Objectives for Relation Extraction

Justin T. Chiu

July 8, 2019

Abstract

Many recent relation extraction systems predict the relationship between an entity and value given their locations in the text. This requires observed locations of mentions, which requires annotations at the word level. Any form of annotation at the word level does not scale as the size of the text and the number of labels increases, and even more so if there is ambiguity. In order to train a probabilistic information extraction without any supervision at the level of text, we specify a model that, for each word, either chooses a triple from a knowledge base to explain or does not use the current word.

1 Introduction

In relation extraction we extract facts from a passage of text. Let's jump straight into a formal description. Given a text $y = [y_0, \ldots, y_T]$ the goal is to predict all facts expressed in that text with respect to the schema of a Knowledge Base (KB). The length of the text T varies: depending on the dataset we may be interested extracting all facts expressed in a sentence or paragraph. All sequences of text are bounded in length. The KB $r = \{(\epsilon_i, \tau_i, \nu_i)\}_{i \in \mathcal{I}}$ is an indexed set of records r_i. Each record $r_i = (\epsilon_i, \tau_i, \nu_i)$ consists of an entity, type, and value triple that is governed by a schema so that $\epsilon_i \in \mathcal{E}, \tau_i \in \mathcal{T}, \nu_i \in \mathcal{V}$. Let ϵ, τ, ν be the entities, types, and values from the KB r in aggregate, i.e. $\epsilon = \{\epsilon_i\}_{i \in \mathcal{I}}$ etc.

We assume supervision only at the proposition level of a KB. We know what the relationship is between entities and values (although this assumption can be relaxed), but we do not know where in text records are realized. Except for Zeng et al. (2018), prior work has either assumed that the locations of entities and values are given as input features or that the locations of entities and values are observed at training time. We relax this assumption and do not use any annotations at the level of text.

We are primarily concerned with the scenario where we have an overcomplete KB schema with respect to a specific passage of text. This fits many scenarios in real world applications: we may have thousands of entities of interest if our KB was pulled from an external source such as Freebase, but the particular document we wish to analyze only discusses tens of entities, only a few of which are present in our KB.

Table 1: Notation

y	$\triangleq$	A vector of words y_t.
r	$\triangleq$	The knowledge base, an indexed set of records each consisting of entities, types, and values.
ϵ	$\triangleq$	The list of all entities in a KB by index. The same entity may appear in multiple indices.
τ	$\triangleq$	The list of all types in a KB by index.
ν	$\triangleq$	The list of all values in a KB by index.
c_t	$\triangleq$	A Bernoulli RV that indicates whether word y_t is content or not.
a_t	$\triangleq$	A Categorical RV that specifies the index of the record word y_t is aligned to.
v_t	$\triangleq$	A Categorical RV that is the canonical value representation of word y_t.

2 Model

Our goal is to produce a set of record triples given text. As we assume that only the KB and text are observed, we can either produce this set directly from the text or identify realizations of records at a more granular level and then aggregate our choices. We choose to pursue the latter since identifying facts at a granular level is a step towards compositional language understanding. This may appear similar to shallow parsing, however we do not expect to obtain any interpretable segmentations of the text. We are only interested in the extractive capabilities of the model. This is more difficult than performing extraction at a higher level since we must justify every extraction with a mention.

Although entity or value mentions may be multi-word expressions, identifying realizations at the word level is no less expressive as we can aggregate our word level choices into sequence level ones.

Loosely inspired by work in language grounding Liang et al. (2009), we focus on identifying value mentions in text. Our model first identifies words that are value mentions, translates the value mention into a canonical value, then aligns the mention to a record in the KB by predicting what entity and the relation type associated with the value mention.

Our information extraction model is denoted $q(r \mid y)$, a distribution over a KB r given some text y. Let f_* be arbitrary learned functions. Our word level model is given by the following: For each word y_t, we have

1. Value mention detector $q(c_t \mid y)$: We classify whether word y_t is a value mention. Each $c_t \sim \text{Bern}(f_c(y))$, where a value of 1 indicates that y_t is a value mention.

2. Value prior $p(v_t \mid \epsilon, \tau)$: If $c_t = 0$, we generate the value distribution ignoring the text.

3. Translation $q(v_t \mid y)$: We translate the word y_t into a value v_t. The $v_t \sim \text{Cat}(f_v(y))$ is the canonical value from the KB schema associated with y_t.

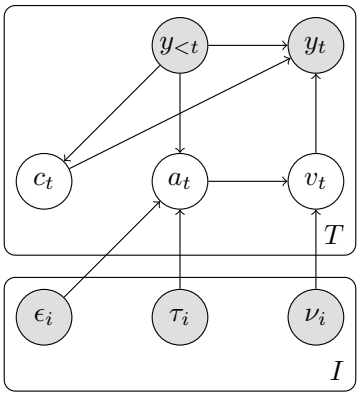

Figure 1: The generative model which produces words given a knowledge base.

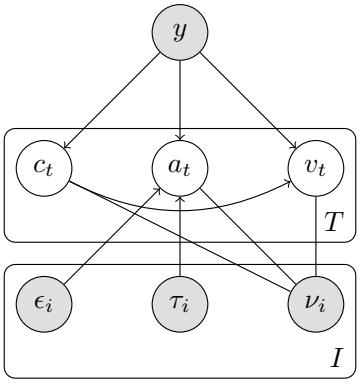

Figure 2: The inference network which predicts word-level values and alignments.

4. Alignment $q(a_t \mid y, \epsilon, \tau)$: We align the word y_t by classifying who (the entity) and what (the relation type) are being talked about. In particular, $a_t \sim \text{Cat}(f_a(y))$ denotes the alignment to the record r_{a_t} given by the index $i = a_t$.

Finally, we aggregate the word level information at the sequence level:

1. Aggregation $\prod_i q(\nu_i \mid a, v, c)$: In order to parameterize a distribution over the KB r, we must then aggregate all of the word-level information into a distribution over facts at the sequence level. We use a deterministic OR potential for each r_i (stuck here, need to read up on this).

3 Three perspectives on training

We can either train q directly on the conditional task or train it to mimic the posterior of a suitable generative model.

Axes of objectives:

1. Proposal distribution: Learned or uniform (or some prior)

2. Probabilistic interpretation: Marginal likelihood or KL
3. Probabilistic interpretation 2: Approximate posterior of a generative model or learn directly.

3.1 Marginal loss

Our first loss, the marginal likelihood, takes the form of

$$\mathcal{L}_{\text{ML}} = \sum_t \log \sum_{a_t} \sum_{c_t} q(a_t, c_t, v_{a_t})$$

where the unobserved alignments and binary choice are marginalized over.

This has the interpretation of maximizing the 'softmax' of the log probabilities. However, the sharpness of is limited by the choice of q. With a uniform $q(a, c)$, $\mathcal{L} = \sum_t \sum_{a_t} q(v_{a_t}) + C$, where C is the normalization term which we can assume to be constant here. The gradient of this term wrt the log probabilities is weighted by the posterior, so a high entropy q will result in a smaller gradient.

One concern is whether the model will learn to align at all. Decomposing this lower bound further, we have

$$\begin{aligned}
\mathcal{L}_{\text{ML}} &= \sum_t \log \sum_{a_t} \sum_{c_t} q(v_{a_t}, a_t, c_t) \\
&= \sum_t \log p(v) q(c_t = 0) + \sum_{a_t} \sum_{c_t} q(v_{a_t}, a_t \mid c_t = 1) q(c_t = 1)
\end{aligned}$$

Given a reasonable initialization of $q(v_{a_t} \mid a_t)$, there is no reason to believe this will not work.

3.2 KL

The second loss, a lower bound on the marginal likelihood, is the following

$$\mathcal{L}_{\text{KL}} = \sum_t \sum_{a_t} \sum_{c_t} q(a_t, c_t) \log q(v_{a_t})$$

The benefit of this loss is that we can approximate it via Monte Carlo sampling. Under a uniform $q(a, c)$, we must maximize the probabilities of the values associated with each alignment equally. This will force the model to explain records that may have nothing to do with the current word, resulting in a large difference between $\mathcal{L}_{\text{ML}}$ and $\mathcal{L}_{\text{KL}}$. With a learned $q(a, c)$, the lower bound can be made much tighter so there is less of a difference.

3.3 Approximating the posterior of a generative model

One issue with the above model is that it is highly unconstrained. There is no constraint that forces the model to extract a fact from any specific word. This may lead to issues seen in other alignment problems where there is an off-by-one error. Depending on our parameterizations of f_*, this behaviour may be exacerbated. We propose to alleviate this

issue by training our q to approximate the posterior of a more constrained generative model. (Philosophically, I believe the model you care about should have the constraints rather than this way.)

The information extraction model itself was inspired by the following generative process: For every word y_t

1. Generate the values of the KB $v \sim p(v)$ assuming the schema is fixed.
2. Choose $c_t \sim \text{Bern}(f(y_{<t})$, which determines whether to generate word y_t from a language model or the KB.
3. If $c_t = 0$, generate $y_t \sim \text{Cat}(g(y_{<t}))$.
4. Otherwise pick an alignment to the KB $a_t \sim \text{Cat}(h(y_{<t}))$ then generate $y_t \sim \text{Cat}(f'(v_{a_t}))$.

We then train our model to approximate the posterior $p(v \mid y)$. Since this generative model does not use every record during generation, the posterior may only explain a subset of all records. This fits our hypothesis that the KB contains a superset of records expressed in text.

The loss is then given by

$$\mathcal{L}_{\text{VI}}$$

The primary benefit of this over a purely conditional approach is semi-supervised information extraction, where values are missing.

$$\begin{aligned}
\arg\max_p \sum_{y'} \sum_{x'} p^*(y', x') \log \frac{p(y', x')}{p^*(y', x')} &= \arg\max_p \sum_{y'} \sum_{x'} p^*(y', x') \log \sum_z p(y', z, x') && (1)\\
&= \arg\max_p \sum_{y'} \sum_{x'} p^*(y', x') \log \sum_z p(y', z, x') && (2)\\
&\geq && (3)
\end{aligned}$$

References

Percy Liang, Michael I. Jordan, and Dan Klein. 2009. Learning semantic correspondences with less supervision. 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 1 - Volume 1*, ACL '09, pages 91–99, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xiangrong Zeng, Daojian Zeng, Shizhu He, Kang Liu, and Jun Zhao. 2018. Extracting relational facts by an end-to-end neural model with copy mechanism. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 506–514, Melbourne, Australia. Association for Computational Linguistics.