Bi-directional Joint Inference for Entity Resolution and Segmentation Using Imperatively-Defined Factor Graphs

Sameer Singh, Karl Schultz*, and Andrew McCallum

Department of Computer Science
University of Massachusetts, Amherst MA 01002 USA
{sameer,kschultz,mccallum}@cs.umass.edu

Abstract. There has been growing interest in using joint inference across multiple subtasks as a mechanism for avoiding the cascading accumulation of errors in traditional pipelines. Several recent papers demonstrate joint inference between the segmentation of entity mentions and their de-duplication, however, they have various weaknesses: inference information flows only in one direction, the number of uncertain hypotheses is severely limited, or the subtasks are only loosely coupled. This paper presents a highly-coupled, bi-directional approach to joint inference based on efficient Markov chain Monte Carlo sampling in a relational conditional random field. The model is specified with our new probabilistic programming language that leverages imperative constructs to define factor graph structure and operation. Experimental results show that our approach provides a dramatic reduction in error while also running faster than the previous state-of-the-art system.

1 Introduction

Advances in machine learning have enabled the research community to build fairly accurate models for individual components of an information extraction and integration system, such as field segmentation and classification, relation extraction, entity resolution, canonicalization, and schema alignment. However, there has been significantly less success combining these components together into a high-accuracy end-to-end system that successfully builds large, useful knowledge bases from unstructured text.

The simplest possible combination is to run each component independently such that they each produce their output using only provided input data. We call this an *isolated* approach. For example, if asked to build a de-duplicated database of bibliographic records given citation strings, one could build field extraction and citation coreference models that operate independently of each other. However, the accuracy of coreference would almost certainly be improved if it could access the output of segmentation, and thus be able to separately compare individual bibliographic fields such as authors, titles, and venues.

* First two authors contributed equally.

For this reason, the most common approach to component combination is a *pipeline*, in which components are run in some order, and later components have access to the output of already-completed earlier components. Here segmentation could be run first, and then coreference would have fruitful access to field boundaries. Nevertheless, pipeline systems also typically fail to produce high-accuracy final output. This is because errors cascade and compound in a pipeline. When an earlier component makes a mistake, later stages consuming its output are also very likely to produce errorful output. Every stage is an opportunity for compounding error—for example, six components each having 90% accuracy may result in about 50% accuracy for the final stage when pipelined.

In order to avoid this brittle accumulation of errors there has been increasing interest in probabilistic models that perform *joint* inference across multiple components of an information processing pipeline, *e.g.* [1]. Here the system does not commit to a single answer in some stage before considering other stages; rather, multiple hypotheses and uncertainty information are exchanged throughout, such that some components can correct the misconceptions of others. The need for joint inference appears not only in extraction and integration, but also in natural language processing, computer vision, robotics, and elsewhere.

Joint inference aims to predict many variables at once, and thus usually leads to complex graphical model structure with large tree-width, making exact inference intractable. Several approximate methods for joint inference have been explored.

One, described by Finkel *et al.* [2], runs a pipeline, but *samples* the output for each component rather than selecting the single most likely output; then the pipeline is run repeatedly so that different combinations of output throughout the pipeline are evaluated. This feed-forward approach to inference is a classic method in Bayesian networks, but has the drawback that it only allows information to flow in one direction. For example, correct coreference of a messy citation with a clean citation provides the opportunity for an alignment between these two citations to help the model correctly segment the messy one. However, feed-forward inference does not support this backward flow of information.

In another approach, pursued by Wellner *et al.* [3], each component produces a weighted N-best list of hypotheses for consumption by other components, and components are re-visited in both directions—both forward and backward through the pipeline. This approach allows information to flow in both directions, however an N-best list is a restrictive approximation for the full distribution of large-output components.

Yet another approach, proposed by Poon and Domingos [4], creates a single Markov random field with factor template structure specified via first-order logic—called a Markov logic network, MLN—and performs randomized approximate inference by MC-SAT [5], a variant of WalkSAT [6]. MC-SAT repeatedly reconsiders variables and factors in the model in an order independent of the pipeline. However, as described below, limitations of first-order logic make it difficult to specify a coreference factor that uses the uncertain output of segmentation. For this reason, in Poon and Domingos [4], citation coreference com-

patibility is measured using features of the *un*-segmented citation (see Sect. 3.2), and inference is not strongly bi-directional.

This paper presents a strong bi-directional approach to inference for joint segmentation and coreference. Like Poon and Domingos [4] we use a conditionally-trained factor graph, with randomized approximate inference that roams freely in both directions. However, rather than employing first-order logic, we leverage our new work in probabilistic programming that uses imperative procedures to define the factor template structure. We term this approach *imperatively-defined factor graphs* (IDFs); they are factor graphs whose template structure, as well as aspects of parameterization and inference, are specified with the power of a Turing-complete language. This added flexibility enables us to include more sophisticated factors between segmentation and coreference that allow information to flow bi-directionally. In spite of being more expressive, IDF's flexibility also allows our model to be more efficient.

We evaluate our approach on segmentation and coreference of the citation strings from the Cora data set. In comparison with Poon and Domingos [4] we reduce coreference error and segmentation error by $\sim 20-25\%$ while also running $3-15$ times faster, providing a new state-of-the-art result. We also analyze the nature of bi-directional inference with separate diagnostic experiments.

2 Background

2.1 Factor Graphs

A factor graph [7] is a bipartite graph over factors and variables. Let factor graph G define a probability distribution over a set of output variables $\mathbf{y}$ conditioned on input variables $\mathbf{x}$. A factor Ψ_i computes a scalar value over the subset of variables $\mathbf{x}_i$ and $\mathbf{y}_i$ that are neighbors of Ψ_i in the graph. Often this real-valued function is defined as the exponential of an inner product over sufficient statistics $\{f_{ik}(\mathbf{x}_i, \mathbf{y}_i)\}$ and parameters $\{\theta_{ik}\}$, where $k \in [1, K_i]$ and K_i is the number of parameters for factor Ψ_i. Let $Z(\mathbf{x})$ be the data-dependent partition function used for normalization. The probability distribution can be written as:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{\Psi_i \in G} \exp\left[\sum_{k=1}^{K_i} \theta_{ik} f_{ik}(\mathbf{x}_i, \mathbf{y}_i)\right].$$

In practice factor graphs often use the same parameters for several factors; this is termed *parameter tying*. A *factor template* T_j consists of parameters $\{\theta_{jk}\}$, sufficient statistic functions $\{f_{jk}\}$, and a description of an arbitrary relationship between variables, yielding a set of satisfying tuples $\{(\mathbf{x}_j, \mathbf{y}_j)\}$. For each of these variable tuples $(\mathbf{x}_i, \mathbf{y}_i)$ that fulfills the relationship, the factor template instantiates a factor that shares $\{\theta_{jk}\}$ and $\{f_{jk}\}$ with all other instantiations of T_j. Let $\mathcal{T}$ be the set of factor templates. In this case the probability distribution becomes:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{T_j \in \mathcal{T}} \prod_{(\mathbf{x}_i, \mathbf{y}_i) \in T_j} \exp\left[\sum_{k=1}^{K_j} \theta_{jk} f_{jk}(\mathbf{x}_i, \mathbf{y}_i)\right].$$

The process of instantiating individual factors from their templates is termed *unrolling*. For a factor Ψ_i that is an instantiation of factor template T_j, the inner product of $\{f_{jk}(\mathbf{x}_i, \mathbf{y}_i)\}$ and parameters $\{\theta_{jk}\}$ is termed the *score* of the factor.

In a pipeline approach to solving multiple tasks the influence between stages is uni-directional down the pipeline, since inference is performed separately for each task in a sequential order. To enable the bi-directional flow of information we add factors connecting variables of different tasks and perform inference simultaneously for the entire model. A factor template defined over variables of different tasks is termed a *joint factor template*, and an instantiation of a joint factor template is called a *joint factor*.

Introducing joint factors usually increases the complexity of the graph because it tends to create many more cycles and a larger tree width. These complex graphs often cause inference to become intractable, which we address by using imperatively-defined factor graphs, as described next.

2.2 Imperatively-Defined Factor Graphs

Imperatively-defined factor graphs (IDFs) are an approach to probabilistic programming that preserves the *declarative* semantics of factor graphs, while leveraging *imperative* constructs (pieces of procedural programming) to greatly aid both efficiency and natural intuition in specifying model structure, inference, and learning. Rather than using declarative languages, such as SQL or first-order logic, model designers have access to a Turing complete language when writing their model specification. A model written as an IDF *is* a factor graph, with all the traditional semantics of factors, variables, possible worlds, scores, and partition functions; IDFs simply provide an extremely flexible language for their succinct specification that also enables efficient inference and learning. A side benefit is that IDFs provide a mechanism whereby model designers can inject both declarative and procedural domain knowledge.

We have developed IDFs in the context of Markov chain Monte Carlo (MCMC) inference, which is a common approach to achieve efficiency in complex factor graphs [8–10] where variable-factor connectivity structure changes during inference. In such situations, fully unrolling the graph (creating the factors that would be necessary to score all possible worlds, as required for belief propagation) would often result in an exponential or super-exponential number of factors. However, in MCMC, we only represent a single possible world at a time. MCMC performs inference by stochastically proposing some change to the current possible world, and then accepting that change with a probability that depends on the ratio of post- and pre-proposal model scores. Calculating these acceptance probabilities is quite efficient because not only do the partition functions, $\mathrm{Z}(\mathbf{x})$, cancel, but the contributions of all factors not touching changed variables also cancel; in fact, in our implementation they are not even created. This allows us to avoid the need to unroll and score the entire graph to evaluate a change, resulting in quite efficient inference.

We summarize four key imperative constructs in IDFs, and argue that they provide a natural interface to central operations in factor graph construction and inference. For more details see [11].

1. *Imperative structure definition* determines the connectivity between factors and their variable arguments. The key operation in efficient MCMC is finding all variable arguments of a factor template given a relevant changed variable. IDFs make this a primitive operation, with the opportunity to define its behavior in a Turing-complete language—dynamically finding all neighboring variables of an instantiated factor given one of its neighbors. For example, this approach allows arbitrary graph-search algorithms to define the arguments of a factor.
2. *Imperative constraint preservation* keeps all explored possible worlds in the feasible region by embedding the necessary control in the procedurally-defined MCMC proposal function. For example, we can avoid the large expense of having factors that aim to enforce transitivity in coreference by instead: (a) initializing to a possible world that obeys transitivity, and (b) implementing a proposal function that is guaranteed to preserve the transitivity constraint.
3. *Imperative variable coordination* also preserves constraints or encourages more fruitful proposals by including in the variable-value setting method of one variable a procedural "hook" that automatically sets other related variable(s) to a compatible value.
4. *Imperative variable-sufficient mapping* allows the data to be represented in natural, convenient variables, and then later to be functionally mapped into the sufficient statistics required for our desired parameterization.

We have implemented IDFs in the FACTORIE toolkit[1] [11]. Typically, IDF programming consists of four distinct stages: (1) defining the data representation, (2) defining the factors for scoring, (3) optionally providing domain knowledge to aid inference (including the flexibility to write the entirety of a proposal function), (4) reading in the data, learning parameters, testing, and evaluating.

For parameter estimation in this paper we use Sample-Rank [12]. This method embeds opportunities for approximate gradient ascent into each MCMC proposal step by performing perceptron-like updates whenever possible worlds, pre- and post-proposal, are ranked differently by their model scores versus their distance to the labeled true world. Sample-Rank's proof of convergence [12] is similar to the proof for perceptron. In this paper, the final values of each parameter are obtained by averaging over the learning period. To evaluate a trained model we search for the MAP configuration by running MCMC with a low temperature applied to the ratio of model scores in the proposal acceptance probability.

[1] Available at http://factorie.cs.umass.edu

3 Bi-directional Joint Inference for Segmentation and Entity Resolution

In some information extraction tasks mentions of entities must be discovered by segmenting them from background text. In other cases the mention strings are provided, but they have internal structure requiring segmentation. Here we address the latter case, in which we jointly segment the contents of many mentions, while simultaneously performing coreference on the mentions.

Consider the task of citation matching in which we are given a large collection of citation strings from the "References" section of research papers. They have different citation styles, different abbreviations, and typographical errors. Many of the citations refer to the same underlying papers. Our job is to find the citations referring to the same paper (*coreference* or *entity resolution*) and also identify the *author*, *title*, and *venue* fields of each citation (*segmentation*).

The tasks of segmentation and entity resolution are often solved in isolation, without access to each other's predictions [13, 14], however, using the results of the other subtask often helps reduce errors. For example, coreference compatibility between two citations can be assessed more accurately if we can compare segmented *title* fields and *venue* fields separately, with different distance measures for each. Also, segmentation accuracy can be improved by accounting for field similarity among multiple coreferent citations. These interdependencies between the two tasks have led others to explore *joint* models of segmentation and coreference of citations with generative models [15], with conditionally-trained models performing bi-directional N-best message-passing between tasks [3], and with conditional models whose bi-directional factors mainly leverage coreference for performing segmentation [4].

Next we present a highly-coupled, bi-directional approach to joint inference for citation segmentation and coreference. We use imperatively-defined factor graphs (IDFs) to specify a single undirected graphical model that performs both tasks. It includes multiple factors that simultaneously examine variables of segmentation and coreference. In the following sections we describe the variables and factors of this model, as well as the MCMC proposal function used.

3.1 Variables

Segmentation Variables. Observed and predicted data for segmentation are represented by three types of variables: `Token`, `Label`, and `Field`. Each observed citation string consists of a sequence of words, each represented by a `Token` whose value is the word string itself. Each `Token` is associated with a corresponding unobserved `Label` variable, whose value is any of the field types, or "None." In addition, consecutive `Tokens` whose `Labels` have the same value are also represented as a `Field` variable, whose value is the string formed by the concatenation of its words. `Fields` are set in coordination with `Labels` through *imperative variable coordination*, and facilitate joint factors between coreference and segmentation. These variables are summarized in Table 1.

Table 1. Variable Types for Segmentation and Coreference: Variable types that are used in the joint model of segmentation and entity resolution.

Segmentation Variable Types	
`Token:`	Observed variable that represents a word in the mention string
`Label:`	Variable that can take any of the field types as a value (or "None")
`Field:`	Indices of consecutive `Tokens` that belong to a particular field
Coreference Variable Types	
`Mention:`	Variable that takes a single `Entity` as its value
`Entity:`	Set of `Mentions` that are coreferent

Coreference Variables. There are two variable types for coreference: `Entity` and `Mention`. Each `Entity` embodies an underlying paper, to which there may be many citations. Each citation is represented by a `Mention` whose coreference involves an assignment to its corresponding `Entity`. Although each `Mention` object *contains* the citation string as a member instance variable, the `Mention`'s *value* (as an IDF random variable) is the `Entity` to which it has been assigned. The `Entity` is accordingly a *set-valued* variable—its value is the set of `Mentions` that have this `Entity` as their value. The values of `Mentions` and `Entities` are synchronized through *imperative variable coordination*. Note that this representation eliminates the need for factors enforcing mutual-exclusion or transitivity since each `Mention` can only have one `Entity` as its value. These variables are also summarized in Table 1.

Connections between Coreference and Segmentation Variables To support joint factors between segmentation and coreference, the above variables contain references to each other. We take advantage of the fact that our random variables are objects in an object-oriented programming language, and represent arbitrary relations as member instance variables. Each `Mention` contains an array of its `Tokens`, as well as a list of its constituent `Fields`. Furthermore, each `Field` has a reference to its enclosing `Mention`, comprising `Tokens`, and adjacent `Fields`. These object-oriented cross-references are used to efficiently find the neighboring variables of a factor (*imperative structure definition*) and to access related variables for calculating sufficient statistics (*imperative variable-sufficient mapping*).

3.2 Factors Templates

Now we define the factor templates that are used to score possible worlds; they are summarized in Table 2. A small example consisting of three mentions and three fields, showing their instantiated factors and neighboring variables, is depicted in Figure 1. Given the neighboring variables of a factor, most of our factor templates employ additional computation to calculate sufficient statistics (features) from these neighbors—leveraging the flexible separation of data representation and parameterization (*imperative variable-sufficient mapping*).

Table 2. Factor Templates of Segmentation and Coreference: Factor templates that are used in the joint model of segmentation and entity resolution.

Segmentation Factors	
`LabelToken:`	Factor between every token and its field type
`LabelNextToken:`	Factor between every label and the next token
`LabelPrevToken:`	Factor between every label and the previous token
`FieldFactor:`	Factor created for every `Field` to allow field-wise features
Coreference Factors	
`Affinity:`	Factor created between coreferent pairs of `Mentions`
`Repulsion:`	Factor created between pairs of `Mentions` that are not coreferent
Joint Factors	
`JntInfBased:`	Factor between `Mention` and `Label` based on joint factors in [4]
`JointAffinity:`	Factor between `Fields` of the same type of coreferent `Mentions`
`JointRepulsion:`	Factor between `Fields` of the same type of non-coref. `Mentions`

Segmentation The segmentation factor templates express preferences about the `Tokens`' `Label` values and their segmentation into `Fields`. We define three factor templates traditional in linear-chain CRFs: factors between each `Label` and its corresponding `Token` (`LabelToken`), plus factors between the `Label` and adjacent `Tokens` on either side (`LabelPrevToken` and `LabelNextToken`).

In addition we define a factor template that examines a `Field` (and therefore has simultaneous access to all its constituent `Tokens`). This allows us to implement features similar to the rest of the segmentation rules of Poon and Domingos' "Isolated" MLN [4]. These include various first-order formulae over the `Tokens` of a `Field`. We also employ features testing the existence of punctuation at the beginning, the end, and within the `Field`. Finally we include features that take the position of the `Field` within the mention string into account. All of these features are taken in conjunction with the field type (`Label`).

Coreference We have two coreference factor templates, which express preferences about partitioning `Mentions` into `Entities`. One measures pairwise affinity between `Mentions` in the same `Entity`; the other measures pairwise repulsion between `Mentions` in different `Entities`. Both factor templates have two `Mentions` as neighbors, and they both share the same *imperative variable-sufficient mapping* function. The number of these factors for a fully-unrolled graph is $O(m^2)$ (where m is the number of the citations). As described above, IDFs do not unroll the graph fully, and only need evaluate factors neighboring *moved* `Mentions`, which is $O(k)$ (where k is the average `Entity` size).

`Affinity` and `Repulsion` factors are scored using the same features (*i.e.* sufficient statistics function), but different parameters. These sufficient statistics are calculated from the un-segmented citation strings. We use the *SimilarTitle* and *SimilarVenue* features as defined in [4]. Furthermore, we add a *SimilarDate* feature that is true when the same year `Token` appears in both `Mentions`, as well as a *DissimilarDate* feature that is true when unequal year `Tokens` appear.

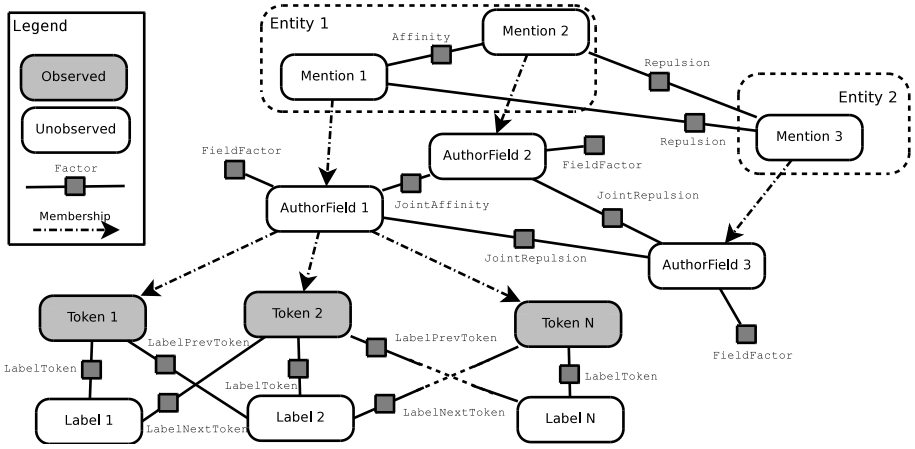


Fig. 1. Model: Variables and factors for joint segmentation and entity resolution shown on a toy example containing two entities and three mentions with a single field in each. Segmentation factors are only shown for one field. `JntInfBased` factors have been omitted for clarity.

Bi-directional Joint factor templates express preferences about both segmentation and coreference simultaneously. In Poon and Domingos [4] all of the interaction between these tasks is captured by four similar rules that use their "JntInfCandidate" predicate. JntInfCandidate(m, i, m') is true if the `Token` trigram starting at position i in `Mention` m also appears in `Mention` m', and the trigram in m is not preceded by punctuation whereas the trigram in m' is. The trigram also must not meet certain "title exclusion" rules described in [4]. Note that JntInfCandidate is pre-calculated from the observed data, independent of segmentation and coreference predictions.

Even though Poon and Domingos' rules containing JntInfCandidate are scored on changes to both coreference and segmentation decisions, there are two reasons it forms only a weak interaction between the tasks. First, these templates only examine pairs of consecutive labels, not whole fields—failing to use information from predicted field range and non-consecutive words in the field. Second, the frequency with which the JntInfCandidate feature appears is quite data-dependent. In the Cora corpus, it occurs only 4973 times—representing an average of < 4 possible coreference candidates per `Mention`, whereas the average `Entity` size is ~ 10. On the other hand, if the feature occurs too often, it can be harmful for coreference. We hypothesize these reasons explain the empirical results in [4], in which the joint factors do not help entity resolution. We further explore this issue in Sect. 5.2.

To achieve stronger interaction between the tasks, we add factor templates that examine predicted `Fields` jointly with coreference decisions (`Mentions`). Our `JointAffinity` factor template defines factors that measure the compatibil-

ity of corresponding `Fields` in coreferent `Mentions`. Similarly `JointRepulsion` factor templates compare the corresponding `Fields` of non-coreferent `Mentions`. Hence the features (sufficient statistics) of these factor templates are able to compare full extracted field strings, and include StringMatch, SubStringMatch, PrefixMatch, SuffixMatch, AnyNTokenMatch, TokenIntersectionSize, etc.

One reason such strongly-joint factor templates have not been used in related work is due to the large number of factor instantiations in the many possible worlds. Such a factor could be created between any two pairs of `Mentions`, and any pair of their possible `Field` segmentations. This leads to $O(m^2n^4)$ factors, where m is the number of `Mentions`, and n is the maximum number of `Tokens` in a `Mention` string. The number of factors then becomes too large to be pre-processed and stored for big datasets, making such factor templates intractable for methods that fully unroll the factor graph or that pre-calculate features. This problem is common in joint inference because factors that represent dependencies between tasks often blow-up in the cross-product of the two hypothesis spaces. Since IDFs never unroll the entire factor graph and allow on-the-fly feature calculation they can efficiently use such factors.

3.3 Proposal Function

Our MCMC proposal function can make changes either to coreference or segmentation. A coreference proposal selects a random `Mention`, then with probability 0.8 moves it to another randomly selected `Entity`, or with probability 0.2 makes it a singleton in a new `Entity`. A segmentation proposal selects a random `Field` and identifies the minimum and maximum amount by which the `Field` can shrink or grow (which depends on the neighboring `Fields`). A new range for the `Field` is selected randomly based on that potential range. When the range of a `Field` is changed, the corresponding `Labels` are automatically adjusted via *imperative variable coordination*. The order of fields within a mention string is fixed: *author*, *title*, then *venue*.

4 Experimental Setup

We use the Cora dataset[2] [16] to evaluate our joint entity resolution and segmentation model. The dataset contains a total of 1,295 citations that refer to 134 ground truth entities. Each citation has three fields (*author*, *title*, and *venue*), with a total of 36,487 tokens. The dataset is divided into the same three folds used by [4]. These folds were not entirely random since they ensure no clusters are split across different folds. Ten runs of three-fold cross-validation are performed on the dataset, unless otherwise specified. Segmentation is evaluated on token-wise precision, recall, and F1. Pairwise coreference decisions are evaluated to obtain the precision, recall, and F1. Cluster recall, defined as the fraction of clusters that are correctly predicted, is calculated to compare with earlier results.

[2] The cleaned version available at http://alchemy.cs.washington.edu/papers/poon07

As a baseline we run isolated coreference and segmentation experiments. These isolated models use only the segmentation and coreference factors, as described in Table 2, respectively (not the "joint factors"). Coreference is initialized to an all-singleton configuration, while segmentation is initialized to an equal-sized three way split of the mention string; we term these "default" configurations. Training consists of 5 loops of 100,000 proposals each. At the beginning of every loop we initialize either to the ground truth or the default configuration, selected randomly. Test-time inference consists of 300,000 proposals (for each task), starting with the default configuration. The temperatures for annealing during training and testing are set to 1.0. During training and inference for baseline isolated tasks only the respective proposal function is used.

Three different types of joint experiments are run to examine several aspects of the bi-directional nature of the joint model. For all of the joint experiments training is performed for 5 loops of 250,000 proposals each. Each task is initialized to either the default or the ground truth configuration, selected randomly, at the beginning of every training loop. Test-time inference consists of a total of 750,000 proposals across both tasks. The temperatures for training and testing are set to 3.0 and 1.0, respectively. During joint inference the proposal function randomly chooses between selecting a coreference or a segmentation proposal.

5 Results

The first joint experiment evaluates the full model including all factor templates and features, and compares these results to the isolated baseline models. The second joint experiment separately evaluates the gains resulting from the `JointInfBased` factors and the fully bi-directional factors described in Sect. 3.2. The third joint experiment examines the behavior of passing predictions between coreference and segmentation in an iterative fashion.

The experiments run very quickly, which can be attributed to *imperative variable coordination* and *imperative structure definition*, as described earlier. Training and inference of the isolated tasks finish within 3 minutes while the joint task takes approximately 18 minutes to run. By comparison, MC-SAT in [4], which does not enforce transitivity constraints for coreference, takes $50 - 90$ minutes. Adding transitivity constraints to the MLN severely increases running time further, as shown in [17].

5.1 Overall Joint Inference

Our results on the Cora dataset are shown in Table 3 and Table 4, demonstrating the benefits of a bi-directional approach to joint inference, with significant improvements on both segmentation and coreference. We also compare with the Fellegi-Sunter coreference model [14]. All improvements of our joint model over our isolated models are statistically significant at 1% using the T-test.

Table 3 shows that our isolated coreference model outperforms the previously published results in [4] on both metrics. Our joint model, which concurrently

Table 3. Cora Coreference: Pairwise precision/recall, F1, and cluster recall, for the coreference task on the *Cora* dataset.

Method	**Prec/Recall**	**F1**	**Cluster Rec.**
Fellegi-Sunter	78.0/97.7	86.7	62.7
Joint MLN	94.3/97.0	95.6	78.1
Isolated IDF	97.09/95.42	**96.22**	**86.01**
Joint IDF	95.34/98.25	**96.71**	**94.62**

Table 4. Cora Segmentation: Token-wise F1 for each field of the segmentation task on the *Cora* dataset.

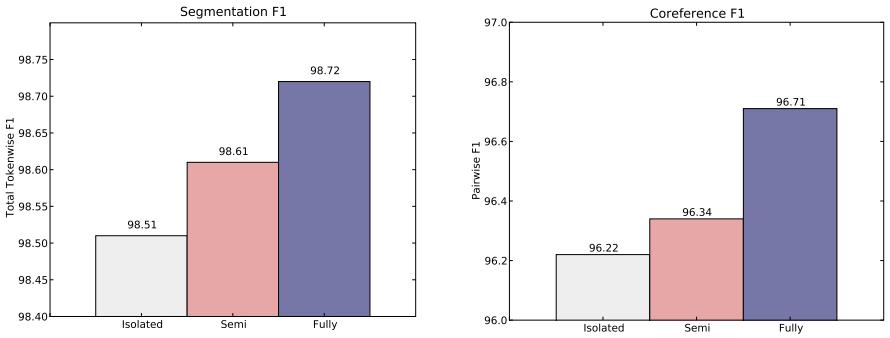
Method	**Author**	**Title**	**Venue**	**Total**
Isolated MLN	99.3	97.3	98.2	98.2
Joint MLN	99.5	97.6	98.3	98.4
Isolated IDF	99.35	97.63	98.58	**98.51**
Joint IDF	99.42	97.99	98.78	**98.72**

solves the segmentation task, outperforms our isolated coreference model, with a 13% error reduction compared to our isolated IDF. It also provides an overall 25.2% error reduction in pairwise coreference F1 in comparison to the joint MLN. In addition, Table 3 shows that the joint approach allows cluster recall to improve substantially, resulting in a 75.4% error reduction compared to the joint MLN, and a 61.5% error reduction compared to our isolated IDF.

Table 4 shows similar improvements on the segmentation task. Our isolated segmentation model significantly outperforms all earlier results, and the joint model uses the coreference predictions to improve segmentation further. In comparison to the joint MLN we provide an overall error reduction in token-wise segmentation F1 of 20.0%. Compared to our isolated IDF the reduction is 14.1%.

5.2 Bi-directionality

We also examine the performance as joint factors are added to our isolated models. The results are shown in Fig. 2. The isolated models produces the lowest scores amongst our models. "Semi-Joint" refers to the model containing the `JointInfBased` factors in addition to the isolated factors. They lead to a larger improvement in segmentation than in coreference, confirming their weaker effect on coreference proposals. When the fully bi-directional factors are also added ("Fully-Joint") both segmentation and coreference scores improve. However, the improvement for coreference is much higher. Recall that the factors added for the "Fully-Joint" model are prohibitively expensive to pre-calculate (as described in section 3.2), which demonstrates the benefit of using an IDF for bi-directional joint inference.



(a) Segmentation F1 (b) Coreference F1

Fig. 2. Adding Joint Factors: F1 of the joint model as different types of factors are added, starting with the base model containing only isolated segmentation and coreference factors. "Semi-Joint" refers to the model containing *weakly* joint factors while the "Fully-Joint" model consists of bi-directional highly-coupled factors.

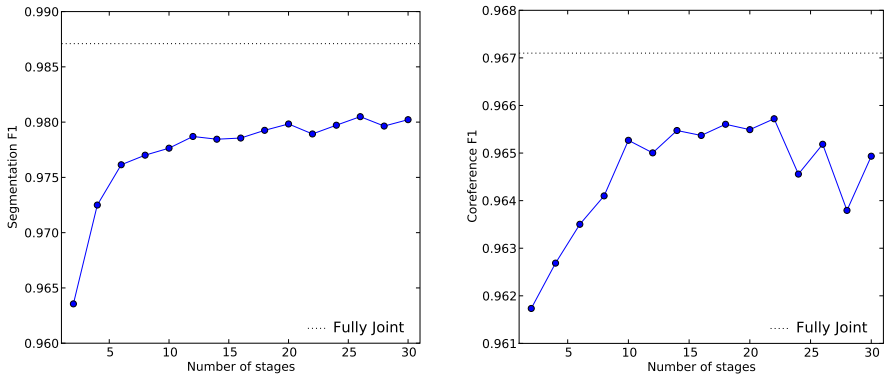
5.3 Iterated Pipeline Comparison

Conventionally, multiple information extraction tasks are solved using a pipeline architecture in which the output predictions of one task are used as input to the next. To minimize error caused due to cascading, multiple iterations of a pipeline can be carried out, such that the output predictions of the last stage of the pipeline feed back to the first stage.

As described earlier, we switch between segmentation and coreference proposals randomly. From the iterated pipeline perspective our method of performing joint inference is similar to repeating a pipeline in which each stage consists of a single proposal. To compare the performance of the fully joint proposal function against an iterated pipeline, we vary the number of pipeline iterations—by changing the total number of stages—while keeping the total number of training and testing proposals constant.

Our results are shown in Fig. 3. Each experiment involves 20 runs of three-fold cross validation using 5 loops of 250,000 training proposals and 750,000 testing proposals. The proposals are evenly divided across the stages, for example, the 2-stage experiment consists of 125,000 training proposals in segmentation followed by 125,000 training proposals in coreference for each of the 5 loops. In comparison, the 10-stage experiment consists of 5 pipelines, in which each pipeline has a segmentation stage of 12,500 proposals followed by a coreference stage of 12,500 proposals. Thus a higher number of total stages leads to a smaller number of proposals per stage. The first stage is always segmentation, to be consistent with earlier work in citation matching [15].

For both tasks our experiments show that the fully joint model gives higher F1 than any of the iterated pipeline results. Notice that the segmentation F1 rises as the number of stages in the pipeline increases. It is possible that segmentation



(a) Segmentation F1 (b) Coreference F1

Fig. 3. Iterated Pipeline: Performance of segmentation and coreference as the number of iterations of the pipeline is varied for a fixed number of total sampling steps over all iterations. The dotted line denotes the model that performs inference jointly, randomly switching between segmentation and coreference proposals.

F1 for a specific number of stages may be better than the fully joint results. However, finding the optimal number of stages can be expensive and we feel that a fully joint model is likely to perform competitively and generalize to a withheld validation set.

6 Related Work

Many researchers have explored issues of joint inference in text processing. McCallum and Jensen [1] present a position paper motivating the need for joint inference, and propose unified undirected graphical models for joint information extraction and data mining, describing several examples of conditional random fields. Many other papers present experimental results with various methods of inference. Sometimes joint inference can be done with standard dynamic-programming methods, for example, joint named entity recognition and parsing [18, 19] via CYK, with parse non-terminal symbols augmented to include named entity information. Another common alternative is feedforward N-best lists, *e.g.* [20, 21]. The feedforward probability distribution can be better approximated by sampling [2], but this flow of information is still uni-directional. Others have passed N-best list information bi-directionally between two tasks [3, 22]. Multi-directional passing of full probability distributions corresponds to loopy belief propagation, which has been used for skip-chains [23]. If joint factors can be expressed as linear constraints, one can employ efficient software packages for integer linear programming (ILP) [24]. When the structure of the model is changing during inference, MCMC provides significant efficiencies [8–10].

Joint citation segmentation and coreference has become somewhat of a standard evaluation task. Pasula *et al.* [15] perform this task using BLOG to define a generative model of research paper entities and their noisily-rendered citations; they perform inference by MCMC. Wellner *et al.* [3] bi-directionally pass N-best lists among conditional random fields for 14-field citation segmentation, coreference, and canonicalization. Poon and Domingos [4] avoid N-best lists with inference via MC-SAT in a Markov logic network. However, the tasks are weakly-coupled, do not enforce transitivity, and only segment into three fields. In this paper, for purposes of comparison, we perform the same three-field task. We leverage the efficient power of IDFs to define bi-directional joint factors that provide reduced error, faster running times, and enforce coreference transitivity.

7 Conclusions and Future Work

In this paper we presented a highly-coupled, bi-directional model for joint inference in citation segmentation and coreference, yielding new state-of-the-art accuracy. We incorporate factors that use coreference to aid segmentation and vice-versa, and do so efficiently using imperatively-defined factor graphs (IDFs). Compared to other joint models for the same tasks, our method results in an error reduction of $20 - 25\%$, providing a new state-of-the-art result, while also running $3 - 15$ times faster. In future work we will explore similar methods for joint inference in newswire named entity extraction and coreference, in which, unlike citations, the number of mentions must be inferred.

Acknowledgements

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #CNS-0551597, in part by Army prime contract number W911NF-07-1-0216 and University of Pennsylvania subaward number 103-548106, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and in part by UPenn NSF medium IIS-0803847. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

References

1. McCallum, A., Jensen, D.: A note on the unification of information extraction and data mining using conditional-probability, relational models. In: IJCAI Workshop on Learning Statistical Models from Relational Data. (2003)
2. Finkel, J.R., Manning, C.D., Ng, A.Y.: Solving the problem of cascading errors: Approximate bayesian inference for linguistic annotation pipelines. In: Conference on Empirical Methods on Natural Language Processing (EMNLP). (2006)
3. Wellner, B., McCallum, A., Peng, F., Hay, M.: An integrated, conditional model of information extraction and coreference with application to citation matching. In: Uncertainty in Artificial Intelligence (UAI). (2004) 593–601

4. Poon, H., Domingos, P.: Joint inference in information extraction. In: AAAI Conference on Artificial Intelligence. (2007) 913–918
5. Poon, H., Domingos, P.: Sound and efficient inference with probabilistic and deterministic dependencies. In: AAAI Conference on Artificial Intelligence. (2006)
6. Selman, B., Kautz, H., Cohen, B.: Local search strategies for satisfiability testing. Discrete Mathematics and Theoretical Computer Science (DIMACS) **26** (1996)
7. Kschischang, F., Frey, B., Loeliger, H.A.: Factor graphs and the sum-product algorithm. Information Theory, IEEE Trans on **47**(2) (Feb 2001) 498–519
8. Culotta, A., McCallum, A.: Tractable learning and inference with high-order representations. In: International Conference on Machine Learning (ICML) Workshop on Open Problems in Statistical Relational Learning. (2006)
9. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning **62**(1-2) (2006) 107–136
10. Milch, B., Marthi, B., Russell, S.: BLOG: Relational Modeling with Unknown Objects. PhD thesis, University of California, Berkeley (2006)
11. McCallum, A., Rohanimanesh, K., Wick, M., Schultz, K., Singh, S.: FACTORIE: Efficient probabilistic programming via imperative declarations of structure, inference and learning. In: NIPS Workshop on Probabilistic Programming. (2008)
12. Rohanimanesh, K., Wick, M., McCallum, A.: Inference and learning in large factor graphs with a rank based objective. Technical Report UM-CS-2009-08, University of Massachusetts, Amherst (2009)
13. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD). (2003)
14. Singla, P., Domingos, P.: Entity resolution with Markov logic. In: International Conference on Data Mining (ICDM). (2006) 572–582
15. Pasula, H., Marthi, B., Milch, B., Russell, S., Shpitser, I.: Identity uncertainty and citation matching. In: Neural Information Processing Systems (NIPS). (2003)
16. McCallum, A., Nigam, K., Rennie, J., Seymore, K.: A machine learning approach to building domain-specific search engines. In: IJCAI. (1999) 661–667
17. Poon, H., Domingos, P., Sumner, M.: A general method for reducing the complexity of relational inference and its application to MCMC. In: AAAI. (2008)
18. Miller, S., Fox, H., Ramshaw, L., Weischedel, R.: A novel use of statistical parsing to extract information from text. In: Applied Natural Language Processing Conference. (2000) 226–233
19. Finkel, J.R., Manning, C.D.: Joint parsing and named entity recognition. In: North American Association of Computational Linguistics (NAACL). (2009)
20. Gildea, D., Jurafsky, D.: Automatic labeling of semantic roles. Computational Linguistics **28** (2002) 245–288
21. Sutton, C., McCallum, A.: Joint parsing and semantic role labeling. In: Conference on Computational Natural Language Learning (CoNLL). (2005)
22. Hollingshead, K., Roark, B.: Pipeline iteration. In: Annual Meeting of the Association of Computational Linguistics (ACL). (2007) 952–959
23. Sutton, C., McCallum, A.: Collective segmentation and labeling of distant entities in information extraction. In: ICML Workshop on Statistical Relational Learning and Its Connections to Other Fields. (2004)
24. Roth, D., Yih, W.: Global inference for entity and relation identification via a linear programming formulation. In Getoor, L., Taskar, B., eds.: Introduction to Statistical Relational Learning, MIT Press (2007)