A Joint Model of Implicit Arguments for Nominal Predicates

Matthew Gerber and **Joyce Y. Chai**
Department of Computer Science
Michigan State University
East Lansing, Michigan, USA
{gerberm2,jchai}@cse.msu.edu

Robert Bart
Computer Science and Engineering
University of Washington
Seattle, Washington, USA
rbart@cs.washington.edu

Abstract

Many prior studies have investigated the recovery of semantic arguments for nominal predicates. The models in many of these studies have assumed that arguments are independent of each other. This assumption simplifies the computational modeling of semantic arguments, but it ignores the joint nature of natural language. This paper presents a preliminary investigation into the joint modeling of implicit arguments for nominal predicates. The joint model uses propositional knowledge extracted from millions of Internet webpages to help guide prediction.

1 Introduction

Much recent work on semantic role labeling has focused on joint models of arguments. This work is motivated by the fact that one argument can either promote or inhibit the presence of another argument. Because most of this work has been done for verbal SRL, nominal SRL has lagged behind somewhat. In particular, the "implicit" nominal SRL model created by Gerber and Chai (2010) does not address joint argument structures. Implicit arguments are similar to standard SRL arguments, a primary difference being their ability to cross sentence boundaries. In the model created by Gerber and Chai, implicit argument candidates are classified independently and a heuristic post-processing method is applied to derive the final structure. This paper presents a preliminary joint implicit argument model.

Consider the following sentences:[1]

(1) [c_1 The president] is currently struggling to manage [c_2 the country's economy].

(2) If he cannot get it under control, [p loss] of [arg_1 the next election] might result.

In Example 2, we are searching for the $iarg_0$ of *loss* (the entity that is losing). The sentence in Example 1 supplies two candidates c_1 and c_2. If one only considers the predicate *loss*, then c_1 and c_2 would both be reasonable fillers for the $iarg_0$: presidents often lose things (e.g., votes and allegiance) and economies often lose things (e.g., jobs and value). However, the sentence in Example 2 supplies additional information. It tells the reader that *the next election* is the entity being lost. Given this information, one would likely prefer c_1 over c_2 because economies don't generally lose elections, whereas presidents often do. This type of inference is common in textual discourses because authors assume a shared knowledge base with their readers. This knowledge base contains information about events and their typical participants (e.g., the fact that presidents lose elections but economies do not).

The model presented in this paper relies on a knowledge base constructed by automatically mining semantic propositions from Internet webpages. These propositions help to identify likely joint implicit argument configurations. In the following section, we review work on joint inference within semantic role labeling. In Sections 4 and 5, we present the joint implicit argument model and its features. Evaluation results for this model are given in Sec-

[1]We will use the notation of Gerber and Chai (2010), where standard nominal arguments are indicated with arg_n and implicit arguments are indicated with $iarg_n$.

tion 6. The joint model contains many simplifying assumptions, which we address in Section 7. We conclude in Section 8.

2 Related work

A number of recent studies have shown that semantic arguments are not independent and that system performance can be improved by taking argument dependencies into account. Consider the following examples, due to Toutanova et al. (2008):

(3) [*Temporal* The day] that [arg_0 the ogre] [*Predicate* cooked] [arg_1 the children] is still remembered.

(4) [arg_1 The meal] that [arg_0 the ogre] [*Predicate* cooked] [*Beneficiary* the children] is still remembered.

These examples demonstrate the importance of inter-argument dependencies. The change from *day* in Example 3 to *meal* in Example 4 affects more than just the *Temporal* label: additionally, the arg_1 changes to *Beneficiary*, even though the underlying text (*the children*) does not change. To capture this dependency, Toutanova el al. first generate an n-best list of argument labels for a predicate instance. They then re-rank this list using joint features that describe multiple arguments simultaneously. The features help prevent globally invalid argument configurations (e.g., ones with multiple arg_0 labels).

Punyakanok et al. (2008) formulate a variety of constraints on argument configurations. For example, arguments are not allowed to overlap the predicate, nor are they allowed to overlap each other. The authors treat these constraints as binary variables within an integer linear program, which is optimized to produce the final labeling.

Ritter et al. (2010) investigated joint selectional preferences. Traditionally, a selectional preference model provides the strength of association between a predicate-argument position and a specific textual expression. Returning to Examples 1 and 2, one sees that the selectional preference for *president* and *economy* in the $iarg_0$ position of *loss* should be high. Ritter et al. extended this single-argument model using a joint formulation of Latent Dirichlet Allocation (LDA) (Blei et al., 2003). In the generative version of joint LDA, text for the argument positions is generated from a common hidden variable. This approach reflects the intuition behind Examples 1 and 2 and would help identify *president* as the $iarg_0$. Training data for the model was drawn from a large corpus of two-argument tuples extracted by the TextRunner system, which we describe next.

Both Ritter et al.'s model and the model described in this paper rely heavily on information extracted by the TextRunner system (Banko et al., 2007). The TextRunner system extracts tuples from Internet webpages in an unsupervised fashion. One key difference between TextRunner and other information extraction systems is that TextRunner does not use a closed set of relations (compare to the work described by ACE (2008)). Instead, the relation set is left open, leading to the notion of Open Information Extraction (OIE). Although OIE often has lower precision than traditional information extraction, it is able to extract a wider variety of relations at precision levels that are often useful (Banko and Etzioni, 2008).

3 Using TextRunner to assess joint argument assignments

Returning again to Examples 1 and 2, one can query TextRunner in the following way:

arg_0 : ?

Predicate : lose[2]

arg_1 : election

In the TextRunner system, arg_0 typically indicates the *Agent* and arg_1 typically indicates the *Theme*. TextRunner provides many tuples in response to this query, two of which are shown below:

(5) Usually, [arg_0 the president's party] [*Predicate* loses] [arg_1 seats in the mid-term election].

(6) [arg_0 The president] [*Predicate* lost] [arg_1 the election].

The tuples present in these sentences give strong indicators about the type of entity that loses elections.

[2]Nominal predicates are mapped to their verbal forms using information provided by the NomBank lexicon.

Given all of the returned tuples, only a single one involves *economy* in the arg_0 position:

(7) Any president will take credit for [arg_0 a good economy] or [*Predicate* lose] [arg_1 an election] over a bad one.

In Example 7, TextRunner has not analyzed the arguments correctly (*president* should be the arg_0, not *economy*).[3] In Section 5, we show how evidence from the tuple lists can be aggregated such that correct analyses (5 and 6) are favored over incorrect analyses (7). The primary contribution of this paper is an exploration of how the aggregated evidence can be used to identify implicit arguments (e.g., *president* in Example 1).

4 Joint model formulation

To simplify the experimental setting, the model described in this paper targets the specific situation where a predicate instance *p* takes an implicit $iarg_0$ and an implicit $iarg_1$.[4] Whereas the model proposed by Gerber and Chai (2010) classifies candidates for these positions independently, the model in this paper classifies joint structures by evaluating the following binary prediction function:

$$P(+|\langle p, iarg_0, c_i, iarg_1, c_j\rangle) \quad (8)$$

Equation 8 gives the probability of the joint assignment of c_i to $iarg_0$ and c_j to $iarg_1$. Given a set of n candidates $c_1, \ldots, c_n \in C$, the best labeling is found by considering all possible assignments of c_i and c_j:

$$\underset{(c_i,c_j)\in C\text{x}C \text{ s.t. } i\neq j}{\arg\max} P(+|\langle p, iarg_0, c_i, iarg_1, c_j\rangle) \quad (9)$$

Consider modified versions of Examples 1 and 2:

(10) [c_1 The president] is currently struggling to manage [c_2 the country's economy].

(11) If he cannot get it under control before [c_3 the next election], a [p loss] might result.

In this case, we are looking for the $iarg_0$ as well as the $iarg_1$ for the *loss* predicate. Three candidates c_1, c_2, and c_3 are marked. The joint model would evaluate the following probabilities, taking the highest scoring to be the final assignment:

$$\begin{aligned}
&P(+|\langle \text{loss}, iarg_0, \text{president}, iarg_1, \text{economy}\rangle)\\
{*}&P(+|\langle \text{loss}, iarg_0, \text{president}, iarg_1, \text{election}\rangle)\\
&P(+|\langle \text{loss}, iarg_0, \text{economy}, iarg_1, \text{president}\rangle)\\
&P(+|\langle \text{loss}, iarg_0, \text{economy}, iarg_1, \text{election}\rangle)\\
&P(+|\langle \text{loss}, iarg_0, \text{election}, iarg_1, \text{president}\rangle)\\
&P(+|\langle \text{loss}, iarg_0, \text{election}, iarg_1, \text{economy}\rangle)
\end{aligned}$$

Intuitively, only the starred item should have a high probability. In the following section, we describe how these probabilities can be estimated using information extracted by TextRunner.

5 Joint model features

As mentioned in Section 2, the TextRunner system has been extracting massive amounts of knowledge in the form of tuples such as the following:

$$\langle \text{president}, \text{lose}, \text{election}\rangle$$

The database of tuples can be queried by supplying one or more of the tuple arguments. For example, the following is a partial result list for the query $\langle \text{president}, \text{lose}, ?\rangle$:

$\langle \text{Kenyan president}, \text{lose}, \text{election}\rangle$
$\langle \text{president's party}, \text{lose seat in}, \text{election}\rangle$
$\langle \text{president}, \text{lose}, \text{ally}\rangle$

The final position in each of these tuples (e.g., *election*) provides a single answer to the question "What might a president lose?". Aggregation begins by generalizing each answer to its WordNet synset (glosses are shown after the arrows):

$\langle \text{Kenyan president}, \text{lose}, \text{election}\rangle \rightarrow$ a vote
$\langle \text{president's party}, \text{lose seat in}, \text{election}\rangle$ (same)
$\langle \text{president}, \text{lose}, \text{ally}\rangle \rightarrow$ friendly nation

In cases where a tuple argument has multiple WordNet senses, the tuple is mapped to the most common sense as listed in the WordNet database.

[3] Banko and Etzioni (2008) cite a precision score of 88% for their system.

[4] This simplifying assumption does not hold for real data, and is addressed further in Section 7.2.

Having mapped each tuple to its synset, each synset is ranked according to the number of tuples that it covers. For the query $\langle \text{president}, \text{lose}, ? \rangle$, this produces the following ranked list of WordNet synsets (only the top five are shown, with the number in parentheses indicating how many tuples are covered):

1. election (77)
2. war (51)
3. vote (39)
4. people (34)
5. support (26)

   ...

The synsets above indicate likely answers to the previous question of "What might a president lose?".

In a similar manner, one can answer a question such as "What might lose an election?" using tuples extracted by TextRunner. The procedure described above produces the following ranked list of WordNet synsets to answer this question:

   ...

9. people (62)
10. Republican (51)
11. Republican party (51)
12. Hillary (50)
13. president (49)

   ...

In this case, the expected answer (*president*) ranks 13th in the list of answer synsets. It is important to note that lower ranked answers are not necessarily incorrect answers. It is a simple fact that a wide variety of entities can lose an election. Items 9-13 are all reasonable answers to the original question of what might lose an election.

The two symmetric questions defined and answered above are closely connected to the implicit argument situation discussed in Examples 10 and 11. In Example 11, one is searching for the implicit $iarg_0$ and $iarg_1$ to the *loss* predicate. Candidates c_i and c_j that truly fill these positions should be compatible with questions in the following forms:

Question: What did c_i lose?

Answer: c_j

Question: What entity lost c_j?

Answer: c_i

If either of these question-answer pairs is not satisfied, then the joint assignment of c_i to $iarg_0$ and c_j to $iarg_1$ should be considered unlikely. Using the first question-answer pair above as an example, satisfaction is determined in the following way:

1. Query TextRunner for $\langle c_i, \text{lose}, ? \rangle$, retrieving the top n tuples.
2. Map the final argument of each tuple to its WordNet synset and rank the synsets by frequency, producing the ranked list A of answer synsets.
3. Map c_j to its most common WordNet synset $synset_{c_j}$ and determine whether $synset_{c_j}$ exists in A. If it does, the question-answer pair is satisfied.

Some additional processing is required to determine whether $synset_{c_j}$ exists in A. This is due to the hierarchical organization of WordNet. For example, suppose that $synset_{c_j}$ is the synset containing "primary election" and A contains synsets paraphrased as follows:

1. election
2. war
3. vote

   ...

$synset_{c_j}$ does not appear directly in this list; however, its existence in the list is implied by the following hypernymy path within WordNet:

$$\text{primary election} \xrightarrow{\text{is-a}} \text{election}$$

Intuitively, if $synset_{c_j}$ is connected to a highly ranked synset in A by a short path, then one has evidence that $synset_{c_j}$ answers the original question.

The evidence is weaker if the path is long, as in the following example:

$$\text{open primary} \xrightarrow{\text{is-a}} \text{direct primary} \xrightarrow{\text{is-a}} \text{primary election} \xrightarrow{\text{is-a}} \text{election}$$

Additionally, a path between more specific synsets (i.e., those lower in the hierarchy) indicates a stronger relationship than a path between more general synsets (i.e., those higher in the hierarchy). These two situations are depicted in Figure 1. The synset similarity metric defined by Wu and Palmer (1994) combines the path length and synset depth intuitions into a single numeric score that is defined as follows:

$$\frac{2 * depth(lca(synset_1, synset_2))}{depth(synset_1) + depth(synset_2)} \qquad (12)$$

In Equation 12, *lca* returns the lowest common ancestor of the two synsets within the WordNet *is-a* hierarchy.

To summarize, Equation 12 indicates the strength of association between $synset_{c_j}$ (e.g., primary election) and a ranked synset $synset_a$ from A that answers a question such as "What might a president lose?". If the association between $synset_{c_j}$ and $synset_a$ is small, then the assignment of c_j to $iarg_1$ is unlikely. The process works similarly for assessing c_i as the filler of $iarg_0$. In what follows, we quantify this intuition with features used to represent the conditioning information in Equation 8.

Feature 1: Maximum association strength. Given the conditioning variables in Equation 8, there are two questions that can be asked:

Question: What did c_i p?

Answer: c_j

Question: What entity p c_j?

Answer: c_i

Each of these questions produces a ranked list of answer synsets using the approach described previously. The synset for each answer string will match zero or more of the answer synsets, and each of these matches will be associated with a similarity score as defined in Equation 12. Feature 1 considers all such similarity scores and selects the maximum. A high value for this feature indicates that one (or both) of the candidates (c_i or c_j) is likely to fill its associated implicit argument position.

Feature 2: Maximum reciprocal rank. Of all the answer matches described for Feature 1, Feature 2 selects the highest ranking and forms the reciprocal rank. Thus, values for Feature 2 are in [0,1] with larger values indicating matches with higher ranked answer synsets.

Feature 3: Number of matches. This feature records the total number of answer string matches from either of the questions described for Feature 1.

Feature 4: Sum reciprocal rank. Feature 2 considers answer synset matches from either of the posed questions; ideally, each question-answer pair should have some influence on the probability estimate in Equation 8. Feature 4 looks at the answer synset matches from each question individually. The match with highest rank for each question is selected, and the reciprocal rank $\frac{2}{r_1 + r_2}$ is computed. The value of this feature is zero if either of the questions fails to produce a matching answer synset.

Features 5 and 6: Local classification scores. The joint model described in this paper does not replace the local prediction model presented by Gerber and Chai (2010). The latter uses a wide variety of important features that cannot be ignored. Like previous joint models (e.g., the one described by Toutanova et al. (2008)), the joint model works on top of the local prediction model, whose scores are incorporated into the joint model as feature-value pairs. Given the local prediction scores for the $iarg_0$ and $iarg_1$ positions in Equation 8, the joint model forms two features: (1) the sum of the scores for c_i filling $iarg_0$ and c_j filling $iarg_1$, and (2) the product of these two scores.

6 Evaluation

We evaluated the joint model described in the previous sections over the manually annotated implicit

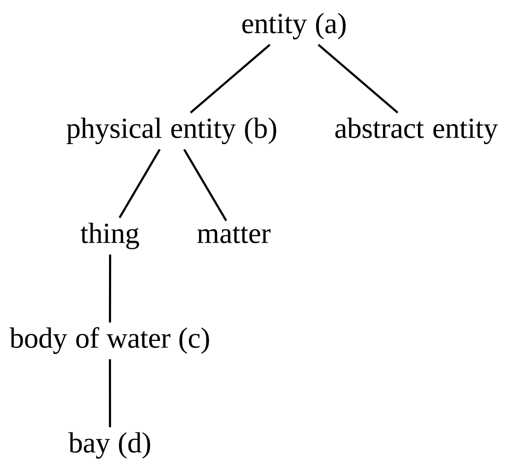

Figure 1: Effect of depth on WordNet synset similarity. All links indicate *is-a* relationships. Although the link distance from (a) to (b) equals the distance from (c) to (d), the latter are more similar due to their lower depth within the WordNet hierarchy.

argument data created by Gerber and Chai (2010). This dataset contains full-text implicit argument annotations for approximately 1,200 predicate instances within the Penn TreeBank. As mentioned in Section 4, all experiments were conducted using predicate instances that take an $iarg_0$ and $iarg_1$ in the ground-truth annotations. We used a tenfold cross-validation setup and the evaluation metrics proposed by Ruppenhofer et al. (2009), which were also used by Gerber and Chai. For each evaluation fold, features were selected using only the corresponding training data and the greedy selection algorithm proposed by Pudil et al. (1994), which starts with an empty feature set and incrementally adds features that provide the highest gains.

For comparison with Gerber and Chai's model, we also evaluated the local prediction model on the evaluation data. Because this model predicted implicit arguments independently, it continued to use the heuristic post-processing algorithm to arrive at the final labeling. However, the prediction threshold t was eliminated because the system could safely assume that a true filler for the $iarg_0$ and $iarg_1$ positions existed.

Table 1 presents the evaluation results. The first thing to note is that these results are not comparable to the results presented by Gerber and Chai (2010). In general, performance is much higher because predicate instances reliably take implicit arguments in the $iarg_0$ and $iarg_1$ positions. The overall performance increase versus the local model is relatively small (approximately 1 percentage point); however, the *bid* predicate in particular showed a substantial increase (greater than 11 percentage points).

7 Discussion

7.1 Example improvement versus local model

The *bid* and *investment* predicates showed the largest increase for the joint model versus the local model. Below, we give an example of the *investment* predicate for which the joint model correctly identified the $iarg_0$ and the local model did not.

(13) [Big investors] can decide to ride out market storms without jettisoning stock.

(14) Most often, [$_c$ they] do just that, because stocks have proved to be the best-performing long-term [*Predicate* investment], attracting about $1 trillion from pension funds alone.

Both models identified the $iarg_1$ as *money* from a prior sentence (not shown). The local model incorrectly predicted *$1 trillion* in Example 14 as the $iarg_0$ for the *investment* event. This mistake demonstrates a fundamental limitation of the local model: it cannot detect simple incompatibilities in the predicted argument structure. It does not know that "money investing money" is a rare or impossible event in the real world.

For the joint model's prediction, consider the constituent marked with c in Example 14. This con-

		Local model			Joint model		
	# Imp. args.	P	R	F_1	P	R	F_1
price	40	65.0	65.0	65.0	67.5	67.5	67.5
sale	34	86.5	86.5	86.5	84.3	84.3	84.3
plan	30	60.0	60.0	60.0	56.7	56.7	56.7
bid	26	66.7	66.7	66.7	78.2	78.2	78.2
fund	18	83.3	83.3	83.3	83.3	83.3	83.3
loss	14	100.0	100.0	100.0	100.0	100.0	100.0
loan	12	63.6	58.3	60.9	50.0	50.0	50.0
investment	8	57.1	50.0	53.3	62.5	62.5	62.5
Overall	182	72.6	71.8	72.2	73.1	73.1	73.1

Table 1: Joint implicit argument evaluation results. The second column gives the total number of implicit arguments in the ground-truth annotations. P, R, and F_1 indicate precision, recall, and f-measure ($\beta = 1$) as defined by Ruppenhofer et al. (2009).

stituent is resolved to *Big investors* in the preceding sentence. Thus, the two relevant questions are as follows:

Question: What did big investors invest?

Answer: money

Question: What entity invested money?

Answer: big investors

The first question produces the following ranked list of answer synsets (the number in parentheses indicates the number of answer tuples mapped to the synset):

money (71)

amount (38)

million (38)

billion (22)

capital (21)

As shown, the answer string of *money* matches the top-ranked answer synset. The second question produces the following ranked list of answer synsets:

company (642)

people (460)

government (275)

business (75)

investor (70)

In this case, the answer string *Big investors* matches the fifth answer synset. The combined evidence of these two question-answer pairs allows the joint system to successfully identify *Big investors* as the $iarg_0$ of the *investment* predicate in Example 14.

7.2 Toward a generally applicable joint model

The joint model presented in this paper assumes that all predicate instances take an $iarg_0$ and $iarg_1$. This assumption clearly does not hold for real data (these positions are often not expressed in the text), but relaxing it will require investigation of the following issues:

1. **Explicit arguments** should also be considered when determining whether a candidate c fills an implicit argument position $iarg_n$. The motivation here is similar to that given elsewhere in this paper: arguments (whether implicit or explicit) are not independent. This is demonstrated by Example 2 at the beginning of this paper, where *election* is an explicit argument to the predicate and affects the implicit argument inference. The model developed in this paper only considers jointly occurring implicit arguments.

2. **Other implicit argument positions (e.g., $iarg_2$, $iarg_3$, etc.)** need to be accounted for as well. This will present a challenge when it comes to extracting the necessary

propositions from TextRunner. Currently, TextRunner only handles tuples of the form $\langle arg_0, p, arg_1 \rangle$. Other argument positions are not directly analyzed by the system; however, because TextRunner also returns the sentence from which a tuple is extracted, these additional argument positions could be extracted in the following way:

(a) For an instance of the *sale* predicate with an arg_0 of *company*, to find likely arg_2 fillers (the entity purchasing the item), query TextRunner with $\langle$company$, sell, ?\rangle$.
(b) Perform standard verbal SRL on the sentences for the resulting tuples, identifying any arg_2 occurrences.
(c) Cluster and rank the arg_2 fillers according to the method described in this paper.

This approach combines Open Information Extraction with traditional information extraction (i.e., verbal SRL).

3. **Computational complexity and probability estimation** is a problem for many joint models. The model presented in this paper quickly becomes computationally intractable when the number of candidates and implicit argument positions becomes moderately large. This is because Equation 9 considers all possible assignments of candidates to implicit argument positions. With as few as thirty candidates and five argument positions (not uncommon), one must evaluate $30!/25! = 17,100,720$ possible assignments. Although this particular formulation is not tractable, one based on dynamic programming or heuristic search might give reasonable results. Efficient estimation of the joint probability via Gibbs sampling would also be a possible approach (Resnik and Hardisty, 2010).

8 Conclusions

Many prior studies have investigated the recovery of semantic arguments for nominal predicates. The models in many of these studies have assumed that the arguments are independent of each other. This assumption simplifies the computational modeling of semantic arguments, but it ignores the joint nature of natural language. In order to take advantage of the information provided by jointly occurring arguments, the independent prediction models must be enhanced.

This paper has presented a preliminary investigation into the joint modeling of implicit arguments for nominal predicates. The model relies heavily on information extracted by the TextRunner extraction system, which pulls propositional tuples from millions of Internet webpages. These tuples encode world knowledge that is necessary for resolving semantic arguments in general and implicit arguments in particular. This paper has proposed methods of aggregating tuple knowledge to guide implicit argument resolution. The aggregated knowledge is applied via a re-ranking model that operates on top of the local prediction model described in previous work.

The performance gain across all predicate instances is relatively small; however, larger gains are observed for the *bid* and *investment* predicates. The improvement in Example 14 shows that the joint model is capable of correcting a bad local prediction using information extracted by the TextRunner system. This type of information is not used by the local prediction model.

Although the results in this paper show that some improvement is possible through the use of a joint model of implicit arguments, a significant amount of future work will be required to make the model widely applicable.

References

ACE, 2008. *The ACE 2008 Evaluation Plan*. NIST, 1.2d edition, August.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of ACL-08: HLT*, pages 28–36, Columbus, Ohio, June. Association for Computational Linguistics.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. 2007. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan.

2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Matthew Gerber and Joyce Chai. 2010. Beyond NomBank: A study of implicit arguments for nominal predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1583–1592, Uppsala, Sweden, July. Association for Computational Linguistics.

P. Pudil, J. Novovicova, and J. Kittler. 1994. Floating search methods in feature selection. *Pattern Recognition Letters*, 15:1119–1125.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Comput. Linguist.*, 34(2):257–287.

Philip Resnik and Eric Hardisty. 2010. Gibbs sampling for the uninitiated. Technical report, University of Maryland, June.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2009. Semeval-2010 task 10: Linking events and their participants in discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 106–111, Boulder, Colorado, June. Association for Computational Linguistics.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Comput. Linguist.*, 34(2):161–191.

Zhibiao Wu and Martha Palmer. 1994. Verb semantics and lexical selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, pages 133–138, Las Cruces, New Mexico, USA, June. Association for Computational Linguistics.