step generates frames. Pairs of spans (s_i, s_j) that stand in a relation r are first partitioned into equivalence classes $\mathcal{C}_h$ according to their anchor span (i.e., $(s_i, s_j) \in \mathcal{C}_i$). It would be possible to produce one frame for each equivalence class $\mathcal{C}_h$, anchored by the common anchoring span s_h, and with slots filled according to each span pair's relation label r. However, as equivalence classes can be arbitrarily large, this would allow for each slot to be filled by arbitrarily many spans (as illustrated in the bottom-left of Figure 5). As the task schema might impose cardinality constraints, further processing is required to ensure that all produced frames are consistent with the task schema. For each equivalence class $\mathcal{C}_h$, we consider all possible *legal frames* – i.e., all frames that are consistent with the task schema and whose slots are filled according to some subset of $\mathcal{C}_h$. Of these legal frames, we retain all *maximally-filled legal frames* (see bottom-right of Figure 5).

Evaluation and Results. To prove the feasability of our proof of concept, we report results with this configuration on the 2009 BioNLP shared task, for which we re-use the original evaluation machin-

[1] We generate negative examples automatically.

Event Class	Precision	Recall	F1
Gene_expression	68.12	57.30	62.25
Transcription	70.59	14.63	24.24
Protein_catabolism	64.00	76.19	69.57
Phosphorylation	65.85	57.45	61.36
Localization	78.57	41.51	54.32
SVT-TOTAL	68.46	50.27	57.97

Table 1: Performance of the proof-of-concept system for biomedical relation extraction (BioNLP'09 dev set)

Sentiment Class	Precision	Recall	F1
Positive	41.07	24.19	28.57
Negative	26.68	7.15	11.00
Neutral	5.83	4.50	5.08

Table 2: Performance of the proof-of-concept system for aspect based sentiment analysis (10-fold crossvalidation on USAGE corpus).

ery. The evaluation calculates the F_1 scores for the individual frames (events in the BioNLP task) using a soft matching for trigger boundaries and approximate recursive matching. Table 1 provides the results of our simple system on that task. Due to the restriction of our proof of concept to non-recursive structures (*cf.* Section 4), we only report on the BioNLP event types where all slots are filled by spans. In comparison to the second-ranked system, which also reports results on dev (Buyko et al., 2009), our performance is slightly lower (1 percentage point less for protein catabolism, 13pp less for gene expression and phosphorylation, but 11pp more for localization). This confirms the general usability of our general method.

Correspondingly, Table 2 provides the current results of the same model on the USAGE corpus for aspect based sentiment analysis (Klinger and Cimiano, 2014), with 10-fold crossvalidation on the English subset. In comparison to previous results, our numbers are very low. Previous work showed that this is tackled by joint inference, which we did not implement yet (Klinger and Cimiano, 2013; Yang and Cardie, 2013). However, this proof-of-concept implementation of the same model already shows the reusability of our framework by only changing the task schema specification. It motivates and enables further research on reusable models across tasks with different needs.

Technical Details and Availability. The framework is implemented in Python, following an object-oriented design for frontend and backends to support easy interchangeability of components. The choice of Python will also help with future integration of neural network models. For the proof-of-concept backend, we use scikit-learn for feature extraction and training (Pedregosa et al., 2011) with crfsuite and liblinear. Tokenization and stemming is done with NLTK (Loper and Bird, 2002), dependency features are extracted with spacy (Honnibal and Johnson, 2015) and dependency graphs are stored and processed using NetworkX (Schult, 2008). The code is available under the Apache 2.0 License.[2]

5 Conclusion and Future Work

This paper presented DERE, a general framework for declarative specification and compilation of template-based slot filling. It addresses the needs of three groups of users: backend model developers, developers of information extraction tools for new use cases and end users of information extraction tools. vEspecially, it simplifies the evaluation and comparison of new information extraction models across tasks as well as the straightforward application of existing models to new tasks. By our general design of spans and frames, it is possible to apply DERE to a large variety of natural language processing tasks, such as unary, binary and n-ary relation extraction, event extraction, semantic role labeling, aspect-based sentiment analysis, etc.

As BRAT annotations are not as expressive as our task schema files, we plan to extend the frontend of DERE by supporting a native, XML-based annotation format in the future. For the backend, our goal is to develop a variety of state-of-the-art models with joint span identification, slot classification, and frame decoding, *e.g.,* neural networks with structured-prediction output layers (Lample et al., 2016; Adel and Schütze, 2017, *i.a.*). Given a variety of different models and tasks, we will be able to address interesting research questions, such a transfer learning and joint learning across tasks and domains. We plan to further analyze the usage of DERE and the possibilities it provides for integrating different model types and configurations in a multi-task oriented shared task.

Acknowledgments This work has been partially funded by the German Research Council (DFG), projects KL 2869/1-1 and PA 1956/4-1.

[2] http://www.ims.uni-stuttgart.de/forschung/ressourcen/werkzeuge/DeRE.en.html

References

H Adel, B Roth, and H Schütze. 2016. Comparing convolutional neural networks to traditional models for slot filling. In *NAACL-HLT*.

H Adel and H Schütze. 2017. Global normalization of convolutional neural networks for joint entity and relation classification. In *EMNLP*.

G Angeli, V Zhong, D Chen, A Chaganty, J Bolton, MJ Premkumar, P Pasupat, S Gupta, and CD Manning. 2016. Bootstrapped self training for knowledge base population. In *TAC*.

Ekaterina Buyko, Erik Faessler, Joachim Wermter, and Udo Hahn. 2009. Event extraction from trimmed dependency graphs. In *BioNLP'09 Shared Task on Event Extraction*.

J Clarke, V Srikumar, M Sammons, and D Roth. 2012. An NLP curator (or: How i learned to stop worrying and love NLP pipelines). In *LREC*.

R Collobert, J Weston, L Bottou, M Karlen, K Kavukcuoglu, and P Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

H Cunningham, V Tablan, A Roberts, and K Bontcheva. 2013. Getting more out of biomedical documents with GATE's full lifecycle open source text analytics. *PLOS Computational Biology*, 9(2).

JR Curran. 2003. Blueprint for a high performance NLP infrastructure. In *Workshop on software engineering and architecture of language technology systems*.

D Das, D Chen, A Martins, N Schneider, and NA Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40:9–56.

GR Doddington, A Mitchell, MA Przybocki, LA Ramshaw, S Strassel, and RM Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *LREC*.

D Ferrucci and A Lally. 2004. UIMA: An architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3–4):327–348.

I Hendrickx, SN Kim, Z Kozareva, P Nakov, D Ó Séaghdha, S Padó, M Pennacchiotti, L Romano, and S Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *SemEval*.

M Honnibal and M Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *EMNLP*.

J Kim, T Ohta, S Pyysalo, Y Kano, and J Tsujii. 2009. Overview of BioNLP'09 shared task on event extraction. In *BioNLP*.

R Klinger and P Cimiano. 2013. Joint and pipeline probabilistic models for fine-grained sentiment analysis: Extracting aspects, subjective phrases and their relations. In *ICDMW*.

R Klinger and P Cimiano. 2014. The USAGE review corpus for fine grained multilingual opinion analysis. In *LREC*.

JD Lafferty, A McCallum, and FCN. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*.

G Lample, M Ballesteros, S Subramanian, K Kawakami, and C Dyer. 2016. Neural architectures for named entity recognition. In *NAACL-HLT*.

E Loper and S Bird. 2002. Nltk: The natural language toolkit. In *Workshop on Effective Tools and Methodologies for Teaching NLP and CL*.

A McCallum, K Schultz, and S Singh. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In *NIPS*.

M Mintz, S Bills, R Snow, and D Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-AFNLP*.

T Morton, J Kottmann, J Baldridge, and G Bierner. 2005. OpenNLP: A java-based NLP toolkit. `http://opennlp.sourceforge.net`.

TH Nguyen, K Cho, and R Grishman. 2016. Joint event extraction via recurrent neural networks. In *NAACL*.

F Pedregosa et al. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

B Rink and S Harabagiu. 2010. UTD: Classifying semantic relations by combining lexical and semantic resources. In *SemEval*.

DA Schult. 2008. Exploring network structure, dynamics, and function using networkx. In *Python in Science Conference*.

P Stenetorp, S Pyysalo, G Topić, T Ohta, S Ananiadou, and J Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *EACL*.

M Surdeanu. 2013. Overview of the TAC2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *TAC*.

D Tang, B Qin, and T Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*.

B Yang and C Cardie. 2013. Joint inference for fine-grained opinion extraction. In *ACL*.

D Zeng, K Liu, S Lai, G Zhou, and J Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING*.